Preface

The aim of this book is to introduce a different way of looking at IT and IS (information technology, information systems), and to suggest some tools to help us do this. The tools derive from philosophy but are orientated to everyday experience of IT and IS.

Those tools will be applied to five areas of research and practice in IT/IS. Allow me to explain, by way of five personal, autobiographical vignettes, what has motivated my involvement in IT/IS and why philosophy is important. This will give a feeling for how this book approaches the seemingly rather heavy topic of 'Philosophical Frameworks for Understanding Information Systems'.

Vignette 1. The Diversity of the World

Working with computer applications in computer-aided design, the health sector, the chemical industry and the surveying profession from 1970 to 1987 it seemed to me that there were four irreducibly different 'aspects of knowledge' that needed to be encapsulated in computer programs or knowledge bases:

1. Items and relationships, such as patients and problems that each patient might have

2. Quantitative and qualitative values, such as the strength of dose of a drug, the date when a problem started, the name of the problem
3. Spatiality, such as proximity to infection
4. Change: events and processes, such as accidents and healing

Each of these aspects of knowledge seemed to deserve a different fundamental approach to representing them, otherwise programming errors would increase. But I doubted whether I had the full set of such aspects, so I sought a fuller set after returning to academic life as a lecturer in 1987.

I did not find one. Most of my colleagues in computer science and artificial intelligence assumed that standard computer languages like C or PROLOG were sufficient: so what's my problem? At the other extreme were those who, like a professor who knew some philosophy and whom I approached with the question of what 'the' correct set of aspects is, replied "There aren't any, they are socially constructed."

Perhaps because of mild Asperger's Syndrome, I could not accept this. Even though I agreed that that human beings do socially construct their categories, I also believed that there is a reality that transcends us and cannot be socially constructed, and that aspects of knowledge were part of that. Privately, I was still curious, and continued to ponder these aspects of knowledge—what aspects there might be, how to distinguish them, how to implement knowledge representation formalisms in computer terms based on each, and the question that underlies these: what aspects as such are. Eventually I expressed these ideas in Basden (1993).

I had always taken an everyday, lifeworld attitude and was a little wary of theorizing, whether of a rationalist, positivist, social constructivist or any other kind. Though I did not realise it at the time, my difficulties related to philosophy: the radical difference between the everyday and the theoretical attitudes of thought and the need to integrate ontology with epistemology. Not until I found a philosophy that treated the everyday attitude with due respect and did not force me into either social constructivism or positivism or naïve realism, did I find an answer that satisfied me.

Vignette 2. Usefulness

Working in the UK chemical industry in the early 1980s, I had a manager who had been given a top-of-the-range PC with the latest software. He found it easy to use, but I remember him standing in his office and asking, "But what the heck do I use it for?" Of course nobody would ask that today. But it cemented into my mind the difference between usability (and other technical qualities) on one hand and useful-

ness on the other—a difference I had felt for the decade before but not expressed clearly. Basden (1983) began to express these issues.

Over the next dozen years, I experienced a succession of new technologies from within—knowledge based systems (KBS), multimedia and virtual reality—all of which failed to develop their expected potential. This was not because of any lack of technical excellence but because of too little attention given to usefulness. Usefulness seemed to me a crucial issue in the human factors and knowledge-based systems (KBS) communities. Yet it did not enter their academic debates and most researchers responded with "So what!" Our paper on it (Castell, Basden, Erdos, Barrows, & Brandon, 1992), though it received an award, elicited no general response.

Suddenly, in reaction to such books as Thomas Landauer's (1996) *The Trouble With Computers* usefulness came onto the agenda. But the debate leapt straight from "So what!" to "I have the solution to ensure usefulness." The intervening stage of trying to understand what usefulness is was bypassed.

What is usefulness? Why is it that some information systems seem to bring both beneficial and detrimental impact, often to different stakeholders? Which stakeholders are important? How do we address unanticipated impacts (whether beneficial or detrimental)? What about longer-term impacts? How can we design for usefulness, predict usefulness, evaluate usefulness? In short, how do we differentiate success from failure?

To address such issues demands a way of understanding that recognises dynamic diversity and also has a strong basis for differentiating right from wrong (benefit from detriment). Objectivist approaches like cost-benefit analysis are too narrow while subjectivist approaches cannot cope with unanticipated impacts, especially of the long-term variety, nor can they differentiate benefit from detriment. Just like knowledge representation, usefulness and the success or failure of IS demands an everyday rather than theoretical approach.

During that period I became an avid player of computer games, both personal games like *Moria*, *ZAngband* and *The Settlers*, and MUDs (multi-user dungeons) on the Internet. Real playability is like usefulness, depending more on content, story and humour than on technical issues like user interface or graphics. So I wanted to find an understanding of usefulness that was not restricted to work life and organisational use but can address the types of issues encountered in all kinds of computer use.

I discovered that the philosophy I referred to above provides a basis for understanding dynamic diversity, has a strong basis for differentiating right from wrong (normativity) and allowed me to go beyond work applications. It transcends objectivism and subjectivism. It was in regard to use of IS that I first found it immensely practical in application, so much so that I have perhaps been captivated by it ever since.

Vignette 3. Knowledge Elicitation and IS Development

I have always enjoyed programming computers. It is creative. What I produce can be elegant and beautiful, as well as doing some good. Developing KBSs, which involves knowledge elicitation, expanded programming to include the challenge of getting to grips with knowledge in all its diversity and coherence.

Knowledge elicitation involves interviewing experts in a field to obtain some of their knowledge with which to construct the KBS. But I found that, rather then taking the conventional approach at the time of eliciting heuristics (rules of thumb that are actually used in practical expertise), better results are obtained by separating the general 'understanding' and 'laws' underlying the heuristics from the contextual and personal factors.

Moreover, the expert is not just a source of knowledge but a human being, whose expertise is part of who s/he is as a person, so knowledge elicitation is not merely knowledge transfer but involves an intimate relationship of mutual trust and respect, in which the experts feel free to open up.

These two approaches to knowledge elicitation was, in retrospect, one reason for what was an embarrassingly high success rate in developing KBSs. But could it be transferred to others? I wanted to understand why seeking 'understanding' and developing a relationship of trust led to success. Again, philosophy was called for. The strategy of seeking 'understanding' presupposes there is in fact something to understand that we have a hope of finding. Subjectivism would say there is nothing, ultimately, 'out there' to understand. Objectivism might allow 'out there' but immediately tries to reduce understanding of it to logic, especially that of the natural sciences. We (Attarwala & Basden, 1985) rejected both, but at the time we had no philosophical basis for doing so. Now, it appears, the philosophy that helped me understand aspects of knowledge and usefulness can also help here.

From the mid 1980s I had to raise my sight above knowledge elicitation to the wider project that is IS development (ISD). While software engineering methods were becoming increasingly structured in the 1970s and 1980s—analyse, specify, design, implement, test: all aiming for automated, formal proofs of a program's correctness—my own practical development went in the opposite direction because the domains for which I developed KBSs, databases, etc.—medical patients, stress corrosion cracking in stainless steel, agricultural planning, business strategy, budget-setting in the construction industry—were ill-structured. No prior specification could be drawn up since the very activity of implementation reveals new knowledge and stimulates users to change their minds about what they want. So we developed our own 'client centred methodology' (Basden, Watson, & Brandon, 1995).

Of course, I was not alone in finding these characteristics of the process of development. But where our approach differed lay firstly in its key notion of responsibility,

not just to the 'customer' but to all stakeholders, to the community and society, to the future, to the domain of application, to the nature of reality itself, and even (some would add) to God. Secondly, it lay in its sensitivity to everyday life of the application, the stakeholders and the entire ISD team. Though until the mid 1990s I did not attempt a philosophical understanding, it now transpires that the philosophy I found so helpful in the first two areas above is helpful in this area too.

Vignette 4. Nature of Computers

When I read Allen Newell's paper *The Knowledge Level* soon after it was published in 1982, I immediately thought, "Yes!" It put into concepts and words what I had intuitively held to be so: that there is a fundamental difference between symbols of our computer programs, databases or KBSs and the knowledge they 'hold' as their content. From a study of citations of this paper over the next 20 years, it seems that many others found the same intuitive agreement. *The Knowledge Level* freed a whole generation of us who worked in the AI and HCI (artificial intelligence, human computer interaction) fields to talk about computer systems at 'the knowledge level', in terms of their content, independently of how it is represented in symbols and their manipulations.

But Newell had in fact taken us further. He argued that the symbol and knowledge levels were just two in a sequence of levels at which computer systems may be described—the physics, the electronics, the digital signals and bits, the symbols and the knowledge. (Later, several of us added a sixth level, variously called tacit or social.) Thus Newell proposed in effect a multi-level understanding of the nature of computers. I have used it as an educational tool to separate out distinct types of issues in HCI, KBS, multimedia, virtual reality; my students find it very useful.

Newell tried to ground the knowledge-symbol difference in philosophy but not very successfully (see Chapter V), and he merely assumed the other levels. However he made a curious philosophical claim that few have noticed:

Computer system levels really exist, as much as anything exists. They are not just a point of view. Thus, to claim that the knowledge level exists is to make a scientific claim, which can range from dead wrong to slightly askew, in the manner of all scientific claims. (Newell, 1982, p. 99)

This is a strong ontological claim. But on what basis could he be right? And if someone else suggests a new level (as for example I did with a 'tacit level') on what basis do we judge the candidate new level? Yet again, the very philosophy I found helpful in the three other areas is one that is able to provide a philosophical

underpinning for Newell's levels. More: it can throw new light on the AI question of whether a computer can 'think'.

Vignette 5. The Information Society

What damage will IT wreak on humanity and the wider environment? Or what good? That is, how do we understand what has been called the information society? It is too early to tell. Not many predicted climate change from cars and planes until recently. If we cannot predict 'scientifically' what the real problems of IT will be, maybe we can at least take note of the ethical dimension? For example, is it not true that selfishness and self-interest mainly bring problems (refer to business writers since the year 2000)? So, maybe humanity's development of IT could be guided by eschewing self-interest, rather than by seeking 'scientific' attempts to plan it or by assuming that we can continue to please ourselves?

On a personal note, I have worked in the green movement for many years. For me green activism followed, and was a result of, my conversion to Christ and a filling with the Holy Spirit around 1970. Over the years I have become convinced that what has needlessly driven us towards destruction of the earth is not just big business nor a conspiracy, but our idolatrous world-view and self-centred attitudes, which pervade every aspect of the way we live and do business. It even pervades the way we carry out research. The root of our problem is 'religious', even while at the same time there are economic and political factors at work. Those in the feminist and anti-globalisation movements know this very well (though they usually use terms other than 'religious'). I use the word 'religious' rather widely, almost as a synonym for 'ideological'; see Chapter II.

Likewise, when we take a societal, global perspective on IT, a 'religious' aspect is inescapable, both for those involved in the practice of the area and for the content of our theories. To understand the issues in this area, I wanted an approach that acknowledges religious commitments and presuppositions, not only in humanity as studied but also in we who study it. Lo! I found that the philosophy that helped me so much in the four other areas acknowledges and opens up religious issues.

'The Whole Story'

Each vignette indicates a distinct area of research and practice in IT/IS in which I have been involved. In each, a particular issue emerged:

- **Vignette 1:** The diversity of meaningful reality that we want to represent or model in computers
- **Vignette 2: E**veryday diverse normativity and repercussions of computer use
- **Vignette 3:** Responsibility to both those we work with and to a diverse reality that transcends us and yet which we can understand
- **Vignette 4: T**he multi-level nature of computers
- **Vignette 5:** Religious root of IT, of those who practise and research it, and of society

Each indicated the need for philosophy—and that one philosophy in particular has been useful to the author.

Because I have been intimately involved in all these areas, I have tended to see them as closely related to each other, to form what we might call 'the whole story that is information technology'. Almost from the start, when programming for my PhD in the 1970s, I would try to reach beyond my immediate area and feel for the others. In the everyday 'whole story', the areas interweave and while we may conceptually distinguish them, we need a way of understanding each area that acknowledges the others. Yet in each, a different community of practice has developed, a different research agenda and way of thinking and different research communities, which seldom speak to or understand each other. What each finds meaningful the others find meaningless. As I argue in Chapter I, this is why philosophy is needed: philosophy not only helps us understand the issues in each area, but it is the discipline that allows us to acknowledge a variety of spheres of meaning in relation to each other.

Unfortunately, I did not find any philosophy that was on offer either useful or attractive. Ancient Greek thinking opposed form and matter. Mediaeval thinking opposed the sacred and secular. Modern thinking opposes control and freedom, being and norms, thing and thought, subject and object. Or it tries to think them together in ways that ultimately are arbitrary.

Throughout the vignettes, I have mentioned a philosophy that has helped me in each area. It is the philosophy of the late Herman Dooyeweerd, a mid-20th century Dutch thinker. It manages to integrate form with matter, sacred with secular, control with freedom, being with norms, thought with thing, and subject with object in a way that does not denature any of them and yet is not arbitrary. At the same time it deliberately takes an everyday attitude even while it also acknowledges and welcomes the results of scientific work. The reason it can integrate things we have long assumed to be incompatible is because Dooyeweerd questioned the most fundamental presuppositions that have lain at the roots of Western thinking over the last 2,500 years.

Purpose of This Work

In this work I try to show that Dooyeweerd is at least interesting enough, to those who work or research in any of these areas of IT/IS, to be considered alongside other approaches. I do not seek to show Dooyeweerd is superior to any of these, let alone replace them. Rather I simply recommend him for further study. To do this I make a proposal (or set of proposals) for the adoption, development, testing and refinement of his approach by IS researchers and practitioners. It is only once this has been achieved that it is right to even begin to evaluate Dooyeweerd against other thinkers, because before then his thought would not have been properly understood.

As mentioned at the start, the central purpose of this work is to propose a new way of looking at IT/IS, which is philosophically sound and yet practical. That is, it develops five philosophical frameworks for understanding IS, one for each area. But it might be of value in other ways too:

- It is unusual in advocating a lifeworld approach in all areas of research and practice, reinterpreting the issues in each in the light of everyday experience
- It aims at integrable frameworks for understanding the various areas with a view to being able to richly understand the 'whole story that is information systems' in a coherent way
- It addresses both technical and non-technical areas at the micro and macro levels
- It demonstrates how philosophy in general may be used to construct (lifeworld-oriented) frameworks for understanding
- It is the only general introduction to Dooyeweerd's philosophy available in the field of IT/IS

For the last reason, that part which explains Dooyeweerd (Chapters II, III) is designed to be used as a reference work relevant to IS which gives pointers into his thought for further study.

What the work specifically offers includes:

- It indicates how philosophy in general can be employed in working out frameworks for understanding in several areas of research and practice in IS.
- Though the framework it develops for each area is new, it discusses how each links to a selection of extant frameworks and significant issues in its area.
- It throws fresh light on some issues in each area, which, even if the reader might not wish to adopt either the framework offered or this particular philosophy, could be useful in stimulating new ideas or strategic directions in research or practice. It also provides a number of practical devices for each area.

- It systematically explains and critically discusses Dooyeweerd's philosophy, placing it in the context of other philosophies.
- It reviews current research in each area that has made use of this philosophy, and sets the research direction for its further application in IS.
- It makes a number of suggestions for critiquing and refining the philosophy itself.

Despite an emphasis on philosophy, this work (especially Chapters IV-VIII) should be readable by those who have only limited understanding of conventional philosophy. Indeed, knowledge of conventional philosophy might not help, because of Dooyeweerd's very different approach. Reference is made to other philosophers, not because the reader is assumed to understand them, but mainly in order to situate Dooyeweerd among them for those readers who do know their work.

The Chapters

Chapter I is preparatory to formulating philosophical frameworks for understanding information systems. It clarifies what is meant by 'information systems', setting out the five areas referred to in the vignettes. It discusses what is meant by 'understanding' and outlines some characteristics of the everyday 'lifeworld'. It explains what is to be expected in 'frameworks', and finally discusses what is meant by 'philosophical' and what the role of philosophy should be in such a work. In doing this, it gives initial reasons why Dooyeweerd is worth exploring.

Chapter II sets out Dooyeweerd's general approach to philosophy and explains in what ways it is so different from most Western thinking. In particular it makes clear that Dooyeweerd offered both a critical and a positive philosophy, first deconstructing three millennia of Western thought and then having enough courage to construct something to be itself critiqued.

Chapter III explains those portions of Dooyeweerd's positive proposal that will be used in formulating frameworks for understanding. It is necessary to introduce this philosophy systematically because the reader needs to be able to grasp it to the extent of being able to apply it in their own fields in ways not discussed here, to understand how it relates to other streams of thinking, and even to properly test and refine it. It explains Dooyeweerd's 'general theory of modal spheres', his approach to being and things, to knowledge, and to human life. It reviews some extant critique of Dooyeweerd.

Chapters IV to VIII then explore how Dooyeweerd's ideas can be used to formulate frameworks for understanding each of the areas indicated by the vignettes (though in a different order). Each chapter begins with a brief discussion of what 'everyday

experience' or 'lifeworld' means in the area, which surfaces several main issues that need to be addressed. Application of Dooyeweerd's thought is then explored in relation to these issues, including offering practical devices that might assist practitioners in the area. Also each chapter contains discussion of some extant ways of seeing the area. The aim of each chapter is not only to formulate a framework for understanding, but also to demonstrate a general approach that anyone might be able to follow, using whatever is their favourite philosophy.

Chapter IV begins with human use of IT artefacts and systems, as introduced in Vignette 2, because we want to place the human being, rather than the technology, at the centre. Three usage relationships are discussed: human computer interaction, engagement with represented content and human living with computers (the first two do not feature in the Vignette), each of which is seen as multi-aspectual human functioning but each exhibiting a different set of issues. (The Dooyeweerdian notion of multi-aspectual functioning is explained in Chapter III.) Both the structure (i.e. nature) of each and the normativity that is operative in each is explored. The framework thus developed provides a way of addressing the tricky problems mentioned in Vignette 2. How this framework can engage with extant frameworks for understanding is discussed, including Walsham's 'Making a World of Difference' and Winograd and Flores' 'Language Action Perspective'.

Chapter V then discusses the nature of computers (Vignette 4) and also information, because the reflective user will at some time ponder this, and because we need some basis for understanding what computers can and cannot do. The latter centres on the artificial intelligence question of whether computers can think or not. Employing Dooyeweerd's notion of the multi-aspectual meaningful whole (explained in Chapter III), it sees computers as multi-level systems, not dissimilar to Newell's theory. Indeed, the framework developed here shows how Dooyeweerd can provide a sound philosophical underpinning for Newell's theory, including his strong ontological claim. Fresh light is shone into Searle's Chinese Room thought-experiment. Dooyeweerd's treatment of performance art is used to understand the nature of computer programs.

Chapter VI formulates a framework for understanding IS development (Vignette 3) as multi-aspectual human functioning in which the post-social aspects are prominent. In fact, four distinct but interwoven multi-aspectual functionings are explored, two reflecting the issues encountered in Vignette 3. The history of perspectives on methodologies in ISD is explained from a Dooyeweerdian perspective, and it is shown how Dooyeweerd can enrich Checkland's well-known soft systems methodology. It is in this area that most application of Dooyeweerd to IS has occurred so far, and the work of several thinkers is discussed.

Chapter VII discusses the technological resources that IS developers make use of—the programming or knowledge representation (KR) languages, code libraries and inter-program protocols. It begins with Brachman's call for 'KR to the people' (KR languages so natural that anyone could, in principle, use them to develop their

own IS). This call is answered by reference to Dooyeweerd's approach to the diversity of the world, with which a detailed proposal is presented for a multi-aspectual development toolkit. My own premature proposal for 'aspects of knowledge' (Vignette 1) is seen as a small subset of this. The framework formulated here acknowledges a transcending reality of immense diversity, and discusses how to future-proof the toolkit to cope with unforeseen requirements. It also, perhaps more usefully, shows how this proposal can be used as a yardstick against which to measure extant proposals, from the relational data model, object orientation, Wand and Weber's proposal, and the employment of Alexander's design patterns in software.

Chapter VIII discusses how Dooyeweerd's philosophy can address the 'macro' level of IS: the global, societal issues. I have called this our technological ecology, in which human living is 'inside' IT (Vignette 5). First, Schuurman's use of Dooyeweerd to define a 'liberating vision for technology' is discussed, which establishes the conditions under which technological development is a blessing rather than curse for humanity. Then the circular relationship between us and IT is examined: though created by us IT nevertheless changes not only the way we live but how we see ourselves. Critiques by feminism that IT has become inscribed with masculinity and by others that it has become inscribed with Western values are outlined and shown to be commensurable with, but a subset of, what Dooyeweerd could offer. The root of problems in this area of technological ecology is traced to various types of religious dysfunction, for which mere economic, social or technological solutions will be ineffective. Throughout this area is the issue of the ultimate Destiny of IT and humankind.

Chapter IX reflects on the proposals and on our use of Dooyeweerd's philosophy. It summarises the five frameworks for understanding, and discusses how they can cohere to help us understand the 'whole story' that is IS. The benefits and limitations of having used Dooyeweerd are discussed, and various suggestions made for critiquing and refining Dooyeweerd's philosophy itself are collected together. The degree to which the whole exercise has been able to meet the requirements set out in Chapter I is discussed, including the issue of whether one overarching framework should be sought. Finally, the very process of our exploration is discussed, including how the approach adopted in this volume could be used by those who wish to employ a different set of aspects, different areas of research and practice, or even an alternative philosophy to that of Dooyeweerd. The book ends with a brief suggestion for the future.

References

Attarwala, F. T., & Basden, A. (1985). A methodology for constructing expert systems. *R&D Management, 15*(2), 141-149.

Basden, A. (1983). On the application of expert systems. *International Journal of Man-Machine Studies, 19*, 461-477.

Basden, A. (1993). Appropriateness. In M. A. Bramer & A. L. Macintosh (Eds.), *Research and development in expert systems X* (pp. 315-328). Cranfield, UK: BHR Group.

Basden, A., Watson, I. D., & Brandon, P. S. (1995). *Client centred: An approach to developing knowledge based systems.* Chilton, UK: Council for the Central Laboratory of the Research Councils.

Castell, A. C., Basden, A., Erdos, G., Barrows, P., & Brandon, P. S. (1992). Knowledge based systems in use: A case study. In *British Computer Society Specialist Group for Knowledge Based Systems, Proceedings from Expert Systems 92 (Applications Stream)*. Swindon, UK: British Computer Society.

Landauer, T. K. (1996). *The trouble with computers: Usefulness, usability and productivity.* Cambridge, MA: Bradford Books, MIT Press.

Newell, A. (1982). The knowledge level. *Artificial Intelligence, 18*, 87-127.

Note

Trademarks

Copyrights