Chapter 1: Data is not Information is not Knowledge.

When Alan is in Turin, he usually pops into Ludwig's fruit shop to buy three green apples. Sometimes Ludwig serves him, sometimes it's Zane the Robot, but occasionally he has to use the fruit machine.

1.1 The First Step

In the twentieth century, business was transacted person to person. A computer was a box in the corner where a priestly caste extracted oracles on stock levels, account balances and delivery times, but real people talked to the customer. In the twenty-first century, basement dwelling geeks heed the call of Cthulhu [1.0] to write a tangle of apps and web pages, replacing customer service with a labyrinth of procrastination and evasion. As companies substitute computers for humans, finding the services we want will require ways of navigating the labyrinth and avoiding the monsters they hide. We want to taste the sharp sweetness of a green apple, not crack our teeth on a lump of plastic from Silicon Valley.

In the labyrinth, at least three different webs of facts, associations and deductions are tangled together, interrelating people, computers and society. In teasing out these webs I will need to pose some questions. I could start with a cacophony: *What is Knowledge? How do robots understand humans? How do words mean what they mean? What is Logic? How does Artificial Intelligence work? How does real intelligence work?* But I'd rather start with a quieter, simpler question: I know how to do my job, you know how to do yours, so how do we get our computers to exchange information so we can work together? I spent the best part of twenty years working on the core of this question, particularly the inner most core - information - only to discover that information does not exist.

Information is not a *thing,* is not something that persist: it is a transient *relationship* between knowledge and data, between facts and the numbers used to write them down. This idea is not new, and can be traced back to the original definer of "information", Claude Shannon [1.1], although Shannon didn't delve into the question of what knowledge is.

Data is a hard, mathematical concept, where numbers are just numbers. Knowledge is a gloopy slime mould that constantly reshapes itself. But Information is a web of sticky, elastic string, dripping between numbers and knowledge - it oozes out from the human brain. Too often, IT has imagined it can solve the wetware problems of being human with the hardware of data processing, all the while ignoring knowledge as "too vague". In this book, I try to put the human back in charge of information, in the hope our information-based society will become more humane.

1.2 Where Are We Today?

In the 1950's, to make the technology work at all, the IT world made some gross simplifying assumptions - you didn't want too many complications when there was no RAM and your only data storage was sound waves in a tube of mercury[1.2]. By the time I first hacked into our college super computer, circa 1973, it had nearly 1 MByte magnetic core memory and was busy doing "data processing". Then gradually Data Processing became Information Processing became Information Technology, "information" conferring higher status than mere "data". Some thirty years ago, memory had become cheap enough for knowledge-based systems to became a thing, though

enthusiasm petered out after a dozen years, to be replaced by a new found enthusiasm for Artificial Intelligence (AI) - ChatGPT appeared as I was writing the second draft. But Data is not Information is not Knowledge - the changes in terminology have been mostly cosmetic, and the simplifying assumptions are still in place.

The computer world starts from mathematics, and the ideas that challenge those assumptions come from mathematics, but not mathematics as it is taught in schools. Lewis Carroll gave a big hint when, in *Alice Through The Looking Glass,* the Red Queen said "Why, sometimes I've believed as many as six impossible things before *breakfast*." I started out studying Pure Mathematics at Imperial College, where I learned that there are numbers bigger than infinity, that parallel lines can cross, and that two might be equal to zero. Some of the impossible things that you will be asked to believe include:

- there are many types of logic;
- logical thinking does not involve intelligence;
- data contains no information;
- quantised fuzzy logic provides a useful way of talking about meaning;
- knowledge and intelligence are different dimensions of the same thing.

You will not actually need to do any maths, only to nod sagely and say "yes, that's interesting" as you wander round my exhibition of data, information and knowledge.

The exhibition is set, fittingly enough, on Exhibition Road, in the old Huxley Building, which, until 1975 was home of the Maths Department of Imperial College - which is where I ~~studied~~ did my degree. The exhibition is housed off the main stairs, split over the three levels of the first two storeys, with data (the lecture rooms) on the ground floor, knowledge exhibited in the seminar rooms of the first floor, and information (the Maths library) on the mezzanine between them. Actually, this is a simplification for the sake of a good story - the actual building structure was much stranger, with one floor rumoured to be a Moebius strip.

And going up a floor is a key image in the story. The main stairs spiralled round an open well, so going up a level brought you back over where you started, a metaphor for going up a level from data to information and then another level to knowledge. So now you know what the books is about, why read on?

1.3 Four Good Reasons to Read On

Before touring the exhibition, I need to persuade you - particularly the professionals among you - that reading this book will be two or three hours well spent.

Reason 1 - Receive an Inoculation Against Tricksy Definitions

"Words mean what I want them to mean" Humpty Dumpty [1.5] was evidently a civil servant. Throughout my adult life, Income Tax was based on what I was paid between 6th April one year and the 5th April the next. I was not paid until the end of the month, so money earned between the 1st and 5th April was taxed in the next financial year. Clear and simple.

I had filled in my Income Tax form, and anyway, tax was paid before any money ended up in my bank account, so job done, and nothing owing. But no, the year that I got my state pension I was sent an extra bill because my pension was taxed when "accrued" - I was taxed on money owed to

me by the government for the first week of April but not yet paid - "Income" as distinct from income. A theme of this book is how definitions are grounded in corporate behaviour, not natural language.

Reason 2 - It might Save your Company a Lot of Money

"Twenty years of investment recouped on the first day of operation". That investment was in the STEP information standards [1.6]. It should have been paraded as a success story. However, because the savings were so large, it was kept confidential, just in case the customer demanded a discount even though the savings had already been factored into the price.

STEP is a standard developed by Aerospace and Automotive companies to help them work with each other across a complex web of partnerships and supply chains. The first twenty five or so years work was estimated at costing $250,000,000 [1.7], excluding testing and production implementations. All the Intellectual Property (IP) has been made available via the international standards organization ISO. And this has still resulted in big saving for the companies using it.

This book describes information and how it can be "standardised" to mean the same thing in different places, and thence shared in order that companies can work with each other. That it needs standardising is a strong warning that "meaning" is not a simple concept.

Reason 3 -Artificial Intelligence Can't Tell Fact from Fiction

If we had trained Chatbots on science-fiction, they would tell us that we can already travel to the stars - or that we will soon be wiped out by robots (Arnie the Terminator was sent back from 2029). It seems that chatbots use a form of statistical plagiarism, choosing phrases written by human authors on the basis of how often they appeared in the training data, rather than from any knowledge about the world. Children learn language not merely by hearing words, but by their interactions with the physical world - their language is grounded in experience. The word "apple" is grounded by biting into one. As humans, we want our everyday language grounded in the world we live in, so this book takes a closer look at how the language used in computers could be so grounded.

Reason 4 - It might help you Drive from Exeter to Aberdeen (Or New York to LA)

"Islands of Automation" is a phrase from the 1990s, signifying that, even though some steps of the processes had been computerised, manual processes were needed to move data from one computer to the next. 2020s "Smart Apps" seem intent on making the phrase relevant again - in early 2023, one neighbour complained of needing six different Apps to find charging points for their electric car. Making sure information is well defined, consistent and grounded is the key enabler to link together an eco-system of Apps. However, as information is a very fluid concept, you need this book to ground that concept and keep Apps anchored to reality.

1.4 Ludwig's Fruit Shop

Now we know roughly what the book is about, and why it is worth reading, we return to Ludwig's

Fruit Shop, a story to get us deeper in to the problem:

When Alan is in Turin, he usually pops into Ludwig's fruit shop to buy three green apples. Sometimes Ludwig serves him, sometimes Zane the Robot serves, but occasionally he has to use the fruit machine.

The problem posed by this story is to explain how the request "Three Green Apples, Please" is fulfilled by Ludwig, Zane or the fruit machine. Alan can confirm they each understand, because who or what ever is serving, he comes out of the shop with three green apples. Our first step is to find out who Alan, Ludwig and Zane are [1.8].

1.4.1 The Characters

Alan in Turin: A bad pun on Alan Turing, a mathematician. In the 19th century, non-Euclidean geometries were discovered, in which parallel lines could cross or slowly get closer. These raised questions about the foundations of Mathematics, and lead on to formalised mathematical theories, such as the axioms for the natural numbers laid out by Giuseppe Peano in 1889 (died 1932, Turin). Early in the 20th century, Alfred North Whitehead - a maths professor at Imperial - and Bertie Russell (Cambridge philosopher) wrote *Principia Mathematica,* whose aims included precisely expressing mathematical propositions in symbolic logic [1.9] - most notoriously proving that one plus one equals two.

These foundations were built - rebuilt - using formal systems, in which mathematical proofs are formulated in terms of mechanical manipulations of logic, removing the possibility of human error. However, by 1931, Kurt Gödel had proved that formalisation had fundamental limitations. His best known result can be summarised as saying that there are mathematical theorems than cannot be proved. Alan Turing proved another of the limitations, that if one has a machine for proving theorems, then there are theorems that cannot be proved in a finite time. This also provided a theoretical basis for computers, and later, besides winning the Second World war, Turing went went on to build one of the first practical electronic computers. This book uses formal systems to argue that the current generation of AI is not intelligent.

Ludwig Wittgenstein was a philosopher (and student of Bertie Russell) most famous for two books: *Tractatus Logico-Philosophus* and *Philosophical Investigations* [1.10]. *Investigations* opens by contrasting St Augustine's view of language, in which words name concepts, with a shopkeeper serving a customer with three green apples. I read this as an explicit repudiation of *Tractatus*, and setting an agenda for an understanding of language in which meaning is exhibited by the behaviour of the people who use it. In this book, I propose Wittgenstein was wrong, and that his two theories of language do not contradict each other. Rather they operate at different levels - are described by different networks of knowledge - one for specific situations and one for general concepts. There is a mapping between them, in which the knowledge of actual situations grounds the meaning of the general concepts. But for this I need to formalise what I mean by networks of knowledge, and what is meant by "different levels".

Zane Gort is a robot in Fritz Leiber's 1961 novel "The Silver Eggheads". The story's main protagonist is Gaspard, an "author" - someone who presses the button that starts the machine that writes a novel (ChatGPT anticipated 60 years). Dissatisfied, he gets a job looking after a collection of brains housed in metal ovoids - the Eggheads of the title. He and Zane take the Eggheads on adventures, who, once they again experience the world, start writing "human" novels. Zane is here

to ask the question "how does a robot know what *three green apples* means?". In particular, he is here to question the idea that "words point to concepts", since a computer has no concept of concept. The fruit machine is there to help focus on the way "words signify things", rather than on the question of how a sentence like "Three green apples, please?" is converted from natural language to a set of symbols for the machine to act on [1.11].

And I am here to ground the narrative in human experience. In particular, my experience of trying to get data from one computer onto another, and then sorting out the mess that results. A technical delinquent - I did technology, not people management - I was some times the developer of practical solutions - geometric algorithms or integrating data from multiple computers; and some times I was a sorcerer - identifying the box in a block diagram that should have been labelled "something magical happens here" and then working out how to do the magic. My presence is a deliberate repudiation of the impersonal "science way of writing". I am here putting the observer back inside the box with the observations, thereby repudiating the idea the knowledge exists out there, and can be understood by a disembodied brain. Or, equivalently, there can be no artificial intelligence without an artificial body - a view related to the strong AI hypothesis.

And, BTW, Wittgenstein was a friend to several Dominican friars in the Cambridge priory. Much later I was a novice in the Oxford priory, where friars continued to give "Wittgenstein-based philosophical therapy". A good chunk of my work with information has used the philosophy I learned there - philosophy used as a meta-theory to test which of the many theories of the world actually work.

1.4.2 Ludwig's Fruit Machine: Outside the Box

Ludwig's Fruit Machine is a secure box, designed to stop people running off with fruit without paying. It has buttons at the front which select the fruit required, a coin slot to pay for them, and a window where one sees a set of boxes storing the fruit, together with a grabber to take the fruit from the boxes to the delivery drawer.

The customer punches three buttons. The first shows the type of fruit wanted - indicated by a picture on on the face of the button. There are buttons labelled 1 to 9 for the number of individual fruits (the machine does not sell grapes or cherries). Finally, there are red, green, yellow and brown buttons to indicate the colour required - brown for customers who want very ripe bananas. The interface is intuitive, meaning that it ignores inputs it does not understand, and in doing so, trains the users to intuitively push one button in each category. It assumes that customers understand what fruit is and how it works.

The coin slot displays how much the customer should pay, and then, when the customer puts in coins, it counts the coins to see whether they have paid the right amount. Behind the coin slot is a series of mechanisms to authenticate the coins. From the synchronic viewpoint [synchronic - at a specific time, contrasted with diachronic, over time - one function of an author is to train the reader in the words they use] - from the synchronic viewpoint coins are a mystery - abstract tokens whose value depends solely on the society that sets their value. The diachronic view makes more sense - originally a lump of metal whose value was determined by its weight; later a lump of fixed weight authenticated by the face of the king/tyrant who issued them. And later still, when people no longer traded metal by weight, a token guaranteed by the state. The historical process starts with the mass of metal defining the coin's value and ends with a value assigned to a symbolic piece of paper - semantic drift made concrete (sic).

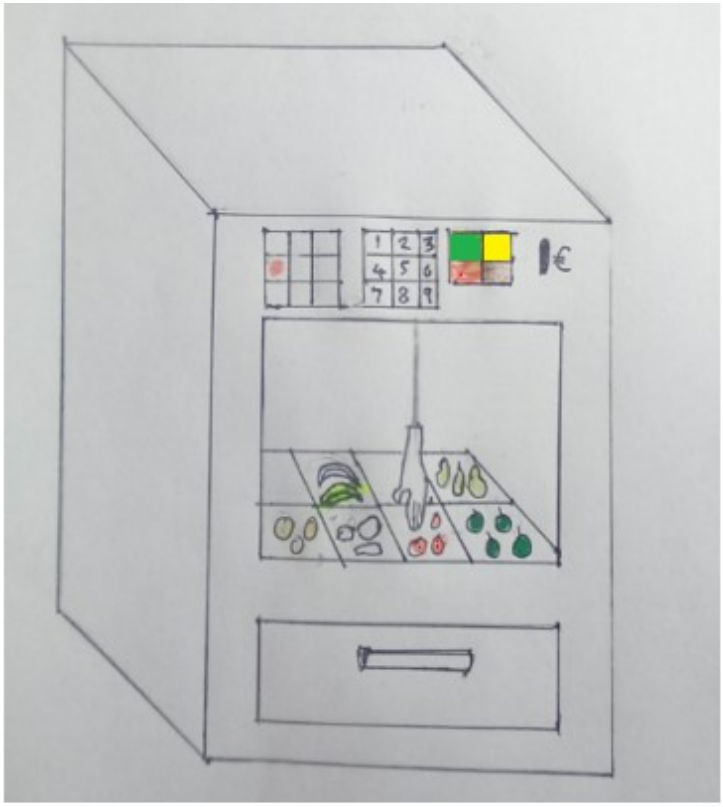

Figure 1.1 Ludwig's Fruit Machine

The computer program that runs the machine is triggered when the value of the coins input is the same as the value indicated by the display. Each fruit button is mapped to a fruit box, and the grabber lines itself up with that fruit box. Sometimes, when Ludwig is in a bad mood, he puts the fruit in the wrong boxes, and to the customers, the fruit machine appears to pick out fruit at random. The computer does not notice, it knows only the mapping from button to box.

To chose the right colour of fruit, the machine has a camera that takes a picture of a fruit, then one of each of four colour samples. It normalises the peak light level in the picture to that of a colour sample, then subtracts the light levels of the colour sample from the levels in the picture. Finally it picks the result with the lowest score. This means that the system works both in daylight - full sunshine or cloudy - and at night when the shop lights are on. Colour is more complicated than it looks. Although the machine gets confused when Ludwig sets his lights to disco mode.

The computer runs a GetFruit-countdown process inside a program loop:

```
        Set accumulator A164            /* A164 the storage location for the number of fruit
loop:   Call GetFruit(Bin number, Colour number)
        Decrement accumulator
        Test non-zero to jump to loop   /* JNZ in 8080/Z80 machine code [1.12]
        STOP
```

The program starts by reading the number of fruit the customer put in and loads it into an accumulator. It picks out a fruit, decrements the accumulator and jumps back to "loop". It stops when the accumulator value is zero. At no point does it use the concept of number or of counting,

although, obviously, the programmer did. (BTW, my 1977 "home computer" - a NASCOM 1 - used a Z80 which I programmed in machine code).

And here is the point of the fruit machine example - it is a simple, mechanical system. At no point is it necessary to attribute intelligence to it. It has no knowledge of the world, coins or fruit, although the person who developed it did - just as a hammer works without having the concept of a nail. Customers might mistakenly think the machine "knows" which fruit is which, but Ludwig knows better. The fruit machine does not pose the question as to how it assigns meaning to Three, Green or Apples, since it has no concept of meaning. Rather, the question is how Zane understands these concepts. Is he merely a machine programmed on the basis of other people's knowledge? Is there someone under the table controlling him? Or is he Artificially Intelligent? - and what do we mean by that?

1.5 The Narrative

So, now we have a story on three levels:

- Inside the box - a description in terms of a mechanism;
- Outside The Box - the buttons and their purposes;
- Outside Outside the Box - the people that know what is going on.

And now we need a Narrative to show how to link them together - a story showing how to make sense of the three levels and their relationship to each other.

You already have the first clue - the way Part 1 is arranged round the stairway of the Old Huxley Building. On the ground floor the basic maths is taught, and correspondingly in Chapter 2 data types are described as a series of tokens such as labelled discs and the mechanisms used to manipulate them. The chapter is inspired by a comment from Wittgenstein, "maths is a game played this way" [1.13]

Knowledge is found on the first floor, is described in chapter 4, and is hinted at though the people whose knowledge this book draws on - particularly Wittgenstein for philosophy and Turing for computing. I don't know where in the building North Whitehead had his office.

And Information - the buttons on the box - is on the mezzanine between data and knowledge, just as chapter 3 sits between chapters 2 and 4. In Ludwig's Fruit Machine, information is provided by the pictures on the buttons, which the machine maps to the box containing the fruit - provided Ludwig puts it in the right box.

And so the narrative appears as we spiral round the staircase, not as a linear journey from one place to another, but in the links that become evident between one level and another - a story in three dimensions. When you get to the end, you are in the same place as you started, but on a higher plane.

1.6 Beware Duck-Rabbits

The drawing below, based on one in Philosophical *Investigations*[1.15], shows a duck - or a rabbit - depending on how you look at it. It appears to swap from one animal to another, and we find it difficult to see it as both animals at the same time.

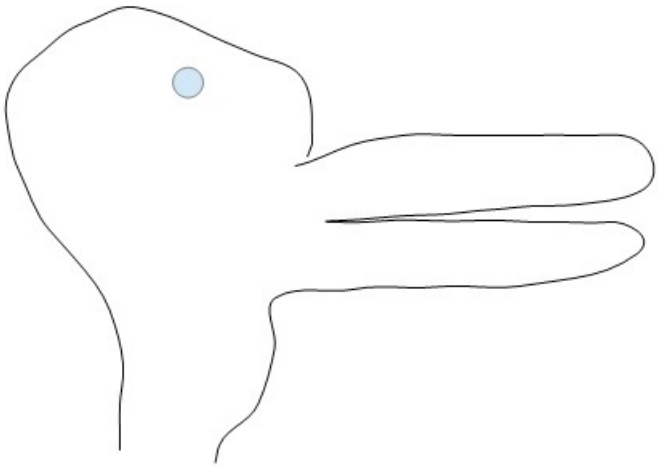

Figure 1.2 A Duck-Rabbit

Duck-rabbits are visual example of how we can flip our way of seeing things between different understandings. Maths is full of duck-rabbits. I once spent three days staring at a proof, and finally, by flipping the way I looked at it, saw it was obvious. One of the hardest things in maths is seeing why things are obvious.

I also had a duck-rabbit moment with Wittgenstein, while walking from the Washeteria to the John Baird pub in Muswell Hill (a wash lasted a pint). Having studied it three years earlier, I suddenly got what his Philosophical Investigations was about, having had a Tractatus view of language previously. It has taken me the next thirty-five years to knit those views back together. In this book, I hope to explain why two contradictory views of language are needed, and how linking them together solves the "grounding problem". This, I hope, will also explain why current AI systems don't work, how to pin down slippery politicians, and so on. And BTW, you don't have to read Wittgenstein, but you do need to keep a track of the ducks and the rabbits.

1.7 Housekeeping

Please ensure you know your safe routes to emergency exits - there is a realistic chance that someone will flee fire or flood while reading this book. Take toilet breaks at your own convenience. Avoid long periods of sitting in front of a screen. Refreshments are not provided. But you know all this - what don't you know about this particular book?

Structure: Chapters are numbered 1 to 12, sections 1.1, 1.2, etc. Chapters may be linked by staircases. Part 1 - the first staircase - has chapters on Data, Information and Knowledge. Knowledge is treated as coming in knitworks (sic) of knowledge and intelligence. And Chapter 5 is a field trip to Product Information Management, which also points the way to the next staircase. Part 2 happens somewhere between the end of chapter 4 and the start of chapter 6. Other field trips to take us to particular applications:

- Trip 2: An Institute in South Carolina
    - Information Exchange - how knowledge is shared (Ch 6);
    - Archiving, or knowledge sharing without being able to talk its creators (Ch 7);

- Trip C: A Research Laboratory in Bristol
  - Level change - what is it? (Ch 8)
  - Integration Architectures - level change in practice (Ch 9);
- Trip IV: The Lobby in the Houses of Parliament
  - Summarising Knowledge with classes - what ontologies really mean (Ch 10);
  - Expanding classes with knowledge - words are not simply pointers to concepts (Ch 11);

Obviously, the final chapter neatly sums everything up, explaining that integration is the intelligent use of knowledge.

Examples: In "The Idler" [1.16], Mark Twain notes that the progress of a story is frequently interrupted by the weather and the need to describe it. Consequently, he provided an appendix of different weathers, allowing the reader to sample some weather whenever they felt like it. I follow this precedent by putting some recurring examples in an appendix to this chapter:

- Computer Aided Design - illustrates the connections between information and inference;
- Bill Of Materials - sharing a single aircraft design across four different companies;
- Spotting Smugglers - mashing together information from different sources.

The examples are shown on a series of display boards - imagine them set up either side of the entrance arch way.

Notes: numbered sequentially by chapter and may cite sources. For practicality, I often cite yellow banana sources such a Wikipedia (thanks guys). Some original sources are professional grade (brown banana), such as ISO 10303 (aka STEP) or ISO 15926, and may run to thousands of pages and cost hundreds of pounds/dollars/euros. Brown banana? [1.17]

- Green banana - introductory material for novices;
- Yellow banana - descriptions for the intelligent reader;
- Brown banana - professional or academic grade - "my brain hurts" material.

This book starts green banana but ripens quickly.

Acknowledgements: Confidentiality agreements means that I can only cite work that is already in the public domain. However, the majority of sources were casual conversations in bars after meetings that happened many years ago. I must therefore thank my various collaborators and adversaries over the years, notably the late Nigel Shaw and his colleagues at Eurostep, Matthew West of Information Junction, and members of PDES Inc and the ProSTEP Association. Academics include Ian Horrocks (aka the editor of OWL), Chris McMahon (Engineering Knowledge) and Emil Lupu (logic-based policy language). Thanks to bosses who left me to play, including Alan Middleditch (computational geometry and formal methods), John Cox (Europe-wide integration) and also Phil Greenway for his bemused look while he wondered what I actually did. BAE colleagues needing special mention include Naiem Dathi (AI), Liz Carver (Human Factors), Julian Johnson (Systems Engineering) and Graham Clark (Process design). Also Monica Nogueira of SAE for commissioning my first two books, and Fergus Kerr O.P. for introducing me to Wittgenstein. The rest are not forgotten, but neither is my word limit.

Trigger Warning: Engineers and academics can be a dour lot when writing, and although they use irony and sarcasm, they may not explicitly flag it as such. This book also includes bad puns. Puns are when you duck-rabbit between two different ways of understanding something.

1.8 Create Your Own Summary

A typical chapter summary is a linear narrative, compressing the text of the chapter into a few short sentences. However, the facts in this chapter are brought together in a loosely woven knitwork, with connections at multiple levels - not linear at all. As will be discussed in chapter 4, a knitwork consists of facts used intelligently, which means it is a job for you, the intelligent reader, to think this through. You may even discover connections that you didn't notice. To make this easier, I have created a game.

Figure 1.3 Summary as a Game

Photocopy the image above onto to card and cut into tokens. Arrange the locations on a ground floor plan of an imaginary Huxley Building and link them with strings labelled with geographic predicates such as *is-next-to.* The ground floor is the WHERE floor. Now add the persons to the floor above - the WHO floor - ideally placing them above the appropriate location. Add person-person relations such as *met* and person-location relations such as *worked-In*. Next arrange the concepts into a Venn diagram on the WHAT floor, adding suitable predicates connecting to other floors. Finally, label the predicates randomly with the numbers 1 to 6. With a dice, follow the summary knowledge network, as if playing Snakes and Ladders. For extra fun, you can add your own WHY layer, or code the knitwork you have created into a Mario Cart race track. As a bonus level, work out why the Author replaced the summary with this game (Hint - rearrange the words "Bloody", "Difficult", and "Too").

Appendix A - Recurring Examples

A.1 Computer Aided Design - Knitting Facts and Algorithms

Computer Aided Design (CAD) covers a class of tool to help engineers develop a design. In this book, the scope is narrowed to cover just those tools that help with mechanical design - tools which replace a drawing board with software that generates 3-D shapes.

In a typical CAD package, drawing a curve starts with the designer inputting a series of control points, and then telling the computer to put a spline through them. A spline used to be a long, flexible strip of wood: weights were sat on the drawing board at the control points and the spline was bent round them to create a smooth curve for the pencil to follow. CAD systems started out by reproducing this procedure electronically, and the terminology of splines and weights persists to this day.

The earliest CAD systems were a form of electronic pencil, reproducing the traditional drawing. However, it was soon realised that by joining up the curves, one could create 3D shapes. With a CAD system, it may take a skilled draftsman only three curves to design the whole skin of an aircraft. Consequently, CAD systems are no longer drawing tools but modelling tools. The don't just replace the drawing board but also replace the woodworking shop. That shop made model 3D parts for mock-ups, which were then used to check that everything fits together. Somewhere in Airbus Filton was - and hopefully still is - a magnificent full-scale wooden mock-up of an A340 wing, over 30m long. Now, such mock-ups are created and tested electronically in CAD systems.

In this book, a CAD model illustrates a simple network of geometric details and algorithms - what will be called a "knitwork" of knowledge and intelligence. The model is stored as a list of geometric data - often just points in space - together with a pointer to the operation that follows: join points into a line, join the lines into a mesh, and fill in the mesh to create a surface. Figure 1A1 below illustrates this. (BTW I used to run the "join the dots" CAD development team).

Figure 1A2 illustrates the connection between starting data and construction software by showing what happens if the operations change following a software update - in this case, although the starting data remained the same, the update wasn't quite right and a different shape resulted.

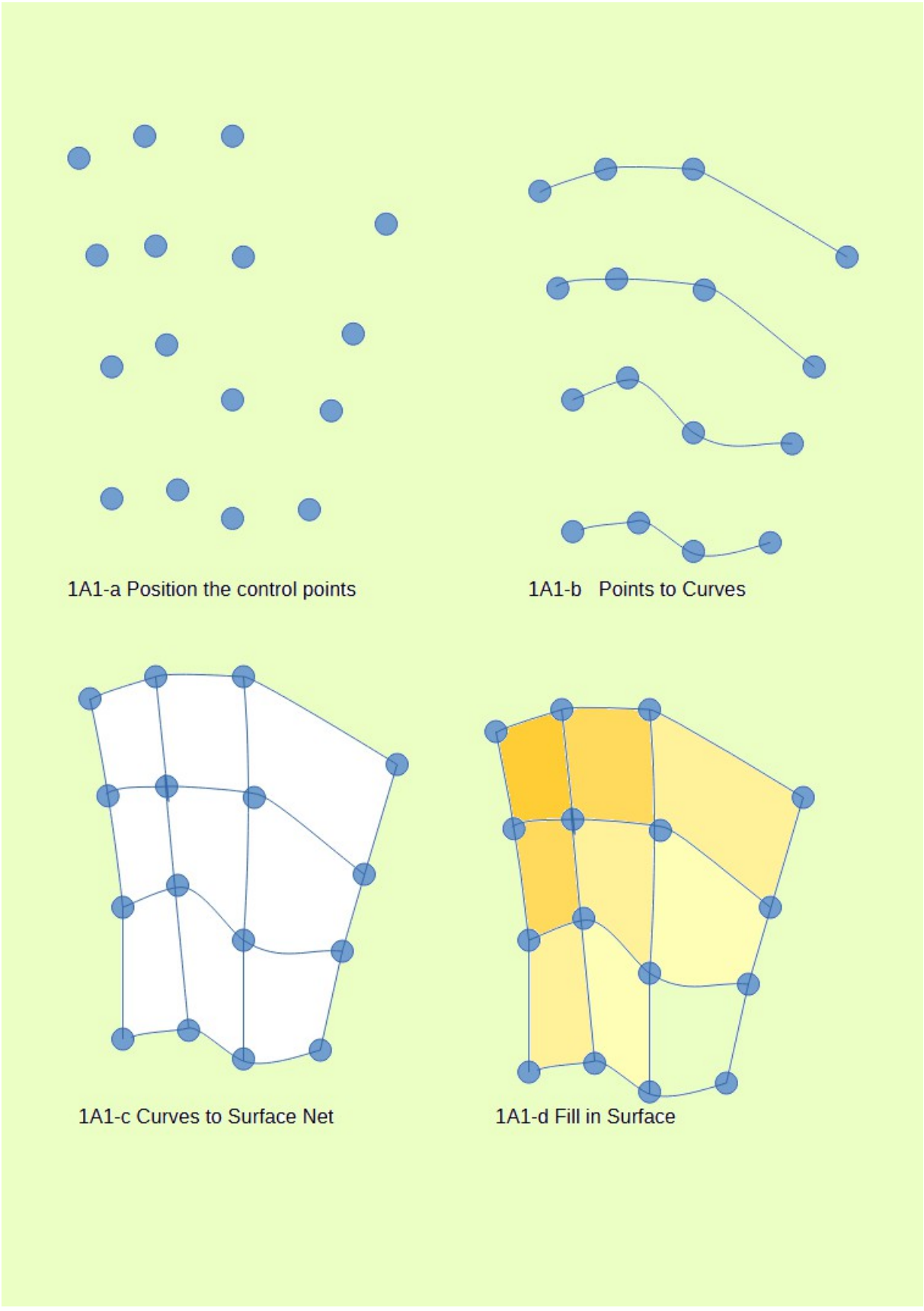

Figure 1A1: Geometric Data to Geometric Knowledge

But the knitwork doesn't stop at the shape. The designer choses the shape to fulfil a particular function. That function links to knowledge about material strength, lightness, manufacturing facilities, and a dozen other properties. How all that knowledge comes together is a whole book in itself [1.17].

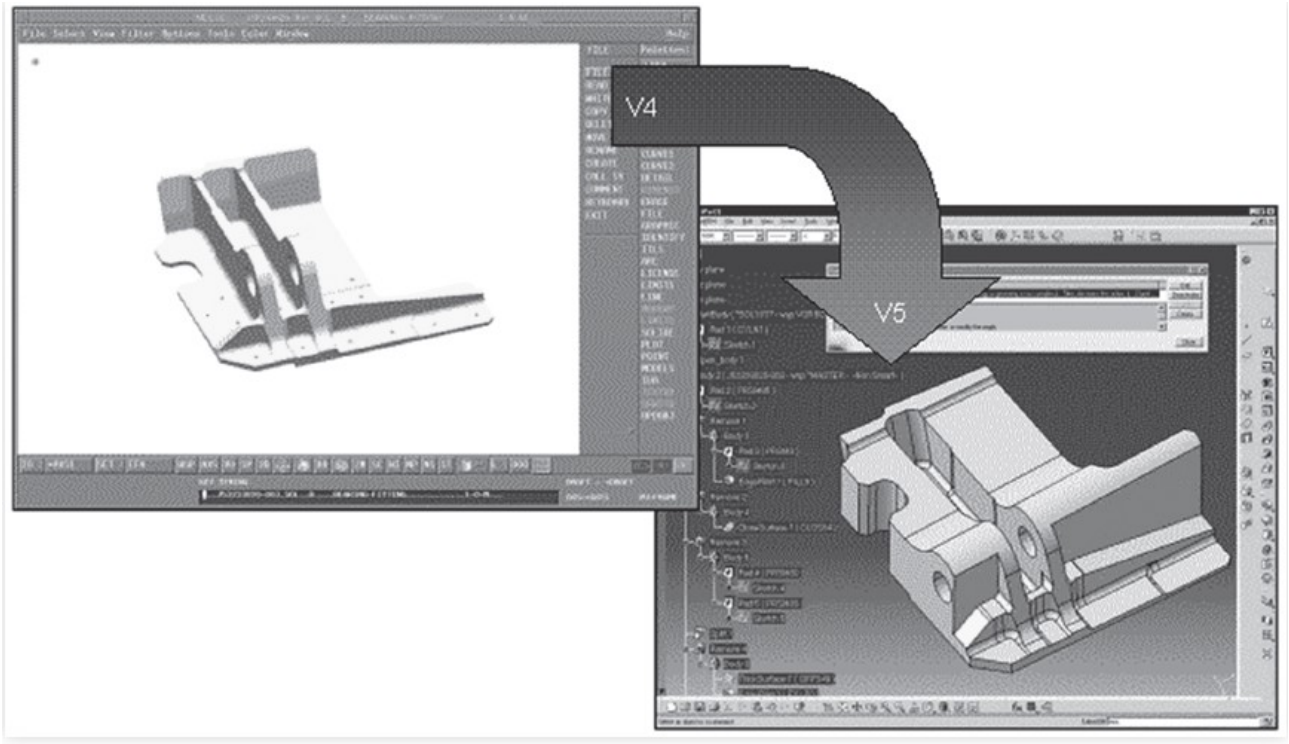

Figure 1A2 - Effect of Software Changes (used in various presentations)

A.2 CAD as a way of Writing Down Knowledge

The example above, "CAD - Knitting Facts and Algorithms", shows how simple geometric facts are knitted together into a complex shape. This example works in the other direction, showing different ways in which knowledge of a shape can be decomposed into simple facts.

It's fairly easy to sketch a simple block (Figure 1A3a) - 12 lines in all. That we see those lines as a block is down to our knowledge of drawing conventions. In CAD, this is called a wire frame model. If you literally cut a cross section through such a model, all you get is a pattern of four dots where the wires are cut through - all knowledge about their relationship to the block is lost.

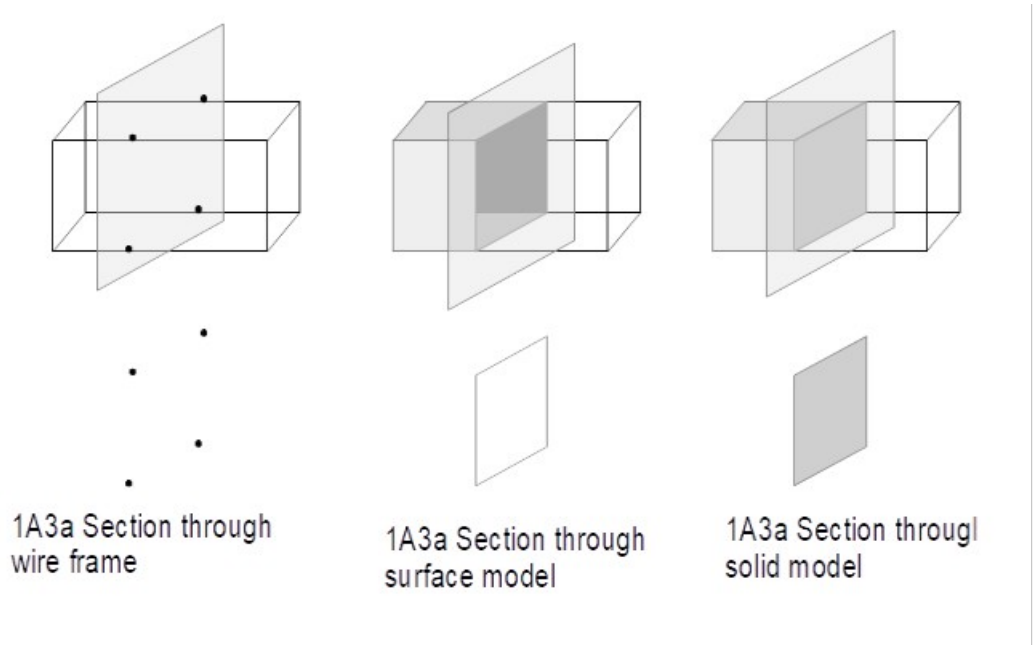

Figure 1A3: Geometric Knowledge to Geometric Data

One can also model a block by folding cardboard into box (1A3b), and this consists of six surfaces joined at the edges. In CAD this is a Surface Model. The cross section for this is a set of four lines, which, for the cut shown, form a rectangle. It is obvious from the outline what is inside the box and what outside, but for more complex shapes, deciding inside or outside can be complicated.

The third approach is sculptural. Start with a large lump of butter, and take six slices off it to leave a block in the middle (1A3c). In this case, the inside is clear - inside is where there is butter. In CAD this is known as Constructive Solid Geometry (CSG). Although this solves the inside/outside problem very effectively, actually drawing the edges takes more calculation.

Writing things down - reducing knowledge to data - itself needs a complex of additional knowledge about how to reverse that process. That knowledge is used to ensure that we can convert data back into knowledge. Information - the mapping from data to knowledge - provides a useful stepping stone, but only a stepping stone. Information does not exist on its own - if you throw the information for a block through a window, it will not break the window. Information is only as solid as our imagination - "*stepping stone*" is only an analogy.

A.3 Bill of Materials

How do you share information across multiple companies? A Bill of Materials (BOM) [1.18] provides a simple illustration. A BOM is a list of parts that make up a product, usually organised into assemblies. (See Figure 1A4)

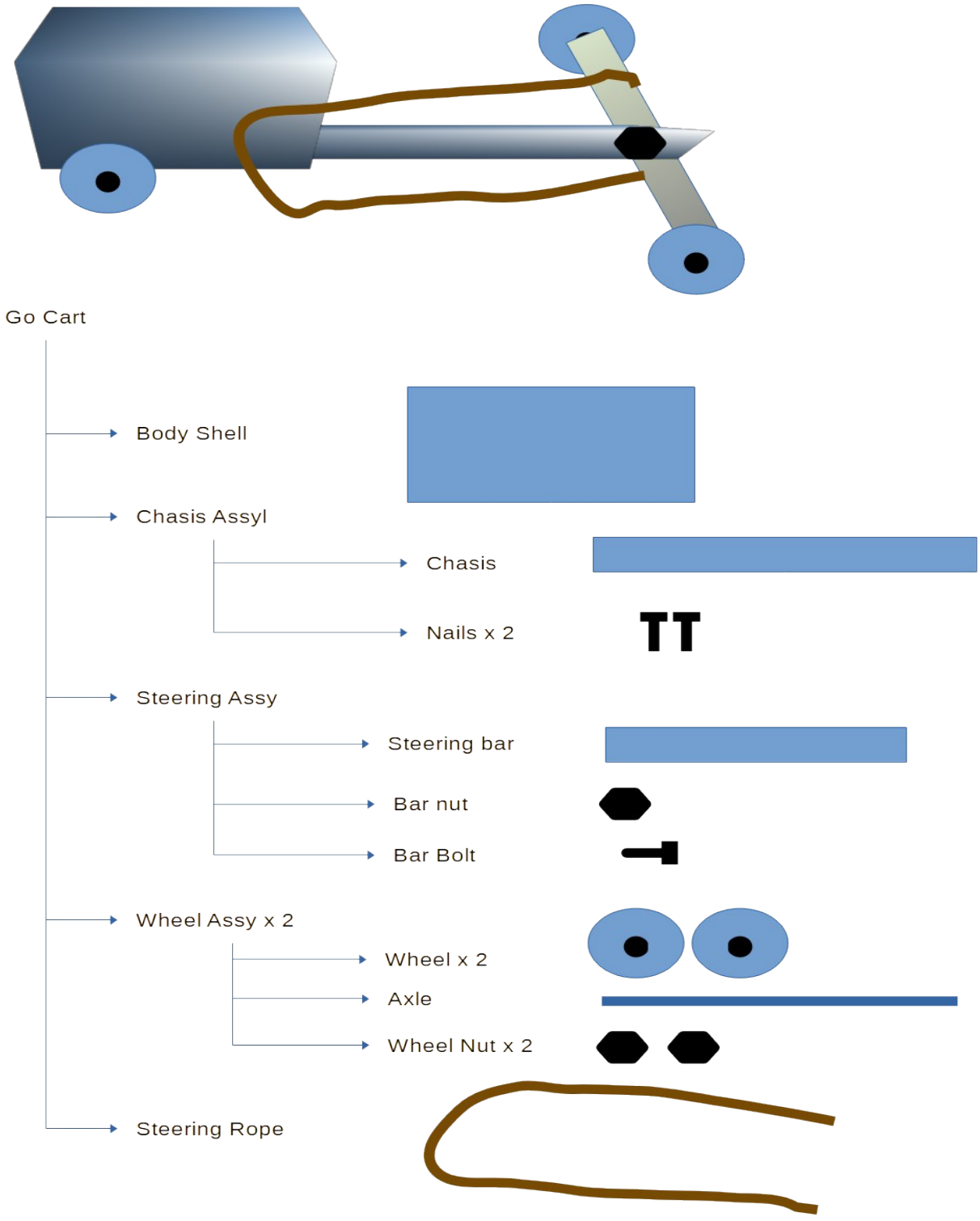

Figure 1A4: Bill of Materials

When you design an aircraft, you might need the designs for a hundred thousand or even a million parts (although a lot of these will be rivets). There are lots of reasons why you want an accurate list of parts, ranging from adding up their weights though ordering from suppliers and on to

airworthiness certification. Moreover, you need accurate lists of parts from each equipment supplier, for example, for the landing gear, the cockpit instrumentation, and - the big item these days - the in-flight entertainment system.

In the project I was involved in [1.19], the BOM was maintained by four partners, who would add both the parts they created and part lists from their sub-contractors and suppliers. The BOM was shared, so that each could run their own production line.

The existing system for sharing was based on a common database schema, but, being the 1990s and Wide Area Networks being a recent development, the process for sharing was

1) print the BOM out in factory A;
2) send document to factory B, C & D;
3) factories B, C, & D would then type in the BOM into their computers;

- a process that took about six weeks. Such a manual process is expensive and error prone, although a report of US experience said that many such "errors" lie in the different terminologies in the factories [1.20]. The delay also meant that people in one design office might be working on data made obsolete by work done in another, and have to redo their work. My mission (which I chose to accept) was to electronically merge and distribute the BOM for factory A.

Taken as an example, a BOM shows how an enterprise distributed across many physical companies can integrate its data. The prime contractor creates a high level assembly which lists a supplier's assembly - a single line in the BOM - and the supplier expands the BOM underneath that line with the list of components in their assembly. The information - the part numbers, the quantity used, and so on - are represented by a list of data items recording individual part designs which are then linked to knowledge about the total product - how the whole aircraft comes together and flies. Different information sources are joined up by expanding each sub-assembly into the list of parts it contains.

A.4 Automatic Mash-Ups

Some ports of entry, such as airports, have systems to spot ne'er-do-wells - smugglers, criminals, spies even - who might be coming into the country. Imagine Bob provides quality control for "Points of Entry Surveillance" - he travels round the country, setting up his own system, and comparing his results to those of the people running the airport.

His system accesses the airport's arrivals list to find all the incoming flights. Airline identifier and flight numbers are used to access the passenger list from each airline and collect the names, addresses and passport details of each passenger. This list is sent to various agencies, such as the police, to see if it contains any person of interest - persons of interest are added to Bob's Watch List. Finally this list is compared to the airport's list to see if they have missed any, or if they have anyone not on Bob's list (Figure 1A5)

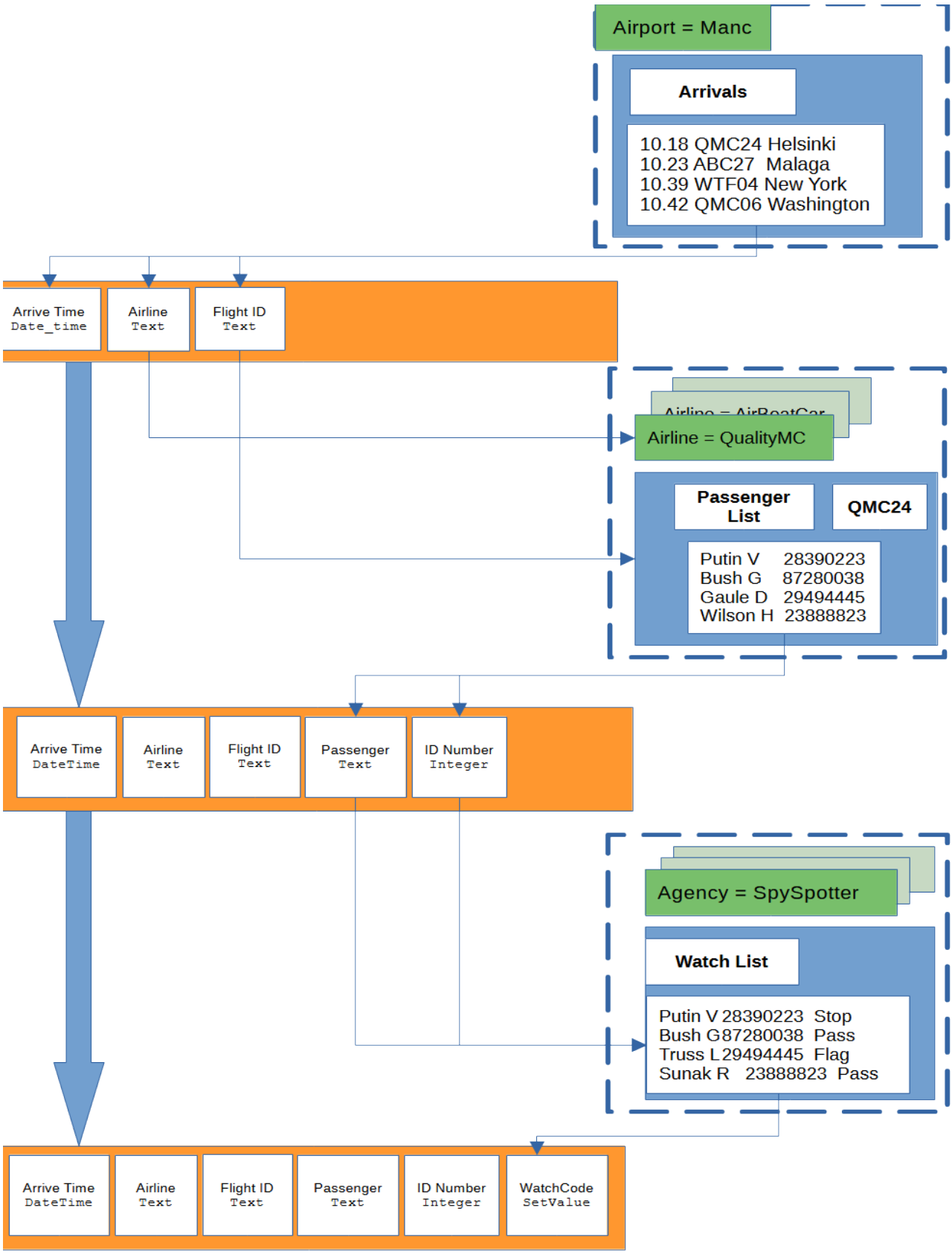

Figure 1A5 Watch List Mash-Up

Bob's query "Is anyone suspicious coming through the airport today?" breaks down into a series of

queries to multiple databases, and his system mashes the results together (hence mash-up). The technical problem is to build the series of queries automatically, and Bob's system uses a version of the "HaLevi Algorithm" [1.21]. This accesses the database schema for each site, and matches columns in one table in one database to the columns in a table in another, creating a chain of connections which together answer the query. Some databases only respond to the exact query, so if Bob is looking for a generic type of ne'er-do-well - someone with a conviction for smuggling say - the algorithm must create a separate query for each subtype - say one each for "commercial smuggling", "drug smuggling" or "arms smuggling", referencing them to the specific law they were convicted under.

Bob's system has to solve a number of other problems. There is a data level problem of making sure the format of the data in each database matches up, so one can compare entries. There is an information level problem, making sure that "Flight Number" in one databases means the same thing as "Flt No" found in another. There is a massive security problem, accessing all the databases, especially those of intelligence agencies - that is why the system sends a list of people asking the agency to tag persons-of-interest, rather than trying to download their data and do it locally. Fortunately, they all know Bob, which is why they will answer his queries.

There is also a knowledge level question, which has gotten much worse since the advent "AI" based systems. Bob regularly looks to see if the airport's watch list includes the international jewel thief George Clooney - aka "Danny Ocean" - or notorious spy Daniel Craig, alias "James Bond". These are a sure sign that the airport watch systems are accessing low quality data sources.

NOTES AND REFERENCES

Wikipedia entries are followed by their accession date.

1.0 I assume that you, the reader, can look up generic references as well I can. For example, Cthulhu gave me about 33 million citations, including Wikipedia, references to H.P. Lovecraft and lots of pictures.

1.1 Wikipedia "Delay-line memory", https://en.wikipedia.org/wiki/Delay-line_memory, 20/9/23

1.2 Exhibition Road, South Kensington, London, opposite the Science Museum

1.3 Wikipedia "Alfred North Whitehead" https://en.wikipedia.org/wiki/Alfred_North_Whitehead 20/09/23

1.4 "Humpty Dumpty" - a character in Lewis Carroll's "Alice through the Looking Glass", with whom Alice discusses semantics and pragmatics. Carroll was the Rev Charles Lutwidge Dodgson's pen name. He lectured in Mathematics at Oxford.

1.5 STEP - STandard for Exchanging Product Model Data - for a general introduction see Wikipedia "ISO 10303" https://en.wikipedia.org/wiki/ISO_10303 (Accessed 27/2/2023).

1.6 Comment at a PDES Inc Technical Advisory Committee meeting circa 2010. PDES Inc is an association of STEP users and vendors. See also https://pdesinc.org/

1.7 See Barker S “Aircraft as a System of Systems: A Business Process Perspective” SAE

International, 2019 ISBN-print 978-0-7680-9402-2

1.8 For more details of these characters, I assume the reader is capable of looking them up for themselves. Fritz Lieber's novel "the Silver Eggheads" 1961 is more enjoyable than an academic introduction to the problems of AI. Original edition details have been swamped by republication blurb.

1.9 Assuming the reader has neither the time or energy to read the original, see Wikipedia "Pincipia Mathematica" https://en.wikipedia.org/wiki/Principia_Mathematica 20/9/23 For further background, try Kline, Maurice "Mathematics", OUP, New York 1980

1.10 Wittgenstein, Ludwig "Philosophical Investigations" Second Edition: Blackwell Publishers, 1958. ISBN 0 631 14670 9

1.11 The need to distinguish the semantics of a word from the problem of understanding natural English arose in an argument with Pat Hayes on Ontolog Forum c. 2007.

1.12 The Intel 8080 and the Zylog Z80 were early (c. 1975) microprocessors. JNZ - Jump non-zero - is a mnemonic for one of their machine code instructions. For "Call GetFruit..." I assume there is a symbolic assembler.

1.13 Wittgenstein, Ludwig, "Remarks on the Foundations of Mathematics", e.g. MIT Press 1983 ISBN 978-0-262-73067-9

1.14 Wikipedia, "Technology readiness level", https://en.wikipedia.org/wiki/Technology_readiness_level 20/9/23

1.15 See *Philosophical Investigations* [1.10] page 14

1.16 Jerome K Jerome & Robert Barr (eds) "The Idler", Vol 1 Feb to July 1892, Chatto & Windus, London 1892

1.17 Barker Sean, "Banana Coding", https://theseanbarker.com/banana-coding/ Accessed 20/9/23

1.18 Wikipedia, "Bill of materials" https://en.wikipedia.org/wiki/Bill_of_materials 20/9/23

1.19 Eurofighter Product Data Exchange, a four language collaboration between British Aerospace, DASA, CASA and Alenia. Originally based on an AECMA Edifact message, then subsequently on the STEP PDM schema, supported by PDES Inc and ProSTEP. See, for example, Graham Spinardi, Robin Williams, Ian Graham “Technical data interchange in the Eurofighter project” Science and Public Policy, Volume 22, Issue 1, February 1995, Pages 29–38, https://doi.org/10.1093/spp/22.1.29, although this paper was published before the exchange was established.

1.20 Apologies, I no longer have the reference. If I remember correctly, the paper discussed an exchange c. 1997 between the Boeing plant in Seattle and the old McDonnell Douglas plant in Long Beach.

1.21 The reference is currently immured in our spare room behind my daughter's "can you look after these while I do a year in Japan" boxes - hopefully to be rediscovered in August.