B usiness
O bject
R eference
O ntology

Program

s i m p l i f y i n g s e m a n t i c s

Working Paper

AS4

The BORO Approach: Strategy-4

Focusing on the Things in the Business

Issue: Version - 4.01 - 01-July-2001

AS4

THE BORO APPROACH: STRATEGY - 4

FOCUSING ON THE THINGS IN THE BUSINESS

CONTENTS

1 Introduction - AS4-1

2 Focusing the re-engineering on things in the business - - - - - - - - - - - - - AS4-1

2.1 The major elements of an information paradigm - AS4-1

2.2 Focusing on 'things in the business' - AS4-2

3 Problems identifying 'things in the business' - - - - - - - - - - - - - - - - - - AS4-2

3.1 The problem with our entity paradigm - AS4-3

3.2 Problems finding 'things in the business' – a simple example - - - - - - - - - - - - AS4-4

3.3 Why this semantics problem exists - AS4-5

4 Ignoring 'things in the business' - AS4-5

4.1 Imposing the data–process distinction onto 'things in the business' - - - - - AS4-6

4.2 Ignoring the difference between understanding and operation - - - - - - - - - - AS4-9

4.3 Distinguishing between operational re-use and generalisation - - - - - - - - - - AS4-9

5 What types of things (in the business) do we re-engineer? - - - - - - - AS4-11

5.1 Particular things - AS4-12

5.2 General types of things - AS4-13

5.3 Relationships between things - AS4-14

5.4 Changes happening to things - AS4-15

6 Our starting point—the entity paradigm - AS4-16

6.1 Engineers often use scientifically 'out-of-date' paradigms - - - - - - - - - - - - - AS4-16

7 Arriving at an object semantics for 'things in the business' - - - - - - - AS4-17

CONTENTS AS4

7.1 The separate evolution of information semantics - AS4-17
7.2 Following the semantic re-engineering route - AS4-18

8 Re-engineering the 'things in the business' -AS4-18

BORO Working Papers - Bibliography -AS4-21

INDEX - AS4-23

AS4

THE BORO APPROACH: STRATEGY - 4

FOCUSING ON THE THINGS IN THE BUSINESS

1 Introduction

AS3—What and How we Re-engineer looked at how we re-engineer entities into objects. In this paper we focus on the specific areas we need to re-engineer.

2 Focusing the re-engineering on things in the business

We focus the re-engineering on one area of the entity paradigm—the 'things in the business'.

2.1 The major elements of an information paradigm

Information paradigms, such as the entity paradigm, are typically divided into the following three elements:

- Technology (or method and materials of construction),
- Syntax (or, structure), and

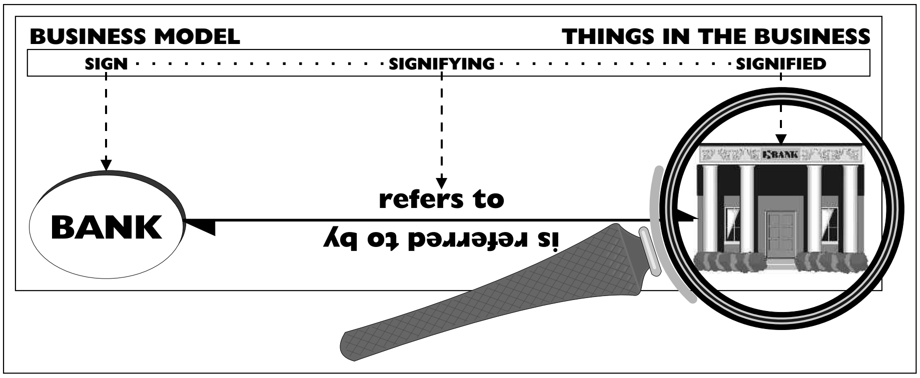
- Semantics (or, meaning).

Computer people naturally focus on the technology element. It seems indubitable that there is a fundamental change going on in information because of the new information technology—computing. And this technology is innovative and exciting. So it is not surprising that some people overlook the two non-technological elements.

2.2 Focusing on 'things in the business'

However, it is in one of these non-technological elements, semantics, that we find the key to business objects—'things in the business'. Look at *Figure AS4–1*, which illustrates the semantic 'signifying relation' between the sign and the signified. It is an obvious truth that we cannot construct signs that reflect a business accurately unless we can see the things in the business clearly. Our re-engineering focuses on developing a much clearer, more accurate, view of these.

Figure AS4–1 Focusing on 'things in the business'



3 Problems identifying 'things in the business'

Most computer people currently assume it is easy to identify the 'things in the business' (the signified) and make sure that the model refers (maps directly) to them. However, if we actually try and identify the things in the business, we come across a problem. Astounding as it may seem, most people cannot clearly and explicitly articulate exactly what these entity 'things in the business' are.

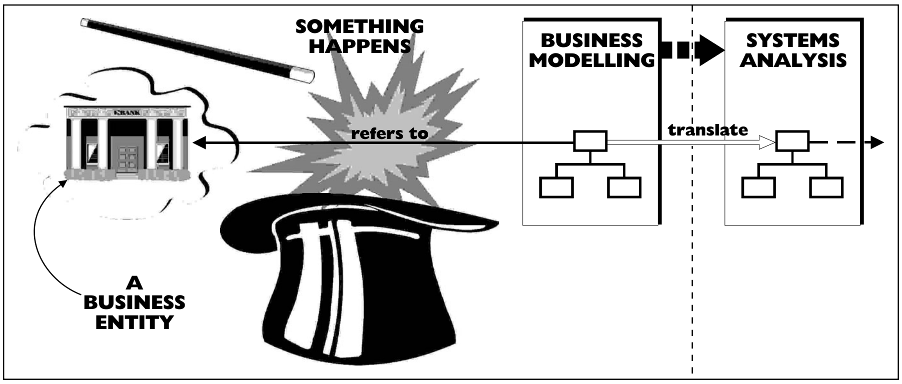
3.1 The problem with our entity paradigm

Many people in the computer industry have lots of experience in constructing business entity models. They must know what a business entity is. One might think that if we asked them, they would tell us what one is. But when we actually ask them, they come up with examples not explanations. They say something like 'an entity is a thing like a car', or 'a company' or 'a foreign exchange deal'. They cannot provide an explanation because their understanding of entities is so deeply ingrained that it is unconscious. And their acceptance of it so complete that asking what an entity is seems to have no practical use.

Most people adopt a similar attitude. They instinctively assume that they and everyone else know what a business entity is. For instance, IT managers expect even the most lowly trainee programmers to come fully equipped with the knowledge of what a business entity is—though not necessarily what the particular entities are for their business. They expect this, despite the fact that the programmers are not consciously aware of what one is nor are they likely to be formally taught this.

Nevertheless, the managers' expectations are justified because it is plain from the trainee programmers' behaviour that they do know. It is as if something magical and mysterious is linking the model to the business entities—as shown in *Figure AS4-2*. It seems odd to me (and if you absorb *The BORO Working Papers*, it will seem odd to you) that companies regularly spend millions of pounds building computer systems that depend on such mysterious semantics.

Figure AS4–2
Magical and mysterious semantics

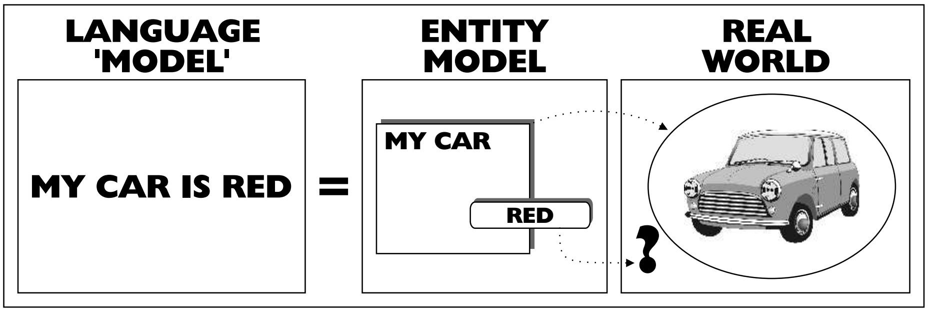


3.2 Problems finding ‘things in the business’ – a simple example

We can appreciate how mysterious our current entity semantics is by looking at this simple example. Consider what a simple entity model of the sentence, ‘my car is red’ would look like. It would have a ‘my car’ entity with a ‘redness’ attribute—as shown in *Figure AS4–3*. The entity ‘my car’ clearly refers to my car, as shown in the diagram.

My car’s redness is more of a problem. It cannot point to my car; that is an entity not an attribute. Apparently, the sign for my car’s redness does not refer to anything. If this is the case, then the model does not directly map onto things in the real world. In *OP4—Business Object Ontology Paradigm*, we will see how resolving fundamental problems, such as these, leads us to the object paradigm.

Figure AS4–3
‘My car is red’



3.3 Why this semantics problem exists

In everyday life, we quite often come across mysterious unexplained ideas. We find an unconscious awareness of something coupled with an inability to discuss it. In the case of business modelling, this is a sure sign of a paradigm. It is a sign of a way of seeing things that is, by its very nature, so deeply embedded in our minds that we are not conscious of it. And we believe so firmly in it that we cannot question it.

Unconscious control is normally an extremely sensible way of dealing with fundamental situations like this. Situations that we are so sure of that they rarely need conscious review. If all our actions had to be under our conscious control, it would take ages to make even the most simple decision. It would be as if the board of directors of a large company insisted on being involved in every decision, from appointing a new chairman to buying a box of rubber bands. The only practical way out is to delegate the control of those situations we are sure of into our unconscious.

The circumstances change when a paradigm needs shifting. The advantages of unconscious control now turn into disadvantages. We can see that here. If we are going to re-engineer the entity paradigm, we need to know what it is. However, whenever we start asking about entities, our unconscious kicks into operation and interrupts the question. It tries to stop the question being consciously considered, often by insinuating that it is irrelevant or obvious. (The Chairman of the Board might use similar tactics to dismiss a question about buying a box of rubber bands.) This makes it difficult to start the re-engineering.

4 Ignoring ‘things in the business’

This unconscious use of the entity paradigm has led to a tendency to ignore the ‘things in the business’ and focus on the signs that refer to them. For example, people working in the computing industry tend to focus on the world of computer information, especially when they are dealing with computers. As a result, they end up imposing the computer world's framework onto the 'things in the business'.

4.1 Imposing the data-process distinction onto 'things in the business'

A good example of this is the way they impose its data-process distinction. Computer technology, unlike paper technology, can both store and process information. So, in the computer world, the distinction between information that is stored—data—and processing information—process—is an important one. But it is only important in the computer system. It is irrelevant to the things in the business that the data and process refer to.

Nevertheless, business modellers impose the data-process distinction onto 'things in the business' with disastrous results. We can illustrate this with a simple example. Most accounting systems have an account movements file. They also have a program that processes the movement records on that file and posts them to the accounts file, updating the balance with the movement. In computing terminology, the account movements and accounts are both data and the accounts movements update program is process. A model of this part of the system would look something like *Figure AS4-4*.

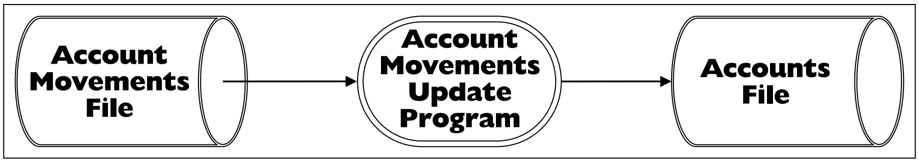


Figure AS4-4 Account movements system model

You will find that business models for accounting systems often have a similar shape to the model in *Figure AS4-4*. Account movements are represented as data and the account movements update of the accounts as a process. This seems a natural way of modelling the business to anyone living in a computer world. It is also the wrong way.

To see this, we need to look at the data-process distinction again. It is a fundamental distinction in a computer system. Data and process are quite different. In an information system, data persists over time; whereas, processes do not. They happen. There is a similar distinction in the real, non-information, world. Things persist over time and changes do not. When people who live in a computer world model the real world, they represent it as data and process. They either ignore the real world's distinction between things and changes or assume that data maps directly onto things and process onto changes—as shown in *Figure AS4–5*.

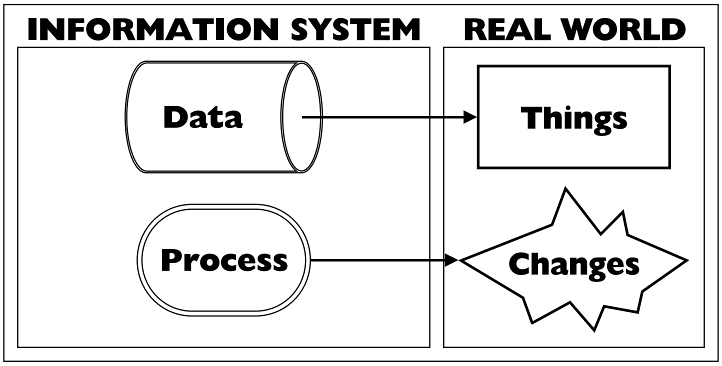
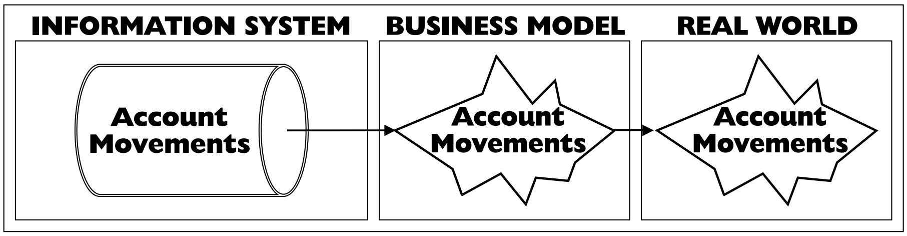


Figure AS4–5 Data-process spuriously reflecting things—changes

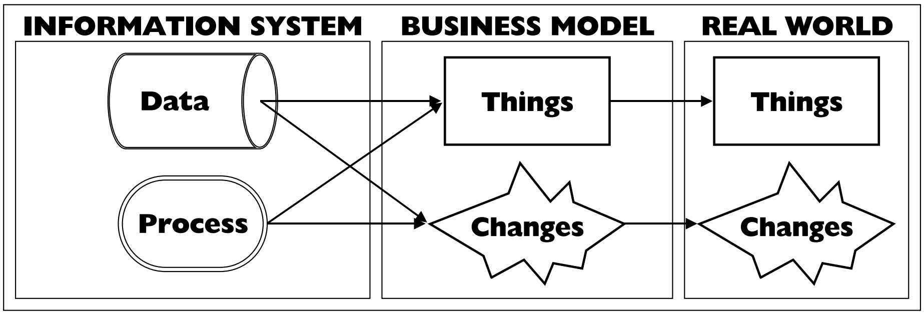
This is a mistake. The business model, and not the information system, is meant to map onto the things in the business (the real world). The problem arises because data does not necessarily map onto things (or process, changes). To see how this causes a problem consider the account movements again. Ask yourself whether the individual account movement records represent a thing or a change in the business? The correct answer is they represent changes. For example, if I pay £100 into my bank account, my paying in is not a thing but a change. And the change is recorded (represented) by data in the form of an account movement record. Once we understand this, we no longer draw the account movement in our business models as data, but as a change. This is illustrated in *Figure AS4–6*.

Figure AS4–6
Account movements business model



This example clearly shows that the distinction between data and process in a computer system is not based on differences between things and changes in the real world. It is based on whether things in the computer system persist or not. The data-process distinction and the things—changes distinction have the same underlying basis. But this in no way implies that data represents things and process represents changes—as suggested incorrectly by *Figure AS4–5*. It turns out that the representing relationship is much more flexible. As illustrated in *Figure AS4–7*, both data and process can represent either things or changes.

Figure AS4–7
Data-process correctly representing things—changes



This provides a simple test of whether a model is representing the business or an information system. If the model's notation classifies changes in the real world as data (as, for instance, the example in *Figure AS4–4* classifies accounting movements as things or data), then it is describing the computer system and not the business. Therefore, it is not a business model. Unfortunately, many so-called business models fall into this category.

These distinctions also highlight one difference between system and business objects. System objects encapsulate data and process into one object. Business objects, on the other hand, deal with the things—changes distinction. As we shall see in *OP4—Business Object Ontology Paradigm*, business objects equivalent of encapsulating data and process are patterns that generalise across the things—changes distinction.

4.2 Ignoring the difference between understanding and operation

This confusion about whether data–process represents things—changes is just part of a wider confusion between understanding things in the business and the operation of things in the computer system. In *AS2—Using Objects to Reflect the Business Accurately* we touched on the distinction—on how business modelling deals with understanding the things in the business; whereas, the other, later, stages of system building were more concerned with the operation of the final system.

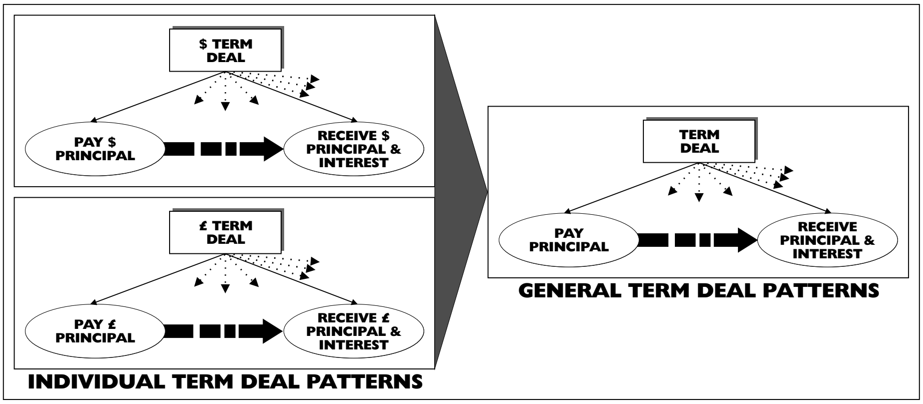
This distinction between understanding and operation is important for business objects. We can get an idea of why, by looking at how it affects a notion dear to people working in O-O re-use. O-O creates an environment where there is more potential for re-use. That is why O-O systems can be simpler and more compact. This re-use works at both an operational and an understanding level. Unless we make a clear distinction between the two levels, we do not take full advantage of O-O's potential. The following example shows how we distinguish between the two.

4.3 Distinguishing between operational re-use and generalisation

Everyone, not just O-O system builders, is familiar with operational re-use. We know what happens when we operationally re-use a pattern or component. We apply it in new situations. Re-use works in a different way at the understanding level, where it is closely tied in with generalisation. The following simple modelling example illustrates this.

Assume that we are building a model of a money market trading system and we are focusing our analysis on $ and £ term deposit placed deals. We notice that these two types of deals have a similar pattern of settlement. For instance, in both cases, the principal is paid away on an agreed date and the principal plus interest is received back after an agreed term. It seems like a sensible idea to consider whether a copy of the program code for the $ term deal could be re-used to process £ term deals. This is an operational approach.

Figure AS4–8 Constructing general patterns


$ TERM DEAL
PAY $ PRINCIPAL
RECEIVE $ PRINCIPAL & INTEREST
£ TERM DEAL
PAY £ PRINCIPAL
RECEIVE £ PRINCIPAL & INTEREST
INDIVIDUAL TERM DEAL PATTERNS
TERM DEAL
PAY PRINCIPAL
RECEIVE PRINCIPAL & INTEREST
GENERAL TERM DEAL PATTERNS

On the other hand, we could think about re-use at an understanding level. Then our chief concern would be finding similarities in the patterns for the two types of things. Patterns that we could use to generalise. Thinking this way we would construct a more general term deal pattern that applies to both of the less general types of deal—as shown in *Figure AS4–8*. We are not really just finding a general pattern. What we have done is constructed a new more general type of thing in the business—a general term deal. We can think of this as building a similarity tree of things as illustrated in *Figure AS4–9*.

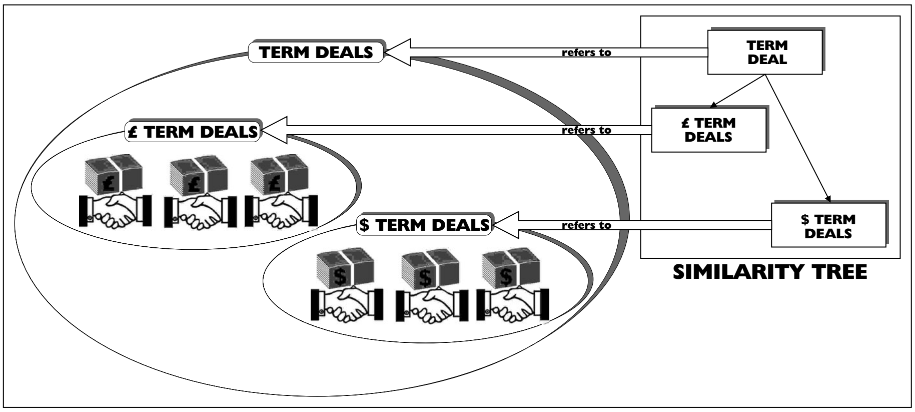


Figure AS4–9 Identifying similar types of 'things in the business'

It is when we start designing the system that we should shift to an operational view of re-use. We can ignore the things in the business and talk of the general term deal pattern (or program code) being re-used for both $ and £ term deals.

5 What types of things (in the business) do we re-engineer?

We are already focused on the things in the business. I have found that the re-engineering to the object paradigm is greatly simplified if we restrict our focus further. All that we need for the re-engineering can be found in a small group of four types of things. When re-engineering these four, we re-engineer the whole paradigm. The four are:

- Particular things,
- General types of things,
- Relationships between things, and
- Changes happening to things.

Stretching a point, we call these the four key types of things (changes are not really things, more a pattern of relationship between things).

At first sight, these four appear simple and appear as if they would not be problematic. But, by the time we have re-engineered to the object paradigm, we shall realise how sophisticated our way of seeing has to be to handle them accurately. When we follow the re-engineering in *O—ONTOLOGY Papers* , we will concentrate on how each paradigm deals with these areas.

5.1 Particular things

Particular things are individual things. For example, see a particular table or a particular chair—as illustrated in *Figure AS4–10*. These are often called physical bodies and are the simplest and most basic items in semantics. An information paradigm needs to explain what makes something particular—what particularity is. This involves more than just saying particular things are individual, concrete, and tangible. The paradigms we look at offer very different explanations of what they are.

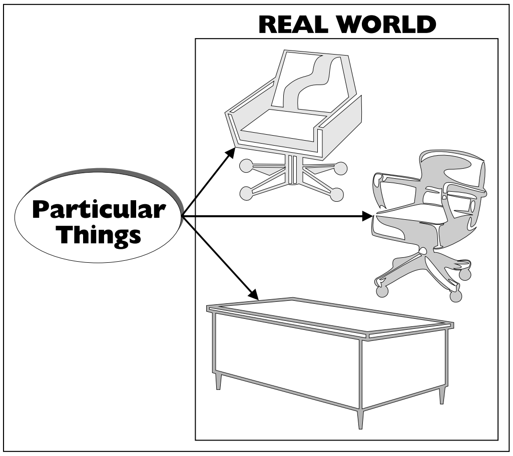


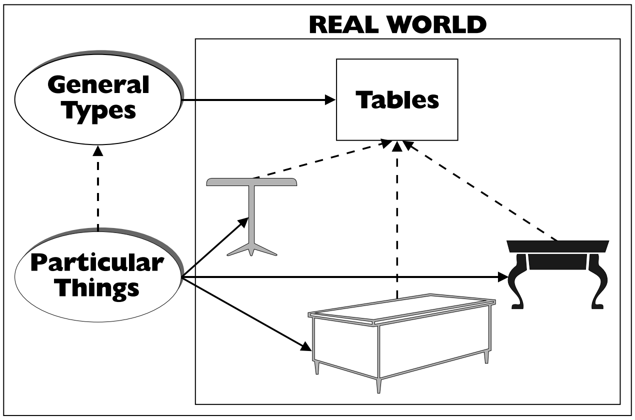
Figure AS4–10
A particular table and two particular chairs

Particular things have one important pattern. We see them as having properties. This was raised in the earlier example of my car (a particular thing) illustrated in *Figure AS4–3*. As we said then, an information paradigm needs to explain what my car's redness is in the real world.

5.2 General types of things

We naturally group particular things into general types. For example, the 'particular' tables—illustrated in *Figure AS4–10*—may be grouped along with other 'particular' tables into the general type, tables. These two are quite different. Particular things are normally concrete and tangible; whereas types, such as table, are normally abstract and intangible. For example, it does not seem to make sense to ask what the type tables feels or looks like. Not surprisingly, we naturally distinguish the general from the particular. We also naturally relate them. For example, when we see a particular table, we naturally classify it as belonging to the general type, tables. This is illustrated in *Figure AS4–11*.

Figure AS4–11 General types and particular things



General types also have a common pattern relating one type to another, the 'more general' pattern. For example, the general type, furniture, is 'more general' than the general type chairs. An information paradigm needs to explain what this more general pattern—illustrated in *Figure AS4–12*—is in the real world.

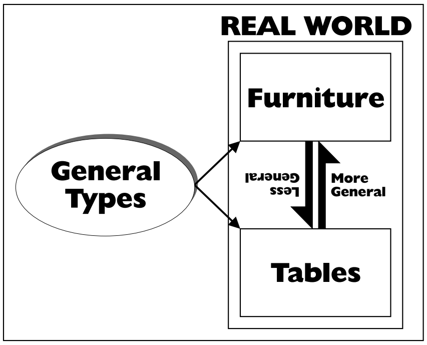


Figure AS4–12
More and less general types

5.3 Relationships between things

The next type of object is relationships. People have relationships—for instance the two ‘particular things’, Queen Elizabeth and Prince Charles, are related: Queen Elizabeth is the mother of Prince Charles. This is an example of a blood relationship. There are other non-blood relationships, for instance a particular chair may be at a particular desk.

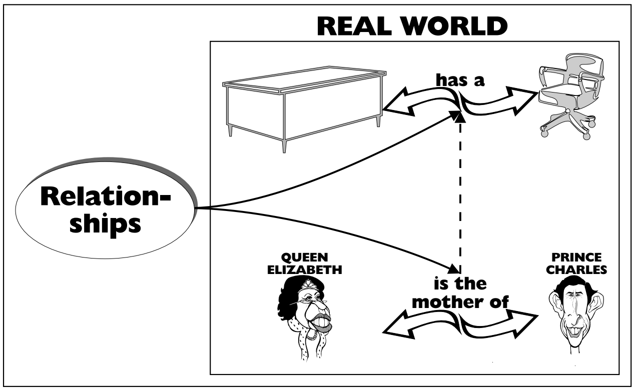


Figure AS4–13
Two examples of relationships

These two examples (illustrated in *Figure AS4–13*) are relationships between particular things—what might be called particular relationships. There are also relationships between general types. For example, we can generalise the ‘Queen Elizabeth is the mother of Prince Charles’ relationship to ‘(the type) mother can be a mother of (the type) children’. These are what might be called general relationships. Seen this way, the particular general pattern applies not only to things and types but also to relationships—as illustrated in *Figure AS4–14*. All this needs to be explained by an information paradigm.

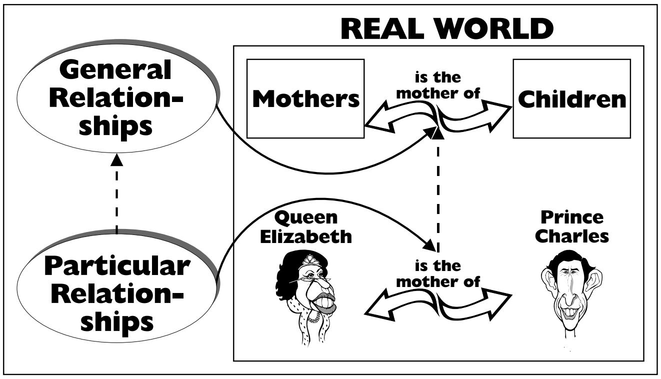


Figure AS4–14 General and particular relationships

5.4 Changes happening to things

At the beginning of this paper, when looking at the data–process distinction, we introduced the things–changes distinction. This provides us with the last type of object: changes. As we said earlier, what distinguishes changes from things are that things persist through time; whereas, changes do not, they happen.

A strong pattern connects changes to things: changes happen to things. Consider for example, a change such as a green tomato turning red. The change (turning red) happens to the tomato (a particular thing). We readily appreciate that the things and changes are two very different types of objects. However, an information paradigm has to give a clear and consistent explanation of what both these objects are and why they are different.

Changes, like things, also have a general–particular pattern. For instance, the change in our example, a particular green tomato turning red, can be generalised to a general type of change, green tomatoes turning red. An information paradigm should explain why changes have the same pattern and what it is.

6 Our starting point—the entity paradigm

Our starting point for the re-engineering is the entity paradigm. In *OP2—Substance Ontology Paradigm*, we see how it and the substance paradigm on which it is based deal with these four key types of things. The substance paradigm was developed by the Ancient Greek Aristotle in the 4th century BC. Some people might find it odd that we currently use an entity paradigm based on something so ancient as the substance paradigm. But, on reflection, however, one should realise that, for practical everyday use, the age of a paradigm is not relevant. The real issue is its suitability for the job. This is why it makes sense for an engineering discipline to use a scientifically 'out-of-date' paradigm.

6.1 Engineers often use scientifically 'out-of-date' paradigms

Once we start looking, we can find many other examples of 'out-of-date' paradigms. We soon recollect that civil and mechanical engineers still use 'out-dated' Newtonian physics although its successor, quantum-mechanical physics, has been available for almost a hundred years. We may be less aware that ship's officers are taught to navigate using a millennia old paradigm of a fixed earth and moving star sphere—one that was superseded over four hundred years ago.

We use these old paradigms because, even though the scientist's latest paradigm may be the most accurate, it is not necessarily the most appropriate for everyday tasks. Unlike scientists, engineers faced with a task have to make a practical decision. That is why they try to choose the most suitable paradigm for the job, no matter how ancient or out-of-date. Information engineers picked the entity paradigm because it was the most appropriate for paper and ink technology. The issue we are facing is that they then imported it wholesale onto computer technology.

7 Arriving at an object semantics for 'things in the business'

Given that the substance paradigm is so old, it should come as no surprise that it is scientifically 'out-of-date. 'Information scientists' have re-engineered a number of new paradigms in the two millennia since it was first formalised. In fact, 'information scientists' were developing the shift to the object paradigm in the first half of this century, well before electronic computers were even invented.

7.1 The separate evolution of information semantics

The easiest way to think about these developments is in terms of the three elements of an information paradigm mentioned at the beginning of this paper:

- Information technology,
- Syntax, and
- Semantics.

Until computers were developed, paper and ink were the leading information technology. And so the entity paradigm was rightly selected as the most practical option for working 'information engineers'. However, at the same time, the world's best thinkers from a variety of fields have been developing semantics. Over time, they have evolved increasingly sophisticated systems. Even though these systems often remained academic, without a practical application, this did not stop them from developing them further. With the development of computing, we have a technology that makes the 'scientifically' advanced systems of semantics a practical proposition.

It is not unusual for people to develop ideas ahead of their times' technology. A well-known example is Leonardo da Vinci. In the 15th century, he drew designs for a helicopter—hundreds of years before the technology needed to build one was available. It was a good idea that could not be put into practice because of the state of technology. In the same way, when object semantics was developed, it was far too rich to work on the then current paper and ink technology.

7.2 Following the semantic re-engineering route

Thinkers have had a long time to develop their 'scientifically' advanced semantics. As you might expect, semantics has moved a long way in the two millennia since the substance paradigm was formalised—in paradigm-speak, there have been a number of shifts. We follow these in *OP—Ontology: Paradigms*.

Following in these thinkers' footsteps is a much easier way of re-engineering than starting from scratch. The shifts have already been worked out for us. All we have to do is understand them. This is just as well, because the computing industry has no real experience of re-engineering or developing semantics.

As our goal here is understanding the object paradigm, rather than the history of semantics, we take a drastically shortened version of the historical path. We just look at the semantics of one intermediate paradigm in the evolution to object semantics—a kind of halfway house. This is the logical paradigm.

8 Re-engineering the 'things in the business'

In this paper we have focused in on the core areas we need to re-engineer. We first focused on the 'things in the business'.

We recognised that system builders currently tend to ignore these 'things in the business'. We saw the undue importance they attached to the data–process distinction, imposing it on the things in the business—confusing it with the things–changes distinction.

We saw that this was part of a wider confusion between understanding the 'things in the business' and the operation of things in the computer system. As an example, we looked at how re-use—a notion dear to O-O people—worked at the operational and understanding levels.

We then refined our focus, identifying the types of things we need to re-engineer to arrive at the object paradigm. We identified four key types of things. We then determined how we were going to arrive at an object semantics. We learned that business entities had already been re-engineered into objects—outside computing. We do not need to re-engineer it from scratch, but can follow in the footsteps of the original re-engineers. We also briefly touched on the origin of the entity paradigm in the Ancient Greek substance paradigm. This is the starting point for our journey to business objects. We stop off at one intermediate paradigm on the way, the logical paradigm. This journey is described in *OP—Ontology: Paradigms*.

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 - M0
M0—The BORO Re-Engineering Methodology: Overview

M01—*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?*

BORO

INDEX

AS4

THE BORO APPROACH: STRATEGY - 4

FOCUSING ON THE THINGS IN THE BUSINESS

A–L

A

accuracy (and inaccuracy) --------------- AS4-12
Aristotle ----------------------------------- AS4-16

B

business objects
    key to ----------------------------------- AS4-2
    vs. system objects ------------------- AS4-8

C

change
    See also four key types of things, changes happening to things
    not persisting over time -------------- AS4-7
computer technology -------------------- AS4-16

D

data–process distinction -- AS4-6–AS4-9, AS4-15, AS4-18

E

entity paradigm
    problem with -------------------------- AS4-3

F

flexibility ----------------------------------- AS4-8
four key types of thing ------------------ AS4-11
    changes happening to things -------- AS4-15
    general types of thing --------------- AS4-13
    particular things ------------- AS4-12–AS4-13
    relationships between things AS4-14–AS4-15

G

general types of thing, *See four key types of thing, general types of thing*
generalisation
    re-use --------------------------------- AS4-9

I

information paradigm ------------- AS4-12–AS4-15
    major elements of -------------- AS4-1, AS4-17
    separate evolution of semantics -----AS4-17

L

lattice structure, *See structure – lattice and tree*

logical paradigm
    halfway house ------------------------ AS4-18

O

object paradigm
    origins of ----------------------------AS4-17
operational level (vs. understanding level)
    See also understanding vs. operational
operational re-use vs. generalisation --- AS4-9

P

paper and ink technology --------- AS4-17–AS4-18
    entity paradigm --------------------- AS4-16
    object paradigm --------------------- AS4-18
particular things, *See four key types of thing, particular things*

R

re-engineer
    things in the business -------------- AS4-18
relationships
    See also four key types of thing, relationships between things
re-use
    operational vs. understanding level -- AS4-9
    potential for ------------------------- AS4-9

S

semantic re-engineering route --------- AS4-18
structure – lattice and tree ------------ AS4-10

T

things in the business
    See also four key types of thing
    forcusing the re-engineering on -------AS4-1
    identifying similar ------------AS4-10–AS4-11
    ignoring -----------------------AS4-5–AS4-9
    problems identifying -----------AS4-2–AS4-4
    re-engineering object semantics for AS4-17–AS4-18
tree structure, *See structure – lattice and tree*

U

understanding
    vs. operational level ----------- AS4-9–AS4-11