B usiness
O bject
R eference
O ntology
Program

simplifying semantics

Working Paper

MW3

Methodology: Worked Example - 3

Re-Engineering Bank Address

Issue: Version - 4.01 - 01-July-2001

Contact

For queries regarding this document, or the BORO Program in general, please use the following email address:

contact@BOROProgram.org

MW3

METHODOLOGY: WORKED EXAMPLE - 3

RE-ENGINEERING BANK ADDRESS

CONTENTS

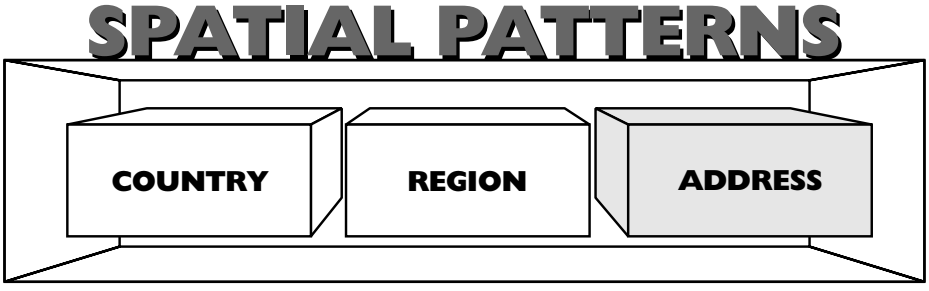
CONTENTS
MW3

RE-ENGINEERING BANK ADDRESS

1 Introduction

We have completed the re-engineering of the first two examples of spatial patterns; country and region. We now move onto the third and final example, address (see *Figure MW3–1*). This example illustrates the power of the basic object model for spatial patterns we have constructed. It shows how it can be re-used to capture the patterns for address, without any changes. It also provides us with an opportunity to construct a truly general model for the naming pattern.

Figure MW3–1
Third and final of three examples of spatial patterns



As you are now quite familiar with the systematic re-engineering approach, we take a high level view. Even though address's entity format is quite different from

the earlier formats, its re-engineering follows the same basic pattern. Like before, we re-use patterns that we have already re-engineered; and, as you have now come to expect, this means we generalise them.

2 Familiarising ourselves with bank address entity formats

We start off, as always, by familiarising ourselves with specific examples of the entity format we are going to re-engineer. *Table MW3–1* has a partial address listing that we use to do this. This is really a partial view of the address fields on the bank file—in other words, the attributes for the bank entity type. We call it bank address, because we are only interested in bank address's spatial patterns. However, we must re-engineer bank to get to bank address.

Table MW3–1: Partial address listing

Bank	001	BarcWest Bank	003	Chase Hanover Bank
Address		Black Horse House Moorgate London United Kingdom		BarcWest Tower Old Broad Street London United Kingdom
Bank	002	NatLand Bank	004	Banco di Guernsey
Address		Listening Buildings 21 Moorgate London England		54th Floor BarcWest Tower Old Broad Street London England

From *Table MW3–1*, we can work out that the entity format for bank address (bank) that is shown in *Table MW3–2*.

Table MW3–2: Bank address entity format

Entity type	Bank
Attribute type #1	Bank code
Attribute type #2	Bank full name
Attribute type #3	Address line 1
Attribute type #4	Address line 2
Attribute type #5	Address line 3
Attribute type #6	Address line 4
Attribute type #7	Address line 5

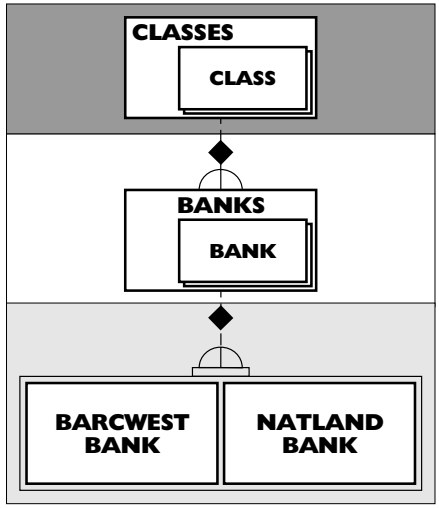
3 Re-Engineering bank

We start by re-engineering, in outline, the bank elements of the entity format. In this example, we look at the bank entity type sign and its attribute types, bank full name and code.

3.1 Re-engineering the bank entity type sign

We follow the rules and start by re-engineering the entity type sign before the attribute type signs. We are not really interested in bank because it is not a spatial or naming pattern; it is just a hook to hang address's spatial patterns on; so, we move through the re-engineering quickly. We re-engineer two individual banks and use these to construct the banks class. We construct signs for these classes and get the object schema in *Figure MW3–2*.

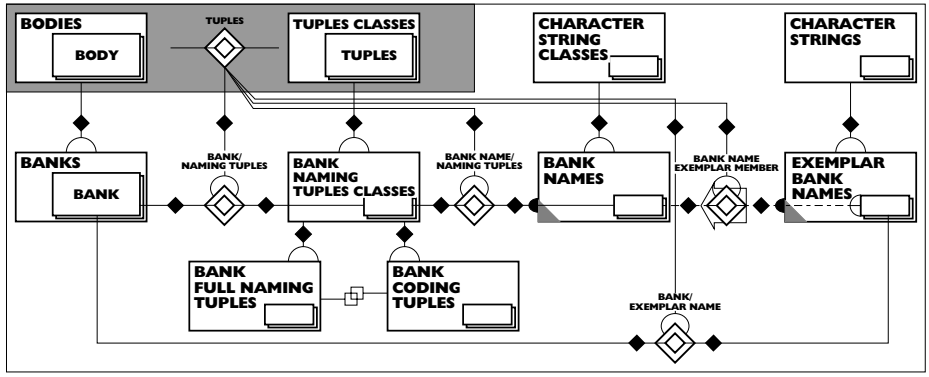
Figure MW3–2
Banks object schema



3.2 Re-engineering bank's full name and code attribute type signs

The bank full name and code attribute types are based on naming not spatial patterns, and they can be used to generalise the naming patterns in the reference ontology model. We start by re-using the geo-political naming pattern (illustrated in MW2's *Figure MW2–21*) as a template to construct the bank naming pattern. This gives us *Figure MW3–3*. We will return to this schema later in the paper to generalise the naming patterns.

Figure MW3–3
Bank naming tuples classes object schema



4 Re-Engineering address

We now move from bank to address and start to re-engineer the five address lines. When most people look at the lines of an address, they see a series of simple attribute signs. When we start analysing these, we will see a very different pattern—one that has been severely distorted to fit into the attribute structure.

4.1 Re-engineering address line one attribute type sign

We follow the rules. We have already re-engineered the entity type sign; so, we can start re-engineering the attribute type signs. We start with the address line one sign. *Table MW3–3* has a partial listing of the individual attributes that we use to start the re-engineering.

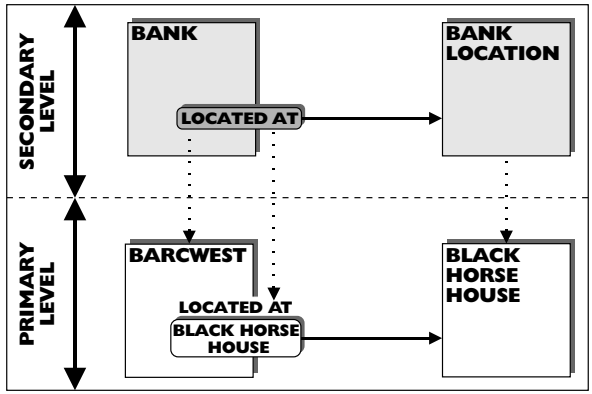
Table MW3–3: Partial address line one listing

Bank	Address line one
BarcWest Bank	Black Horse House
NatLand Bank	Listening Buildings
Chase Hanover Bank	BarcWest Tower
Banco di Guernsey	54th Floor

4.1.1 Re-engineering the individual attribute signs

Following the rules, we re-engineer some individual address line one attribute signs before we re-engineer the attribute type sign. We pick BarcWest's—Black Horse House—from *Table MW3–3*. What does this sign refer to? Within the entity paradigm, it not only refers to a 'located at' attribute of BarcWest but also implicitly to Black Horse House, where BarcWest is located. It is the implicit relational attribute sign described by *Figure MW3–4*.

Figure MW3–4
Bank 'located at' location attribute

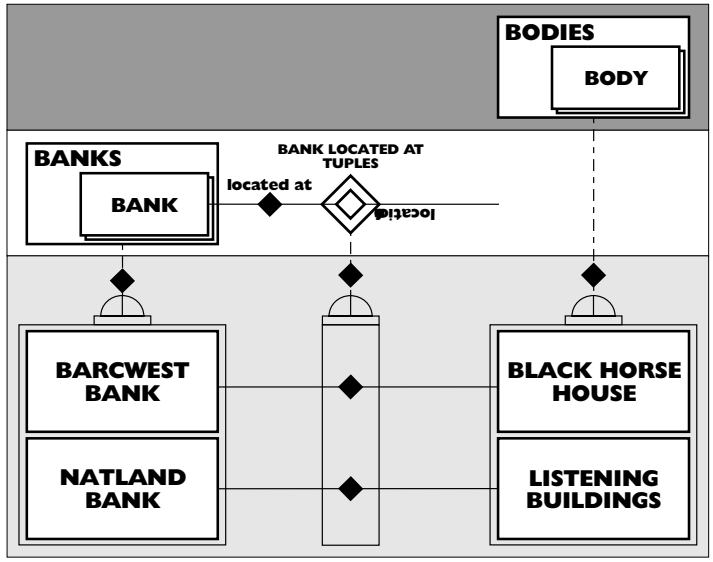


We start the re-engineering with the Black Horse House entity. This is a building on Moorgate, one of the streets in the City of London. In the object paradigm it persists through time and has a continuous four-dimensional extension; so, it is a physical body object; in other words, a sub-class of the framework class, bodies.

Then we re-engineer the 'located at' connection it has with BarcWest Bank. This is transformed into the couple <BarcWest Bank, Black Horse House> belonging to a bank 'located at' tuples class.

We pick another attribute sign from *Table MW3–3*—NatLand Bank's Listening Buildings—and re-engineer it. It follows the same pattern. The sign also refers to a building on Moorgate, one where NatLand Bank is located. We construct signs for both these objects and their sub-class connection to bodies. This gives the object schema in *Figure MW3–5*.

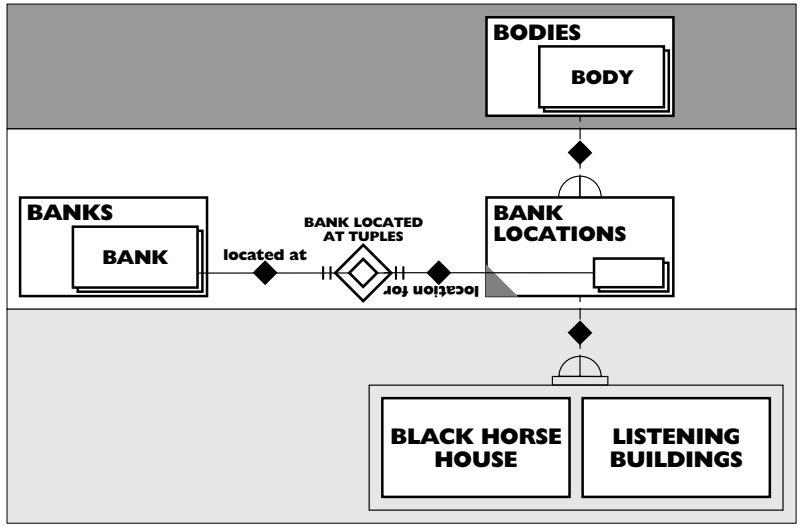
Figure MW3-5
Bank 'located at' tuples object schema



4.1.2 Re-engineering the attribute type sign

We now re-engineer the address line one attribute type sign. We have done part of the work with the bank 'located at' tuples class. We now need to construct a class for Black Horse House and Listening Buildings. Address line one is not really a very informative name for the class, so we use 'bank locations'. We construct a sign for it and get the object schema in *Figure MW3-6*. You will notice that bank locations is derived from the bank 'located at' tuples class.

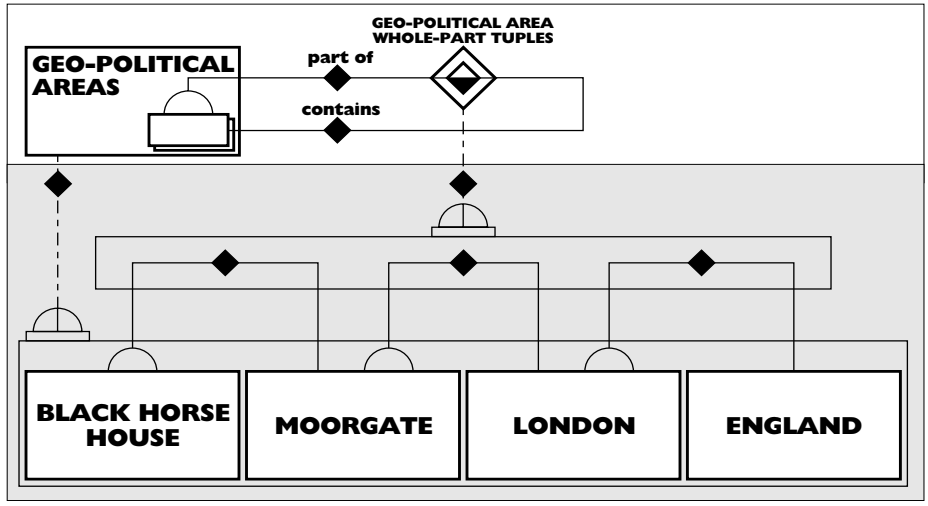
Figure MW3–6
Bank locations
object schema



If we think carefully, maybe stretching our imaginations a little, we can see that bank locations are a type of a geo-political area. They are not as big as countries or regions, but they work in the same way, occupying a defined space. Once we recognise this, we see that bank locations is a sub-class of geo-political areas. This gives us an opportunity to generalise and compact.

We can generalise the 'bank location for' class place up the super–sub-class hierarchy to geo-political areas following the compacting pattern we have used in earlier examples and discussed in *BG2— Constructing Signs for Business Objects' Patterns* (illustrated in its *Figure BG2–46*). This makes bank locations redundant; it is just those geo-political areas at which a bank is located (shown in the object schema in *Figure MW3–7*).

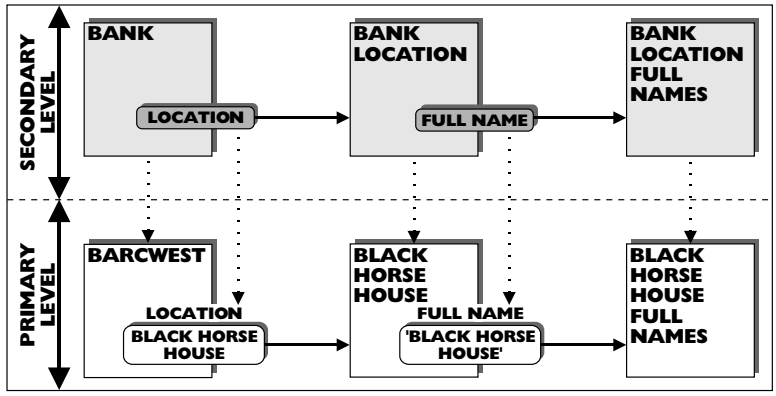
Figure MW3–7 Compacted bank 'located at' tuples object schema



4.1.3 Address line one's naming pattern

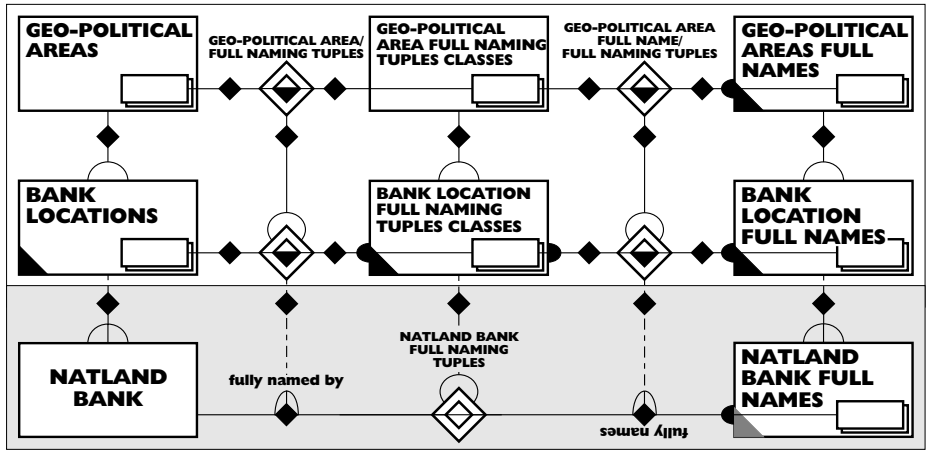
The address line one (or 'located at') attribute sign needs one last important bit of analysis. Look again at BarcWest's attribute sign. It contains the character string 'Black Horse House'. The particular characters in the string are important. They make up the full name of the location. This means we need to revise the entity model of the attribute in *Figure MW3–4*; we have to add in the bank full names entity as shown in *Figure MW3–8*. We now need to re-engineer this revised entity model.

Figure MW3–8 Bank full name and location attributes



However, if we look carefully at the patterns in our lexicon, we will see that the naming pattern has already been re-engineered. In the region example, we generalised the naming pattern, including the full naming pattern, up to geo-political area level. Because the bank location is a sub-class of geo-political area, it inherits its full naming patterns. Bank location—as a geo-political area—already has a full name. This is illustrated in *Figure MW3–9*. The re-engineering does not need to construct any new objects.

Figure MW3–9 Bank locations full name object schema



However, it does introduce a new re-engineering pattern for attribute type signs, such as bank location. We are familiar with the re-engineering pattern for relational attribute type signs, which gives us a tuples class and its place class. We are also familiar with the re-engineering pattern for naming attribute type signs, which gives us a naming tuples class and its name class. Here, we have a re-engineering that combines the two patterns—a sort of naming relational pattern. This is a useful pattern to have when re-engineering. Naming relational attribute type signs are common in entity based systems.

This also is a good illustration of the semantic accuracy required when re-engineering. If we had not been careful, we could have easily stopped the analysis at bank location. Then we would not have recognised that there was a full name pattern to re-engineer—even though it now stares us in the face.

4.2 Re-engineering the other address lines attributetype signs

Because the re-engineering of the other address lines follows a similar pattern, we do it here in bulk. This also helps us to see a connecting pattern between the lines.

Like bank locations, with a little stretching of the imagination, we can see the other address lines as full names for geo-political areas. BarcWest's 'Old Broad Street' and 'London' are larger than bank locations and smaller than countries or regions, but still the same type of object. Address line four, 'United Kingdom', is a country—so, by our previous re-engineering, already a geo-political area. This has other implications that we will examine later.

These other address lines do not have exactly the same pattern as address line one. Unlike it, they do not have a 'located at' connection with a bank, and so are not bank locations. Otherwise, their patterns are the same. At the application level, the other address lines are geo-political areas, and so, like address line one, inherit its full naming patterns. This means we do not need to re-engineer any new application level objects for them (illustrated by the schema for BarcWest's address in *Figure MW3-10*).

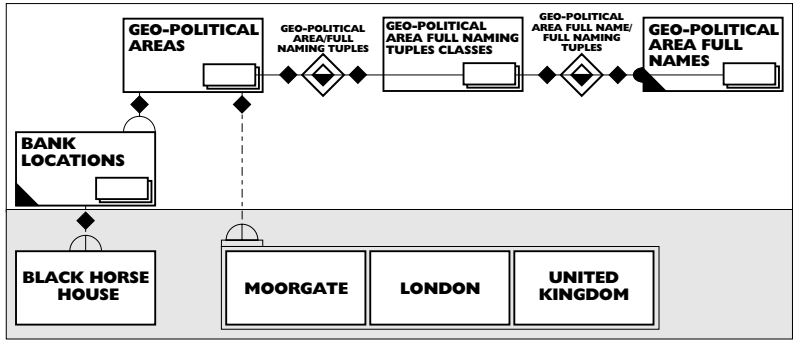


Figure MW3-10 BarcWest's address object schema

5 Nested address lines

We have re-engineered all the explicit patterns for these bank address lines, but there is still one vital implicit pattern that our accurate semantic analysis needs to pick up.

If we look at the listing of addresses in *Table MW3–1* again, it appears that there cannot be anything special about which line the objects appear on. For example, one of them, 'BarcWest Tower', appears as address line one in Chase Hanover's address and as address line two in Banco di Guernsey's address. It cannot be intrinsically both a line one and a line two object.

However, the order of the lines is important. This is easy to show; look at the addresses in *Table MW3–4*. They are clearly not valid addresses; yet, all I have done is move the position of the lines. The reason they are not valid addresses is the positioning of the lines reflects an implicit pattern, and changing the order of the lines breaks it. We are so familiar with this implicit pattern that we automatically read it in. We only notice that it existed when it is not there.

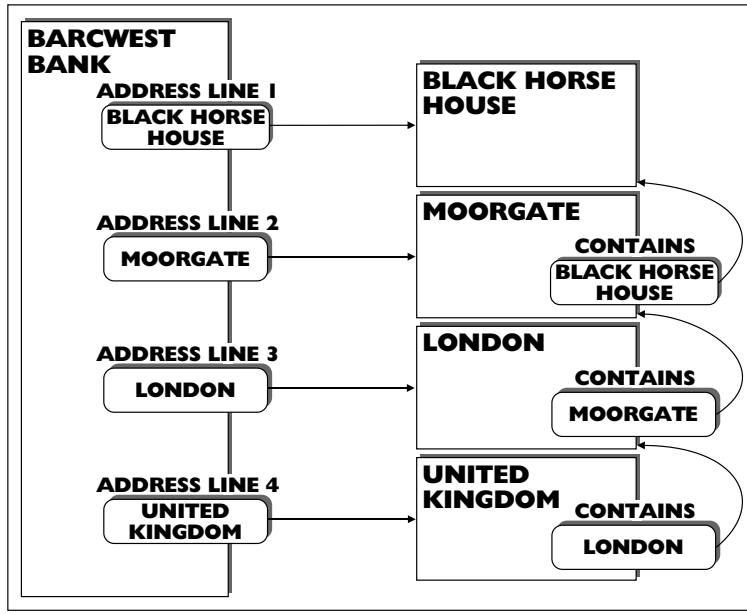
Table MW3–4: Altered address lines

Bank	001	BarcWest Bank	003	Chase Hanover Bank
Address		United Kingdom		London
		Moorgate		BarcWest Tower
		Black Horse House		United Kingdom
		London		Old Broad Street

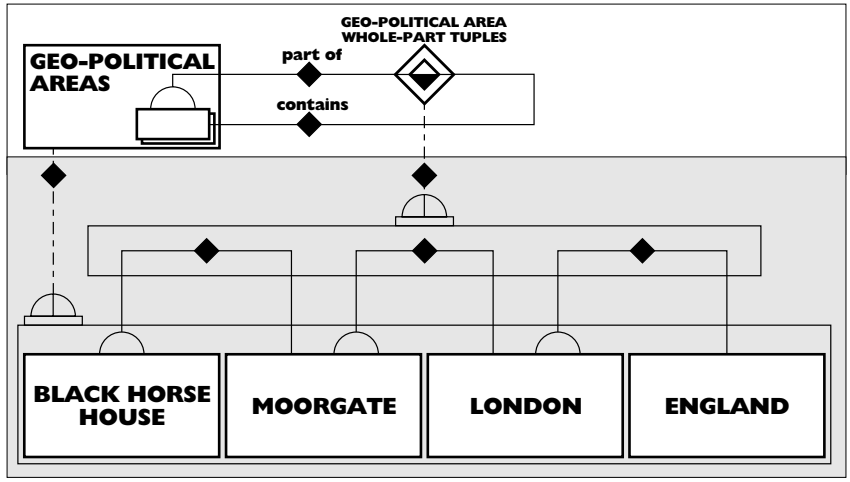
5.1 Implicit whole–part tuples

A little thought shows that the order of the address lines implies a whole–part connection between them. For example, because London is on the line before United Kingdom, this implies it is part of United Kingdom. We broke the order of this whole–part pattern when we moved the lines around. *Figure MW3–11* shows an entity view of the whole–part pattern for BarcWest's address.

Figure MW3-11 BarcWest's address's implicit whole-part patterns



This is not how Aristotle's entity paradigm structure was originally meant to work. It has been 'bent'. We are now so familiar with this 'bending' that we automatically supply the implicit whole-part patterns. However, we do not need to do any bending in the object paradigm. In fact, the strong reference principle demands that we map the whole-part connections directly and explicitly into the ontology model. *Figure MW3-12* illustrates the result.



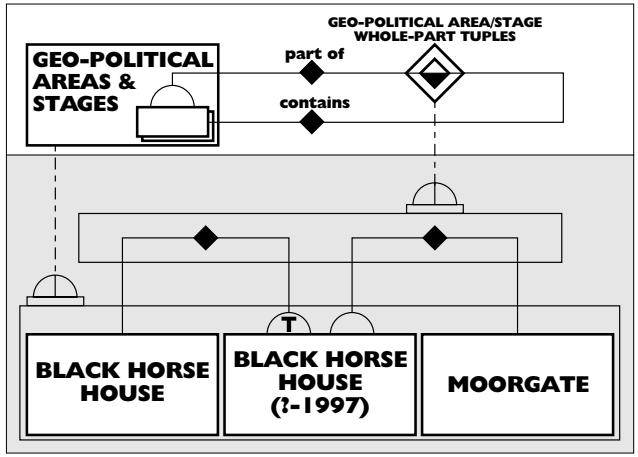
FigureMW3-12 BarcWest's address's whole-part tuples object schema

5.2 More accurate whole-part pattern

You have probably recognised that this modelling of the whole-part pattern is not as accurate as it should be. In *MW1—Re-Engineering Country*, when we re-engineered country's nested whole-part patterns, we saw how Scotland was only 'part of' the Great Britain for part of its life. In other words, a stage of Scotland—and not Scotland—was part of the United Kingdom. This is illustrated in its *Figure MW1-54*.

The address lines are geo-political areas and so could have a similar pattern to Scotland. Though it is unlikely, there is no reason why Black Horse House should not be bought by a foreign billionaire and transported to another country. In this case, Black Horse House would have a similar pattern to Scotland. It would only be part of Moorgate for part of its life. In other words, a stage (state) of Black Horse House would be part of Moorgate. We do not need a new pattern to capture this. We can re-use the geo-political areas and stages nesting pattern. *Figure MW3-13* illustrates what the more accurate reference model looks like.

Figure MW3–13
More accurate whole–part patterns

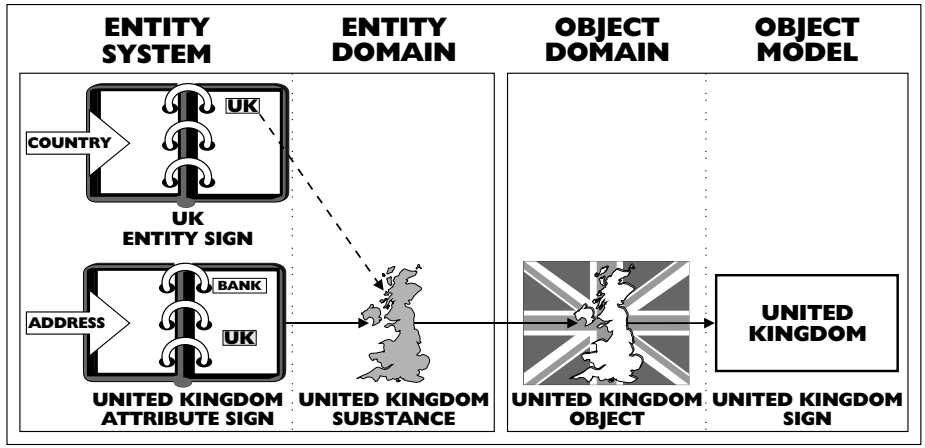


5.3 Re-engineering the same object twice

When we were re-engineering the other address lines, we noted that one of the address lines—BarcWest's address line four—was the full name for a country—'United Kingdom'. This raises an important point. It is impractical to apply the strong reference principle consistently in the constrained entity environment. So, sometimes, there is more than one sign for an entity. This is a case in point. United Kingdom on the country file and United Kingdom in BarcWest's address both refer to the same entity.

Under the strong reference principle, we should re-engineer these two entity signs into one sign in the reference object ontology model. Where there is duplication in the existing system, we *do not* replicate it in the model. So we do not re-engineer a new sign in the model for the United Kingdom, but re-use the existing one. *Figure MW3–14* shows the re-engineering pattern.

Figure MW3–14 United Kingdom re-engineering pattern

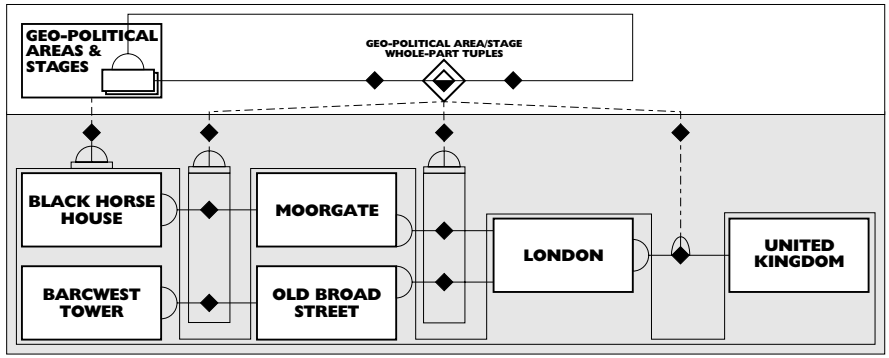


Duplication of signs occurs with a vengeance within address. Consider the list of address line attributes from *Table MW3–1* in *Table MW3–5*. It is plain that the four Londons (and three Englands) in the various addresses are full names for the same entities. *Figure MW3–15* shows what the re-engineered undistorted de-duplicated reference model looks like for two of the addresses.

Table MW3–5: Selected address line attributes listing

Bank	Address Line No.	Location
BarcWest Bank	3	London
NatLand Bank	3	London
	4	England
Chase Hanover Bank	3	London
Banco di Guernsey	4	London
	5	England

Figure MW3–15 BarcWest and Chase Hanover Bank addresses

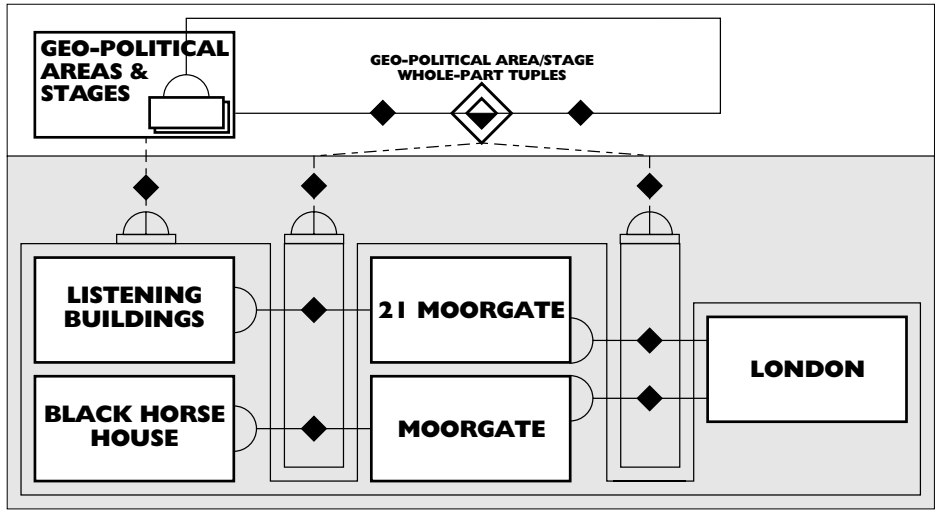


This de-duplication leads to compacting. Transforming two (or more) signs in the entity model into one object sign makes the reference model smaller and leaner. It also makes maintenance of the implemented system easier. In the existing system, when an address entity changes its full name then this change has to be applied to all its full name signs. Let's say London changed its name to Londres; then, every bank address line that contains 'London' would need to be changed. There are four of these in this small example; there could be hundreds, or even thousands, more in a working entity oriented system. However, in a system based on the reference object ontology model, there would only be one and so the change would only need to be done once.

5.4 One name referring to two objects

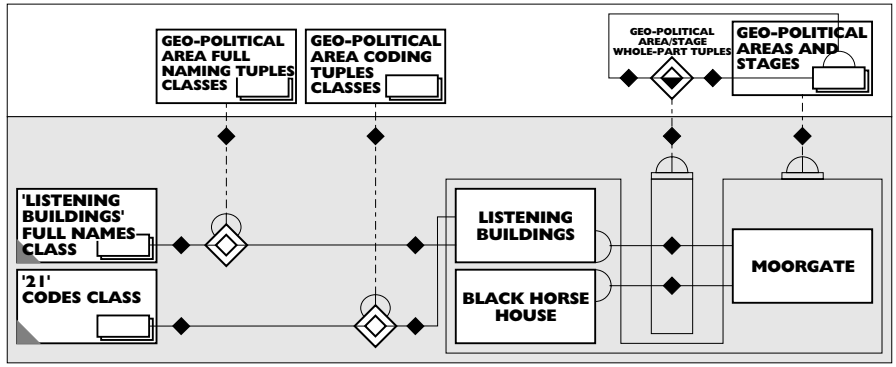
Addresses contain a number of implicit patterns and it can take some time to uncover. For example, one of the addresses in *Table MW3–1* contains another implicit pattern. Consider two address lines; BarcWest Bank's address line two—'Moorgate'—and NatLand Bank's address line two—'21 Moorgate'. We automatically recognise that BarcWest and NatLand Banks are both located on the same street—Moorgate. However, this recognition has not been captured by the re-engineering (shown in their model in *Figure MW3–16*). Listening Buildings is not signed as a part of Moorgate.

Figure MW3–16
Moorgate
object schema



To capture this, we have to divide the name '21 Moorgate' into two names—'21' and 'Moorgate'. The name '21' is another way of indicating the NatLand Bank's location, Listening Buildings. It is not a full name, so we treat it as a code. 'Moorgate' is a duplication of BarcWest Bank's address line two—Moorgate—the full name for a street. The new model is shown in *Figure MW3–17*. This clearly shows that the two buildings are both located in Moorgate.

Figure MW3–17
New Moorgate
object schema



5.5 Address joining events

When we re-engineered country and region, the nesting pattern went hand in hand with joining events. For example, the stage of Scotland that is part of Great Britain was created by the 1707 Scottish Act of Union Joining Event. When we re-engineered region, we generalised the patterns to the geo-political area level. Addresses, as geo-political areas, inherit the joining event patterns.

These patterns are useful for addresses. When counties' boundaries are moved and towns end up in new counties—a reasonably regular occurrence nowadays—this is a joining event. It can be modelled re-using the geo-political area joining patterns.

6 Generalising name

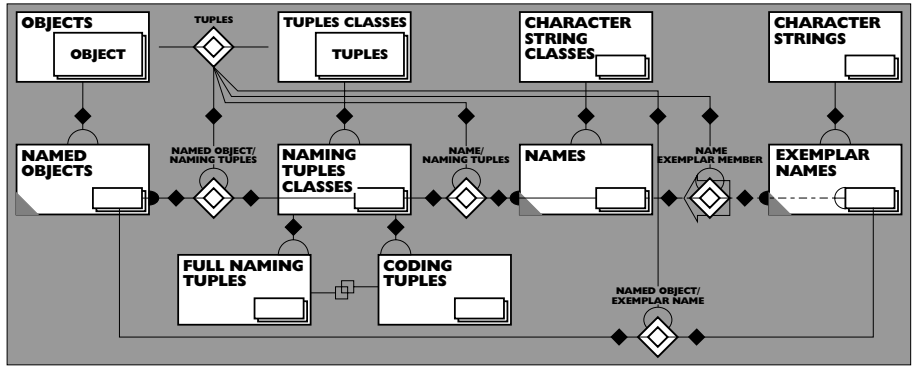
We now turn our attention from spatial to naming patterns. At the beginning of this paper, when we were re-engineering the bank full name and code attribute type signs, I noted that we would use their object ontology model to generalise the naming pattern. We do this now; we generalise naming above and away from geo-political areas and bank. This gives us a separate re-usable web of completely general naming patterns.

6.1 A general reference model for naming patterns

The bank and geo-political area naming models are the same shape. It seems likely that all naming models share the same shaped pattern. So there is an opportunity to generalise to a naming pattern. We take this now. We start with the general naming tuple and build up its connecting patterns. We derive from it the class of named objects—those objects that are connected by the named object/naming tuple class place. In theory, any object can be named; so, this named object class is a sub-class of the framework class, objects. We also derive the names class—those objects that are connected by a class place link to the name/naming tuples class. This names class is a sub-class of character strings.

This gives us the general reference model for the naming pattern shown in *Figure MW3–18*. As you can see, it is independent of both geo-political areas and banks. Furthermore, it has 15 objects of which only 10 are specific to names. This pattern is ubiquitous; its model will be re-used a large number of times in almost every re-engineering. Therefore, it makes sense to put its model into the reference ontology (which we discussed in *MO1—The BORO Approach to Re-Engineering Ontologies*) by classifying its objects as framework level. As you can see, this has been done in *Figure MW3–18*.

Figure MW3–18 Generalised naming tuples object schema



Undoubtedly, when the naming reference model is re-used, there will be opportunities to enhance it. For example, currency symbols, such as '£' and '$', which are not currently catered for, might be found as attributes of the currency entities. These would be generalised into the naming pattern. They would probably be generalised to a new sub-class of the naming tuples class—the symbol naming tuples classes. In this way, the reference model naturally grows as we expand our explicit understanding of the naming pattern.

6.2 Compacting with the general naming reference model

This general naming model leads to substantial compacting. Most of the systems that I have worked on have had well over 100 entity types (files). Almost all of these had name attribute type;, many have had two (a full name and code), some have had more. So there would usually be well over 200 name attribute

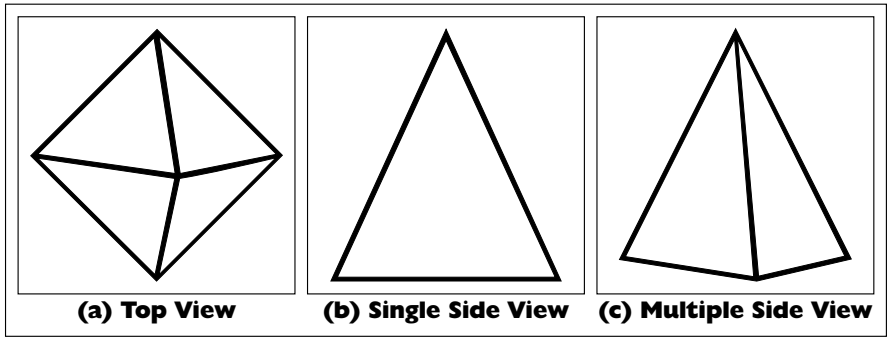
types in a system. We re-engineered these by transforming their entity types into classes that were sub-classes of the named objects class. Because they then inherited the general naming pattern, no other re-engineering was necessary. As well as simplifying the re-engineering process, this compacts the model. The original 200 name attribute types are compacted into the naming pattern's 10 objects. Furthermore, adding additional name attribute types would not increase the size of the model.

7 An aspect of a pattern

The address format was originally designed for paper and ink technology; it fits naturally onto the cover of an envelope. It also transfers effortlessly into the entity paradigm's framework. The re-engineering of address gives us a good insight into how much some patterns have had to be distorted to fit onto paper and ink technology and into the entity paradigm. The patterns we end up with in the object ontology model are very different from the patterns we start with in the existing system.

These radically changing patterns can seem odd. I find it useful to think of them in terms of this analogy. Consider a three-dimensional pyramid, but assume that we can only see it from one aspect at a time. If we look from the top—*Figure MW3-19* (a)—it has a diamond outline. If we look at if from the ground, directly facing one side—*Figure MW3-19* (b)—we see a triangle. If we move so that we are not directly facing a side – *Figure MW3-19* (c)—we see an odd shaped four-sided figure. Because we can only see the pyramid in two dimensions, we only see one aspect of it at one time.

FigureMW3–19
Aspects of a three-dimensional pyramid



Assume now we were to try and build a model of the pyramid. We would not build three models, one for each of the aspects; we would build one model of the complete pyramid. We could re-create the aspects we looked at earlier by taking a particular view of our three-dimensional model. The radical differences between the entity patterns and the object patterns are like the radical differences between any one of the two-dimensional views and the three-dimensional model. In the same way that only looking at the pyramid in two-dimensions views gives us an aspect, so the two-dimensional constraints of the entity paradigm produce partial views of business objects.

8 Summary

This re-engineering of address has completed the basic object model for spatial patterns; it is ready for re-use in future re-engineering projects. More importantly, this and previous worked examples have given us a feel for how the systematic re-engineering process works. We now know the standard steps in the process and are familiar with a number of common re-engineering patterns.

As discussed in the previous section, address has also given us a useful example of how using the entity paradigm (the paradigm behind address) can distort the overall shape of business objects. We have seen how fitting address into the entity paradigm's constraints fixes one particular view. We have also seen how re-engineering frees it from that constraint, giving us a full rounded view of the

business objects. Furthermore, we have seen how general objects that enable substantial re-use, such as the naming objects, are constructed.

In the next worked example (*MW4—Re-Engineering Time*), we re-engineer some of the entity paradigm's temporal patterns into an object model for time. Like space, time is fundamental to most business paradigms and so the model is very re-usable.

BORO Working Papers - Bibliography

The BORO Working Papers

Volume A
A—The BORO Approach

Book AS
AS—The BORO Approach: Strategy

AS1—*An Overview of the Strategy*
AS2—*Using Objects to Reflect the Business Accurately*
AS3—*What and How we Re-engineer*
AS4—*Focusing on the Things in the Business*

Volume - O
O—ONTOLOGY Papers

Book - OP
OP—Ontology: Paradigms

OP1—*Entity Ontology Paradigm*
OP2—*Substance Ontology Paradigm*
OP3—*Logical Ontology Paradigm*
OP4—*Business Object Ontology Paradigm*

Volume - B
B—Business Ontology

Book - BO
BO—Business Ontology: Overview

BO1—*Business Ontology - Some Core Concepts*

Book - BG
BG—Business Ontology: Graphical Notation
Constructing Signs for Business Objects

Graphical Notation I
BG1— *Constructing Signs for Business Objects*
Graphical Notation II
BG2— *Constructing Signs for Business Objects' Patterns*

Volume - M
M—The BORO Re-Engineering Methodology

Book - MO
MO—The BORO Re-Engineering Methodology: Overview

MO1—*The BORO Approach to Re-Engineering Ontologies*

Book - MW
MW—The BORO Methodology: Worked Examples

Worked Example 1
MW1—*Re-Engineering Country*
Worked Example 2
MW2—*Re-Engineering Region*
Worked Example 3
MW3— *Re-Engineering Bank Address*
Worked Example 4
MW4—*Re-Engineering Time*

Book - MA
MA—The BORO Re-Engineering Methodology: Applications

MA1—*Starting a Re-Engineering Project*
MA2—*Using Business Objects to Re-engineer the Business*

Book - MC
MC—The BORO Re-Engineering Methodology: Case Histories

Case History 1
MC1—*What is Pump Facility PF101?*

INDEX

METHODOLOGY: WORKED EXAMPLE - 3

RE-ENGINEERING BANK ADDRESS

A-P

R

S

W