Business
Object
Reference
Ontology
Program

simplifying semantics

# Working Paper

## OP1

# ONTOLOGY: PARADIGM-1

# ENTITY ONTOLOGY PARADIGM

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*


# OP1

## ONTOLOGY: PARADIGM - 1

# ENTITY ONTOLOGY PARADIGM

## CONTENTS




# ENTITY ONTOLOGY PARADIGM

## 1 Introduction

In *AS—The BORO Approach: Strategy*, we looked at our strategy for re-engineering the entity foundations of information. Here in *OP—Ontology: Paradigms* , we describe this re-engineering. In this paper, we start by examining our current starting point—the entity paradigm—concentrating on the four key types of things we identified in *AS4—Focusing on the Things in the Business*. In the following papers we follow the entity paradigm's evolution into the object paradigm.

The entity paradigm is what most computer systems' information is based on. Because our aim is to re-engineer the business paradigms embedded in our existing computer systems, it is a natural starting point. Most people who work with it, instinctively recognise that the entity paradigm is a tool for doing things rather than understanding them. To use a distinction raised in the *AS2—Using Objects to Reflect the Business Accurately*, it works at an operational rather than an understanding level.

The entity paradigm is a simplified version of a powerful paradigm developed by the Ancient Greek Aristotle, which we call the substance paradigm. It is a good

approximation to the way most of us see the world. It was simplified into the entity paradigm to work more effectively with paper and ink technology.

In *AS4—Focusing on the Things in the Business*, we noted that most of us cannot explain what an entity is. What we do here is drag the entity paradigm up to the surface, making ourselves conscious of it. Sometimes people are surprised when they first see it in the cold light of day—some even find it difficult to accept that it is what they have been working with.

We start this paper by looking at the entity paradigm. We look at its fundamental particles, the patterns of re-use it enables, and how it relates to paper and computer information. Then we look at the elements of the substance paradigm that were eliminated when the entity paradigm was simplified, and why they were removed. This gives us an insight into the nature of entity-oriented systems. In the next paper, *OP2—Substance Ontology Paradigm*, we turn our full attention onto the substance paradigm, particularly its semantics.

# 2 The entity paradigm's fundamental particles

We start by looking at the entity paradigm's fundamental particles:

- The entity, and
- The attribute.

The entity particle is more fundamental, so we start with it.

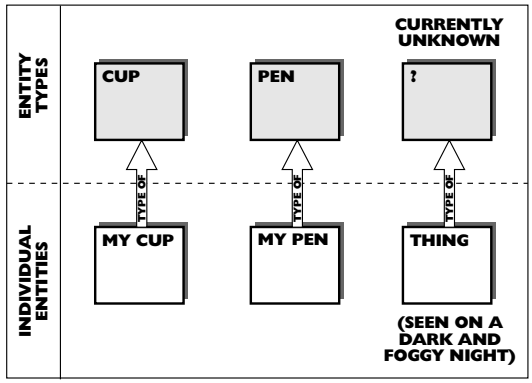
## 2.1 Individual entities

We naturally divide the world into concrete particular things. When we look around a room, our eyes receive a continuous stream of data. We unconsciously analyse this stream and consciously see chairs, tables and so on. These are individual entities.

## 2.2 Entity types

Individual entities are not the only type of entity. We also naturally group together individual entities that are, in some sense, the same. We then say that the entities in the group all belong to the same entity type. Individual entities naturally belong to entity types. They also always belong to one. For instance, we might catch a fleeting glimpse of something (it may be a fox or a dog) on a dark and foggy night. Even though we cannot say what entity type the thing belongs to, it still has one—as shown in *Figure OP1–1*.

Figure OP1–1 Individual entities naturally belong to entity types




## 2.3 Attributes belonging to entities

Attributes have the same two-tier pattern as entities. Just as there are individual entities and entity types, so there are:

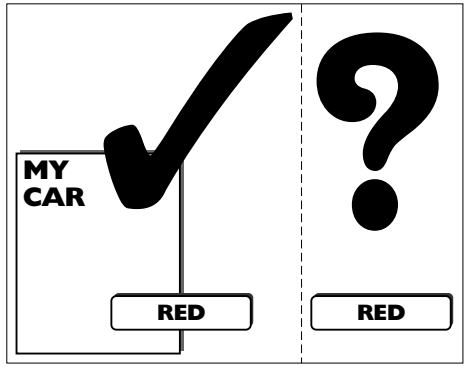
- Individual attributes, and
- Attribute types.

### 2.3.1 Individual attributes

We naturally see that individual entities have attributes (also called properties or qualities). Someone looking at my car naturally notices it is red (in other words, it has a red attribute/property/quality). We always see individual attributes belonging to individual entities; this is one of their inherent features.

Notice how difficult it is to see the particular red property existing on its own (out in the real world) with no entity attached to it—as illustrated by *Figure OP1–2*. This seems impossible. It is important to remember that we are talking about individual attributes out in the real world. We can, of course, imagine the general idea of red on its own; but that is in our head.

Figure OP1–2 Individual attributes always belong to an individual entity




If we think about it, we can see that the individual attributes determine how an entity appears. All of my car's appearance is based on its properties, its individual attributes. If my car looks red, then it has the property of being red—it has a red attribute. The entity itself is not red, being red is the function of the individual red attribute.

### 2.3.2 Attribute types

The relationship between individual attributes and attribute types is similar to that between individual entities and entity types. Every individual attribute belongs to an attribute type and an attribute type can have a number of individual attributes. Also the relationship between an attribute type and its entity type is similar to the relationship between an individual entity and its individual attributes. This is shown schematically in *Figure OP1–3*. An example using my red car is given in *Figure OP1–4*.

Figure OP1-3
Entities and attributes

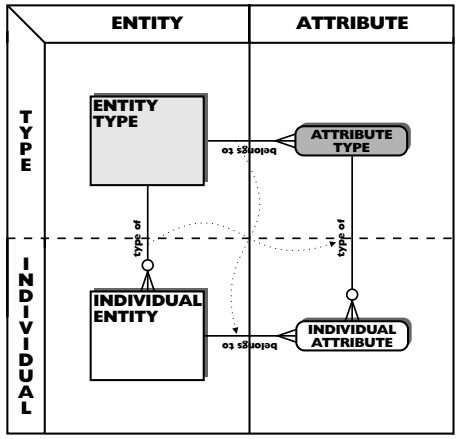
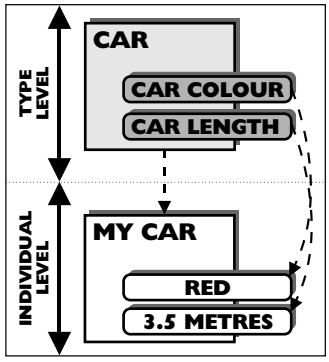



Figure OP1-4
My red car




# 3 The entity framework and its (re-)use of patterns

The entity paradigm is a good example of how a framework enables the (re-)use of general patterns.

## 3.1 Re-use working down the entity framework's hierarchy

Re-use works its way down the entity framework. The general entity and attribute patterns are used to construct all the patterns at the type level. These are then used (and so their embedded general patterns re-used) to construct the individual level patterns.

We can see this happening in *Figure OP1–5*. There, the general entity-attribute pattern provides a framework for the staff member entity type patterns. This in turn provides a framework for the individual staff member patterns.

Figure OP1–5
The entity paradigm's general structure

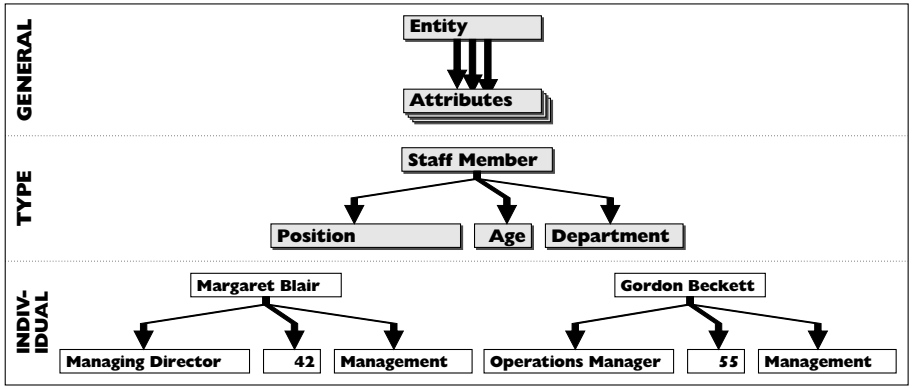
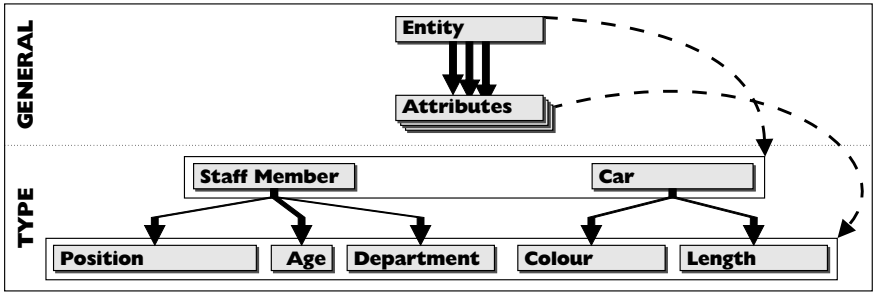



*Figure OP1–6* illustrates how re-use operates at the general level. We can see that the entity paradigm does not determine what the actual entity and attribute types are, just the framework in which they live. The various entity and attribute types work in a similar way. They do not determine what the actual entities and attributes are, just their framework.

Figure OP1–6
Re-using the general entity-attribute pattern



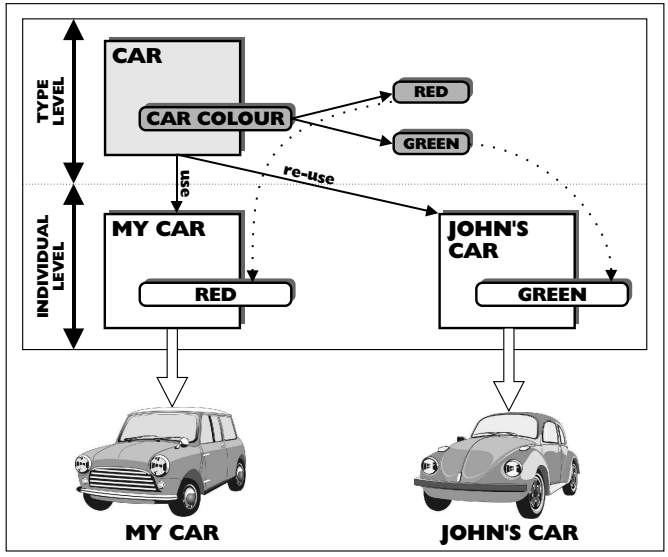

This type of framework derives its power from the repeated use (and re-use) of a pattern from a higher level at a lower level. For example, the general entity-attribute pattern is repeatedly used in the construction of entity types and their attribute types. There is a similar pattern of re-use between the type and individual levels. An individual entity and its attributes are constructed using the


patterns from its entity type and associated attribute types. This usually means that the patterns for the entity type and its associated attribute types are re-used many times. The general entity-attribute pattern is embedded in the entity type and attribute type patterns. So, when the individual entities and their attributes are constructed using type level patterns, the general pattern is also implicitly used. In this way it pervades all levels of the framework.

The entity–attribute pattern is generative. As well as applying to the existing framework, the general pattern can be used to 'generate', when required, new type and individual level patterns. Type level patterns are generative as well; they can be used to 'generate' new individual level patterns.

It might be easier to see this with a simple example. Assume that I have constructed a car entity type pattern and used it to construct an individual entity pattern for my car. Also assume that I have constructed a car colour attribute type for the entity type and, based on this pattern, a red attribute at the individual level for the 'my car entity'. If I now see John's car, I can generate its pattern from the existing framework. I can construct an individual entity for it, using the car entity type and car colour attribute type patterns—as shown in *Figure OP1–7*. When I use the car entity pattern to construct the entity John's car, I also automatically commit myself to constructing an attribute using the car colour attribute type. Notice also that 'John's car' entity inherits the general entity–attribute pattern through the car entity type pattern.

Figure OP1-7
The generative type level




We will appreciate the generative nature of the type level better if we consider what would have happened if there had been no middle type level in the framework, just the top general entity-attribute level. Without the type level, whenever we saw a car for the first time we would have to regard it as a unique entity with its own variety of individual attributes. This means we would have to go through the kind of analysis that we do for completely new and unknown types of things. So, when we see John's car for the first time, we would have to construct its entity and attributes without the guidance of a car entity type. We would not know that it was a car nor that it has a colour attribute. This would not only make life complicated, but very time consuming.

# 4 The entity paradigm and the file-record paradigm

The entity paradigm is closely related to what we will call the file-record paradigm.


## 4.1 Where does the file-record paradigm come from?

Computer users often talk about information being stored in files and records, rather than entities and attributes. However, the terms, 'file' and 'record', are rooted in paper and ink technology. Well before the invention of computer technology, records were made on pieces of paper and kept in files. These paper records and files were based on the way in which information is naturally stored on paper in rows and columns. We distinguish paper's framework of rows and columns from computing's files and records by calling it the tables paradigm. These paradigms are all closely linked. For example, their fundamental particles all map directly onto one another—as shown by *Table OP1–1*.

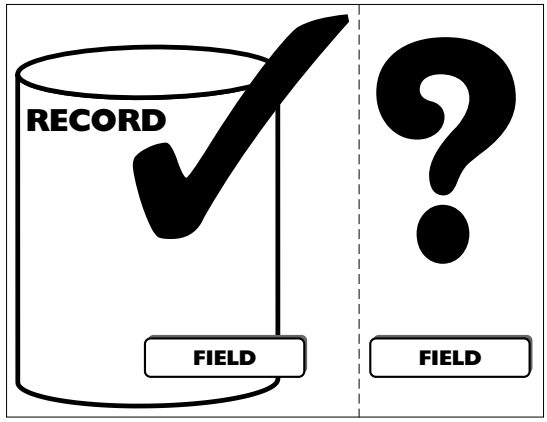
Table OP1–1: Closely linked fundamental particles

| Tables paradigm | File–record paradigm | Entity paradigm | Example |
|---|---|---|---|
| Rows | Computer record | Individual entity | My car |
| Element | Computer field | Individual attribute | Red |
| Table | Computer file | Entity type | Car |
| Column | Computer field type | Attribute type | Car colour |

## 4.2 Reinforcing the entity paradigm

Computer-literate people's training in file and record patterns reinforces the entity paradigm, and vice versa. For example, it appears to make no sense to talk about an individual field existing apart from its record—as illustrated in *Figure OP1–8*. Just as it makes no sense to talk about individual attributes existing without their individual entities (shown in *Figure OP1–2*). What could such a field or attribute be?

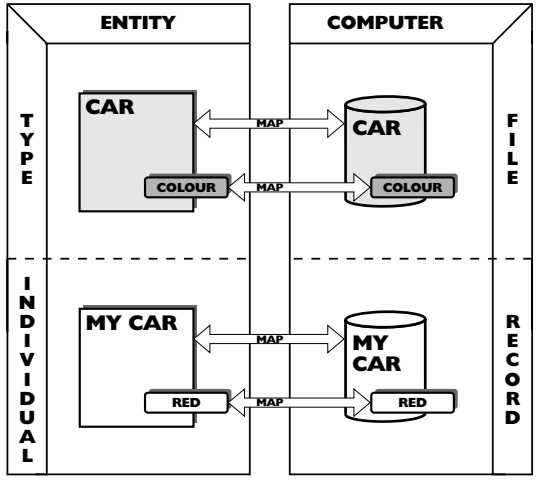
Figure OP1–8
An individual computer field without its computer record




# 5 Mapping entities and attributes onto files and records

These intimate links between the paradigms (shown in *Table OP1–1*) lead many system builders to see computing's file-record paradigm as a physical implementation of the entity paradigm—as illustrated in *Figure OP1–9*.

Figure OP1–9
Physically implementing the entity paradigm



## 5.1 Staff example

This apparently close mapping leads many people to the false assumption that files and records directly reflect the real world's entities and attributes. This simple staff example shows how wrong this assumption can be.

### 5.1.1 Two incompatible entity formats

Consider the two lists in *Table OP1–2*, one of salespersons and the other of account managers. They could equally well be combined into one staff list – as shown in *Table OP1–3*. Let's assume that these two lists are from computer systems whose records and fields directly reflect entities and attributes. We can then work out what the systems' entity formats are and, by implication, the entities and attributes they reflect. The formats are shown in *Figure OP1–10.*

Table OP1–2: Salespersons and Account Managers lists

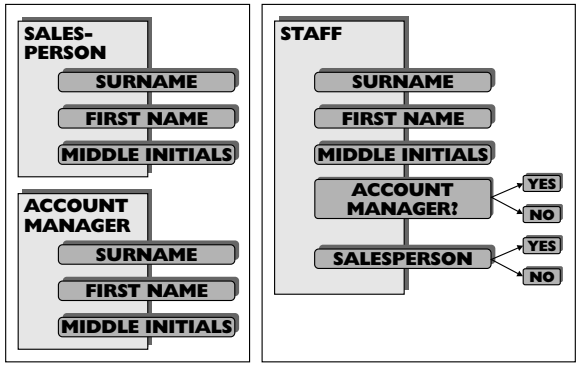
| Salespersons | | |
|---|---|---|
| **Surname** | **First name** | **Middle initials** |
| Bottomley | Margaret | V.G. |
| Clarke | Michael | L. |
| Heseltine | Kenneth | J. |
| Howard | Cecil | S.F.A. |
| Thatcher | Virginia | J.K. |

| Account Managers | | |
|---|---|---|
| **Surname** | **First name** | **Middle initials** |
| Beckett | Gordon | O.B. |
| Blair | Margaret | J. |
| Brown | Claire | F.L. |
| Short | Tony | L. |
| Thatcher | Virginia | J.K. |

Table OP1–3: Staff list

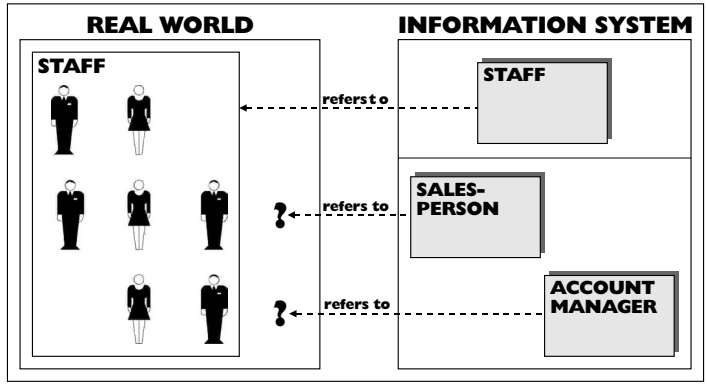
| Staff | | | | |
|---|---|---|---|---|
| **Surname** | **First name** | **Middle initials** | **Salesperson indicator** | **Account Manager indicator** |
| Beckett | Gordon | O.B. | | Yes |
| Blair | Margaret | J. | | Yes |
| Bottomley | Margaret | V.G. | Yes | |
| Brown | Claire | F.L. | | Yes |
| Clarke | Michael | L. | Yes | |
| Heseltine | Kenneth | J. | Yes | |
| Howard | Cecil | S.F.A. | Yes | |
| Short | Tony | L. | | Yes |
| Thatcher | Virginia | J.K. | Yes | Yes |

Figure OP1–10
The two assumed entity formats



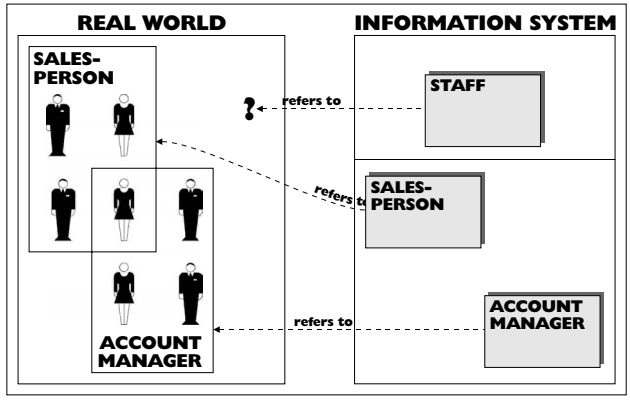

As the business only has one entity structure, it cannot be reflected by both the entity formats in *Figure OP1–10*. Only one of them can be 'right'. If we compare the formats of the two systems, then we immediately see a major difference. In the first system, there are signs for two entity types, Salesperson and Account Manager. Whereas, in the second format, there is only one entity type sign, Staff. Which of these entity type signs actually refer to real entity types in the business? If the business has a Staff entity type, then the first format is 'wrong'; its two entity type signs do not reflect the business—as shown in *Figure OP1–11*.

Figure OP1–11
Staff world view




If Salesperson and Account Manager are entity types then the Staff format—shown in *Figure OP1–12*—is 'wrong'.

Figure OP1–12
Salesperson and Account Manager's world view




There is another contradiction in *Figure OP1–10*. Look at it again, and compare the two systems' entity formats. You may notice that, in the first system, Salesperson and Account Manager are signed as entity types; but, in the second system, they are signed as attribute types (the Salesperson and Account Manager indicators). If the entity paradigm reflects the structure of the world, then the world divides irrevocably into entities and attributes. Something cannot be both an entity and an attribute. However, here we have two systems; one with Salesperson signed as an entity type and the other with it signed as an attribute type. Which type is it—entity or attribute?

### 5.1.2 Not 'wrong' but different objectives

It turns out that neither format is 'wrong'. What is wrong is our assumption that a computer system's files and records necessarily reflect the world's entities and attributes. What dictates the structure of these files and records is not the world they describe but what we want to do with them in our system. The structures of the two systems are different because they serve different purposes. For example, if we wanted to send the same letter to all staff, it would be easier to use the list from the staff system. If we were sending different letters to Salespersons and Account Managers, it would be easier to use the two lists from the other system.

This example shows that the decision on whether to use a record or a field in a computer system does not necessarily involve distinguishing between entities and attributes in the outside world. It is more about different ways of handling the data inside the information system.

# 6 The substance paradigm's secondary hierarchy

When the substance paradigm was simplified into the entity paradigm, the structurally key secondary hierarchy was eliminated. This is part of the reason why the staff example's files and records do not map directly onto the outside world; and why the entity paradigm focuses on the data inside information systems rather than the things in the business. We now look at the substance paradigm, particularly its secondary hierarchy. We see how it was simplified and how the simplification changes what we see.

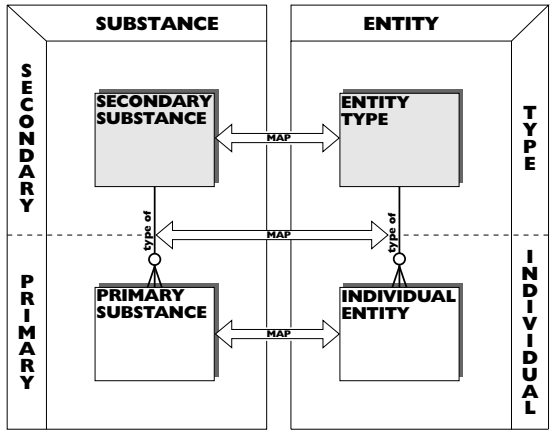
## 6.1 The substance paradigm

The substance paradigm is ancient. It was constructed by the Ancient Greek Aristotle in the 4th century BC and subsequently developed by his followers. It is, like much of Aristotle's work, a rationalisation of people's intuitive ideas. These

ideas are still with us today. Most people see the world through the eyes of the substance paradigm.

The substance and entity paradigms share similar fundamental particles. The substance paradigm's particles are substances and attributes, where substance corresponds to entity (as shown in *Figure OP1–13*) and attribute, not surprisingly, to attribute. (Students of Aristotle use the words 'substance' and 'entity' almost interchangeably; however, we use 'substance' for Aristotle's paradigm and 'entity' for the entity paradigm.) As we shall see in the next section, the main difference is that, at the secondary level (the substance paradigm's name for the type level), the substance paradigm is more powerful.

Figure OP1–13
The substance particle corresponds to the entity particle




## 6.2 The secondary level hierarchy

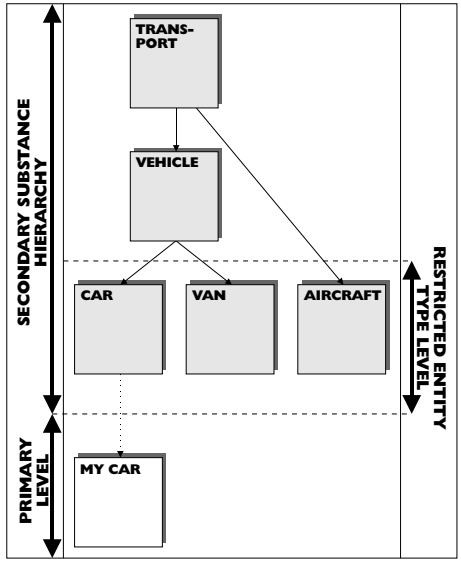
We now look at the part of the substance paradigm that was 'simplified' out of the entity paradigm—its secondary level hierarchy. This can be divided into two interlinked hierarchies:

- The secondary substance hierarchy, and
- The secondary attribute hierarchy.

### 6.2.1 Secondary substance hierarchy

We use my car to illustrate what the secondary substance hierarchy is. My car is a car—in substance-speak, my car's primary substance has a car secondary substance. Cars are a type of vehicle—vans are another type. Similarly, vehicles are a type of transport—aircraft and ships are other types. From a substance paradigm viewpoint these types are all secondary substances and the paradigm recognises that they form a hierarchy (shown in *Figure OP1–14*). In the entity paradigm, there is no hierarchy and entity types are restricted to a single level—also indicated in *Figure OP1–14*.

Figure OP1–14
A substance hierarchy



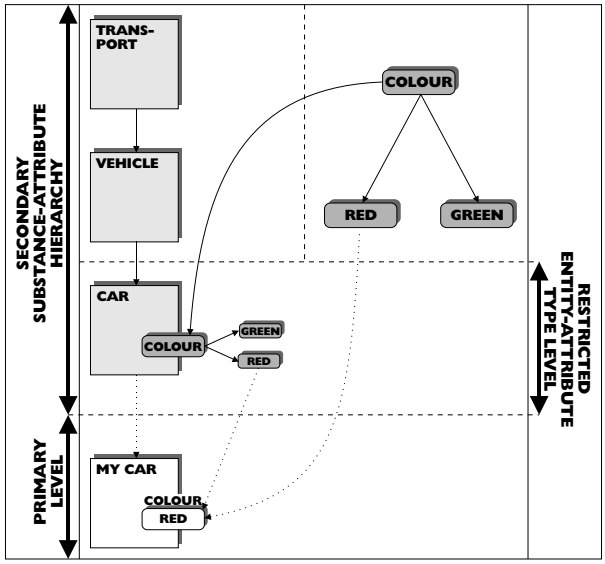

### 6.2.2 Secondary attribute hierarchy

It is not only secondary substances that have a hierarchy, so do secondary attributes. My car is red and cars are coloured. In substance-speak, my car's primary substance has a red primary attribute and secondary car substance has a colour attribute. In this respect, the substance and entity paradigm are similar. But other substances are red (apples, fire engines, etc.), so there are other red primary attributes and these red attributes have something in common, they are

all red. In the substance paradigm this is explained by having an independent red secondary attribute.

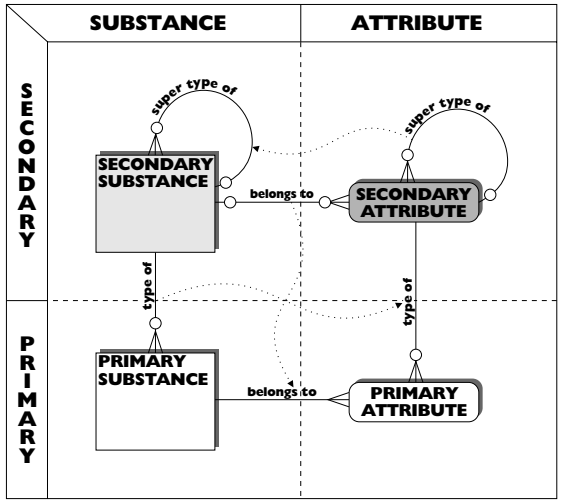
Similarly, other secondary substances (as well as cars) are coloured; so there is an independent colour attribute that has the independent red attribute (and the other individual colour attributes) as part of it. This results in the framework shown in *Figure OP1–15*. My car's red attribute can be called its colour attribute—it is inherited from the secondary level car's colour attribute. We then say the value of my car's colour attribute is red. The red-colour relationship, independent of my car, is reflected by red 'belonging to' colour in the secondary attribute hierarchy.

Figure OP1–15 Substance–attribute framework




The schema of the relationships between the substance paradigm's particles in *Figure OP1–16* shows these secondary level hierarchies. Compare this with *Figure OP1–3* and you can clearly see that the substance paradigm has more structure at the secondary level than the entity paradigm. This structure is what was removed by the entity paradigm's simplification.

Figure OP1-16 Schema of substance paradigm's particles and levels

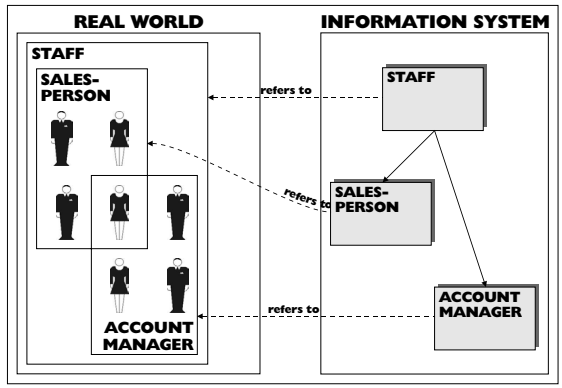



## 6.3 The substance paradigm's solution to the staff example

We can use the earlier staff example to illustrate the substance paradigm's superior semantics. We can show how its secondary hierarchy enables it to reflect the real world more accurately.

When we look at the staff example through substance spectacles, we discover a more consistent system. In this, individual staff (such as Margaret Bottomley) are primary substances and Staff, Salesperson and Account Manager are secondary substances. There is a substance hierarchy in which the Salesperson and Account Manager substances both 'belong to' the Staff substance as shown in *Figure OP1-17*. When the entity paradigm was simplified, hierarchies such as these were flattened, making it difficult to reflect the real world accurately and consistently.

Figure OP1–17
Aristotelian staff hierarchy




## 6.4 Why the substance paradigm was simplified

Why did the secondary hierarchy have to be simplified? When the entity paradigm was being developed, paper and ink were the prevailing information technology. Its two-dimensional structure could not handle the more sophisticated structure of the substance paradigm. This had to be simplified, so that it could operate within paper and ink technology's rows and columns.

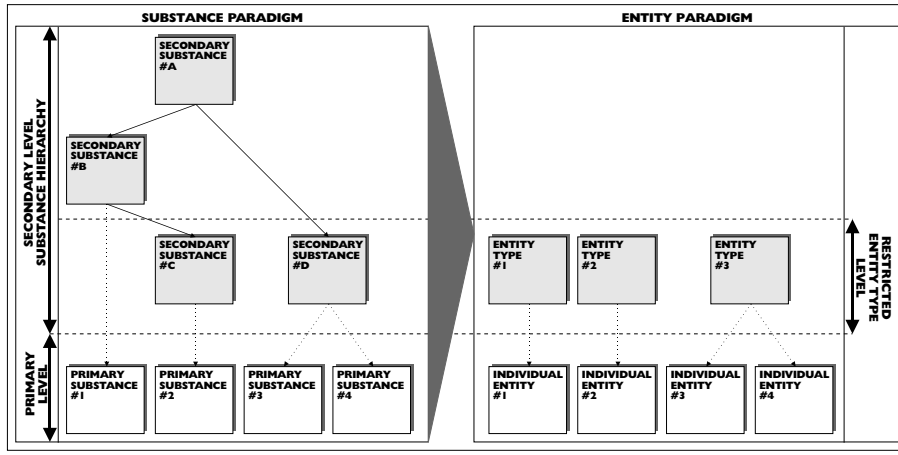
The entity paradigm developed, within the constraints of paper and ink technology, in response to operational needs. It did not develop in the same rational way as the substance paradigm. There was, as far as we know, no-one working out what the entity paradigm's fundamental particles were and what they meant. However, as most people see the world through the substance paradigm, it was a natural starting point for the many minds that contributed to the entity paradigm's development. The substance paradigm was too sophisticated for paper and ink technology and so the question was—how to simplify it?

## 6.5 How the substance paradigm's particles were simplified

The entity paradigm has a basic structure of three levels as shown in *Figure OP1–5*. This results in a framework restricted to a flat entity type level as shown on the right-hand side of *Figure OP1–18*. Compare this with secondary substance's unconstrained multiple-level hierarchy on the left-hand side of the figure.

*Figure OP1-18* makes it clear that it is really only the substance paradigm's secondary level that needed simplifying. Its primary level can be directly translated into the individual entity level. Whereas, its secondary level hierarchy cannot be directly translated into the flat entity type level. To fit the substance paradigm into the entity paradigm's framework shape, its secondary level—both substance and attribute—has to be flattened down to a single level.

Figure OP1-18 \The paradigms' different structures




## 6.5.1 Selecting the natural type level entity

For secondary substances, this means that we have to select a single layer, slicing through the hierarchy, to stand as entity types. We can illustrate this with an example. Consider the secondary hierarchy in *Figure OP1-19*. It has a band across it indicating the selected layer of substances. *Figure OP1-20* shows this layer transformed into entity types. The substances above the selected layer in the hierarchy disappear, while the substances below the selected layer in the hierarchy are transmuted into attributes. A similar transmutation happened in the staff example. The Account Manager and Salesperson substance in *Figure OP1-17* are flattened into attributes in the staff entity system in *Figure OP1-10*.

Figure OP1-19 Selecting a secondary substance layer

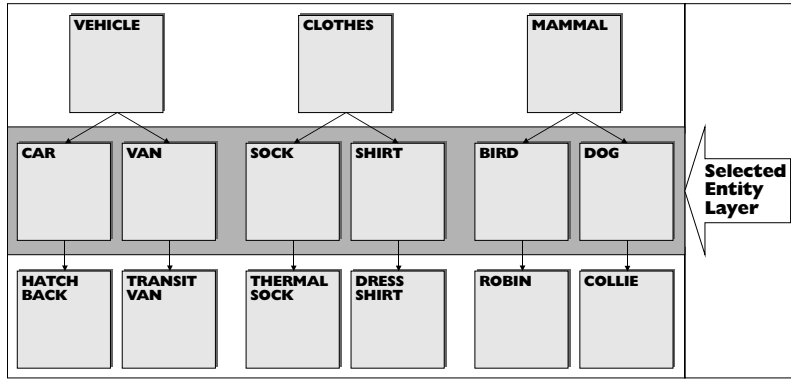
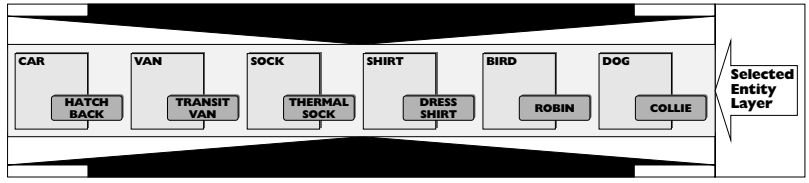



Figure OP1-20 The transformed entities




Selecting a layer in the hierarchy is not as unnatural as it may look. Psychologists have shown that we immediately allocate a thing to a natural level of 'substance'. For instance, most people immediately classify a robin in their garden as a bird rather than the lower level robin or higher level animal. However, this 'natural' level is not a feature of the world. It can vary from person to person. For instance, a bird watcher, unlike other people, is more likely to immediately classify a robin in his or her garden as a robin (maybe even the variety of robin) rather than a bird.

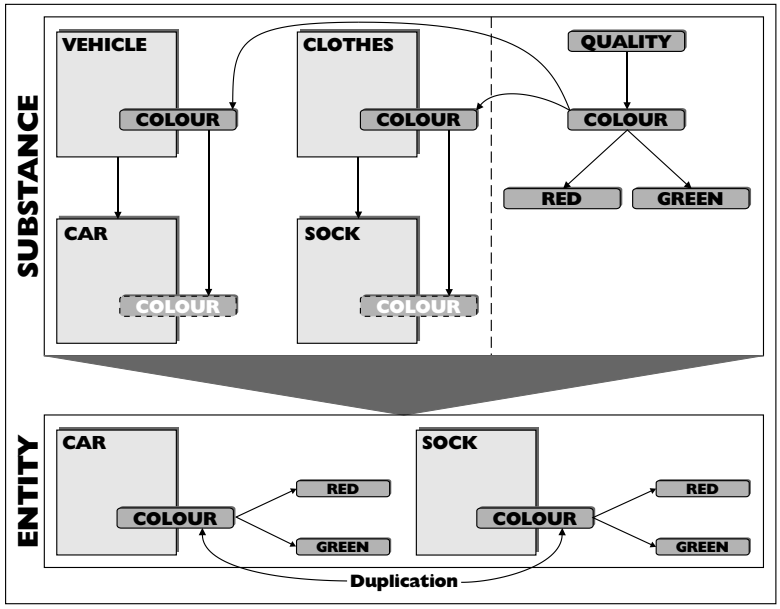
The natural level of classification helps us to decide how to flatten the secondary substance hierarchy into an entity type layer. It is also a vital factor in the design of good user interfaces. Items presented to users on a screen need to be at their 'natural' level.

### 6.5.2 Making attributes dependent

It is not just the secondary substance hierarchy we need to flatten; we also need to flatten the secondary attribute hierarchy to fit it into the entity structure. Not only do we have to flatten it, but we also have to make it completely depend-

ent on its corresponding entity type. We can see how this happens in the colour hierarchy example shown in *Figure OP1–21*.

Figure OP1–21 Flattening the secondary attribute hierarchy




You may have noticed that the dependent attributes belonging to substances above the selected layer, such as vehicle's colour in the example, disappear along with their substances. Notice also that the independent colour attribute hierarchy disappears completely. This means that it can no longer be re-used across the car and sock entity types. In this example, the simplification creates a need for two dependent instances of each colour attribute—one for each entity. This is an example of a general constraint in the entity paradigm. When an attribute has a fixed range of values—as the colour attribute does here—the substance paradigm's independent attribute hierarchy has to be re-constructed anew as dependent attributes for each of the entity types. This significantly reduces the re-use potential.

# 7 Simplifying the substance paradigm's treatment of relationships

It is not just the substance paradigm's secondary hierarchy that was simplified away. So was part of its treatment of relationships—one of the four key types of things we identified in *AS4—Focusing on the Things in the Business*. We now look at how the substance paradigm handles relationships and how this was simplified for the entity paradigm.

## 7.1 Aristotle's relations and co-relations

In the substance paradigm there are two particles—substances and attributes. Everything has to be one or another of these (or some combination). So we have to use one or another particle to describe relationships such as:

Queen Elizabeth is the mother of Prince Charles

The only practical solution, within the paradigm, is to treat relationships as relational attributes. As 'is the mother of Prince Charles' is the predicate of the sentence, it is an attribute. As 'Queen Elizabeth' is the subject, it is the substance that the attribute belongs to.

However this relationship can also be described as:

Prince Charles is the son of Queen Elizabeth

This and the earlier sentence describe the same relationship. But, a subject–predicate analysis of this sentence gives a different result. In this case, as 'is the son of Queen Elizabeth' is the predicate of the sentence, it is an attribute. As 'Prince Charles' is the subject, it is the substance that the attribute belongs to. We appear to have a problem—two different relational attributes for the same relationship.

Aristotle was well aware of this problem. In *Categories*, he wrote:

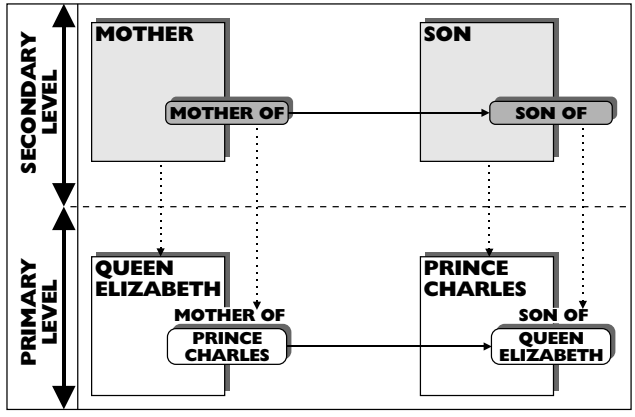
> *Let us now turn to Relation. We call a thing relative, when it is said to be such as it is from being of some other thing or, if not, from its being related to something in some other way. Thus 'the greater' is said to be greater by reference to something outside it. For, indeed, when we call a thing 'greater' we mean by that greater than something. 'The double' is called what it is from its being the double of something. For 'double' means double of something. And so with all terms of that kind.*

His 'solution' to this problem was facile. He suggested that we give each relational attribute a co-relational or correlational attribute (also called a correlative):

> *All relatives have their correlatives. 'Slave' means the slave of a master, and 'master' in turn implies slave. 'Double' means double of its half, just as 'half' means half of its double. By 'greater', again, we mean greater than this or that thing which is less, by 'less' less than that which is greater. So it is with all relative terms.*

In our 'Queen Elizabeth is the mother of Prince Charles' example, the correlation of the 'mother of' attribute is the 'son of' attribute. This is illustrated in *Figure OP1–22*.

Figure OP1–22 Relational and correlational attributes

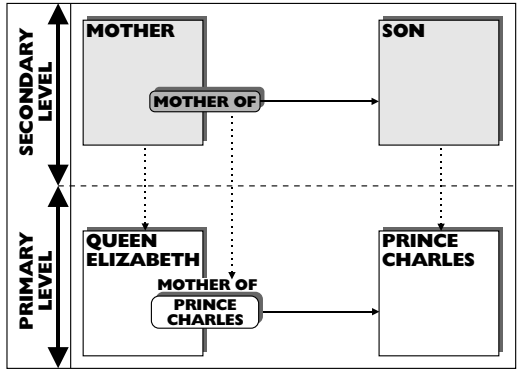



We will briefly look at correlational attributes again in *OP3—Logical Ontology Paradigm*, when we re-engineer the relational attribute pattern into its logical counterpart.

## 7.2 The entity paradigm's simplified treatment of relations

Aristotle was concerned with seeing the real world clearly and accurately. The entity paradigm is more concerned with effectively implementing information in paper and ink technology. This gave it a problem with relational attributes having correlational attributes. The additional correlational attributes are—as far as it is concerned – duplicates. The relational attributes contain all the information it needs. Its solution is to drop all correlational attributes. The entity paradigm's treatment of the 'Queen Elizabeth is the mother of Prince Charles' example looks like *Figure OP1-23*.

Figure OP1-23 Relational attributes without correlational attributes




## 7.3 The problem with relations as attributes

As in the earlier staff example, here we have a weakness in the paradigm's particles leading to a distorted view of the world. The major distortion caused by the relational attribute particle is clearly visible in *Figures OP1-22* and *OP1-23*. In both figures, the most important part of the relationship, the link between the two substances is drawn as a line. But this link is only implicit in the substance and entity paradigms. Attributes, by their nature, belong to only one substance. Neither paradigm has a particle that can capture explicitly the vital linking element. Because neither paradigm is powerful enough to explicitly reveal this, there is no chance of it being reflected accurately in a business entity model.

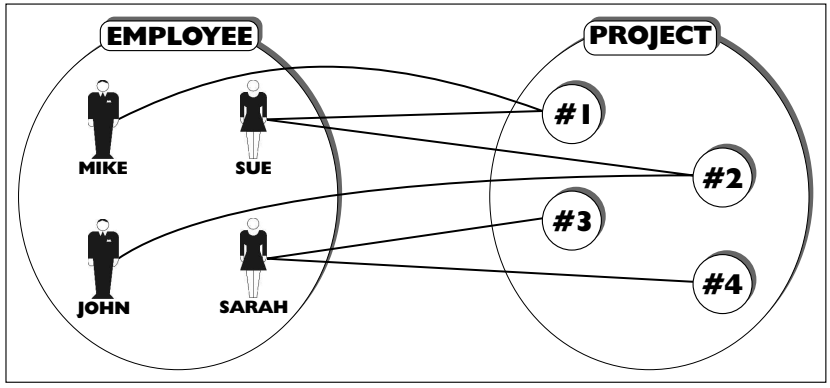
There are other distortions. We can follow the substance paradigm and assume that each relational attribute has a correlational attribute. But then we end up with two fundamental particles to handle a single relationship pattern in the business. The entity paradigm avoids this problem by dropping the correlational attribute. But this has its own problems. We can see this in our example. The 'mother of/son of' relationship really involves two entity's, Queen Elizabeth and Prince Charles, and we can only choose one of these entities as the home for the attribute. In *Figure OP1–23*, we chose Queen Elizabeth as its home; but there is no reason why we should not have chosen Prince Charles. The real world only has a single pattern; so this choice is an indication that we are not capturing its patterns accurately.

## 7.4 Entity business modelling's problem with many-to-many relations

As we have just seen, relational attributes—even without correlations—are an awkward pattern. This has had an unhealthy impact on entity business modelling. One good example is the way in which modellers typically resolve many-to-many relations in entity-oriented models. Consider this example.

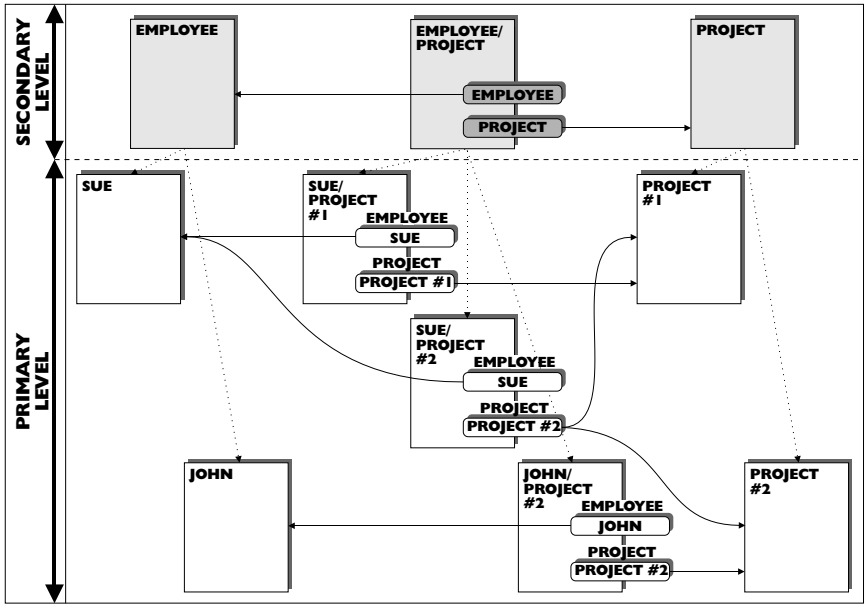
Assume a system records the employees of a company and the projects that they are working on. Assume also that an employee can work on several projects and that a project typically has many employees working on it. Then the situation, from an entity-oriented point of view, is shown in the Venn diagram in *Figure OP1–24*.

Figure OP1-24
Venn diagram
for employee/
project system




A problem now arises. The attribute pattern is not strong enough to reflect a many-to-many relation such as 'employee works on project'. The traditional solution is to build the model as if there was a new 'relational' pseudo entity, employee-project, with employee and project attributes as shown in *Figure OP1-25*.

Figure OP1-25
Entity model for
employee/
project relation



From a practical point of view, this model can be used to build a working system. But because we have assumed the new entity exists rather than mapped it, we are faced with awkward questions:

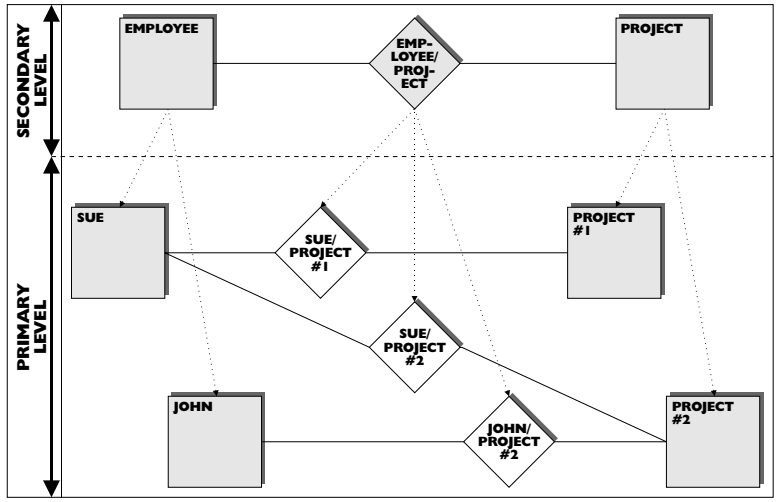
- What is an employee–project entity?
- Is an employee-project as much of an entity as an employee and a project?

The answer to the second question is clearly no. The many-to-many relation has forced us to build the model as if there were employee–project entities, even though there is no external evidence for them. It is the weakness of the attribute pattern when it comes to relationships that forces us into this position. This weakness means that not only can we not reflect the structure of the world directly but we have to construct false 'pseudo' entities.

## 7.5 A partial solution: entity–attribute–relation modelling

Modellers have recognised this problem of fitting relations into the attribute pattern for some time. They have found a way around the problem that keeps most of the entity paradigm intact. It involves having a new particle to handle relations—a relation particle—with its own new patterns. This extended paradigm is known as the entity-attribute–relation (E–A–R) paradigm. We can see how it works in our example. We now model the problem employee-project relation as a relation not a new entity as shown in *Figure OP1–26*.

Figure OP1-26 Entity-attribute-relation model for employee-project system




Introducing this new particle takes two steps in the right direction. It makes the structure of relations more explicit in the model and it recognises that relations are not entities. But its overall semantics is still solidly based on entities. We shall see that this is shaky ground when we investigate the logical paradigm in *OP3—Logical Ontology Paradigm*.

## 7.6 Another partial solution: O-O programming languages' group attributes

O-O programming languages adopt a different and apparently simpler solution to the many-to-many relation problem. They allow an object's 'attribute' to have a group of values and so point to many programming 'objects'. This eliminates the need to create extra 'objects' to resolve many-to-many connections. However, while it works from an operational system point of view, from a semantic, modelling, point of view, it does not. This becomes clear when we ask for a semantic explanation of what an attribute with a group of values refers to. There isn't one.

# 8 Our current way of seeing stored information

The entity paradigm is the natural way for literate people to see information stored on paper or computer systems. It has its foundations in the substance paradigm; our current way of seeing things. And it evolved out of our culture's thousands of years of experience in storing information on paper in lists and tables. Just think how easy most of us find it to draw up lists and tables. We naturally move from the substance paradigm for information in our minds to the entity paradigm for information stored outside them.

## 8.1 The four key types of things

However, if we assess the entity paradigm's semantic power in terms of the four key types of things identified in *AS4—Focusing on the Things in the Business*, we can see the deleterious effect its simplification has had. It mainly affects generality, where there are two changes and both are for the worse. First, simplifying the secondary hierarchy away has removed the apparatus for handling more and less general types (shown in AS4's *Figure AS4–12*). Secondly, it is not practical to accurately reflect the distinction between entity types and attribute types in information systems. The staff example showed how attribute type signs can refer to entity types. *Figures OP1–19* and *OP1–20* illustrated how this inevitably happens as the secondary hierarchy is simplified.

## 8.2 Learning to ignore the semantic problems

However, the simplified entity paradigm was, and is, a very successful means of managing business information. It may have problems reflecting the structure of the world directly. But, while the paradigm is successful, it does not seem to make sense to pursue these problems. In fact, the best policy seems to be—learn to ignore them.

This is how most people work with the entity paradigm. To see this just ask yourself whether the staff example above proves to you that the entity paradigm is inherently wrong. I suspect many of you will say that it does not; that all it high-

lights is an academic issue about semantics. In this way, we legitimise ignoring the problem.

However, as we go through the re-engineering to the object paradigm and we get a better understanding of the semantic issues, we will develop a different perspective on this. It will become clearer and clearer that if we ignore these semantic issues, then we will miss a big opportunity for improving business modelling and so computer systems.

# 9 The next paper

In the next paper, *OP2—Substance Ontology Paradigm*, we look in more detail at how the substance paradigm addresses these semantic issues.

## 9 The next paper

# 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?*

# ENTITY ONTOLOGY PARADIGM

## A-F

### A



### C



### D



### E



### F

## O



## P



## R



## S



## T



## U



## V