Appendix B.
Ontological data models: a new database approach to improve archaeological data management and incorporate Fuzzy Set Theory

Introduction

In Chapters 6 and 7, we have seen that we can express the inexactness of our archaeological knowledge or our uncertainty about archaeological data using Fuzzy Set Theory (FST). If we express this knowledge, however, we should also take it into account in our analyses and conclusions as well (Niccolucci & Hermon 2017). This is only possible if the information about our uncertainty about our claims is available. This means that this information needs to be stored in our databases, especially if it is to be available to others who need it to evaluate our conclusions.

In traditional database approaches, based on relational databases, such information is often stored in comment fields. This presents several, related, problems. Information in comment fields is almost impossible to take along in analyses. Certainly, the database does not distinguish between an artefact that has been classified as 'type A' and one that bears the same classification, but has an attached comment saying that it could also be 'type B'. In subsequent analyses, both will be counted as certain instances of 'type A', losing the information about uncertainty and skewing results.

Furthermore, even if the conclusions are not directly skewed, it is very difficult for someone going back to the data to unearth information about exactness. This is because they need to be aware of the existence of the comment in order to read it. Since not all cases have such comments attached, it is almost unavoidable for a re-user to go back through the entire database; a time-consuming process.

Ontological data modelling, the database approach that I will discuss in this appendix, allows us to deal with these problems, as well as provide us with more

flexible and accurate tools for data handling and storage. Most importantly for this thesis, however, is that it provides the way by which FST's potential can be operationalised on larger datasets. So rather than introducing a second, unrelated, new tool for archaeologists to get acquainted with, this second tool supports the implementation of the first. By changing the way we approach databases, we can open up the way to incorporate FST into our analyses of larger datasets as well as ensure the durability of its benefits for archaeological knowledge creation. In the following sections, I will introduce the concept of the ontological data model, the way in which they work and how they differ from traditional relational databases.

Ontological data models

Why the name 'ontology'?

For those with a philosophical bent, 'ontology' is a confusing name for database approaches. There is little common ground between the philosophical study of being and this approach to data management. The name was chosen because, fundamentally, an ontological data model aims to describe the data as closely to how they are in reality as possible (Gruber 1995). In traditional relational databases data are fitted into a structure which is designed to facilitate answering certain questions. With an ontological data model, the nature of each datum is described as accurately as possible and it is these descriptions which form the information stored in the system. This is still done using a structure, but the structure is designed to represent the data, rather than the data being fitted into a structure. To illustrate the difference, let us consider sixteen sherds that used to belong to a single vessel. A relational database would archive this as a single vessel using a data field 'number of fragments' to register the value '16'. An ontological data model would describe the sixteen sherds individually, as this is how they have come to us, and for each sherd record the relation of it having once been part of the original single vessel. An individual description such as this allows for more accurate descriptions of the data, which, in some cases, could enable subtler research questions to be answered.

Building an ontological data model

The basic concepts of ontological data models are *classes*, *properties* and *individuals*. Formal definitions and discussions of the building blocks of ontological data models can be found at the OWL specification section of the World Wide Web Consortium's website (W3C, see list of abbreviations). Classes are used to describe entities in reality. Concepts, such as 'tree' or 'archaeological context', can all be classes in ontological data models. Not all classes are equally general. To reflect this, classes are organised in a class hierarchy. 'Beech tree' can be a class, but it is more specific than 'tree'. In a class hierarchy, it would therefore be lower down than 'tree' but connected to it through a hierarchical relation. The hierarchical relations between a class and classes higher up in the hierarchy are 'is a' relations. For example, a beech tree is a tree and this is expressed by 'tree' appearing above 'beech tree' in the hierarchy. All trees (including beech trees) are plants though. This means that 'plant' is a superclass of 'tree'. Because all hierarchical class relations are 'is a' relations, it means that 'beech tree' is a subclass of 'plant', just like 'tree' is. This is, of course, exactly as we would want it, since it mirrors our conceptualisation of reality. As a consequence of this, anything defined as a property of a class is also a property of all its subclasses. This inheritance principle means that we can efficiently define our model, since we only have to define properties at the highest level in the hierarchy where they are appropriate. Inheritance ensures that it is propagated down the hierarchy.

Organising concepts in this way ultimately yields a class hierarchy that branches out downwards into ever more specific subclasses and concentrates upwards to ever more general concepts. At the very top we have the class 'thing', which is about as general as we can get.

Defining the classes is an important step in designing a data model. In order to optimise reusability and interoperability, standards have been developed for class definitions. An important one for the field of archaeology is the CIDOC CRM (conceptual reference model, http://www.cidoc-crm.org). In this CRM, concepts are described and defined in a standardised way. These definitions can be imported from the CIDOC CRM into your ontologies, which will ensure that the ontology is compatible with others using the same concepts, but it also saves a lot of re-inventing the wheel. In the CRM, concepts are described using description logic. This means that they are somewhat difficult to read for people used to natural language. An important advantage of using description logic is that it is machine readable. This possibility of

vocabulary being machine readable means that many tasks can be automated, for example data importing.

Creating class hierarchies is all well and good, but for it to be useful, an ontological data model needs to be able to store data. We are not (just) interested in the concept of an amphora, we want to record specific amphorae. Classes are defined in the abstract, but they are concretised through their *instances*. A particular amphora is an instance of the class amphora. We use the ontological data model to describe instances. So if we are recording a Dragendorff 16 plate, we say (among other things) that this is an instance of the class 'plate' and an instance of the class 'Dragendorff 16 vessel'.

Because of the inheritance principle, an instance of a class is automatically also an instance of its superclasses and inherits all the properties defined for these. We will always try to be as specific as possible with our ascriptions, in order to record as much information as possible. It sometimes happens, however, that we know that a sherd is a fragment of *terra sigillata*, but cannot pinpoint a specific form type due to fragmentation or the condition the sherd is in. In this case we record it as an instance of a higher level class, meaning that it inherits all the properties of its superclasses, but not those defined for the more detailed appellations. This is, of course, completely in agreement with intuition and current practice.

Ontological data models do not just consist of a class hierarchy. They are defined in two planes: the 'vertical' class hierarchy and a 'horizontal' plane of relations between them. This horizontal plane consists of *object properties* and *data properties*. Object properties are used to define the relations between classes, whereas data properties link instances to specific values, such as weights or dimensions.

Object properties are used to link between classes in different branches of the hierarchy. That is, they describe relations between classes that are not related hierarchically through 'is a' relations. For example, the object property 'made' links instances of the class 'maker' to those of the class 'artefact'. The reverse is also true, if a maker made an artefact, then the artefact was made by the maker. Object properties have inverse versions in which the order of instances is reversed (see the discussion of triple statements below). These inverse properties often do not need to be explicitly defined. Only when they become necessary do they need to be made explicit. Before that time, they exist implicitly. Once they are defined as the inverse of an existing property, the system can automatically apply the inverse property to those instances between which the original property was defined.

Because of the inheritance principle, an object property that connects instances of two classes, can automatically also connect instances of their subclasses. This means that when we define an object property ‘has_Form_Type’ that links an artefact to an instance of the class ‘Form_Type’, which denotes typological classifications, we do not need to specify that this applies to every single sub-variant within every typology, because these would inherit this potential link from their superclass. The result of this is that the model can be efficiently defined with as little doubled specifications as possible.

It is also possible to define a single object property, but use it multiple times, not only for different instances of the class, but also to link different classes. For example, if we have the object property ‘was_found_in_Archaeological_Context’ linking artefacts to the context they were found in, rather than define a separate object property to document the relation between skeletal material and the context that was found in, we can modify the object property ‘was_found_in_Archaeological_Context’ to not only allow it to apply to instances of the class ‘Artefact’, but also to instances of the class ‘Skeletal _Material’. We can expand the scope of a property by specifying extra classes it can link to others, known as its domain. In Figure B.1 we can see that the object property ‘was_found_in_Archaeological_Context’ has two classes as its domain, as it links both of those to their ‘Archaeological_Context’, which is its range.

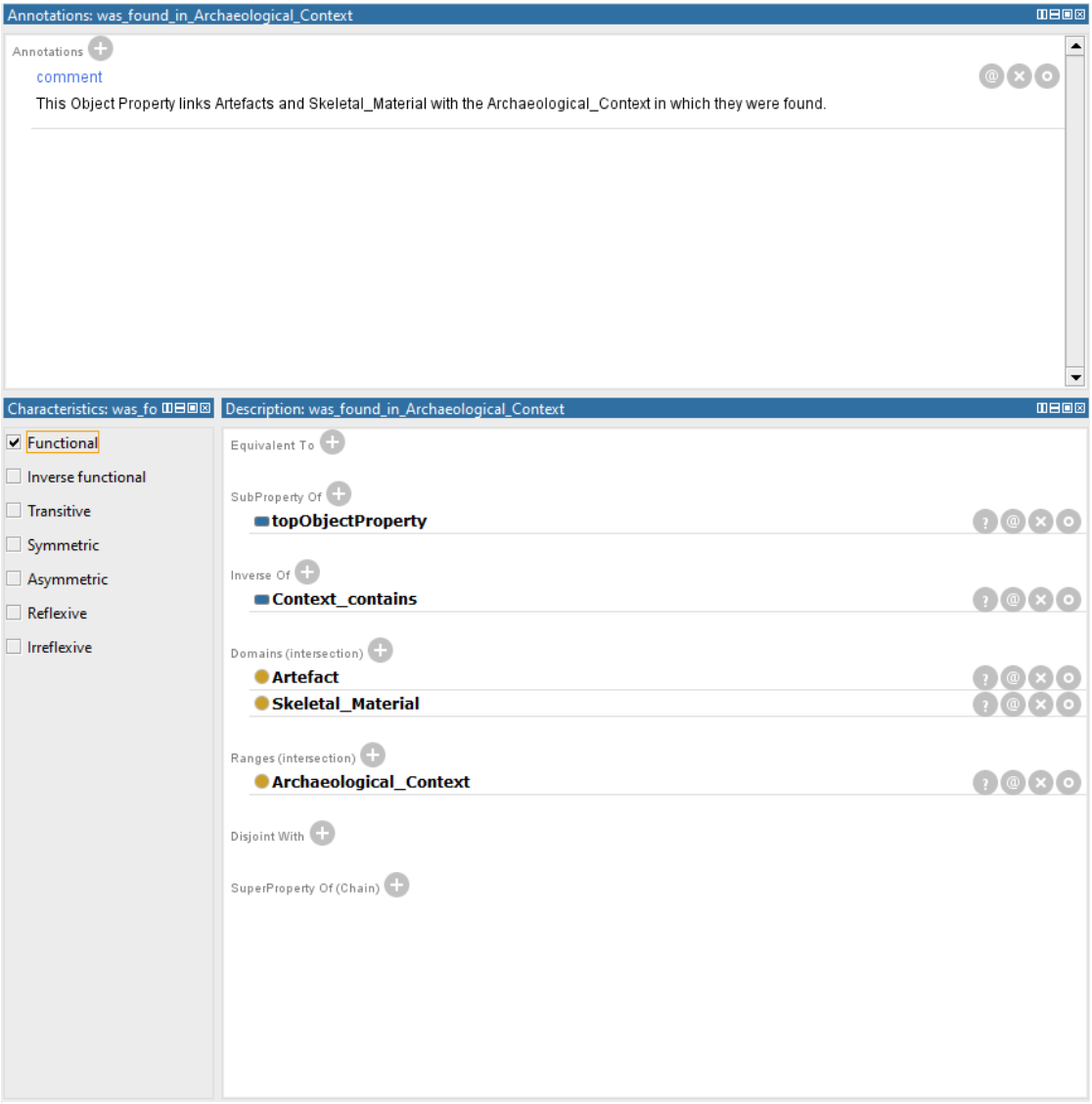

Figure B.1: Screenshot from the Protégé ontology editor showing the definition of the 'was_found_in_Archaeological_Context' object property. Note the double domains and the fact that it has its inverse property specified.

Conversely, an object property can also link an instance of a single class to instances of multiple other classes. In this case, the object property is defined with multiple classes as its range. If we were to take the inverse of Figure B.1, we would get the object property 'Context_contains', which has a single domain 'Archaeological_Context' and multiple classes as its range.

Data properties are used to link instances of classes to specific values. If we wanted to document an artefact's weight, we could use, for instance, a data property 'has_weight' to link the artefact to a numerical field, in which we register its weight. Of course, we would need to specify the units in which this weight is measured, but this can be done in the definition stage. As with object properties, data properties can have multiple classes as a domain. This means we do not need to re-define data properties

for every single class, we can simply include more classes in the domain of a data property.

Instances of classes, properties and data property values are combined into RDF (Resource Description Framework) triples, or triple statements. These triple statements are the form in which actual data gets stored in an ontology. A triple statement consists of a subject, a predicate and an object. The subject of an RDF triple is always an instance of a class. The predicate is formed by either an object property or a data property. Depending on whether the predicate was an object property or a data property, the object of a triple statement is an instance of a class or a data value respectively. Returning to Figure B.1, a hypothetical triple statement based on this would be 'Artefact_1', 'was_found_in_Archaeological_Context', 'Context_1' (subject, predicate, object).

RDF triples map out relations in the three-dimensional network of the data model. Seen in this way, RDF triples are actually a graph. As it is triple statements that get stored in the database, what we are actually storing is a 3D graph mapping the ways in which the data we are documenting relate to one another. If defined appropriately, this is as accurate a description of the nature of data as possible.

Instances of classes can be linked through multiple properties. Archaeological contexts can contain multiple artefacts (documented by object properties) as well as have dimensions (documented by data properties). In all these properties the corresponding instance of the 'Archaeological_Context' class acts as the subject of the triple statement (or the object of the statement if the inverse object property is used). The values that are the object of triple statements using data properties as their predicate are not re-used in the same way, but indicate unique values that characterise this particular case. The same value could, of course, be used multiple times, but each data property triple statement refers to a unique value, whereas when object properties refer to the same thing, they refer to the same instance. So all triple statements asserting that a sherd is of a Dragendorff 16 type link to the 'Dragendorff 16' instance of the 'Form_Type' class, whereas all triple statements asserting that a sherd weighs 40g link to individual values.

In some cases, properties are defined such that only one unique value is possible. For example, in western countries, a person is allowed only one legal spouse or registered partner. In this case, a *functional property* is defined, which limits the number of individuals to which a person can be linked using this property (W3C).

Similarly, an artefact can only have been found at one site, so this property would be an example of a functional property.

Another special type of property is the so-called transitive property. Declaring a property as transitive means that when this property is used to link a pair of instances *a* and *b* and a pair *b* and *c*, that *a* and *c* are also linked by this property. This can be illustrated with the object property 'is_part_of'. An archaeological context can be reconstructed as being part of an ancient building (e.g. a posthole for a centurion's quarters) while the building is part of a larger structure (e.g. a barrack block) and so on. The archaeological context in this example is part of both the building and the larger structure. Thanks to the transitive nature of the object property 'is_part_of', this does not have to be specified separately, but is resolved through the internal logic of the model.

Advantages of ontological data models

When characterising ontological data models, above, I noted that they allow a description of data close to their being. This was phrased in opposition to traditional relational database approaches. This subsection discusses further advantages of using ontological data approaches. Many of these are advantages over relational databases, but they are also beneficial in isolation (Cf. Martinez-Cruz 2012).

One of the benefits of ontologies is that they are very flexible tools. Traditionally, once data has been entered into a database, the structure of the database cannot be altered without potentially compromising its functionality. With an ontological data model, it is possible to define new relations in the model. This will require extra work, but provided sufficient care is taken, there are no fundamental obstructions to changes to the structure after it has been populated. Such flexibility is crucial when collating several datasets for analysis. Ontological data models allow several existing databases to be spliced into a single data model, so long as clear mapping rules are defined. This facilitates data re-use as datasets can be combined and data models carefully adapted to new research questions. Theoretically, any new project that a researcher embarks upon could benefit from data from earlier projects as new data can be added to an existing database while the old data is made available for new questions by adapting the model structure.

The splicing of data files into one ontological data model can be automated. With the proper programming knowhow, code scripts can be written which detail, for instance, the column to property relations, such a script can transfer data from a spreadsheet into an ontology. Given the magnitude of archaeological spreadsheets even for excavations or research projects of modest size, this means that transferring to this new ontological database approach does not only provide theoretical advantages, but is also realistic practically, since it need not involve hundreds of hours of data transfer. Of course, one would still need (access to someone with) the required programming skills.

Perhaps the greatest advantage of ontological data models results from the way in which such models are constructed. Because the data model is like a 3D map and the data gets input and stored as a graph denoting certain connections within the data, new and expanded querying options become available to the researcher. For instance, it becomes possible to query relationships between, for example, two artefacts, which is not supported in relational databases (cf. Turbek 2008).

Because it is possible to query and search the structure of the model itself, it is also possible to find information that was implicit in the data, but not readily accessible. Through Description Logic reasoners, the system is able to suggest links that were not apparent to the researcher, but which were present in the data. For instance, the system could suggest that duplicates exist within the model, based on similar identifiers. This is a valuable feature for a data model, especially when splicing together several databases, since duplicates become a risk in such cases.

The graph-like nature of an ontological data model is exploited in the Visual Query Interface developed by Yi Hong (see van Helden *et al*. 2018). This tool was designed to be an accessible interface between the non-IT-specialist and native query languages used to query databases. Because query languages are rather complex, it is often difficult for non-specialists (such as archaeologists) to write syntactically correct queries. To mitigate this difficulty, user interfaces, such as search- or query forms, are often incorporated to allow non-specialists to create queries without having to write proper syntaxes. The problem with most of these interfaces is that they restrict the query options of the user. Hong developed the VQI to be as expressive as possible, while still removing the need to personally write syntaxes. By using the graph-like nature of the ontological data model, the VQI asks the user to draw the graph structure they are interested in in yEd, a free to download graph editing program. This graph is then translated into a SPARQL (a native query language) query, which is used to

retrieve the results from the actual data model and present them in an Excel compatible format. In this way, the nature of ontological data models allows more expressive query writing for non-specialists, expanding the usefulness of data models, since they can be used for an increasingly wide array of questions.

Incorporating fuzziness

As referred to above, Niccolucci and Hermon have called attention to the importance of expressions of our confidence, or lack thereof, in our archaeological classifications being useably documented alongside the data we are classifying. They have made a suggestion of how this could be defined in Description Logic and CIDOC CRM concepts (Niccolucci & Hermon 2017: 283-284), but they do not explicitly discuss how this would be implemented in actual data models. A very promising way in which this could be incorporated into an ontological data model is through reification vocabulary. Reification vocabulary allows the user to treat triple statements as the subject of another triple statement. It provides the means to attach notes to asserted relations.

This is important, because the traditional way of attaching notes often does not work in ontological data models. In relational databases, a few central elements of the data are unique and information about these unique elements is documented in tables. So an artefact identification number will be unique and for each such number a typological classification will be documented. If two artefacts are of the same typological type, this is documented twice, once per artefact. As such, it is possible to attach a note to the documented typological identification and register the confidence, or uncertainty, about the classification in this note (of course with all the drawbacks of hiding such information in notes, discussed above).

In ontological data models, the same information would be documented with the RDF triple connecting, for example, an instance of the class ‘Artefact’ to the specific instance of the class ‘Typological_Classification’. If two artefacts are of the same typological type, both are linked with the same instance of ‘Typological_Classification’. There is therefore no way to differentiate between the two uses of the instance of ‘Typological_Classification’ other than through the triple statements. Any judgement of uncertainty or reliability of a characterisation that would

need to be documented needs to be related to the RDF triple which documents that characterisation. This is why reification vocabulary is an important tool.

As said, reification vocabulary allows an RDF triple to be used as the subject of a new triple statement. Figure B.2 gives a visual representation of the process by which a judgement of uncertainty could be documented by reification vocabulary. Because the uncertainty coefficient is expressed numerically and standardised, any queries in which the characterisation of the original triple statement comes up as a result can be modified according to the stated coefficient, or at least it can be automatically flagged up.

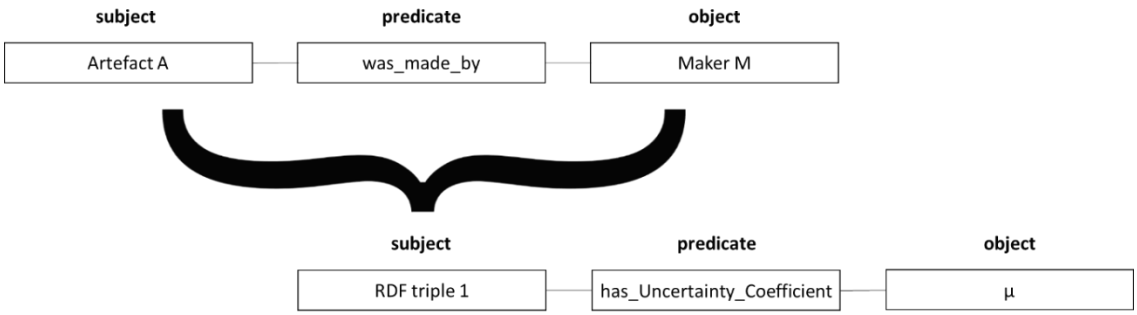

Figure B.2: Visual representation of reification vocabulary being used to document uncertainty of an attribution of a maker to an artefact.

Technically, the way this can be done is by specifying a new class, for instance 'RDF triple', for which three object properties are defined: 'has_Subject', 'has_Predicate' and 'has_Object'. The first statement in Figure B.2 could be represented by 'RDF triple 1', which is linked through the three object properties to the relevant instances of the relevant classes. Then 'RDF triple 1' could be used as the subject for a new triple statement. This is a slightly more convoluted way of representing data, meaning that there is an argument to be made to restrict this method to those elements where it is necessary, but fundamentally, the structure of the data model is unaffected.

Sample ontology

To illustrate the benefits of ontological data models, I have constructed such a model using the Stanford Protégé ontology editor. Protégé is a free to download, open-source and cross-platform ontology editing program. The data for the data model were drawn from two datasets from Roman *Arae Flaviae*, modern day Rottweil in Germany. The data come from two publications, Regina Franke's (2003) publication of the excavations within the area of Forts I and II in the Nikolausfeld area, digitised by

Penelope Allison for her Engendering Roman Spaces project (Allison 2013), and Robert Fecher's (2010) publication of the Kapellenösch cemeteries, for which he provided me access to the digital catalogue. At this stage, the data themselves are not of interest. Description of their nature and context is done in Chapter 7, where this is of relevance for the discussion of fuzzy set analytical approaches.

Caveats

It is important at this point to be frank about the limitations of this specific ontology. It does not incorporate some of the features discussed in this appendix or the thesis. This is in large part due to the priorities I had when researching and writing Part III of this thesis. Rather than producing a solution that was ready to roll out and be adopted by archaeology as a panacea, my goal has been to explore and explain potential and to spread enthusiasm, which will hopefully lead to further work being done down the line. It is at that stage that all of the potential discussed in this part of the thesis will, all being well, be integrated into a grand multipurpose tool.

Despite my description of the benefits of using standardised CIDOC CRM vocabulary to define classes and properties, the sample ontology does not consistently do so. The CIDOC CRM has been consulted in the process of designing the data model, but many of my own vocabulary remains. This is due to the fact that in the process of learning the ideas behind ontological data models and their design, I was already putting together an ontology, learning as I went (see van Helden *et al*. 2018). This practice ontology formed the backbone of the current sample ontology, because I judged the time required to develop and fine-tune a new one could be more fruitfully spent on exploring wider potential rather than pursuing perfection in a limited area.

As it stands, the ontology discussed in this appendix also does not incorporate reification vocabulary. This is partly because, as is, Protégé does not support reification vocabulary in a user-friendly way. Such hurdles are not insurmountable, though. More important for this ontology not including it at the moment is that it is not necessary for the analyses illustrated in Chapter 7 of the thesis. The datasets do not, of themselves, employ numerical μ-values to express uncertainty. This means that, in order to incorporate this information through reification vocabulary, it would need to be 'translated' first, before being entered into the data model using reification vocabulary.

All of this is possible, but since this information does not get used in Chapter 7's analyses, the effort spent on it would yield no further results than the knowledge that it can be done. This is not significantly more useful to the reader of this thesis than my explanation in the previous section. I therefore decided to focus in this thesis on the exploration and explanation of the potential of both ontological data models and FST and defer the actual incorporation of reification vocabulary to a 'further work' section and with this, postponing the true utility of this dataset to a future project, where this work will have to be done.

Finally, it is important in this section to note that for the analyses discussed in Chapter 7, this ontology is not crucial, at least not for that chapter's discussion. This means that the fuzzy sets used in Chapter 7 and their associated μ-values are not stored in the ontology. Furthermore, an automated dating algorithm is not incorporated into a front-end data entry interface as suggested Chapter 7, since no such interface has been designed. Again, this is because the current aim is to discuss potential and raise the profile of both ontological data models and FST. Because I cannot predict the uses others might want to put ontological data models and FST to, this seems a more productive avenue than to present a polished product with limited options. Hopefully, half measures presented with enthusiasm will inspire more work in these fields than finished black boxes where no further work is required.

The design

Describing a data model in natural language is rather inefficient. This would result in a long list of often obvious descriptions with a lot of repetition of standard phrases. The structure of the data model is much more easily understood visually for the horizontal plane and from an interactive hierarchy for the vertical. Therefore, the horizontal plane of the ontological data model is detailed in Figure B.3. A digital version of this is available as Appendices C and D (in respectively .JPG and .GML (yEd's format)). The first is just a digital version of Figure B.3. The second is a Graph Markup Language file, which means it can be manipulated in yEd, which makes some of the properties more easily distinguishable. For the vertical dimensional, Appendix E is the OWL file, readable in Protégé, in which the class hierarchy can be explored by extending and collapsing the tree.

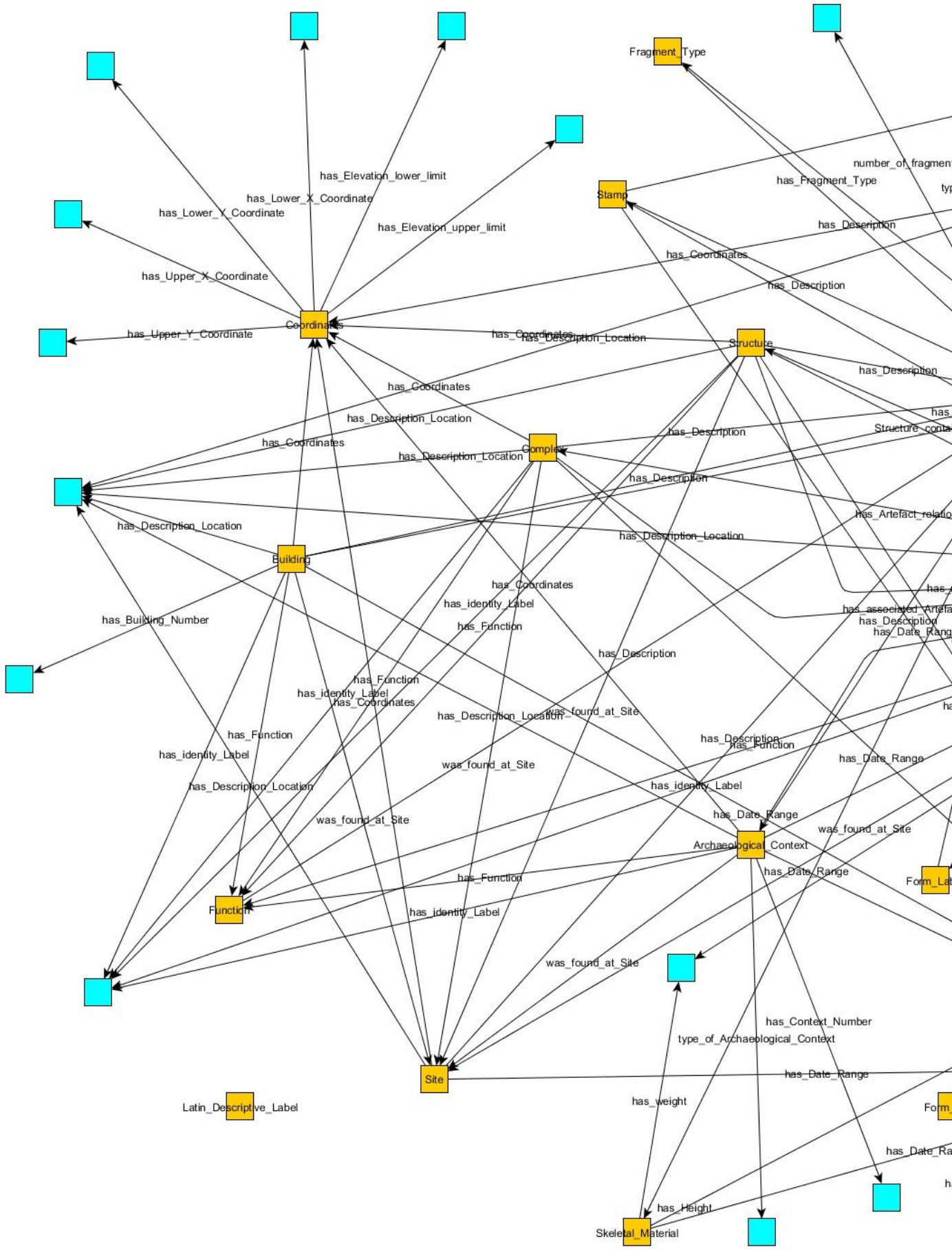
Fragment_Type
has_Elevation_lower_limit
has_Lower_X_Coordinate
has_Lower_Y_Coordinate
has_Elevation_upper_limit
Stamp
has_Fragment_Type
has_Description
has_Coordinates
has_Upper_X_Coordinate
has_Description
Coordinates
has_Upper_Y_Coordinate
has_Coordinates
has_Description_Location
Structure
has_Description
has_Coordinates
has_Description_Location
has_Description
has_Coordinates
Complex
has_Description_Location
has_Description
has_Description_Location
has_Description_Location
Building
has_Coordinates
has_identity_Label
has_Building_Number
has_Function
has_Description
has_Function
has_identity_Label
has_Coordinates
has_Description_Location
was_found_at_Site
has_Function
has_identity_Label
has_Description_Location
was_found_at_Site
has_Description
has_Function
has_Date_Range
has_identity_Label
was_found_at_Site
has_Date_Range
Archaeological_Context
was_found_at_Site
has_Date_Range
has_Function
Function
has_identity_Label
was_found_at_Site
has_Context_Number
type_of_Archaeological_Context
Site
has_Date_Range
Latin_Descriptive_Label
has_weight
has_Height
Skeletal_Material

Figure B.3: Overview of the horizontal plane of the ontological data model. Arrows represent properties and are labelled with the name of the property. Yellow nodes are classes, cyan ones are information entered in data properties. Where an arrow connects two classes (yellow nodes), it represents an object property. If it connects a yellow node to a cyan one, the arrow represents a data property. Cyan nodes are generally not labelled because they just have individual data in them (e.g. an amount in centimetres), whereas yellow nodes are labelled with their class name. Properties that have 'Thing' as their domain or have RDF triples as their domain are not shown as arrows. This explains, for example, why the cyan nodes in the top right-hand corner are not connected to any yellow nodes, as the information in them pertains to other triple statements. Modified from van Helden et al. *2018, figure 5*

Conclusions

In this appendix we have seen the role ontological data models can play in archaeology. By describing data in a way that is as close to their nature as possible, they provide a more accurate description than current approaches often do. Analogous to the way in which FST allows for our analytical concepts to more closely approximate the concepts of our thinking, ontological data models allow data description to resemble the way they are given to us in reality.

The flexibility of ontological data models allows us to adapt models after data have already been entered. Of course, extra care needs to be taken in such cases, but the existence of this possibility opens up options for much wider re-use of datasets. When combined with agreed-upon vocabulary of conceptual reference models (such as CIDOC) and the relative ease with which (with appropriate IT support) data models can be spliced into one, the potential for accumulating larger datasets through combining different ones becomes apparent. In theory, a researcher could use all of their data collected for earlier projects on a new project, adapting the data model to fit their new questions.

Most importantly in the context of this thesis, however, ontological data models also have an important role to play in the incorporation of FST into archaeology. Through the use of reification vocabulary, ontological data models can be used to store important information about the certainty, or lack thereof, of characterisations of the data, for example typological identifications. This information, that is often lost or glossed over in traditional databases, is of vital importance to accurately represent archaeologists' knowledge of the data. Without this information, the analyses and interpretations based on the data collected could be seriously skewed, or at least misrepresented as better founded than was actually the case.

By adopting ontological data models as a tool to describe and store archaeological data, we would be embracing an approach that more accurately describes the nature of

the data as they exist in reality, but is also more flexible than the systems that are currently widespread. This in itself might be reason enough to consider their incorporation into archaeological practice, but when we consider how their use would enable fuzzy set approaches in archaeology by allowing the recording and storage of the crucial μ-values, the case becomes stronger still. By dovetailing together, Fuzzy Set Theory and ontological data approaches strengthen the argument for each other's adoption. This is also why these two concepts which, at first glance, seemed to be quite distinct should really be seen together.