Uncovering Empirical Modelling

by

Karl George King

Thesis

Submitted to The University of Warwick

for the degree of

Master of Science (Research)

Department of Computer Science

September 2006
To Laura, with thanks
# Contents

List of Figures \hspace{6cm} v

Acknowledgments \hspace{3cm} vii

Declarations \hspace{4cm} viii

Abstract \hspace{6cm} ix

Abbreviations \hspace{8cm} x

## Chapter 1 Introduction

1.1 Thesis overview \hspace{2cm} 2
1.1.1 Background and Philosophy \hspace{2cm} 3
1.1.2 Empirical Modelling Principles \hspace{2cm} 3
1.1.3 Empirical Modelling Activity \hspace{2cm} 4
1.1.4 Empirical Modelling Tools \hspace{2cm} 4
1.1.5 Conclusions and Future Research \hspace{2cm} 5

## Chapter 2 Background and Philosophy

2.1 Introduction \hspace{2cm} 6
2.2 Preamble \hspace{2cm} 6
2.3 Challenging the analytic tradition \hspace{2cm} 7
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2.4</td>
<td>Adjusting possibilities for other observations</td>
<td>52</td>
</tr>
<tr>
<td>4.2.5</td>
<td>Reflecting on the exercise</td>
<td>52</td>
</tr>
<tr>
<td>4.3</td>
<td>Music Modelling</td>
<td>54</td>
</tr>
<tr>
<td>4.4</td>
<td>Timetabling</td>
<td>58</td>
</tr>
<tr>
<td>4.5</td>
<td>Developing Intelligent Agents</td>
<td>61</td>
</tr>
<tr>
<td>4.6</td>
<td>Summary</td>
<td>65</td>
</tr>
<tr>
<td>Chapter 5</td>
<td>Empirical Modelling Tools</td>
<td>67</td>
</tr>
<tr>
<td>5.1</td>
<td>Brief background to tkeden</td>
<td>67</td>
</tr>
<tr>
<td>5.2</td>
<td>Core Notations</td>
<td>69</td>
</tr>
<tr>
<td>5.2.1</td>
<td>eden</td>
<td>69</td>
</tr>
<tr>
<td>5.2.2</td>
<td>DoNaLD</td>
<td>71</td>
</tr>
<tr>
<td>5.2.3</td>
<td>SCOUT</td>
<td>75</td>
</tr>
<tr>
<td>5.2.4</td>
<td>Other notations</td>
<td>78</td>
</tr>
<tr>
<td>5.3</td>
<td>Evaluating tkeden</td>
<td>79</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Cognitive Dimensions</td>
<td>79</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Abstraction gradient</td>
<td>81</td>
</tr>
<tr>
<td>5.3.3</td>
<td>Closeness of mapping</td>
<td>81</td>
</tr>
<tr>
<td>5.3.4</td>
<td>Consistency</td>
<td>82</td>
</tr>
<tr>
<td>5.3.5</td>
<td>Diffuseness/Terseness</td>
<td>83</td>
</tr>
<tr>
<td>5.3.6</td>
<td>Error proneness</td>
<td>84</td>
</tr>
<tr>
<td>5.3.7</td>
<td>Hard mental operations</td>
<td>84</td>
</tr>
<tr>
<td>5.3.8</td>
<td>Hidden dependencies</td>
<td>85</td>
</tr>
<tr>
<td>5.3.9</td>
<td>Premature commitment</td>
<td>86</td>
</tr>
<tr>
<td>5.3.10</td>
<td>Progressive evaluation</td>
<td>86</td>
</tr>
<tr>
<td>5.3.11</td>
<td>Role expressiveness</td>
<td>87</td>
</tr>
<tr>
<td>5.3.12</td>
<td>Secondary notation</td>
<td>87</td>
</tr>
</tbody>
</table>
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>The alpha and beta semantic relations [Smi96]</td>
<td>11</td>
</tr>
<tr>
<td>3.1</td>
<td>Cayley Diagrams - Richard Cartwright PhD thesis [Car98]</td>
<td>29</td>
</tr>
<tr>
<td>3.2</td>
<td>Photograph of two tumblers (jugs)</td>
<td>31</td>
</tr>
<tr>
<td>3.3</td>
<td>Screenshot of the Jugs model</td>
<td>33</td>
</tr>
<tr>
<td>3.4</td>
<td>An alternative semantic diagram for human computing</td>
<td>37</td>
</tr>
<tr>
<td>4.1</td>
<td>Sudoku puzzle example – taken from the Guardian newspaper [Weba]</td>
<td>50</td>
</tr>
<tr>
<td>4.2</td>
<td>The Sudoku model in action</td>
<td>53</td>
</tr>
<tr>
<td>4.3</td>
<td>The simple music model</td>
<td>56</td>
</tr>
<tr>
<td>4.4</td>
<td>The interface of the re-developed Temposcope [Kin04b]</td>
<td>60</td>
</tr>
<tr>
<td>4.5</td>
<td>Keer’s Ant model [Kee05]</td>
<td>64</td>
</tr>
<tr>
<td>4.6</td>
<td>Stein’s Car model [Ste05]</td>
<td>65</td>
</tr>
<tr>
<td>5.1</td>
<td>The tkeden input window</td>
<td>68</td>
</tr>
<tr>
<td>5.2</td>
<td>The Guernsey flag as defined in DoNaLD</td>
<td>72</td>
</tr>
<tr>
<td>5.3</td>
<td>The Guernsey flag interface in SCOUT</td>
<td>75</td>
</tr>
<tr>
<td>5.4</td>
<td>drawScout version 1.0 with two created windows</td>
<td>90</td>
</tr>
<tr>
<td>5.5</td>
<td>An extended drawScout aiding intelligibility of the sisyphus lift model</td>
<td>92</td>
</tr>
<tr>
<td>5.6</td>
<td>The first drawSlide presentation</td>
<td>93</td>
</tr>
<tr>
<td>5.7</td>
<td>The original drawSlide presentation in edenSlide</td>
<td>95</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

5.8  The original implementation of VST ..................................... 96
5.9  Use of VST in demonstrating Care’s planimeter model ............ 98
5.10 The Imagine Eden tool .......................................................... 99
5.11 Step 1 – creating the paddles and window for the ball ........... 101
5.12 The Pong model with scoreboards ......................................... 105

6.1  drawDonald prototype ......................................................... 115
6.2  A possible interface for the Eden Active Scripting Environment (EASE) ................................................................. 117
Acknowledgments

First and foremost, I must acknowledge the helpful guidance provided to me throughout the past year by my supervisor Meurig Beynon. I must also acknowledge the support provided by Steve Russ, particularly in my engagement with philosophical material. I would also like to thank fellow researchers in the group, who have aided the development of my thoughts through useful discussion, most notably Antony Harfield, Charlie Care and Ashley Ward.

I am also indebted to the work of previous research students in the group, without whose contribution this thesis would not have been possible.
Declarations

This thesis is presented in accordance with regulations for the degree of Master of Science by research. It has been composed solely by the author and has not been submitted in any previous application for any degree. All work in this thesis has been undertaken by the author unless otherwise stated.
Abstract

Empirical Modelling is the name given to an ongoing research programme at the University of Warwick which began in 1983 with the development of ARCA, a definitive (definition-based) notation for interactive Cayley diagrams (cf. [Bey83]). ARCA led to much interest in definitive notations and motivated the subsequent development of the Evaluator of DEfinite Notations (EDEN) in 1987 by Edward Yung, and the introduction of the notion of Modelling With Definitive Scripts (MWDS) [Bey85]. Traditional construals of computation were found to be insufficient in accounting for the semantic richness of MWDS and principles were introduced to distinguish such activity from conventional programming. As the principles and tools developed, consideration was given to the implicit philosophy of the approach, and strong connections were identified with the Radical Empiricism of William James [Bey03]. However, the distinctions between Empirical Modelling and traditional approaches to computing are both profound and subtle, and communicating the nature of the subject matter has proven difficult.

This thesis explores the breadth of Empirical Modelling from its underlying philosophical orientation to the current tool support. Its central theme is uncovering EM and it proceeds to do so by:

- Exploring connections between phenomenology and the philosophical orientation underlying EM.
- Developing a fresh account of Empirical Modelling activity and the semantics of computer-based artefacts.
- Considering areas in which EM principles and tools can be usefully applied.
- Describing work undertaken by the author in improving the current tool support and exploring alternative ways of MWDS.

The thesis concludes that whilst there is much still to be done in articulating the approach and realising the aspiration of supporting 'modelling in the stream-of-thought', Empirical Modelling provides a practical means of exploring problems that are not amenable to formalisation.
Abbreviations

AOP  Agent Oriented Parser

DoNaLD  A Definitive Notation for Line Drawing

DMT  Dependency Modelling Tool

EASE  Eden Active Scripting Environment

EDEN  The Evaluator of DEfinitive Notations

EM  Empirical Modelling

IA  Intelligent Agents

MWDS  Modelling With Definitive Scripts

SCOUT  A notation for SCreen layOUT

VST  Visual Symbol Table
Chapter 1

Introduction

This thesis is the result of a year of research into the nature of Empirical Modelling, principally motivated by a desire to address the difficulty of communicating the subject matter. In the chapters that follow, I consider the breadth of Empirical Modelling, from its philosophical orientation and principles, to the modelling activity itself and current tool support. This thesis promotes a shift away from previous theses, which have focused particularly on dependency and the technical aspects of tool support for Empirical Modelling, toward a broader account of the subject matter, based on observation, experiment and human experience.

Whilst the current state of the principles and philosophy is largely the result of in-depth consideration of practical work, this thesis emphasises that it is important that the philosophy and principles now take a more prominent role in the development of tool support for Empirical Modelling activity. The current tools and models do illustrate much potential for useful application, but there are many improvements that need to be made in working toward the aspiration of supporting “modelling in the stream-of-thought” [Kin04a]. It may be important to develop further technologies that complement the alternative view of computation and computing promoted by EM. Though such development work is beyond the scope
of this thesis, one objective in this thesis is to inform the future tool support for Empirical Modelling activity.

The four key research objectives are to:

1. Embellish Empirical Modelling philosophy by exploring possible connections with phenomenology.

2. Use phenomenological ideas to develop a fresh account of Empirical Modelling activity and the semantics of computer-based artefacts.

3. Explore potential tool developments that may provide better support for Empirical Modelling activity.

4. Illustrate particular areas of application in which Empirical Modelling principles and tools can be usefully applied.

Further to these objectives, consideration is given throughout to communicating Empirical Modelling.

1.1 Thesis overview

The thesis comprises five chapters:

- Chapter 2: Background and Philosophy
- Chapter 3: Empirical Modelling Principles
- Chapter 4: Empirical Modelling Activity
- Chapter 5: Empirical Modelling Tools
- Chapter 6: Conclusions and future directions
1.1.1 Background and Philosophy

Chapter 2 is devoted to a consideration of the implicit philosophical orientation of the discipline of computer science and Empirical Modelling philosophy. Exploration of the philosophical underpinnings of the subject matter is a relatively recent endeavour, dominated by the connections made by Meurig Beynon with the Radical Empiricism of William James [Bey03]. Through the research carried out by the author over the past year, it has become apparent that phenomenological ideas, particularly those of such philosophers as Heidegger and Merleau-Ponty can also be usefully applied in understanding the nature of Empirical Modelling. Whilst phenomenology and William James's Radical Empiricism do not share the same agenda, the emphasis placed on the primacy of experience in both philosophies does provide a suitable point of contact for legitimately drawing ideas from each.

A key challenge in articulating the nature of Empirical Modelling is to convey the need for an alternative view of computation to that traditionally adopted in the discipline of computer science, and this forms a central theme of the chapter. Interestingly, the pertinent philosophical issues are similar in character to those recognised as problematic by William James and Edmund Husserl over one hundred years ago.

1.1.2 Empirical Modelling Principles

Chapter 3 considers the key principles of Empirical Modelling and explores their relationship with the alternative philosophical orientation articulated in the previous chapter. Drawing on key ideas from William James, Martin Heidegger and Michael Polanyi, and particularly the Jamesian notion that one experience can know another, the semantics of computer-based artefacts is reconsidered.

The Jugs model is then introduced to illustrate the nature of Empirical
1.1.3 Empirical Modelling Activity

Chapter 4 revisits the nature of EM activity before considering particular areas of application with reference to their amenability to formalisation. Four areas of application are considered with reference to the author’s model making for Sudoku and music, previous research exploring timetabling, and modelling undertaken by Keer and Stein in developing Intelligent Agents. The central theme of the chapter is to illustrate the potential for EM application wherever the problem domain or part thereof is not amenable to formalisation.

1.1.4 Empirical Modelling Tools

Chapter 5 describes the principal Empirical Modelling tool, tkeden, and evaluates its usability with Petre and Green’s Cognitive Dimensions [GP96]. The author’s work in exploring tool improvements and alternative ways of modelling with tkeden is then considered.

The tkeden tool has been essential in the development of the subject matter and the focus of the majority of Empirical Modelling research. This thesis is one of the first to shift the emphasis away from the tools and technical aspects of dependency toward a broader consideration of how the activity of Empirical Modelling, as understood in principle, can be best supported in practice. Following this shift in emphasis and in keeping with the spirit of the approach, the potential tool developments have been explored using the existing tool support. A future consideration for the research group will be whether a better understanding of Empirical Modelling can inform the development of a completely new environment to
support the aspiration of facilitating *modelling in the stream-of-thought* [Kin04a].

1.1.5 Conclusions and Future Research

The final chapter considers some of the key challenges for future Empirical Modelling research. Communicating the nature of the research and securing funding for continued exploration and development will be key challenges in the short term. In the long term, there is a need to consider further areas of application and to develop further tools to support Empirical Modelling activity.
Chapter 2

Background and Philosophy

2.1 Introduction

This chapter considers background to the activity of Empirical Modelling (EM) and its underlying philosophical orientation. The chapter provides the reader with some context for the later consideration of EM activity, but its principal aim is to establish the key philosophical differences between EM and traditional approaches in computer science.

2.2 Preamble

It has been observed that whilst it is possible to engage with Empirical Modelling principles and tools without considering philosophical issues, they are essential to appreciating its nature. In communicating EM, a key challenge is to overcome the barriers created by traditional ways of thinking about the nature of computing, computation and the semantics of computational artefacts. As philosophical issues are traditionally viewed as beyond the remit of mainstream computer science, the challenge is even greater, as it is necessary prior to presenting an alternative stance,
to justify engaging in philosophical debate about the nature of computation in the first instance, and to articulate limiting assumptions of the traditional, implicit orientation in the second. Following this structure, the chapter begins by justifying the need to challenge traditional ways of thinking and to engage with philosophical issues, highlights problematic aspects of the tradition and presents the alternative stance promoted by EM.

2.3 Challenging the analytic tradition

A key problem is that existing construals of computation embody an implicit philosophical orientation – a legacy from the time of Alan Turing when analytic philosophy was particularly influential (cf. [Aye90]). As Winograd and Flores observe in Understanding Computers and Cognition [WF86], whilst computer scientists would not consider themselves philosophers, their ways of thinking nevertheless embody a philosophical tendency that should be challenged as a source of understanding. In itself, this may not appear problematic. Certainly, much has been achieved within the discipline under the traditional orientation. However, the problem is that it has become the only way of thinking about computing for many in the discipline of computer science; something that is apparent from widely accepted construals, where prominence has been given to language and logic, and computation has traditionally been defined in terms of abstract algorithmic behaviour (see Denning et al in [DCG+89]).

The inadequacy of traditional construals of computation has been particularly exposed in recent work in the field of cognitive science, where the validity of computationalism, defined by Matthias Scheutz as “the view that mental states are computational states”, has been widely debated and largely rejected. As Scheutz explains in [Sch02]:

7
Computation fails as an explanatory notion for mind, the critics claim, because computation, assumed to be defined solely in abstract syntactic terms, necessarily neglects the real-time, embodied, real-world constraints with which cognitive systems intrinsically cope. (p.ix) [Sch02]

The fact that all of these issues are as important to computational artefacts as they are to human cognition, as Scheutz goes on to observe, motivates re-examination of traditional construals of computation, and has led computationalists to seek a successor notion. Traditional construals of computation have also been found problematic by others in the discipline of computer science (cf. [Weg97] and [Wes96]). However, much as there will be no silver bullet for software development [Bro87], it is unlikely that any single construal of computation would be appropriate in all contexts; something which is perhaps evident from the large number of different construals in widespread use (see [Smi02]).

In the context of Empirical Modelling, we are centrally concerned with human computing – human engagement with computer-based artefacts – in which the issues identified by Scheutz are both important and pertinent. Following this, EM research seeks to provide computational support for thinking. As in the field of computationalism, existing construals have been found inadequate in accounting for the richness of model building in EM, and experience of this has led to the development of an alternative stance on computing, computation, and the semantics of computational artefacts.

At this point it is useful to consider particular aspects of the tradition in more detail to illustrate the nature of the problem.
2.4 Key problems

2.4.1 The intentional character of computation

Perhaps the most problematic aspect of traditional construals of computation is that they fail to deal adequately with one of its most fundamental properties. As Smith explains in [Smi02]:

> It is widely (if tacitly) recognised that computation is in one way or another a symbolic or representational or information-based or semantical – that is, as philosophers would say, an intentional – phenomenon. Somehow or other, though in ways we do not yet understand, the states of a computer can model or simulate or represent or stand for or carry information about or signify other states in the world (or at least can be taken by people to do so). (p.32) [Smi02]

The term *intentional* is worth further consideration at this point as it is often confused, as Tim Crane observes in [Cra03], with “the ordinary ideas of intention, intending and acting intentionally”. The term *intentional* is most simply described traditionally as the “aboutness” or “directedness” of states of mind. When I think about what I ate for lunch, or about going to the supermarket this evening, my thoughts are respectively about or directed at my lunch and going to the supermarket respectively. Furthermore, whilst the latter involves my intention, namely to go to the supermarket this evening, the former is simply “about” some thing, in this case my lunch. It follows that it is not a requirement for intentional states to have anything to do with ordinary intentions. As Tim Crane explains in [Cra03]:

> Intentions in the ordinary sense are intentional states, but most intentional states have little to do with intentions. (p.25) [Cra03]
Computational artefacts, in so far as they can be perceived to be *about* or to represent or stand for other states in the world, can be said to be intentional in some sense. However, there is clearly some distinction to be drawn between the nature of the intentionality of human thoughts and that of computational artefacts. John Haugeland uses the terms *original* and *derivative* intentionality to distinguish between these two types:

Derivative intentionality (or meaning) is the intentionality that something has only by virtue of having it *conferred* on it by something else that has it. A common example is the intentionality of words and sentences, which (according to this view) they have only because it is conferred on them by thoughts, which already have it. (p.161) [Hau02]

This further serves to illustrate an earlier point about the unlikelihood of a single construal of computation appropriate to all contexts. In *human computing*, it is more appropriate to think about intentionality in derivative rather than original terms. Much like that of sentences, the meaning of computational artefacts is better considered as the result of human interaction and interpretation. In contrast, in the fields of cognitive science and AI, more emphasis is placed on explaining the nature of original intentionality, as the ultimate goal is the possibility of developing mechanical minds that might exhibit such a property. Throughout this thesis, the *intentional* character of computational artefacts should be interpreted as the derivative rather than the original type.

### 2.4.2 Cartesian dualism

Perhaps more problematic because it is so implicit to the philosophical tradition and commonsense philosophy more generally is that there is a tendency to think
about computers and cognition as two entirely separate domains, as Winograd and Flores observe in [WF86]. It was mentioned earlier that the significant semantics of computational artefacts is traditionally understood in terms of its abstract algorithmic behaviour, rather than its relationship to that which it is directed at in human interaction and engagement therewith. Brian Cantwell Smith distinguishes between two semantic relations in the context of computer artefacts, alpha and beta (see Figure 2.1).

The alpha relation denotes the traditional view of the semantics of computer programs. It is the relation between the program and the abstract process that it implements. The beta relation is the relation between the abstract process of the program and the world. However, considering the semantics of computer-based artefacts in this way is somewhat detached from the everyday experience of their development and implementation, and disguises the more primitive nature of human engagement with computers. It also creates an irreconcilable dualistic divide between the artefact and that which it is about.
Considering computational artefacts under the Alpha relation is to deem the program to be independent of human interpretation and interaction, and objective. However, paraphrasing Winograd and Flores, whilst computer scientists in theorising about computational artefacts and describing their semantics in abstract terms consider themselves to be discovering how things really are, in doing so they create a detachment from everyday experience of interacting with the artefact. This resonates with the arguments presented by Heidegger in *The Origin of the Work of Art* [Hei01]:

> Colour shines and wants only to shine. When we analyse it in rational terms by measuring its wavelengths, it is gone. It shows itself only when it remains undisclosed and unexplained. (p.45) [Hei01]

However, that is not the only dualistic problem. Acknowledging the intentional character of computational artefacts is problematic in itself when presented in dualistic terms, and yet, for the purposes of practicality, it is often necessary that we distinguish between the computational artefact on the one hand and that which it is about on the other. We will return to this point in detailing the resolutions provided by William James and Martin Heidegger, but what is important to recognise is that in the context of human computing, the detached and dualistic theoretical viewpoint is inappropriate as it disguises the primacy of experience.

**2.4.3 Rationalistic epistemology**

The traditional philosophical orientation is also characterised by strong rationalistic tendencies. Rationalism, in contrast to empiricism, considers reason to be the primary source of all knowledge. Central to rationalism is systematic justification, something which is apparent in the discipline of computer science where emphasis
has been placed on abstract and mathematical explanation. However, there are also rationalistic tendencies to be found in the discipline’s approach to knowledge representation, and, as Beynon observes in [Bey03], the classical view of knowledge is intimately bound with language, rather than engagement with the world:

Knowledge is seen as something to be possessed that can be expressed and recorded as a proposition, as in “I know the telephone number of staff member X”. (p.3) [Bey03]

This rationalistic, reductionist view of knowledge is inappropriate in the context of human computing where a computer user, as Beynon goes on to describe, is likely to seek more from their computer-based models “than has been consciously encoded by way of ‘use cases’”. Much as we can engage with the world in new and different ways, so in the context of human computing, where the computer serves to support human creativity, it is more appropriate to adopt a more open view of knowledge. Importantly though, as Willard McCarty remarks in [McC05], the view of computers as nothing more than “knowledge jukeboxes” is a gross misunderstanding of human knowledge. Furthermore, it understates the potential for computers to support human thinking.

2.5 An alternative construal

Whilst exploration of philosophical foundations for Empirical Modelling is ongoing, significant connections have already been identified with the Radical Empiricism of William James. This section begins by considering a more appropriate view of the nature of computing. It then presents pertinent Jamesian ideas in relation to the problematic aspects of the traditional orientation described above and embellishes
the ideas, drawing from the phenomenological thinking of Heidegger, Merleau-Ponty, and Polanyi.

2.5.1 A broader view of computers and computing

The traditional view of computing as the systematic development of computer programs that implement algorithmic behaviours is seemingly detached from modern practice where those using computers have little concern for such matters. A more appropriate view is to consider computers in the context of the world rather than as they have traditionally been regarded, as an object of interest in themselves. This view is shared by Brian Cantwell Smith in [Smi02]:

Computers turn out in the end to be rather like cars; objects of inestimable social and political and economic and personal importance, but not in and of themselves, *qua* themselves, the focus of enduring scientific or intellectual inquiry. (p.51) [Smi02]

Much as it is possible to consider computers in broader terms, it is possible to recharacterise the nature of computing. As Brian Cantwell Smith goes on to say:

Rather, what computers are, I now believe - and what the considerable and impressive body of practice associated with them amounts to - is neither more nor less than the *full fledged social construction and development of intentional artefacts*. (p.52) [Smi02]

The nature of Empirical Modelling is best considered under this broader notion of computing as the development of intentional artefacts, and it is from this definition that we can begin our account of the alternative orientation promoted by EM.
2.5.2 Returning to experience – overcoming dualism

Recognising computers in the context of the world, and the pragmatic value of computation as related to its intentional character, the alternative orientation must challenge the dualistic tendencies of the tradition. To talk about computers in abstract terms is to step back from the practical experience of engaging with them. What is apparent is that the majority of those interacting with computer programs have little concern for its operational semantics. Instead, they are motivated by a more pragmatic concern of it providing reliable and useful support as they carry out particular activities. Accordingly, focusing on computers in isolation is inappropriate in the context of human computing. As Beynon argues in [Bey03], computer-based artefacts are more usefully regarded as a source of experience.

The Radical Empiricism of William James provides an appropriate philosophical foundation for this perspective. William James considered dualism man-made, and recognised the problems this had caused throughout the history of philosophical discourse. In *A World of Pure Experience* James explains:

> Throughout the history of philosophy the subject and its object have been treated as absolutely discontinuous entities; and thereupon the presence of the latter to the former, or the 'apprehension' by the former of the latter, has assumed a paradoxical character which all sorts of theories had to be invented to overcome. (p.52) [JBBS76]

Similarly in the discipline of computer science, under the traditional perspective, the dualistic stance leads to the same outcome – irreconcilable problems in accounting for how abstract algorithmic behaviour relates to what the model is *about*. Instead, and following the Jamesian *Weltanschauung*, the answer is to be found by returning to something more primitive – the primacy of experience. As James observes in [JBBS76]:

15
The world may well exist by itself, but we know nothing of this because for us it is exclusively an object of experience. [JBBS76]

Returning to experience, according to James, enables all of the problems introduced by dualistic tendencies to be overcome. Contrary to popular belief and discourse deriving from the philosophy of Descartes, there is no primary distinction to be made between subject and object, or consciousness and content, or knower and known. Instead, according to James, there is something more fundamental to be found in the primacy of experience that is neither mental nor physical, something which James refers to as ‘pure experience’, defining it in The thing and its relations thus [JBBS76]:

‘Pure experience’ is the name which I gave to the immediate flux of life which furnishes the material to our later reflection with its conceptual categories. Only new-born babes, or men in semi-coma from sleep, drugs, illnesses, or blows, may be assumed to have an experience pure in the literal sense of a that which is not yet any definite what, tho’ ready to be all sorts of whats; full both of oneness and of manyness, but in respects that don’t appear; changing throughout, yet so confusedly that its phases interpenetrate and no points, either of distinction or of identity, can be caught. Pure experience in this state is but another name for feeling or sensation. But the flux of it no sooner comes than it tends to fill itself with emphases, and these salient parts become identified and fixed and abstracted; so that experience now flows as if shot through with adjectives and nouns and prepositions and conjunctions. Its purity is only a relative term, meaning the proportional amount of unverbalized sensation which it still embodies. (p.93) [JBBS76]
The key difference between Radical Empiricism and traditional empiricism is in part apparent from the latter sentences of the quotation above. Contrary to the emphasis placed in traditional empiricism on disjunctive relations, James asserts that “the relations between things, conjunctive as well as disjunctive are just as much matters of direct particular experience, neither more so nor less so, than the things themselves” [JBBS76]. For the purposes of human computing, it is useful to regard a computer-based artefact primarily as a source of experience, and to understand its intentionality with reference to conjunctive relations formed throughout our interaction with it.

Conjunctive relations are worthy of further consideration at this point. They are described by James in *A World of Pure Experience* thus:

Relations are of different degrees of intimacy. Merely to be ‘with’ one another in a universe of discourse is the most external relation that terms can have, and seems to involve nothing whatever as to farther consequences. Simultaneity and time-interval come next, and then space-adjacency and distance. After them, similarity and difference, carrying the possibility of many inferences. Then relations of activity, tying terms into series involving change, tendency, resistance, and the causal order generally. Finally, the relation experienced between terms that form states of mind, and are immediately conscious of continuing each other. (p.44) [JBBS76]

When relations are taken as parts of experience themselves, it is possible to further understand the nature of human interaction with computers. As I write this, throughout my interaction with the computer, my direct experience comprises several elements, and all of the things in front of me – my keyboard, mouse, monitor, speakers, printer, camera, notepad, pen, desk, the family photographs on my wall...
and my thoughts serve to make up my personal experience of writing. In my experience, each element is here and now, their similarities as well as their differences are apparent; I have the expectation that pressing the keys will indivisibly result in words appearing on the page in front of me, and all the while, my thoughts and previous experiences are present and join together in my states of mind.

Following his account of conjunctive relations, James continues by describing two types of knowing that are central to his thinking and apparent from his earlier writings. James distinguishes between perceptual and conceptual knowing, where a mind “enjoys direct ‘acquaintance’ with a present object”, and where “the mind has ‘knowledge-about’ an object not immediately there”. For example, as I sit here at my desk, my mind enjoys direct acquaintance with my keyboard, my desk fan, my copy of *Essays in Radical Empiricism*. If I were to leave the room, and to recall these items in my thoughts, I would revisit my knowledge ‘about’ them. Crucially following Radical Empiricism, my thought would be considered an experience in itself, directed on the very same keyboard, desk fan, and copy of *Essays in Radical Empiricism*, an experience that, in Jamesian terms, “knows another”.

Just as it is possible for us to manipulate our thoughts, to shape them in ways that are not possible *in the world*, the development and manipulation of computer-based artefacts facilitates human thinking and understanding by providing support for creativity, exploration and variation. As Willard McCarty argues in [McC05], the traditional perspective of computers is inappropriate in this respect, as it neglects the role of such aspects in the development of human knowledge:

Although efficient access to data is an essential function of computing, the greater potential is for *computers as modelling machines*, *not knowledge jukeboxes*. To think of them as *only* the latter is pro-
foundly to misunderstand human knowledge. (p.27) [McC05]

Computers are more appropriately considered as providing the means for the making of meaning and the development of knowledge and understanding. Following this, further light is shed on the nature of the intentionality of computer based artefacts; it can be said that experience of interacting with a computer-based artefact is “an experience that knows another” – the experience of interacting with that which the artefact is about.

It is unfortunate that language encourages the use of dualistic terms. However, it should be recognised that the duality is dissolved when everything is considered in terms of experience. This is neither in the world nor in the mind but a blend of the two. Following Heidegger, as interpreted by Winograd and Flores, “existence is interpretation” and “interpretation is existence” [WF86].

Clearly there is a lot more to be said about the ontological status of computer based artefacts. For the moment however, it can be simply understood that computer-based artefacts relate to something other than themselves, and whether conceptual or perceptual in character, should be understood in terms of experience.

2.5.3 Phenomenology

James is not the only philosopher to emphasise the primacy of experience. It also forms the foundation of phenomenology, which, as Husserl originally intended it, has much in common with James’s Radical Empiricism (cf. [Mor00]). In parallel with the original motivations for exploring alternative perspectives on the nature of computing and computation, phenomenology was introduced as “an attempt to bring philosophy back from abstract metaphysical speculation wrapped up in pseudo-problems, in order to come into contact with the matters themselves, with concrete living experience” [Mor00]. The phenomenology of Heidegger is partic-
ularly interesting as in regard to the structure of intentionality, as Agre observes in [Agr97], Heidegger rejects the Cartesian starting point that is apparent in the work of Husserl, emphasising instead the primacy of practical engagement with the world. Moran provides useful description of the difference between the Husserlian view of things in the world and the Heideggerean perspective in [Mor00]:

Heidegger agrees with Husserl that intentionality is a defining characteristic of all lived experiences (Erlebnisse), but, against Husserl, he emphasises the practical embodied nature of these experiences. Our lived experiences are practical bodily encounters with things in our environment: for example, in moving around a room I am in an encounter with a ‘thing in the environment’, a chair, not chair-sensations. Hence, I can genuinely say “the chair is uncomfortable” and grasp the mode of being of the chair for me, for my living. The chair’s being is one of discomfort for me. (p. 232) [Mor00]

Heidegger’s viewpoint is more appropriate in considering the experience of interacting with computer-based artefacts as to adopt that of Husserl is to step back from the primacy of our practical engagement with the world. Rather than considering the experience of computer-based artefacts in terms of sense data, it is more appropriate to adopt a more holistic viewpoint. Computer-based artefacts, like the chair in Moran’s example, have some identity and mode of being throughout interaction.

A further distinction is made by Heidegger in regard to the nature with which we engage with things in the world. Heidegger identifies two modes in which things in the world present themselves to us in our practical everyday engagement therewith:
1. Readiness to hand (Zuhandenheit) – The idea that things in the world primarily present themselves as ‘available’ (see Dreyfus [Dre91]) to us in the sense of being somehow useful in our practical engagement.

2. Presence at hand (Vorhandenheit) – The idea that it is through a secondary, more theoretical mode of engagement that we see things in themselves.

As explained earlier, it is the theoretical mode of engagement that is seen as superior to the practical engaged viewpoint in the discipline of computer science. Adopting a broader view of computation such as that proposed by Brian Cantwell Smith where computers are to be understood in the context of the world requires consideration of pragmatic human engagement with the world. Moran provides a useful description of presence at hand [Mor00] which resonates strongly with the traditional perspective and further serves to illustrate the alternative viewpoint that is more appropriate to human computing:

   This theoretical way of viewing things leads to science, to the pure interest in examining things as they are, bracketed from their connections and engagements with our interests. (p.233) [Mor00]

However, in the context of human computing, the connections and engagements with our interests are particularly important. The value of computers lies in their practical value for supporting our various activities.

A further Heideggerean idea that is appropriate to the discussion is the notion of breakdown. An aspiration in human computing is to provide support for ‘modelling in the stream-of-thought’ where the modeller interacts with and interprets the model in an holistic manner, with each of its parts being understood in terms of its participation in the model as a whole and in relation to that which the model refers. However, this is somewhat idealistic; it is common when interacting
with computers to experience *breakdown* where the computer suddenly becomes present to hand. When this occurs, the individual parts of the model become present in themselves in much the same manner as that which Polanyi identifies in [Pol61]:

The essential feature is throughout the fact that *particulars can be noticed in two different ways*. We can be aware of them uncomprehendingly, i.e. in themselves; or understandingly, in their participation in a comprehensive entity. In the first case we focus our attention on the isolated particulars, in the second our attention is directed beyond them to the entity to which they contribute. In the first case therefore we may say that we are aware of the particulars focally, in the second, that we notice them subsidiarily in terms of their participation in a whole. (p.461) [Pol61]

Where “our attention is directed beyond” the individual parts of the artefact “to the entity to which they contribute”, and where they are understood directly in relation to that to which they refer, the modeller could be said to be ‘modelling in the stream-of-thought’. In this mode of interaction, the attention of the modeller is concentrated on the activity in which they are engaged, with little concern for their tools in themselves. This resonates with the aspiration of *invisible computing* (cf [Nor98]), but extends the concept to the realm of artefact development as well as manipulation and end-user interaction. When we are aware of the parts of the model (traditionally variables) we can relate to them in different ways when in the first instance we consider them in terms of what they are *about* and in the second we consider them in themselves. What is interesting is that in traditional procedural programming languages, the relationship between variables and what the artefact is *about* is often difficult to establish. However, in applications such
as the spreadsheet, whilst it is likely that there will be an underlying variable that corresponds to the content of a particular cell, this is of little concern to the user for whom there is a direct relationship between a cell and that which it is about, for example where a cell indicates the total revenue for a business. Central to the alternative position is the notion that the whole is more than the sum of its individual parts, and this provides another motivation for adopting a broader view of computers where computers are not studied in themselves, but more comprehensively in the context of the world.

2.5.4 The making of meaning

So far, we have concentrated on the intentionality of computer-based artefacts and its nature. However, it is also necessary to consider how that intentionality comes about. The role of human interaction and interpretation is often neglected in computer science in its emphasis on the theoretical viewpoint. This has been the case since the computer was invented. Having ‘done the hard part’ in designing and creating the computer, all that remained was what was considered to be the trivial task of programming them [CKA04]. However, it was soon found that this was difficult, and the emphasis placed in computer science on theory over practice only serves to disguise this.

The work of David Gooding in investigating the development of scientific theory is useful in understanding the process by which humans develop computer-based artefacts in the context of human computing, and also provides an alternative view of knowledge, based on observation and experiment. It is also helpful in considering the ontological status of artefacts in a state of completion (cf. discussion of “systemness” in [LN01]).

Gooding, like Heidegger and James, also considers practical engagement
with the world to be primary, and rejects the superior status given to detached theoretical observation, particularly in analytical philosophy. A central idea in traditional computer science is that the operational semantics of computer programs can be described in abstract terms. Following this, programs in a state of completion are considered knowable independent of human interpretation. This is analogous to the development of scientific theory as Gooding discusses at length in [Goo90]:

As scientists introduce more visual, symbolic or conceptual modes of understanding, so the procedural, non-discursive basis experience becomes less important to communication and understanding. Representations (images, devices, words) gain a degree of independence of observational practices. Phenomena – and possible phenomena – are first made ‘easy to see’ and then ‘evident’. They finally acquire the property of being ‘out there’, as given independently of human action. This is because the role of human intervention soon drops out of the account altogether. (p.89) [Goo90]

The superior status given to the theoretical viewpoint disguises the part played by humans in its development. In the context of human computing however, the role of the human is vital, and the process of “coming to know” in some instances is perhaps even more important than the artefact in itself. This is central to Gooding’s orientation and apparent from the introduction to [Goo90] where he seeks to make the role of observation and experiment in the development of scientific theory explicit:

Modern philosophy cannot understand how what scientists do gives them power over nature as well as our imaginations. It lacks a plausible theory of observation.
This book proposes an alternative view based on aspects of scientific work largely neglected by modern, especially analytical, philosophy. These are the agency of observers and the way in which their observation of nature is mediated by their interactions with each other, with their instrumentation and with the material world. (p.xii) [Goo90]

Gooding introduces the notion of a construal to describe the development of human understanding throughout the process of experimentation. As Gooding relates in [Goo90], construing “may be thought of as a process of modelling phenomena while the conceptual necessities of theory are held at arms length”. Construals are best described as “embodiments of phenomena” – “communicable representations” of experience, constructed empirically through exploratory interaction guided by the situated interpretation of observation. Similarly, computer-based artefacts can be regarded as construals in the context of human computing – they are intentional in character and are usefully considered as work in progress, something that is always open to future embellishment and re-interpretation.

2.6 Summary

The central theme of this chapter has been to provide an overview of an alternative view of computers that is more appropriate to the context of human computing where human interaction and engagement plays a vital role in the development of computer based artefacts. As traditional approaches in computer science have been shaped by the analytic tradition, it is important that they are challenged and that alternative perspectives are developed, especially in areas where mainstream conceptions of computing and computation are insufficient for interpreting everyday practice.
Perhaps the biggest problem in interpreting human computing using traditional construals is that whilst they are useful in understanding the operational semantics of computer based artefacts in terms of the algorithmic behaviour that they implement, they fail to acknowledge and adequately account for their intentional character – the fact that they represent or stand for other things in the world, whether conceptual or perceptual. This is particularly problematic in the context of human computing, as rather than this aspect being a by-product of the development of a computer based artefact, it should be regarded as its significant semantics.

Another issue is the dualistic tendencies inherent in the traditional approach. The separation between the objective world of the computer and the subjective world of human cognition, and that between computer based artefacts and that which they are about leads to the problem of reconciling abstract processes with real world phenomenon. However, this problem is largely artificial and, much as we do not require any concept of gravity to observe and experience its effect, it is unnecessary for us to understand the behaviour of artefacts in abstract terms to engage with them. Rather, computer based artefacts are more adequately regarded as a source of experience in human computing, and their relationship with the experience of that which the artefact intends, more appropriately considered as one of “knowing the other” in the Jamesian sense.

The traditional rationalistic approaches to epistemological aspects of computing have also led to the view of knowledge being connected with language, expressible in a particular linguistic form as in “I know the colour of the sea”, and of computer programs as being knowable in terms of their operational semantics. However on both counts in the context of human computing, these views of knowledge are inadequate and grossly inhibit the potential of computers. For example,
following the work of Polanyi [Pol67], it is clear that there is a considerable amount of knowledge that remains tacit, and as Brian Cantwell Smith argues in [Smi96], content is simply not reducible to any form. In respect of the idea that computer programs are known when their operational semantics are described in abstract terms, again there are problems when relating this to everyday experience. For example, my experience of using a particular program locally is very different when compared with using the very same program over a slow remote connection, yet the semantics would be considered identical. This is also the case when there are environmental changes. For example, sunlight on my screen can greatly detract from my experience of using a program.

Significantly, many of these issues are similar in character to those encountered in the field of philosophy over one hundred years ago, with both James and Husserl recognising that the subject had become “too abstract and academic” [JBBS76], and that there was a need to get back to the things themselves. As such, many of the ideas from Radical Empiricism and from areas of phenomenology stand in direct opposition to the analytic tradition and warrant further research in the development of alternative approaches to computing.

In the next chapter, we will consider Empirical Modelling in more detail. The general (alternative) philosophical orientation outlined here will serve as a basis for the discussion that follows.
Chapter 3

Empirical Modelling Principles

The purpose of this chapter is to consider Empirical Modelling principles, and to provide a fresh account of the nature of EM models drawing on the phenomenological ideas introduced previously. Experience has suggested that the link between Empirical Modelling philosophy and Empirical Modelling principles and tools is often unclear to those engaging with the subject matter. Following this, a central theme in this chapter is to identify and discuss these links to address the danger of subverting one of the key aims of Empirical Modelling research – to bridge the divide between traditional computer science and modern computing practice.

3.1 Brief background

Empirical Modelling as a subject matter has evolved out of many years of research at the University of Warwick. It began in 1983 with the development of ARCA, a notation for interactive Cayley diagrams (see Figure 3.1) [Bey83]. ARCA\textsuperscript{1} was the first example of what was subsequently described as a\textsuperscript{2} definitive (definition-based) notation to be developed at Warwick. ARCA led to much interest in definitive no-

\textsuperscript{1}Named after ARthur CAyley

\textsuperscript{2}Named after ARthur CAyley
tations, and motivated the subsequent development of EDEN\textsuperscript{2} in 1987 by Edward Yung [Yun87]. Experience of modelling with the tools exposed many limitations of traditional construals of computation, as they were found to be insufficient in capturing the semantic richness of the activity. This led to the introduction of principles to differentiate the work from the mainstream, and these have served to influence the development and interpretation of the tools and models to the present day. The focus on definitive notations, and particularly the concept of dependency however, has remained central. As the principles and tools developed, consideration was given to the implicit philosophy of the approach, and strong connections have been identified with the Radical Empiricism of William James in [Bey03], as discussed earlier.

Despite the top-down approach adopted in presenting Empirical Modelling in this thesis, the current understanding of EM (and the interpretation articulated in this thesis) should be regarded as a particular construal arising from engagement in the practical model building activity; a construal subject to future

\textsuperscript{2}Engine for DEfinite Notations - This forms the core of the primary research tool, Tkeden
re-interpretation and embellishment. The core ideas and principles of Empirical Modelling will now be considered in detail in relation to the Jugs model.

3.2 State-as-experienced

Perhaps the most fundamental concept in EM is that of state-as-experienced. The term was originally introduced to distinguish the nature of state in Empirical Modelling from that understood in mainstream computer science. Following traditional interpretations of computational artefacts, states are typically characterised in relation to the behaviour of a computer program, or more generally to a space of potential states. In contrast, in Empirical Modelling the primary semantic relation is the result of personal subjective experience and relates to the association between a computer-based artefact and that which it is currently about for the modeller. Abstract semantic relations considered to be an intrinsic property of a computer program such as those developed under a traditional perspective are of little concern in Empirical Modelling, where the emphasis is placed on everyday interaction which, following Heidegger, is practical, situated and operates largely without reflection (cf. [Hei62]).

State-as-experienced encompasses the holistic experience of interacting with the computer-based artefact. It is the actual, situated experience of interacting with computer-based artefacts with which Empirical Modelling is concerned. The experience of interacting with a particular artefact in any given situation will be unique. For example, the experience of interacting with a program on a computer in the department of computer science could be entirely different when using the very same program on the same machine but over a network. It can often be difficult to maintain meaningful interaction if there are problems with network connectivity, and yet under the traditional interpretation of computer programs,
the semantics of the artefact in each situation would be considered identical. State-as-experienced is best regarded as the holistic, unreflective personal experience of artefacts and the world in general, along with all its conjunctive relations.

The nature of state-as-experienced can be made apparent by practical example. Consider the photograph in Figure 3.2 which depicts two glass tumblers in a particular state. The photograph is in itself about particular tumblers and will serve to illustrate the point. Looking at the tumblers you will be able to perceive immediate associations with your personal experience of using a tumbler, and might even correctly assume as a result of your observation that the content of the tumbler is in fact orange cordial. However, you would be unlikely to realise that the wood in the background is the right hand side panel of my wardrobe, or that the tumbler on the left fell from a shelf in my kitchen soon after the photograph.
was taken and shattered into several pieces.

Equally, I will be unaware of your personal experience as you see the photograph for the first time. You will likely be reading this in a context with which I am unfamiliar, and so your experience of the photograph will be entirely different from mine as you sit in your own chair in your familiar surroundings. The example is not to imply that Empirical Modelling philosophy promotes solipsism however, as it is possible for me to point you toward elements of my own experience through communication, as has been demonstrated, and for you to point me toward your experience.

The tumblers in the photograph should be considered as in a state-to-be-experienced. The photograph in itself is inadequate in conveying that which can only be determined through observation, experiment and interaction. Furthermore, the way in which we construe potential interactions with each tumbler is determined by previous observations, experiments, and interactions.

3.3 Construal

It has already been mentioned extensively in relation to state-as-experienced, but the notion of a construal is another central concept in Empirical Modelling that merits further consideration. An EM model can be said to embody a particular construal (cf. [Goo90]). As discussed in the previous chapter, a construal is a particular understanding of a situation, “a communicable representation” developed empirically as the result of situated observation, interpretation and experiment. It is always subject to future embellishment and re-interpretation.

The Jugs model depicted in Figure 3.3 was originally developed by Meurig Beynon in 1988, but was further developed by Joanna Pavelin in 2002 (see [Pav02]). The screenshot shows two jugs in a particular state, similar to that which can be
observed in the photograph of the two tumblers (see Figure 3.2). The model embodies a simple construal of the experience of interacting with jugs ‘in the world’. The jugs can be filled, emptied and it is possible to pour from one jug to the other.

The Jugs model is particularly interesting due to its simplicity. However, it should not be regarded as a finished product. Construals are always subject to future embellishment and reinterpretation. For example, it may be useful for certain purposes to make the model more realistic. Such developments as those undertaken by Jaratsri Rungrattanaubol to allow for pixel level jug contents and the evaporation of liquid over time at a given temperature\(^3\) serve to illustrate the manner in which a construal can be embellished and re-interpreted for other purposes\(^4\).

Before considering the model itself in further detail, it is necessary to introduce three key principles of Empirical Modelling – observables, dependency and agency.

---

\(^3\)This model, and the original jugs model can be found in the EM projects archive – [http://www2.warwick.ac.uk/fac/sci/dcs/research/em/projects/](http://www2.warwick.ac.uk/fac/sci/dcs/research/em/projects/).

\(^4\)Other developments to the basic jugs model are considered in [Bey05a].
3.4 Observables, dependency and agency

Practical experience of model building has suggested that state-as-experienced can be usefully construed as comprising the following elements:

- **Observables** – something which has a value or status to which an identity can be ascribed. For example, the content and capacity of each jug are observables in the Jugs model, along with the fullness of each jug and the colour of the liquid.

- **Dependency** – a relationship between observables such that interaction with one observable leads indivisibly in our experience to a change in the other. For example, in the Jugs model there is clearly a relationship between the capacity of a jug and its current content which will indivisibly determine its fullness.

- **Agency** – an agent is projected on to the referent as something that can change the state of the model in some way by manipulating observables and the relationships between them. For example, there is a ‘filling’ agent in the jugs model that can increase the content of a jug until it is full by manipulating the content observable.

It is important to recognise how these principles differ from mainstream conceptions of computer-based artefacts and to consider how they relate to the alternative philosophical orientation discussed in the previous chapter. This alternative orientation emphasised the *intentional* character of computer-based artefacts and the need to consider these artefacts in the broader context of the world. It also stressed the importance of situated interpretation of observation in the development of understanding. The principles are consistent with this perspective, and
each relates directly to observation, interpretation and experimentation. Observables, dependencies and agents should not be reduced to variables, constraint relations and sophisticated abstractions in line with traditional interpretations of computer-based artefacts [Bey03]. To do so is to draw a dualistic distinction between the model and that which it is, at least to the modeller, about (cf. the discussion of dualistic problems in the previous chapter).

The important feature of the principles is that they have a meaning for the modeller that enables a relationship to be established between one experience and another. For example, in the Jugs model it is possible for the modeller to regard the content observable of jug A in the model as that of the glass tumbler depicted in the photograph earlier. The particular value of this in the context of human computing is that it facilitates modelling in the stream-of-thought where the modeller is primarily concerned with the practical model building activity, and not with the computer in isolation. In this mode of interaction, the model, whether as a whole or in its parts, is understood in terms of that which it is about.

### 3.5 Semantics in EM

The previous chapter established that traditional approaches to semantics in computer science are not appropriate in the context of EM. Having introduced the key Empirical Modelling principles, it is now possible to consider a more appropriate semantic framework for considering EM models. The diagram of the traditional conception of semantics in the discipline of computer science provided by Brian Cantwell Smith in [Smi96] and included in the previous chapter in Figure 2.1 can be revisited to develop a more appropriate framework for considering the semantics of EM models. The crucial addition to the diagram is the observer/modeller – the individual observing, interacting, experimenting and embellishing the model.
Following the research presented in the previous chapter, the alternative
diagram of semantics for human computing should include the following elements:

- The two types of knowing identified by William James in the Principles of Psychology [Jam90] and apparent in his discussion of the cognitive relation in *A World of Pure Experience* [JBB76], namely knowledge by acquaintance and knowledge about.

- The idea of breakdown as introduced by Heidegger and connected with computer-based artefacts by Dreyfus and Winograd and Flores.

- The notion that computer-based artefacts are intentional in character as stated explicitly in [Smi02] by Brian Cantwell Smith.

- Polanyi’s idea that “particulars can be noticed in two different ways”; in themselves or understandingly in their role as part of a comprehensive entity. [Pol61]

- The Empirical Modelling ideas that artefacts are best regarded as a source of experience and that the parts of the model themselves can be observed in relation to that which they are about.

On this basis, the diagram has been significantly revised into something more appropriate for the context of human computing (see Figure 3.4). Central to the revised diagram is the Jamesian idea that one experience can know another (cf. section 2.5.2). In his discussion of the cognitive relation in *A World of Pure Experience*, James considers the example of thinking about 'Memorial Hall', “an object not immediately there”, and explains how his ‘knowledge about’ the hall can know the experience of the hall [JBB76]:

36
Figure 3.4: An alternative semantic diagram for human computing

If I can lead you to the hall, and tell you of its history and present uses; if in its presence I feel my idea, however imperfect it may have been, to have led hither and to be now terminated; if the associates of the image and of the felt hall run parallel so that each term of the one context corresponds serially, as I walk, with an answering term of the other; why then my soul was prophetic, and my idea must be, and by common consent would be, called cognizant of reality. That percept was what I meant, for into it my idea has passed by conjunctive experiences of sameness and fulfilled intention. Nowhere is there jar, but every later moment continues and corroborates an earlier one. (pp.55–56) [JBB56]

The left side of the diagram corresponds to the two types of knowing, and the intentionality of thoughts or ideas in the original sense. When the observer or
modeller is “truly thinking of” the house, this experience can be said to know the experience of ‘direct acquaintance’ with the house.

The right side of the diagram corresponds to the intentionality of computer-based artefacts, which is considered to be of the derivative type in this thesis (cf. section 2.4.1). The experience of interacting with the artefact can be said to know the experience of direct acquaintance with the house when the modeller/observer perceives a direct correlation between the two during interaction. Following the ideas of Polanyi presented in section 2.5.3, the modeller/observer will be aware of the particulars of the artefact "subsidiarily in terms of their participation in a whole" in this mode of interaction. However, the artefact can also be ‘noticed’ or experienced in itself when the modeller/observer is focally aware of its particulars. This might occur after a breakdown in the Heideggerean sense (cf. [McC05] whereby the particulars of an artefact become suddenly apparent after an unexpected change in state). Following such breakdown, the modeller might decide to manipulate particulars of the model by modifying the underlying code, for example, to correct some aspect of the visualisation. These two modes of experiencing the artefact are comparable to Rungrattanaubol’s internal and external relations as introduced in [Run02].

Despite the distinction that is made between the modeller/observer and the artefact, subject matter and thought in the diagram, it should not be interpreted in dualistic terms. Notice that the only relationship between each of the elements is experiential in character, as denoted by the dotted arrows. In the context of human computing, computer-based artefacts should be considered in relation to the modeller/observer, and not as objects of interest in isolation. Contrary to the traditional perspective as discussed in the previous chapter, meaning is not intrinsic to the artefact. Rather, the artefact acquires meaning through the emergence of
reliable patterns of state-change during interaction, and the correlation between these state-changes and those ‘in the world’. The semantic relations are to be established in this manner, and will be personal and subjective. The extent to which experience of interacting with the model knows the experience of interacting with the referent or that which the model is about will vary between individuals.

For example, someone approaching a primitive computer-based artefact for the first time would be unlikely to know it in the same way as the modeller, for whom strong experiential relationships have been formed throughout interaction. However, that is not to say that the artefact will be entirely meaningless for someone interacting with it for the first time, as certain aspects will be given in experience (cf. [JBBS76]). For example, the relationship between the jugs in the Jugs model and jugs in the world may be immediately given in experience through conjunctive relations. The artefact presents itself as something whose meaning is to be made through observation and experiment, guided by situated interpretation (cf. [Goo90]).

There is one final point for consideration in relation to the revised semantic diagram. In human computing it is appropriate to maintain a meaningful relationship between artefact and referent even when the particulars of the artefact are noticed focally [Pol61]. In Empirical Modelling, this is facilitated by definitive scripts which capture the modeller’s construal of the referent in terms of observables, dependencies and agents which can be perceived as operating in both domains. It is useful at this point to consider the practical example of the Jugs model in further detail.
3.6 The Jugs model

The Jugs model was developed using definitive notations. The model comprises a series of definitions that collectively embody a particular construal of jugs. In line with EM principles, the model includes a number of observables and definitions that embody the modeller’s current expectations about state-change resulting from interaction. The following definitions correspond to the screenshot of the model presented in Figure 3.3:

```
contentA = 2;
capA = 5;
contentB = 5;
capB = 7;
A full is capA==contentA;
B full is capB==contentB;
```

It is important to note that the aim of engaging in Empirical Modelling is not to produce a model of how things actually and objectively are, but to explore particular construals. Whilst the primitive jugs model does not include some of the observables key to making the model a realistic reflection of jugs in the world, it does provide an experience that in a limited sense *knows* the experience of interacting with real jugs. The value of engaging with the EM Jugs model is that it provides an environment in which the jugs can be explored in an open manner.

A key difference between interaction with conventional programs and interaction in the world is that in the latter situation we enjoy comparative freedom. A traditional computer program developed under a ‘use-case’ approach would offer only a limited number of possibilities for exploration and experimentation in
relation to Jugs. However, following Heidegger, as humans are principally concerned with engaging in their everyday activities where anything may present itself opportunistically as a tool fit for a particular purpose, the ‘use-case’ mentality is inappropriate in the context of human computing. Whilst a computer program to simulate jugs use might provide functionality for pouring, filling and emptying, it would be unlikely to support less common interactions. For example, I recently used a glass tumbler and a newspaper to catch and remove a spider from the house.

There are obvious situations in which restriction is appropriate – for example it would not be advisable to offer such freedom of interaction in secure applications such as online banking. However in the context of human computing, it is precisely the possibility of open exploration and experiment that will enable the full potential of computers to be realised. It is in this respect that Empirical Modelling principles and tools provide radically different practical support from traditional approaches in computer science.

The point is further illustrated by a specific and well cited practical example in relation to the Jugs model. Whilst in a conventional jugs program it would be unlikely that a user would be granted access to manipulation of state other than that prescribed by ‘use cases’, it would be perfectly legitimate in Empirical Modelling to make the following direct redefinition of the content of jug A:

```
contentA = 10;
```

If we take the previous illustrative definitions to be the modeller’s current construal, then we would consider jug A to be full when its content is equal to its capacity. However, in this instance the content of the jug has been manipulated directly through redefinition, resulting in a visibly overfull jug. Such a state would be
considered meaningless under traditional interpretations, and not something that would even occur under the prescribed modes of interaction.

However, it is quite possible for the artefact to be considered meaningful by the modeller if for example the content of the jug is considered to be a tower of coins rather than liquid. Whether the result of a deliberate redefinition or blind variation in the sense of Vincenti [Vin90], the artefact invokes a new experience for observation, interpretation and further experiment.

Depending on the particular purposes of the modeller it may be appropriate to keep the existing definition and to introduce a new observable relating to the extent to which the jug is full, and for the ‘tower of coins’ example to be further pursued.

In Empirical Modelling, the development of the model proceeds through the redefinition and embellishment of the model on the fly, facilitated by the tkeden environment. The artefact offers itself as an experience, and its meaning is to be made through situated observation, experiment and interpretation. In this respect, model development is usefully regarded as a process of construing in the sense of Gooding in [Goo90].

For further details about the jugs model and the benefit of Empirical Modelling for this particular application, it is useful to refer to [WMBJ05] where four different constructions of the jugs model are considered and their relative merits compared.

### 3.7 Modelling with Definitive Scripts (MWDS)

The example of the jugs model has provided some demonstration of the potential benefit of Modelling with Definitive Scripts (MWDS). The concept of MWDS will now be briefly discussed.
A definitive script comprises a series of definitions of the form:

\[
\text{observable\_name is expression}
\]

Expressions can be made up of observables, functions of observables or values. Each definition associates observables/values on the right-hand-side (RHS) with the observable named on the left-hand-side (LHS) such that the current status or value of the LHS is indivisibly determined by the RHS. In this way, the definitions capture the expectations of the modeller in relation to observables on the LHS of the definitions. It is important to note that each definition relating one observable to another establishes a unidirectional dependency between the LHS and RHS such that a change in the value/status of observables on the RHS leads to an indivisible change in the value/status of the LHS. In reality of course, this \textit{indivisibility} is merely \textit{perceived indivisibility} as it takes time for the computer to re-evaluate each of the expressions following a redefinition.

The meaning of each observable is to be made through observation, interpretation and interaction, and, on account of this, the \textit{intentional} character of a definitive script is personal and subjective.

### 3.8 Summary

This chapter has introduced the key principles underlying Empirical Modelling activity. Perhaps the most fundamental of these is the notion of \textit{state-as-experienced}, introduced to distinguish the nature of \textit{state} in EM from that understood in mainstream computer science. It is the holistic, situated experience of interacting with computer-based artefacts with which Empirical Modelling is principally concerned. The notion was illustrated with reference to a photograph of two glass tumblers, where the initially subjective character of such experience was exposed.
Models in EM can be said to embody a particular construal – a particular understanding of a problem situation, "a communicable representation" developed empirically through observation, experiment and interaction, always subject to embellishment and reinterpretation in light of new experience. The notion of construal in relation to EM was illustrated with reference to the Jugs model, which has been embellished in various ways in line with particular construals of jugs (cf. [Bey05a]).

The principles of observables, dependency and agency provide the foundation for Empirical Modelling and each of these corresponds directly to an aspect of observation, experiment and interpretation in the modeller’s construal of that with which they are concerned. To reduce observables, dependencies and agents to variables, constraint relations and sophisticated abstractions respectively is to misunderstand the nature of Empirical Modelling, drawing a dualistic distinction between the model and that which it is about for the modeller.

In line with this alternative perspective, it is also necessary to reconsider the semantics or intentionality of computer-based artefacts in the context of EM and human computing more generally. The ideas of James, Heidegger and Polanyi presented in chapter 2 were used in conjunction to develop the revised semantic diagram depicted in Figure 3.4. Central to this alternative view is the Jamesian idea that one experience can know another, and each of the semantic relations illustrated can be considered in this way as experiential in character. Polanyi’s idea that particulars can be noticed in different ways is incorporated in the diagram to highlight two modes of interacting with computer-based artefacts – one where the particulars are noticed focally, and one where they are noticed “subsidiarily in terms of their participation in the whole”. A transition from the latter mode to the former can be considered a breakdown in the Heideggerean sense [McC05].
The purpose of EM activity is not to produce a model of how things actually and objectively are, but to explore particular construals. In contrast to traditional approaches to the development of computer-based artefacts such as those which proceed through the identification of ‘use cases’, EM encourages open exploration and embellishment throughout the model-building activity, embracing a more holistic view of human interaction with the world which resonates with that of Heidegger in [Hei62]. In EM, this creative exploration is facilitated by Modelling With Definitive Scripts (MWDS).

The next chapter will consider the nature of Empirical Modelling activity in further detail and discuss areas in which the principles and tools can provide particularly effective support.
Chapter 4

Empirical Modelling Activity

This chapter is concerned with the activity of Empirical Modelling and illustrates some of the areas in which Empirical Modelling principles and tools can be usefully applied. It begins by considering the nature of Empirical Modelling activity and then introduces model building exercises undertaken by the author during the course of this research. This is followed by comments on the author’s earlier work with timetabling and discussion of two projects undertaken by others that illustrate the use of EM principles and tools in developing Intelligent Agents (IA). The central theme of the chapter is to consider situations where it is appropriate to adopt an Empirical Modelling approach. It concludes that the extent to which it is appropriate to engage in Empirical Modelling activity when approaching a particular problem is dependent on the amenability of the problem situation to formalisation.

4.1 Introduction

The use of the term intentional throughout this thesis must be approached with care, particularly as Empirical Modelling activity is not necessarily intentional in
the ordinary sense (cf. section 2.4.1). The model building activity has previously been described as amethodical [Won03] i.e. the modeller proceeds without a particular method, process or outcome in mind. The modeller begins only with the motivation to undertake the modelling activity for a particular purpose, which may itself be subject to change throughout the course of the activity.

In direct contrast to traditional approaches to the development of computer-based artefacts such as that embodied in the waterfall model (cf. [Art06a]), Empirical Modelling proceeds in an unstructured manner. Furthermore, unlike traditional approaches, where it is common to circumscribe the problem domain prior to implementation, Empirical Modelling requires little by way of commitment to particular construals. EM activity is an evolutionary process whereby the artefact under construction develops with the modeller’s construal of that which the model is about, i.e. the referent.

The motivations for engaging in model building will vary, but can be generally characterised as the desire to utilise computer-based support for thinking about a particular problem. In contrast to the tradition where the emphasis is placed almost entirely on the finished product or program, in EM it is the model building activity itself that is of equal if not greater importance. In EM, there is less concern for exploring what can be (efficiently) automated in favour of understanding what cannot (cf. [BRM06]).

This thesis contends that the areas in which Empirical Modelling principles and tools can be most usefully applied is dependent on the degree to which the problem situation and aims of those wishing to utilise computer-based support are compatible with EM philosophy. More specifically, Empirical Modelling is particularly useful where the problem of concern or part thereof is not amenable to formalisation. This is where it is inappropriate to approach the problem in terms
of language, logic and rules (cf. the (analytic) tradition as discussed in chapter 2). Some areas of particular interest in this regard will now be considered in relation to practical case studies in model development by the author and others engaging in Empirical Modelling research.

4.2 Sudoku modelling

The sudoku modelling was an exercise in embodying personal understanding within a computational artefact in conjunction with skill development. The artefact itself serves as simple cognitive support for individuals as they attempt to find the solution to a sudoku puzzle. It is important to recognise from the outset that the purpose of the exercise was not to develop a computer program that could provide the solution to any given logically solvable sudoku puzzle. Rather, it was to explore ways in which the puzzle solving exercise could be usefully supported by the computer, and how the skills of the human solver could be embodied in a computer-based artefact. What follows is an account of the modelling activity that is phenomenological in spirit. It is deliberately written in the first person to convey the initially personal and subjective nature of human understanding.

4.2.1 The naive solver support

The first stage of the modelling was to develop a grid that could be used throughout the modelling process. For the purposes of this exercise, I developed a standard 9x9 sudoku grid and created agents to respond to interactions so that digits could be entered and removed from the empty squares.

At the beginning of the exercise, I had no previous experience of solving sudoku puzzles and so understood the problem of solving the puzzle only in terms...
of the instructions provided, such as those found in the Times:

Fill the grid so that every column, every row and every 3x3 box contains the digits 1 to 9. [Webb]

It was on this basis that I made the naive preliminary assumption that the puzzle would proceed through the identification of squares for which there was only one possibility based only on the digits present in the relevant column, row, and box/region. On this basis, the model was embellished to indicate the possibilities for a given square. However, I soon realised through interaction with the model that this approach was inadequate in supporting a human solver in solving the majority of sudoku puzzles in their entirety and that it would be necessary to adopt a broader approach to the puzzle solving exercise.

4.2.2 Looking at rows and columns adjacent to regions

I became aware that there were other ways in which the correct position of each digit could be determined. For example, consider the puzzle in Figure 4.1.

When looking at a particular group of adjacent regions such as those formed by the top three rows, I found it was often possible to determine the position of a particular digit by noticing that the digit was present in two of the three regions and that there was only one space in which the digit could be placed in the third region. This approach can be adopted to determining the position of the digit ‘8’ in the middle region of the top row. Notice also, that a similar pattern can be found in the bottom three regions for the digit ‘1’. I realised that a similar approach could be adopted when looking at adjacent regions formed by three consecutive columns. For example, it is possible to determine the position of the digit ‘7’ in the bottom region formed by the middle three columns.
Figure 4.1: Sudoku puzzle example – taken from the Guardian newspaper [Weba]
Following these observations, I embellished the naive solver support with further observables and agents that could ‘watch’ particular groups of regions and provide clues relating to particular digits. Interestingly, having discovered this particular approach to the puzzle solving activity, I found that it was much easier to solve sudoku puzzles than I had initially thought. It is significantly easier to notice situations in which it is appropriate to adopt this approach than to notice squares for which there is only one possibility as a result of the digits present in the relevant row, column and region, as I had included in the naive modeller support.

4.2.3 Looking at rows and columns

Up to this point, the modelling exercise had focused on the possibilities for squares in themselves, looking only at the digits present in the relevant column, row and region, and the interplay between rows and columns in vertically and horizontally adjacent regions. However, I became aware of another scenario when looking at the puzzle depicted in Figure 4.1 in more detail. I noticed that it was possible to determine the position of the digit ‘8’ in the right most column as there was only one square in the column where it is possible to place the digit. I noticed that this was also appropriate when looking at individual rows and regions. For example, it is possible to determine the location of the digit ‘3’ in the region in the top left corner by looking at the regions immediately below and to the right.

Following this I further embellished the model to include agents watching the current possibilities for empty squares in the columns, rows and regions to indicate when it is possible to place a digit in this manner.
4.2.4 Adjusting possibilities for other observations

I subsequently realised that it was necessary in certain sudoku puzzles to make multiple inferences to determine the position of particular digits, so I further embellished the model with additional agents that could notice situations in which a digit had to be in a particular column or row in a particular region. The agent developed in the early naive modeller support was redefined so that it considered these observations and adjusted its reporting of the current possibilities accordingly.

4.2.5 Reflecting on the exercise

An interesting aspect of the modelling exercise in understanding the nature of solving sudoku puzzles was to be found with further experience. At first, solving even the most simple puzzles was a cognitively demanding exercise, often requiring double checking before committing a digit to a particular square. This was made even more difficult under my initial construal of the problem solving exercise, as the possibilities for any particular square involved looking at all of the digits in that row, column and region.

However, by the end of the exercise and through much experience of puzzle solving, the problem evolved into a process of noticing and responding to particular patterns (cf. Merleau-Ponty’s phenomenological consideration of skill acquisition as interpreted by Hubert Dreyfus in [Dre02]). Rather than considering all of the digits before making a decision, it became possible to respond in certain ways to particular situations.

This is particularly interesting as it illustrates that the difficulty of a puzzle is the result of the experience of the solver and the cognitive support at their disposal, rather than it being intrinsic to the puzzle. The exercise also makes clear
Figure 4.2: The Sudoku model in action

the distinction between the puzzle in itself and the logical next step, to the solver’s construal of the puzzle and the extent to which they perceive the next possibility.

As a problem-solving aid, the model itself (see Figure 4.2) is by no means complete in the sense that it would not provide suitable clues in all scenarios that might arise\(^1\). The flexibility with which the model can be embellished by the familiar modeller enables it to be readily modified in light of new experience. For example, shortly after the modelling exercise when looking at another sudoku puzzle, a further scenario arose whereby it was necessary to determine that a pair of digits should be placed in a particular pair of cells in a row, column or region so

\(^1\)See [Wil05] for further information about solving sudoku puzzles.
that these digits could be eliminated as a possibility from the remaining squares. In this way, the modelling exercise illustrates the use of EM tools and principles for the development of understanding and the support it can provide for human thinking in, for example, the design of a computer program for solving sudoku using human solving approaches.

As the principal aim of the exercise was to explore the heuristic problem solving activity, it was particularly appropriate to adopt an Empirical Modelling approach in this instance. Whilst conventional approaches may be suitable in developing an efficient, automated sudoku solver, they are less appropriate in exploring the human problem solving exercise as it is not amenable to formalisation. By engaging in the modelling exercise I was able to develop and improve my ability to solve sudoku puzzles through interaction with the model. The observables that were introduced served as a cognitive aid in further developing my understanding of how I could approach the problem solving activity. Furthermore, having developed the artefact as an EM model, it is possible to readily explore other aspects of sudoku. For example, the model can be adapted to support those building sudoku puzzles, or to explore other forms of the sudoku puzzle such as variants involving larger or smaller grids.

### 4.3 Music Modelling

In the sudoku modelling exercise, whilst the scope of the human problem solving activity being explored evolved with skill development, the context of the sudoku puzzle was fixed and could be formally defined. Another area in which Empirical Modelling can be usefully applied is in situations where the context itself is ill-defined. In modelling music, where meanings are subject to individual interpretation, the context is difficult to determine and is dynamic throughout the exercise.
Whilst this particular study became quite specifically concerned with Schubert’s harmonic style, this was not the sole and original aim at the outset. It was necessary during the course of the modelling exercise to negotiate a stable objective for the activity and several aspects were explored in relation to particular musical compositions, becoming more or less important as areas of interest as the activity progressed.

The initial motivations for exploring the potential application of EM principles and tools to the study of music were threefold:

- To engage in a collaborative model building exercise to better understand Empirical Modelling outside of the one-modeller scenario.
- To use the tool support to develop basic harmonic analysis.
- To explore the rich relationships between the experiences of an interactive artefact, the music, and that which the music is about.

The study began with a consideration of Schubert’s *Der König in Thule*. The aim of the exercise was initially to explore ways in which, given a particular set of notes, the current chord could be determined. The initial model comprised a keyboard and a small database of basic chord triads. Clicking on the keys of the keyboard would cause redefinition and a basic query of the database to determine the current chord. As an extension to this, an additional database was created of the notes in each of the simple major and minor scales. An additional query was then created that could determine possible keys for the current notes or triads. This method of determining possible keys from the presence of each triad in major and minor scales follows Tovey’s alternative model of tonality as developed in [Tov44]. Whilst the keyboard had been originally developed as a means of simultaneously controlling and making visible the current selection of notes for testing purposes, the artefact
Figure 4.3: The simple music model

The keyboard interface acquired further meaning when the model was embellished to support the input of the music for exploring semi-automated analysis of the progression of keys throughout *Der König in Thule*. The music was input bar by bar into the model using the keyboard interface. The resulting definitions were then grouped to follow the observable phrases in the piece. A further query was then developed to look for common possible keys across the phrase. This proved largely successful for *Der König in Thule*, but the model was later found to be too simplistic to generalise.

In parallel with these developments, Meurig Beynon undertook a larger study of Schubert's *Erlköning*, a piece of particular interest due to its semantic richness. The piece is based on Goethe's poem of the same name that was inspired by the true story of a father attempting to save his sick son from imminent death by transporting him to a doctor in a distant town late at night on horseback. The semi-automated harmonic analysis as used for *Der König in Thule* was initially intended to support a similar analysis of *Erlköning*. However the relative...
complexity of the harmonic devices in the latter meant that the analysis was more effectively carried out by hand. This is particularly characteristic of the nature of the modelling exercise – the appropriate balance in the partnership between human and computer is to be established throughout interaction.

The author explored integration of the model with standard musical notation, but this became less important as the harmonic analysis became the central interest. The Der König in Thule model was then further embellished to incorporate some of the artefacts developed by Beynon. Particularly interesting was the integration with Beynon’s model of the classical cycle of twelve major/minor keys, as it highlighted the relative harmonic complexity of Erlkönig. In the Der König in Thule model, key transitions are easily described with reference to the classical cycle of keys, where the major keys are associated with relative minors, for example C major and A minor (both of which have no flats or sharps). However, as Beynon observes in [Bey06], Erlkönig exhibits a particularly characteristic feature of Schubert’s harmonic style whereby the music involves transitions between major and minor keys with the same tonic note (such as A major and A minor). This prompted Beynon to experimentally embellish the cycle of keys model into something more appropriate to capturing Schubert’s harmonic style as described in [Bey06].

EM principles and tools provide particularly useful support in exploring the harmonic structure of Der König in Thule and Erlkönig when compared with traditional programming languages. This is largely attributable to the nature of the modelling exercise, which is quite different in character to the conventional development of a computer program. As Beynon observes in [Bey05b]:

Where conventional computer programming is like specifying a composition for an established instrument, EM more closely resembles
rehearsing a composition whilst it is being written on an instrument that is in the process of being devised. Even when an EM exercise has a specific objective, aspects of the modelling typically spawn artefacts of ephemeral interest, incidental extensions of the instrument with other potential applications, and imaginative ideas about directions for further exploration. [Bey05b]

In the music modelling, it was particularly useful when ideas arose to explore particular elements in further detail, and to integrate artefacts that had been developed with completely different purposes in mind. It was not clear at the beginning of the exercise in which direction the modelling would proceed. For this reason, it was particularly appropriate to adopt an Empirical Modelling approach as both the context of the problem situation and the objectives of the exercise were unstable and to be determined and refined throughout the modelling exercise rather than established from the outset. In this way, Empirical Modelling principles and tools could provide useful support more broadly for humanities computing (cf. [BKR06]).

4.4 Timetabling

In the music modelling, the context of the problem situation was ill-defined and dynamic due to the hermeneutic nature of music. However, the context can also be dynamic in situations where it is not possible to preconceive all agency, such as in the problem of timetabling. Timetabling can be characterised as an ‘open-ended problem’, one for which the scope of the problem-solving exercise is difficult to determine due to the unpredictability of future events. Furthermore, certain variants of the timetabling problem have been shown to be NP-hard (cf. [BWM+00]), and
this motivates the use of heuristic approaches to finding a suitable solution. On this basis, it is once again appropriate to adopt an Empirical Modelling approach to the problem solving activity.

The timetabling work carried out previously by the author (cf. [Kin04b]) involved the re-development of the Temposcope (cf. [Run02]), an artefact used to support the timetabling of third year project presentations in the department of computer science. The interface of the re-developed Temposcope is depicted in Figure 4.4. The modelling of timetabling is particularly interesting as it involves consideration of conceptual observables such as availability, rather than perceptual observables like the content of jug A in the earlier example.

The problem of timetabling is simplistically one of matching resource availabilities; however, the quality of any particular timetable when compared with another is largely subjective. For example, there is often the need to consider practical concerns beyond the immediate timetabling problem such as the relative location of resources, and preferences for particular locations. As a result, it is appropriate to adopt a semi-automated approach to the activity for many timetabling problems. In this situation, the computer needs to provide support for observation and experiment in finding the ‘best’ solution, and for exploring alternative arrangements should unforeseen issues arise.

The decision to re-develop the Temposcope rather than re-engineer the existing model was largely due to the complexity of the previous implementation and its incompatibility with the author’s personal construal of the timetabling problem. Whilst the Temposcope provided some support for automated timetabling (cf. [BWM+00]), the re-implementation was more concerned with providing more effective support to the manual timetabler. A particular feature of the artefact is the support it provides to the timetabler when unexpected requirements arise.
Figure 4.4: The interface of the re-developed Temposcope [Kin04b]
For example, whilst it is standard practice for each third year project student to present in a separate timetable slot, in timetabling the presentations in 2004 an unforeseen requirement arose when a supervisor requested that two students who had been working on overlapping projects present their work together. As the two students had previously been allocated different second assessors this created a new requirement that the three academic staff involved all be available at the same time. In contrast to traditional approaches where it is often necessary to fully circumscribe the problem domain prior to implementation, it is possible to embellish the model on the fly to account for this new experience.

Adopting an EM approach to the timetabling problem supports the timetabler in embellishing their construal of the problem throughout the development of the timetable. An artefact developed using Empirical Modelling principles and tools can be embellished to support particular aspects of the timetabling activity as the timetable is under construction. For example, in the re-implemented Temposcope, an *ad hoc* interface was created to compare the availabilities of multiple resources. Rather than this being envisaged at the beginning of the activity, it was developed during the timetabling activity itself as a cognitive aid in developing a suitable timetable.

### 4.5 Developing Intelligent Agents

Empirical Modelling principles and tools can also be usefully applied in developing ‘Intelligent Agents’ (IA). In such development, it is necessary to establish both the context and the recipes for action therein. Empirical Modelling enables the heuristic development of both context and action in an holistic manner which does not require premature commitment to particular construals of either domain. The instability of both the context for action and the action itself means that
the problem is not amenable to formalisation. Approaching the development of IA under a formal, logicist approach is inappropriate as it neglects the role that experience plays in informing intelligence, as illustrated by Beynon in [Bey99].

It is more appropriate to adopt a view of intelligence and cognition such as that of Maturana, as Winograd and Flores discuss in [WF86]. Maturana stresses the situated nature of cognition and the role that observation plays in intelligent behaviour. Winograd and Flores summarise Maturana’s view in [WF86]:

Every organism is engaged in a pattern of activity that is triggered by changes in its medium, and that has the potential to change the structure of the organism (and hence to change its future behaviour).

(p.71) [WF86]

On this basis, it is appropriate for the development of IA to proceed in an evolutionary manner whereby both the context for action and the response of the agent is developed and refined with the progression of the modelling activity. The support that EM tools provide for embellishment on the fly, and the ease with which artefacts can be developed in isolation and later integrated into the model is particularly appropriate in this respect. Two recent EM studies by Keer and Stein can be used to illustrate the use of EM principles and tools for the development of IA², the interfaces of which are depicted in Figures 4.5 and 4.6 respectively.

In Keer’s Ant model, an ant embarks on a search for food from its nest, navigating with reference to observable obstacles of different colours and heights within a context created by the modeller. If successful, it then attempts to return to the nest laden with food, such as in the situation depicted in Figure 4.5. To

²Whilst the models presented in this section are not the work of the author, they are included because they are interesting examples of another area in which Empirical Modelling principles and tools can be usefully applied. For more detailed information about the models and their operation, the reader should refer to [Kee05] and [Ste05].
ensure that it can return later to collect more food, it *remembers* its route by observing the observable features of the context (the colours and relative heights of its surroundings) *en route* back to the nest. When it successfully returns to the nest, it attempts to retrace its steps to the food by aligning what it remembers about the surroundings with the current view and correcting its path as it goes. EM principles and tools provided useful support for the development of the Ant model as it enabled Keer to create new contexts on the fly for the ant to explore. What was particularly interesting about the model was that the ant displayed intelligence beyond that which was expected. Keer introduced new contexts for the ant to explore that had not been considered in its development, for example a narrow tunnel resembling a maze. The ant was able to find the food at the end of the tunnel and return to its nest despite the fact that its immediate surroundings were the same height and colour throughout its exploration.

Stein’s Car model is similar in terms of the interplay between agent and context; each car navigates around a user defined context by *observing* the observables in its immediate surroundings, and adjusting its speed and direction appropriately to avoid collision and to stay on the appropriate side of the road. EM principles and tools supported Stein in automating the behaviour of each car after extensive manual experimentation. The legacy of Stein’s manual interaction remains apparent in the model. Those interacting with it are able to take control of the cars at any point during the interaction, and to subsequently return the responsibility to the computer. This blend of automatic, semi-automatic and fully manual development is particularly characteristic of EM, and illustrates its potential for human computing in exploring what can’t be automated in the spirit of [BRM06].

In both models it is possible to create and refine the contexts on the fly, and to modify the agent’s response to particular scenarios through redefinition. In
this manner it is possible to incrementally embellish the intelligence of the agent through observation and experiment. By defining particular contexts and observing the agent’s response it is possible to experimentally develop its intelligence through redefinition. The modeller is free to experiment with different definitions until the agent responds appropriately to the context, and can then move on to explore further scenarios. Using EM tool support it is possible to explore particular aspects in isolation within the model itself. When the modeller is comfortable with particular definitions, they can be integrated into the model using dependency.

The particular value of adopting an Empirical Modelling approach in developing IA is that it provides effective support for the exploratory development of both context and agent. As both are to be stabilised throughout the modelling exercise, neither the context or the agent is amenable to formalisation. For example, the modeller may stumble upon a particular scenario through blind variation (cf. [Vin90]), and realise the agent’s limited intelligence only when it responds in an unexpected manner. Importantly, the role of experience in developing intelligence
should be recognised in the development of IA.

4.6 Summary

The central theme of this chapter has been to illustrate areas in which Empirical Modelling principles and tools can be usefully applied. Four problem situations were considered with reference to their amenability to formalisation – Sudoku, music, timetabling and Intelligent Agents (IA). In each instance, the problem situation or part thereof was not amenable to formalisation and it was argued that it was appropriate to use Empirical Modelling principles and tools.

In the Sudoku modelling, EM supported skill development by providing an environment in which observables could be readily introduced to assist the human solver. In the music modelling, EM provided a means of exploring various areas of interest in relation to Schubert’s *Erlkönig* and *Der König in Thule*, without premature commitment to particular perspectives or research directions. The
timetabling modelling demonstrated that EM principles and tools can also be useful in application areas where it is difficult to preconceive all agency and where unforeseen requirements might arise at any time during the modelling activity. Finally in the development of IA, EM provided effective support for the exploratory development of both context and agent, supporting the modeller in experimenting with particular scenarios.

The next chapter considers the principal tool used to support Empirical Modelling activity, evaluates its usability and considers the author’s work in exploring alternative ways of model building with the tool.
Chapter 5

Empirical Modelling Tools

This chapter considers tkeden, the primary tool used in Empirical Modelling research. Whilst tkeden does provide a suitable environment for supporting Empirical Modelling activity through close mapping with the key principles, there are several usability barriers to be addressed in realising the aspiration of supporting modelling in the stream-of-thought. A significant amount of work has been carried out on developing the EDEN core; however, prior to this research, relatively little work had been undertaken in developing the tkeden interface and in considering alternative approaches to modelling with the tool. After a brief discussion of the background to the tkeden tool and consideration of its current usability limitations against the cognitive dimensions framework of Green and Petre [GP96], the author’s work in exploring alternative interfaces and ways of modelling is presented.

5.1 Brief background to tkeden

Tkeden has been the principal Empirical Modelling tool for a number of years. Its development dates back to 1987 when Edward Yung developed the first version
of EDEN, the Evaluator of DEminutive Notations\textsuperscript{1}. The advent of EDEN enabled the development of a number of definitive notations, the earliest of which communicated with EDEN through UNIX pipelines as discussed by Ashley Ward in [War04]. Two of these early notations, namely DoNaLD (the definitive notation for line drawing originally implemented by Edward Yung) and SCOUT (a definitive notation for describing SCreen layOUT developed by Simon Yung) were eventually integrated into the first version of tkeden in 1996 and largely replaced Edward’s terminal-based implementation which later became known as ttyeden. The ‘tk’ part of the name ‘tkeden’ refers to its use of the Tcl/Tk toolkit in all of the graphical components of the tool (cf. [BBW03]).

The current version of tkeden is 1.66 and its interface is depicted in Figure 5.1. The tkeden session includes by default the eden, DoNaLD and SCOUT notations, as well as sasami, a definition-based notation for 3D modelling, and aop, an agent oriented parser notation which enables the development of additional notations.

It is beyond the scope of this thesis to consider the historical development

\textsuperscript{1}Further details about the original version of EDEN can be found in Edward Yung’s final year report, [Yun87].
or technical operation of tkeden at length. For this information, the reader should refer to Ashley Ward’s PhD thesis [War04]. Is is however useful at this point to introduce some of the core notations to provide the necessary background for the discussion of the author’s work that follows.

## 5.2 Core Notations

### 5.2.1 eden

The eden notation combines definitive and procedural elements to provide a non-traditional general purpose language for the development of computer-based artefacts. It is syntactically similar to the C programming language. There are several features of eden that serve to provide more appropriate support for Empirical Modelling activity. These include the following:

- In the C programming language, memory allocation and management is the responsibility of the programmer. In eden, this is controlled automatically, enabling the modeller to concentrate on the activity rather than the tool.

- eden includes a generalisation of the dependency concept that is a key feature of spreadsheet applications. For example, in the context of a spreadsheet it may be necessary to calculate the average of a column of numbers, and to do this a simple formula could be entered into a particular cell to establish a direct link with the contents of the cells in the column. A change to any of the numbers in the column would result in a direct change in the cell displaying the column average. In much the same way, in eden it is possible to introduce definitions that create relationships between observables such that a change in one observable leads to a direct change in another. The
observables can be integers, characters, floating point numbers, strings or lists. There is also a special undefined type denoted by the ‘@’ symbol. This is particularly useful in the modelling activity as observables can be introduced in definitions without being previously defined.

- The eden language provides procedural elements that enable the automation of agency within the model as a direct response to changes of state in particular observables. These procedural agents can be linked to particular observables such a way that they are triggered to act whenever the observables change.

- It is possible to create additional notations on the fly using the agent oriented parser (aop) notation.

The following definitions from the jugs model considered earlier serve as a simple example of a family of definitions in eden:

```plaintext
/* Observables */
contentA=0;
capA=5;
contentB=0;
capB=7;

/* Dependencies */
Afull is capA==contentA;
Bfull is capB==contentB;

/* Agent triggered by button pressing */
proc wmenu1_button : wmenu1_mouse_1 {
    if (wmenu1_mouse_1[2]==4)
```
To develop interface components for the models it is necessary to use the DoNaLD and SCOUT notations. The example which follows is intended to illustrate basic definitions in the DoNaLD and SCOUT notations, necessary background for the evaluation of tkeden's usability and the discussion of the author's work in exploring alternative interfaces that follows.

### 5.2.2 DoNaLD

The DoNaLD notation was originally implemented by Edward Yung following the development of EDEN. It itself, it is a purely definition-based notation for line drawing and as Ashley Ward explains in [War04], “agency is limited to that of the modeller” and so DoNaLD provides “a 1-agent modelling environment”.

The following example illustrates the use of the DoNaLD notation to define a simple Guernsey flag\(^2\), as depicted in Figure 5.2:

```plaintext
%donald

## declare the flag and positional observables
rectangle flag
point bottomleftcorner, toprightcorner
int botleftcorner.X, botleftcorner.Y
int toprightcorner.X, toprightcorner.Y
int dimX, dimY
real midX, midY
botleftcorner.X=100
botleftcorner.Y=350
```

\(^2\)Note that the gold cross has been deliberately simplified for the purpose of this example.
Figure 5.2: The Guernsey flag as defined in DoNaLD
toprightcorner\_X=900
topleftcorner\_Y=800
dimX=toprightcorner\_X-bopleftcorner\_X
dimY=toprightcorner\_Y-bopleftcorner\_Y
midX=0.5*dimX+bopleftcorner\_X
midY=0.5*dimY+bopleftcorner\_Y
bopleftcorner={bopleftcorner\_X,bopleftcorner\_Y}
topleftcorner={toprightcorner\_X,toprightcorner\_Y}

## define the flag
flag=rectangle(bopleftcorner,toprightcorner)
?A\_flag=\"fill=solid, color=white\";

## declare lines for the cross of St George
line georgevert, georgehoriz

## define the lines and their attributes
georgevert=[[midX,bopleftcorner\_Y],
{midX,toprightcorner\_Y}]
georgehoriz=[[bopleftcorner\_X, midY],
{toprightcorner\_X, midY}]
?A\_georgevert=\"fill=solid, color=red, linewidth=40\";
?A\_georgehoriz=\"fill=solid, color=red, linewidth=40\";

## declare and define the yellow cross
line vert, horiz
int gapvert, gaphoriz
gapvert=40
The example is intended to highlight particular aspects of the DoNaLD notation useful for the evaluation of eden that follows, rather than to provide a comprehensive illustration of its capabilities. Unlike the eden notation, where the use of semicolon terminators is required at the end of each definition, notice that line breaks are used in DoNaLD. The only instance in which the semicolon terminator is used in DoNaLD is when attributes are defined, such as the colour and line width of the vertical and horizontal lines forming the cross of St George. This is because the attributes are defined in eden. The ‘?’ symbol at the beginning of each definition denotes an escape to the eden notation. In comparing the example DoNaLD script with the earlier eden example, the reader may also notice that the type of each observable must be defined explicitly in DoNaLD prior to use. In eden, this is unnecessary.

Whilst the DoNaLD notation is effective in supporting the development of line drawing models, it provides limited support for developing an interface through which to interact with the model. For this, it is necessary to use the SCOUT notation.
5.2.3 SCOUT

The SCOUT notation was developed by Simon Yung and incorporated into tkeden as the primary language for defining screen layout. The notation supports the creation of text, textbox, picture and DoNaLD windows which all utilise the Tcl/Tk canvas widget\(^3\). Like DoNaLD, it is a definitive notation, developed in line with the concept of MWDS (cf. Chapter 3).

To illustrate the SCOUT notation, consider the development of a basic interface for the Guernsey flag developed previously in DoNaLD (see Figure 5.3). The SCOUT definitions used to produce this interface are as follows:

\begin{verbatim}
%scout
window flag = {

\end{verbatim}

\(^3\)For more information about Tcl/Tk widgets, the reader should refer to [BBW03].
Notice that like DoNaLD, SCOUT requires type declarations. However, notice also that SCOUT requires the use of semicolon terminators like eden.

In order to make the model interactive, agency can then be added in eden through the use of a procedure that is triggered by mouse clicks on the sensitive button window. The following agent definitions hide the yellow cross when the mouse button is pressed down to reveal the England flag. The Guernsey flag is
revealed again when the button is released:

```eden
proc buttonpress : button_mouse_1 {
    if (button_mouse_1[2]==4)
        clicked = 1;
    if (button_mouse_1[2]==5)
        clicked = 0;
}

proc hidecross : clicked {
    if (clicked == 1)
        execute("A_vert="color=red"");
        A_horiz="color=red";"");
    if (clicked == 0)
        execute(""
            A_vert="linewidth=10, color=yellow";
            A_horiz="linewidth=10, color=yellow";"");
}
```

**The importance of context**

Whilst the principal aim of the Guernsey flag example was to illustrate key aspects of the *eden*, DoNaLD and SCOUT notations prior to the usability evaluation of tkeden that follows, it can also be used to highlight the role of context in the semantic consideration of computer-based artefacts. It has been a central theme of the thesis to illustrate that the significant semantics of a computer-based artefact should be considered as that hermeneutically established by those interacting with
it. However, the importance of the context in which this interpretation occurs (cf. Heidegger’s discussion of signs in [Hei62]) has yet to be explicitly considered.

In the absence of further context, the fact that the simple artefact depicts the Guernsey flag will only be apparent to those familiar with it, though its similarity to the English flag may be recognised from the outset, or through interaction when the yellow cross is removed. However, the model acquires further meaning when considered in relation to the author, who was born in Guernsey but now resides in England. The artefact can also be interpreted in the context of its historical significance – the two states of the model illustrate the flag before and after 1985 when the gold cross, an early Norman symbol, was introduced to give a unique identity to the island in respect of the time when Guernsey was part of Normandy [Web04].

A significant limitation of the traditional approach to semantics in the discipline of computer science is that context is largely ignored. Whilst the operational behaviour of an artefact in two different situations can be considered identical, independent of interpretation and context, this does not account for the meaning of the artefact for the modeller or those interacting with it. Whilst the Guernsey flag artefact is too simple to serve any further practical purpose, the example does illustrate the context-dependent nature of interpretation in relation to the meaning of computer-based artefacts.

5.2.4 Other notations

The development of the agent oriented parser (AOP), first by Chris Brown [Bro01] and later by Antony Harfield [Har03] has provided the possibility of developing additional notations within the tkeden environment on the fly. Several auxiliary notations have been developed using the AOP, and now that it is part of the tkeden
environment it is likely that more will be produced.

Notations that have been developed using the agent oriented parser include Beynon’s eddi, a definitive database notation, Care’s physical, a notation for physical quantities, and the aop notation itself, also developed by Care to support the development of additional notations.

5.3 Evaluating tkeden

Experience suggests that definitive notations provide good support for Empirical Modelling activity. However, there is a distinction to be made between the support that the tool provides for EM activity and its usability as a modelling environment. The latter will now be considered with reference to the ‘cognitive dimensions framework’, developed by Green and Petre (cf. [GP96]).

5.3.1 Cognitive Dimensions

The cognitive dimensions framework is described by Green and Petre as “a broad-brush evaluation technique for interactive devices and non-interactive notations”. It comprises the following dimensions:

- Abstraction gradient – What are the minimum and maximum levels of abstraction? Can fragments be encapsulated?
- Closeness of mapping – What ‘programming games’ need to be learned?
- Consistency – When some of the language has been learnt, how much of the rest can be inferred?

The dimension list and descriptions have been extracted directly from [GP96] (p.9). The reader should refer to [GP96] for a more detailed explanation of the dimensions.
• Diffuseness – How many symbols or graphic entities are required to express a meaning?

• Error-proneness – Does the design of the notation induce ‘careless’ mistakes.

• Hard mental operations – Are there places where the user needs to resort to fingers or pencilled annotation to keep track of what’s happening?

• Hidden dependencies – Is every dependency overtly indicated in both directions? Is the indication perceptual or only symbolic?

• Premature commitment – Do programmers have to make decisions before they have the information they need?

• Progressive evaluation – Can a partially complete program be executed to obtain feedback on “how am I doing”?

• Role-expressiveness – Can the reader see how each component of a program relates to the whole?

• Secondary notation – Can programmers use layout, colour, or other cues to convey extra meaning, above and beyond the ‘official’ semantics of the language?

• Viscosity – How much effort is required to perform a single change?

• Visibility – Is every part of the code simultaneously visible (assuming a large enough display), or is it at least possible to juxtapose any two parts side-by-side at will? If the code is dispersed, is it at least possible to know in what order to read it?

The cognitive dimensions provide a useful framework to consider ways in which the usability of tkeden could be improved. It is important to note however, as
Green and Petre are keen to stress in [GP96], that there are certain trade-offs between the dimensions. In addressing a usability issue highlighted by a particular dimension, it is likely that another will be affected. Following this, addressing concerns in particular dimensions is likely to result in usability issues in other dimensions. With this in mind, the usability of tkeden as a modelling environment will now be considered.

5.3.2 Abstraction gradient

Tkeden provides limited support for ‘abstractions’ where elements can be grouped and treated as a single entity, and much of the interaction is carried out through the redefinition of relevant observables. However, this does present several advantages in keeping with EM philosophy:

- The low-level nature of interactions carried out with tkeden facilitates flexible control over the artefact under development.

- Conjunctive relations between the experience of interacting with observables in the world and in the model are easily formed.

The trade-off however is that the low-level nature of the interaction can make it difficult to create large scale models as there is little support for grouping particular elements that can be treated as a whole.

5.3.3 Closeness of mapping

The dimension of the ‘closeness of mapping between the problem world and the program world’ is open to re-interpretation in line with the arguments of William James. Following James, it was argued that, contrary to the traditional dualistic perspective, the relationship between the artefact and its referent, and so the
mapping between the problem world and program world, is experienced in a single relation (cf. Figure 3.4).

Tkeden provides good support for Empirical Modelling activity whereby artefacts are open to embellishment and redefinition on the fly. This was illustrated by the earlier example of changing the content observable in the Jugs model such that it exceeded the capacity, an interaction not preconceived by the modeller in the original construal. Simple definitions in eden embody relationships that can be construed as dependencies in the world. Furthermore, individual notations are often domain specific, providing good mapping between the definitive script and the world.

However, the eden language itself contains much syntactic legacy from C. Whilst some of these elements act as secondary notation, for example the use of curly braces, which aid the modeller in distinguishing between sections of script, others, such as the use of semicolon terminators, are largely superfluous, as illustrated by their exclusion from the DoNaLD notation. This leads to difficulties when developing complicated models that require elements of automatic agency, as a significant amount of syntactic clutter is required.

5.3.4 Consistency

As Green and Petre observe in [GP96], the consistency dimension is difficult to define. They propose that the significance of the dimension is best considered with reference to the following question:

When a person knows some of the language structure, how much of the rest can be successfully guessed? (p.18) [GP96]

It is the author’s view, from personal experience of engaging with definitive notations, that tkeden should be considered consistent when consistency is defined
in this manner. Whilst the notations themselves are syntactically varied, it has been found possible to correctly guess unknown aspects of a notation with limited knowledge. This is largely due to the fact that the notations are consistent with the concept of definitive notations.

Consistency differences at the syntactic level between notations are largely unproblematic in themselves; as already mentioned, notations are often domain specific and so their syntax is developed with the aim of supporting good mapping. However, there are certain inconsistencies that do prove problematic. One example is that whilst the origin of DoNaLD windows is situated at the bottom left corner of the window, with the y-axis extending vertically and the x-axis horizontally, the origin of SCOUT windows is located at the top left corner. During the course of model building this led to particular problems in interpreting mouse interaction with DoNaLD windows, as it had been expected that it would be consistent with SCOUT in this respect.

5.3.5 Diffuseness/Terseness

It has been demonstrated that the use of dependency in tkeden can have a significant effect on script terseness when compared with traditional procedural programming languages, where it is often necessary to implement dependency maintenance directly for particular variables. However, where complex agents are introduced that carry out redefinitions in notations other than eden, terseness, as measured by the number of characters required, can be comparable to traditional languages. On this basis, the usability of tkeden against this dimension can be considered as good if not better than traditional programming languages.
5.3.6 Error proneness

In considering the dimension of error proneness in relation to tkeden a number of trade-offs are apparent. Whilst the development of domain specific notations often results in good mapping and useful support for specific practical purposes, syntactical inconsistencies between the notations often lead to ‘careless mistakes’. A common example is that whilst eden requires the use of semicolon terminators, using them in DoNaLD results in error messages, unless the definition involves an escape to eden as denoted by the prefix ‘?’. Another example is that one of the common character combinations used to denote comments in eden, of the form /* comment *//, cannot be used in DoNaLD.

As tkeden provides immediate feedback in regard to errors in syntax, it is arguable that these issues are only problematic to a limited extent. However, inconsistencies in syntax do result in unnecessary errors, and this presents a barrier to those engaging with the tools, particularly those using the tools for the first time.

5.3.7 Hard mental operations

Tkeden provides good support for reducing the extent to which the modeller must engage in hard mental operations. When faced with a difficult problem, the modeller is encouraged to use the tool to support the development of an appropriate solution through observation and experiment. Redefinitions can be introduced on the fly, and the modeller can concentrate on particular aspects of the model in isolation before they are integrated with the model. Rather than thinking about the problem on paper, the modeller is able to find an appropriate solution through a series of small interactions. In this way, the cognitive burden on the modeller is significantly reduced, as illustrated by Beynon in [Bey01] with reference to the
development of the Clock model.

As Beynon observes in [Bey01], using tkeden in this way also enables the modeller to record the history of the problem solving activity:

The end result is also much more satisfactory – not only do I derive the correct answer, but I construct an environment in which my mistakes and misconceptions have been captured and recorded to an extent that is otherwise problematic. (p.15) [Bey01]

5.3.8 Hidden dependencies

In [GP96], Petre and Green use the example of the spreadsheet to illustrate the dimension of hidden dependencies:

Today the classic example is the spreadsheet. The formula in a cell tells which other cells it takes its value from, but does not tell which other cells take their value from it. In the worst case the whole sheet must be scrutinised cell by cell before one can be certain that it is safe to change a cell. (p.24) [GP96]

Relationships in tkeden are stored in the symbol table and are traceable in their entirety. When the modeller ‘observes’ an observable, tkeden returns its current definition which reveals the names of those observables that it “takes its value from”. However, it also informs the modeller about the observables that “take their value from it” and shows any agents that are triggered by changes in the observable. This would lead to the conclusion that the performance of tkeden against this dimension is particularly good when compared with traditional procedural programming languages such as C.

That said, the dependency relationships in tkeden are only visible to a limited extent. The environment discloses only the immediate relationships for a
particular observable. This is unproblematic for small models where it is readily possible to trace changes in state. However, for larger models, where there are often several related observables it can become difficult to ascertain problematic definitions when unexpected changes arise, particularly when engaging with models that have been developed by someone else.

5.3.9 Premature commitment

Tkeden is particularly effective in eliminating the need to look ahead during the modelling process. Definitions can be introduced and redefined on the fly. Observables can also take the value undefined, denoted by the '@' symbol, which enables them to be introduced without premature commitment to assigning values.

5.3.10 Progressive evaluation

Tkeden also provides excellent support for progressive evaluation. The effect of interaction can be readily ascertained through the inspection of relevant observables, as the dependency maintainer responds to the changes to ensure that state is consistent with the definitions introduced by the modeller. This supports Empirical Modelling activity and the aspiration of modelling in the stream-of-thought by eliminating the need to engage in the traditional procedural cycle of coding, compiling and running the program.

The environment also enables the modeller to develop particular aspects of the artefact in isolation before they are integrated with the observables in the model (cf. subsection 5.3.7). For example, the modeller might develop a function with the aid of ‘test’ definitions before it is integrated with the rest of the model. Tkeden stores the history of interaction in the form of the definitions introduced by the modeller and so he/she is also able to evaluate the steps taken in arriving
5.3.11 Role expressiveness

The extent to which it is possible to understand the role of particular observables in a modeller’s construal is debatable, as it is largely dependent on the following factors:

- The use of ‘meaningful’ observable names and secondary notation by the modeller.
- The previous experience of the individual interpreting the model.
- The similarity between the modeller’s construal of the referent and that of the individual interacting with the model.

Furthermore, it is particularly necessary when considering this dimension to distinguish between the script in itself, and the interactive artefact. Following the Empirical Modelling Weltanschauung, the meaning of particular observables and the artefact as a whole is something to be established through observation and interaction. This is an area for further consideration, particularly in regard to collaborative modelling which is briefly discussed in the final chapter.

5.3.12 Secondary notation

Extensive use can be made of secondary notation in the tkeden environment. Tkeden supports the use of comments, indentation, and it is possible to create additional descriptive observables as an aid to the intelligibility of the model. However, with the exception of the last example of using observables, secondary notation is lost when definitions are accepted in the tkeden input window, as only the definitions are stored in the symbol table. Again, this is particularly problematic when
interacting with models developed by other people, as the secondary notation is often useful in understanding the modeller’s construal.

### 5.3.13 Viscosity

Dependency maintenance in tkeden can be used as a means of reducing viscosity – the amount of work that the modeller has to do to invoke a small change in state. With experience, a modeller can become particularly adept at introducing definitions that enable flexible access to all of the observables in the model. Those familiar with a particular model and the notations with which it has been developed should be able to manipulate the model with relative ease on the fly.

### 5.3.14 Visibility

Somewhat ironically, given the observation-based orientation promoted by Empirical Modelling, one of the fundamental usability issues of tkeden relates to the visibility it provides to the modeller. For example, if the redefinition of a particular observable results in an unexpected change in state, breakdown may occur, whereby the model itself becomes the focus of attention. In a large model such as the Temposcope (cf. [Run02]), where there are many agents and dependencies, several steps may be required to make the relevant definitions observable to inspect the problem.

### 5.4 Modelling with tkeden: new directions

Whilst significant development work has been carried out on the EDEN core in recent years, for example in porting it to the linux, Windows and Mac OS X platforms from Solaris, developing a distributed version and various feature de-
developments and bug fixes, the current interface and mode of interaction is much the same as that of early versions of tkeden. This section describes extensions to tkeden that were conceived and prototyped by the author to explore alternative ways of modelling with the tools. Consideration is also given to the ways in which the developments address some of the usability concerns identified in the previous section.

5.4.1 drawScout

The original implementation of drawScout was developed by the author in 2004 and is described in [Kin04b]. The drawScout tool enabled drag-and-drop creation of SCOUT windows for the first time, and served as a convenient means of displaying particular observables throughout interaction. The original implementation was limited, and only supported the creation of text windows. The original interface of the drawScout tool is depicted in Figure 5.4.

The drawingBoard screen contains a single SCOUT window sensitive to mouse actions. Click and release actions over the window trigger a procedural agent that interprets the interaction, and, based on the current mode of interaction (one of create, manipulate or use), creates or redefines observables and window definitions as appropriate.

The development was originally motivated by the issues of visibility considered in the previous section, referred to in [Kin04b] as the “observability” problems. However, other applications became apparent once the prototype version had been developed. For example, it is possible to create ad hoc interfaces with the tool throughout development activity.

The application of drawScout as a tool to aid model intelligibility was also demonstrated in [Kin04b] in relation to Sisyphus, a decision support model for
Figure 5.4: drawScout version 1.0 with two created windows
elevator configuration⁵ (cf. Figure 5.5). The key problem with the model was the fact that its existing interface conveyed very little to those interacting with it. As the author explained in [Kin04b], this placed significant demands on those using the model:

The interface to the sisyphus model is highly reticent. It provides the user with a list of options or fixes, enabling them to make blind changes to the configuration of an elevator in an attempt to satisfy pre-defined constraints. Once a satisfactory configuration has been achieved, sisyphus announces this fact, displaying “All Constraints Satisfied” on the screen. It is not easy for a user to observe the result of changes without asking sisyphus for the information, and without having quite an in-depth knowledge of how all the variables are related. How is the general user meant to know which variables have changed as the result of a fix? (p.50) [Kin04b]

By creating a number of SCOUT windows with the drawScout tool and creating a dependency between the text displayed and the current value or status of observables of particular interest, the intelligibility of the model was improved as the effects of interaction become immediately visible. As an extension, additional definitions were introduced to change the background colour of windows containing observables when their value changed as the result of interaction (see Figure 5.5). Automating the definition of windows, the drawScout tool reduces the cognitive burden placed on the modeller by removing the need to remember particular SCOUT attributes and the need to declare observables before using them. It also reduces error proneness resulting from syntactic differences between SCOUT and the

⁵This model was developed by Paul Ness, Mike Joy and Simon Yung. It can be found in the EM projects archive.
5.4.2 drawSlide

The first of the extensions was to develop the capability for using drawScout as a presentation tool. It is often necessary to demonstrate EM models in formal presentations at conferences and workshops. However, the support for developing slideshow presentations prior to this research was limited.

As the original implementation of drawScout was based on a single screen,
the development involved adapting the core window management functions to allow for multiple pages. In drawSlide, each window became recognised as belonging to the page that was current when the window was created. Buttons were added to the modeSwitcher window to enable basic transition between windows to occur through redefinition. The resulting tool and its potential was demonstrated in a group seminar in Autumn 2004, at which the screenshot in Figure 5.6 was taken.

Figure 5.6: The first drawSlide presentation

The drawSlide tool is intended as an alternative to Microsoft PowerPoint as a support for presenting Empirical Modelling. When compared with PowerPoint, described by Edward Tufte as “a competent slide manager and projector for low-resolution materials” [Tuf04], drawSlide should be considered particularly suitable for demonstrating EM. As discussed in [Kin04a], slides developed using drawSlide can be considerably more dynamic than those of a standard PowerPoint presentation. Models can be integrated into the presentation using dependency,
and can be manipulated during the presentation on the fly. Additional slides and demonstrations can also be introduced during the presentation. The drawSlide tool was later integrated with the tkeden interface to create the edenSlide utility.

5.4.3 edenSlide

The limited work on the tkeden interface since the original implementation prompted a re-evaluation of its suitability in supporting Empirical Modelling activity. The tkeden input window, as depicted in Figure 5.1, is defined in a single edenio.tcl file in the library of each version of the tool. Research into the capabilities of the Tcl/Tk toolkit, in which the interface had been constructed, led to some initial experimentation into alternative interfaces. During this experimentation, it was realised that SCOUT screens could be integrated into the main tkeden input window interface with only minor engineering.

A previous disadvantage of drawSlide was that it comprised two windows, namely the drawingBoard and the modeSwitcher, in addition to the tkeden interface. Whilst this was adequate for the limited ad hoc interaction intended by the original implementation of drawScout, it was a cumbersome format for developing and giving presentations. However, the technical investigation into the tkeden input window interface enabled the integration of drawSlide with the tkeden input window to form the edenSlide environment.

The edenSlide environment is depicted in Figure 5.7. It comprises a reduced input window (a single line at the bottom), a large drawSlide screen, and the buttons from the original modeSwitcher, redesigned as Tk widgets. Whilst the functionality of edenSlide remains the same as the original drawSlide implementation, the interface changes have resulted in a greatly improved presentation environment. This demonstrates the critical role play by the interface in distin-
guishing one application from another. It is a key aspiration of EM to dissolve the
duality between the field of Human Computer Interaction (HCI) and mainstream
computer science in considering the semantics of computer-based artefacts. To
enable meaningful interaction it is useful to consider human, computer and inter-
face as they are all essential in the semantics of computer-based artefacts. The
edenSlide tool was used to support the presentation of a short paper [Kin04a] at

5.4.4 Visual Symbol Table (VST)

In addition to improving the existing support for constructing presentations in
tkeden, the development of the edenSlide environment highlighted the possibility
of further addressing tkeden visibility issues through the integration of drawScout
and the EDEN symboltable. The motivation was to create an interface whereby
user-defined definitions belonging to the model under construction could be made automatically visible in drawScout windows within the interface.

The initial implementation (the interface of which is depicted in Figure 5.8) provides a means of making model definitions visible. A series of 'definition windows' can be created by the modeller to display the current definition of a particular observable. When an observable is redefined, the VST updates the appropriate window such that it always displays the current definition. The key problem with the original implementation was that the definition windows had to be created by the modeller through definition. Furthermore, triggering the creation of VST windows following the introduction of new definitions proved problematic, as there was no way to create an observable that indicated updates to the symboltable. The following definition was intended to enable the update:

```
symboltablesize is symboltable()#;
```

However, it was found unsuitable as the definition of the symboltable built-in function (symboltable()) is not updated with redefinition. To overcome this problem most effectively would require development to the EDEN core and for
tkeden to be recompiled. However, as the purpose of the exercise was largely to explore the possibilities for alternative ways of model building, a notation was developed in Care’s aop notation to overcome the problem for the prototype: vst. Making eden definitions in vst results not only in the redefinition being made in EDEN, but also triggers the creation of a new VST window for new observables or agents.

The prototype was tested using some of the key definitions from Care’s Planimeter model (cf. [Car04]). It was then adapted by Care for use as a support in explaining the planimeter model at [BRC+05]. The adaptation utilised the core VST functionality, but separated the VST screen from the input window for the purposes of demonstration. It was also modified to use different colours to highlight the distinction between assignment (‘=’) and dependency (‘is’), and so that clicking on a definition window would clear the input window before pasting the definition contained. The demonstration is depicted in Figure 5.9.

Further development work to the VST was considered but deemed beyond the scope of this research due to its added complexity. It is considered in the final chapter of this thesis as a potential direction for future research.

### 5.4.5 Imagine Eden prototype

The ‘Imagine Eden’ prototype was motivated by the work undertaken by Chris Roe in exploring the potential of introducing the dependency concept in Logotron’s educational software package Imagine Logo, as demonstrated by Roe in [Roe05]. Imagine Logo is an event driven programming environment based on the Logo language as originally developed by Wally Feurzeig and Seymour Papert in 1967. The prototype built on the drawSlide implementation to support event driven modelling in tkeden. It is depicted in Figure 5.10.
Figure 5.9: Use of VST in demonstrating Care's planimeter model
The significant developments to the drawSlide tool were the introduction of attribute boxes to display the current definition of a particular SCOUT window in terms of its observable attributes, and the provision of text boxes in which actions in response to click, release and key press actions could be introduced in the form of definitions. In [Roe05], Roe demonstrated the use of Imagine Logo to create a simple game of Pong. This motivated investigation into whether the game of Pong could be easily developed in the Imagine Eden prototype. It is important to note that the example is not intended as further illustration of Empirical Modelling activity. Rather, it is to illustrate an alternative, arguably easier way of interacting with the tools to produce a computer-based artefact.
Developing the Pong game

The game of Pong was first developed by Atari and released in November 1972 [art06b]. Based loosely on the game of table tennis, it features two paddles at either end of the screen that are controlled by two players along a vertical plane, and a ball that moves in a continuous straight-line trajectory changing direction when coming into contact with the screen edges or the paddles. When the ball touches either the far left or right side of the screen, the player at the other end of the screen receives a point.

Developing the Pong game in tkeden using only definitions is a non-trivial exercise for the novice, requiring knowledge of, in the following example, three notations as well as basic concepts of modelling with definitive scripts. However, the Imagine Eden environment greatly simplifies the process of model building, requiring relatively few interactions and definitions:

- **Step 1** – Select the Create mode and drag-and-drop to create two SCOUT text windows for the paddles. Select the DONALD option and drag-and-drop to create a SCOUT DoNaLD window. (See Figure 5.11)

- **Step 2** – Resize the first window to an appropriate size and introduce the following definitions in the text input window to the right of the attribute boxes:

```plaintext
/* definitions for sorting out the bats */
DS_win2.X2=DS_win2.X1+(DS_win1.X2–DS_win1.X1);

DS_win1bgcolor is "red";
DS_win1string="";
DS_win2bgcolor is "blue";
```

100
Figure 5.11: Step 1 – creating the paddles and window for the ball

```
DS_win2_string="";
```

- **Step 3** – Clear the input window and enter the following definitions to define the ball:

```
/* Definitions for the ball */

%donald
circle ball
ball=circle({490,520},480)
?A_ball="fill=solid , color=yellow";
%eden

DS_win3_bgcolour is DS_drawingboard_bgcolour;
DS_win3_bdcolour is DS_win3_bgcolour;
```
A large yellow circle will appear in the DoNaLD screen. This window can be closed.

- **Step 4** – Introduce the following redefinition for the DoNaLD SCOUT window created in step 1:

```plaintext
%scout

window DS_win3 = {
    type: DONALD
    box: [{ DS_win3_X1, DS_win3_Y1 },
          { DS_win3_X2, DS_win3_Y2 }]
    pict "DoNaLD"
    bgcolor: DS_win3bgcolor
    fgcolor: DS_win3fgcolour
    bdcolor: DS_win3bdcolour
    border: DS_win3border
    relief: DS_win3relief
    sensitive: ON
};
```

- **Step 5** – In manipulate mode, position the paddles at either end of the screen. Click on the drawingBoard and enter the following definitions into the 'On KeyPress' field at the bottom of the modeSwitcher screen:

```plaintext
/* KeyPress definitions - drawingboard*/
if (DS_drawingboard_key1[1]==113) {
    DS_win1_Y1=DS_win1_Y1-15;
    DS_win1_Y2=DS_win1_Y2-15;
}
```

\[6\] Due to an unresolved issue with viewports in the implementation of SCOUT, it is not possible in the prototype to simply redefine the pict attribute in the relevant box.
It will now be possible to move the paddles up and down when the mouse is positioned over the drawingboard using the ‘Q’ and ‘A’ keys for paddle 1 and ‘P’ and ‘L’ for paddle 2.

- **Step 6** – Click on the ‘ball’ and introduce the following definitions in the ‘On Release’ field and confirm to get the ball to move:

```plaintext
DS_ballMoveX = 0.5;
DS_ballMoveY = 0.3;
DS_directionX = DS_ballMoveX;
DS_directionY = DS_ballMoveY;

if (DS_mode == DS_useMode) {
    if (DS_win3_X1 <= DS_win1_X2) {
        if (DS_win3_Y1 >= DS_win1_Y1 &&
```
Step 7 – The model can be further extended by adding score boards and buttons to control the beginning and end of the game. Figure 5.12 depicts a game in progress within the model.

Whilst the prototype Imagine Eden is rather primitive in its operation and limited
in the extra support it provides to the modeller, it would be possible in a full implementation to introduce a higher level of interaction whereby actions could be demonstrated by the user and recorded in a similar manner to spreadsheet macros. Common functionality could also be implemented as part of the tool to provide a simple means of repeating actions such as the movement of the ball. However, despite its limitations, the prototype does demonstrate an alternative way of model building with tkeden. Imagine Eden combines the benefits of drawScout with the ease of use associated with event-driven environments. It increases visibility and lowers the cognitive burden on the modeller in remembering SCOUT syntax, reducing error proneness. Such a tool may be of particular use in specific application areas such as educational technology, where the standard tkeden environment would be too complicated.
5.5 Summary

This chapter has introduced tkeden, the primary tool used in Empirical Modelling research, and considered the author’s work in exploring alternative interfaces and ways of modelling with the tool. Whilst tkeden provides suitable support for Empirical Modelling activity, with good mapping to the principles, analysis of its usability against the ‘cognitive dimensions framework’ of Green and Petre has highlighted certain issues that will need to be addressed in future tool improvements.

The author’s work in developing drawScout and the subsequent prototyping of VST and Imagine Eden has illustrated alternative ways of modelling that may address some of the usability issues, providing better support for Empirical Modelling practice.

The next and final chapter concludes the thesis by considering the extent to which the objectives, as expressed in chapter 1, have been met. It also considers future research directions arising from the work undertaken by the author during the course of this research.
Chapter 6

Conclusions and future directions

6.1 Introduction

This thesis has explored the nature of Empirical Modelling from its underlying philosophical orientation and high level motivations through to development work undertaken by this author in addressing some of the usability issues in relation to the primary Empirical Modelling tool, tkeden.

The objectives of this research as stated in the introduction were to:

1. Embellish Empirical Modelling philosophy by exploring possible connections with phenomenology.

2. Use phenomenological ideas to develop a fresh account of Empirical Modelling activity and the semantics of computer-based artefacts.

3. Explore potential tool developments that may provide better support for Empirical Modelling activity.

4. Illustrate particular areas of application in which Empirical Modelling principles and tools can be usefully applied.
It was stated that consideration would also be given throughout to communicating Empirical Modelling. The extent to which these objectives have been addressed and future research directions arising from the work will now be considered.

6.2 Philosophy and principles

Prior to this research, the relationship between phenomenological ideas and Empirical Modelling had not been considered in depth. The research presented in this thesis serves to verify that there are significant points of contact between EM philosophy, the Radical Empiricism of William James and areas of phenomenological thought, particularly that of Heidegger and Merleau-Ponty.

The phenomenological ideas of Heidegger in conjunction with those of Polanyi served as a useful support for revisiting the nature of semantics in EM and helped to clarify the philosophical stance in relation to the modelling activity. Central to this was the introduction of the intentional character of computer-based artefacts, the nature of the intentionality, and the way in which such intentionality comes into being.

The phenomenological ideas also supported the clarification of the manner in which EM principles relate to the philosophy and to the modelling activity which formed the basis of chapters 3 and 4.

6.3 Empirical Modelling Activity

Phenomenological thinking was useful in clarifying the nature of Empirical Modelling activity, and in further distinguishing it from traditional approaches in computing. Following this, it was argued that the appropriateness of adopting an EM approach to a particular problem situation is dependent on its amenability to
formalisation. The areas in which EM principles and tools can be most usefully applied are those where the problem situation or part thereof is not amenable to formalisation, and where the aims of those wishing to utilise computer-based support are compatible with the general philosophical orientation.

The four areas of application presented in this thesis illustrate particular ways in which EM principles and tools can facilitate ‘thinking with computers’.

6.4 Empirical Modelling Tools

The development work exploring potential directions for model building with tkeden and the analysis of tkeden in relation to the cognitive dimensions illustrate that several usability issues can be easily overcome. The work has also demonstrated that it is possible to explore novel ways of model building with tkeden that can potentially inform future tool development.

Whilst the tools have served to influence the development of Empirical Modelling principles to date, it is essential that the ideas and philosophical orientation now play a more prominent role in future tool development. The emphasis should be placed less on understanding the nature of modelling with tkeden, and more on providing effective tool support for Empirical Modelling activity.

6.5 Future research directions

The research undertaken by the author has highlighted particular areas for further research. This section provides an overview of some of the key areas and provides some comments arising from the author’s work.
6.5.1 EM philosophy and principles

Phenomenology

Phenomenology has been active in philosophy since it was first introduced by Husserl. However, one has only to look at the diverse writings of mainstream phenomenologists to realise that there is much disagreement about the fundamental ideas. This thesis has presented aspects of phenomenology that are considered most pertinent to EM, and has considered their significance in connection with the development of computer-based artefacts in general. The research presented in this thesis demonstrates the potential benefits of exploring phenomenology in further detail. Certainly, in over one hundred years of thinking and writing about the nature of experience it is likely that there will be other points of contact with EM.

Despite the differences between the Radical Empiricism of William James and phenomenology in general, to read them in conjunction may support the interpretation of each in its own right. Whilst it is beyond the scope of EM research to identify the points of contact between the two branches of philosophy in full, there is much to suggest that this would be a profitable exercise (cf. [Wil69]). The various sources that have been uncovered during the course of this research suggest that Radical Empiricism has not been given the attention it deserves in philosophical discourse (cf. [EIT96]). There has been much misinterpretation by critics, evident even from responses provided by William James himself (for example, in the essay *Is Radical Empiricism Solipsistic?* in [JBBS76]). It is possible that in revisiting Radical Empiricism in the light of phenomenological thinking and Empirical Modelling that the central ideas could be developed in full, completing what James initially set out to achieve.

Empirical Modelling faces similar challenges in overcoming mainstream con-
ceptions in the discipline of computer science and the sciences in general. In the absence of further development of Radical Empiricism, the embellishment of phenomenological connections with Empirical Modelling will serve to raise the profile of its philosophical orientation, making it approachable to a wider audience.

**Exploring earlier writings of James**

It was not possible during the course of this research to explore the previous writings of James in full. However, the notion of two types of knowledge – knowledge by acquaintance, and knowledge about an object not immediately present – are both evident in Radical Empiricism and James’s earlier work. In further eliciting the nature of Empirical Modelling, it will be useful to consider James’s earlier writings in more detail.

The metaphors introduced by Peter Naur in [Nau01] to describe some of the key aspects of the thinking of James in his article entitled *The stream of consciousness* (cf. [Jam90]) are particularly interesting in this regard and are included along with Naur’s descriptions below in full:

**Metaphor 1:** *The mental activity is like an octopus jumping in a pile of rags.* This metaphor is meant to indicate the way in which the state of consciousness at any moment has a field of central awareness, that part of the rag pile in which the body of the octopus is located. The arms of the octopus stretch out into other parts of the rag pile, those part presenting themselves vaguely, as the fringe of the central field. The rags of the pile are the mental objects that may come to the conscious awareness. They are of all colours and shapes. The jumping about of the octopus indicates how the state of consciousness changes from one moment to the next.
Metaphor 2: A person’s insight is like a site of buildings in incomplete state of construction. This metaphor is meant to indicate the mixture of order and inconsistency characterizing any person’s insights. These insights group themselves in many ways, the groups being mutually dependent by many degrees, some closely, some slightly. As an incomplete building may be employed as a shelter, so the insights had by a person in any particular field may be useful even if restricted in scope. And as the unfinished buildings of a site may conform to no plan, so a person may conform to no plan, so a person may go through life having incoherent insights.

Metaphor 3: A person’s utterances relate to the person’s insights as the splashes over the waves to the rolling sea below. This metaphor is meant to indicate the ephemeral character of our verbal utterances, their being formed, not as a copy of the insight already in verbal form, but as a result of an activity of formulation taking place at the moment of the utterance. (p.86) [Nau01]

The first metaphor is consistent with James’s later idea of conjunctive relations and the notion of state-as-experienced in EM. The second metaphor resonates with the epistemological stance promoted by EM and helps to further illustrate the nature EM activity and how EM models should be regarded as always subject to future embellishment and reinterpretation. The third metaphor is useful in illustrating the nature of language as viewed from an EM perspective and resonates with the discussion of content and form in [Smi96].

It is thought that further exploration of James’s earlier writings will support those thinking about Radical Empiricism in relation to EM, and also support the communication of the philosophical orientation promoted by EM.
Foundations for EM principles

Whilst EM principles appear compatible with Radical Empiricism and the phenomenological ideas presented in this thesis, the notions of observables, dependency and agency are not mentioned explicitly in the works of James considered during the course of this research. This in itself is unproblematic – the model building exercise serves as evidence that the principles are comprehensive and applicable to all areas considered to date in the research. However, an appropriate philosophical foundation for the principles will give them further credibility, and might also assist in future interpretation of model building and in communication with external parties. (cf. Gooding’s desire for a “theory of observation” in [Goo90])

It is thought that Bertrand Russell’s consideration of causal relationships (cf. [Rus48]) may provide interesting material in relation to dependency and triggered agents. Another idea is to explore the possibility of considering a definitive script as an alternative form of phenomenological description. However, it has not been possible to identify significant connections during the course of this research and determining appropriate philosophical foundations for EM principles is an area for ongoing investigation.

6.5.2 EM activity

Collaborative modelling

Most Empirical Modelling research to date has concentrated on supporting a single modeller. This is in keeping with the view of the semantics of computer-based artefacts as personal and subjective. However there is a need for further research into the nature of the modelling exercise when it is undertaken collaboratively. For example, in this situation, a further dimension is added to any semantic consid-
eration of the model under development. There is also a need to consider the ontological status of models under collaborative development.

Following the phenomenological idea of objectivity as inter-subjective confirmation and acceptance (cf. [Mor00]) and the Jamesian idea of pointing it should be possible to consider both the semantic and ontological issues in relation to collaborative modelling in a manner consistent with the single modeller scenario. However, it is beyond the scope of this thesis to consider this in further detail due to the limited experience of collaborative development with EM principles and tools.

6.5.3 EM tools

drawDonald

Following the drawScout developments and the author’s experience of model building using DoNaLD, it was thought useful to develop a drag-and-drop tool for the ad hoc creation of line drawings. Whilst it was not possible to develop a full prototype during the course of this research, a simple proof of concept version has been developed and will now be briefly discussed.

Following the same principle as drawScout, the drawDonald proof of concept comprises a SCOUT window sensitive to mouse click and release actions. Clicking, dragging and releasing on the sensitive window enables the modeller to draw circles of various sizes and to readily create extensive line drawings without the need to learn DoNaLD syntax. This has similar advantages to those of the drawScout tool as discussed earlier (cf. chapter 5).

The proof of concept version was slightly extended such to enable the creation of boxes comprising four points and four lines, however it was not completed and it is currently only possible to create a single box. A line drawing created with
the drawDonald proof of concept prototype is depicted in Figure 6.1.

**Eden Active Scripting Environment**

The Eden Active Scripting Environment (EASE) is an idea for further extending the VST tool to provide better support for Empirical Modelling. Due to the complexity and scale of the necessary development work, it was considered beyond the scope of this research to develop the tool in full, but it will now be described in concept.

The initial idea for EASE came from the observation that many of those modelling with tkeden use a text editor to support their model building. Using
a text editor enables the modeller to control the structure and format of their script. As previously mentioned, such ‘secondary notation’ is lost when definitions are stored in the EDEN symbol table.

The VST tool can be used to make the current definitions visible (cf. Chapter 5), but it does not offer the flexibility that the text editor provides to the modeller. The first step in building EASE would be to extend the VST tool to support the editing of definitions in place. The resulting interface would resemble a single column spreadsheet. The ‘input’ part of the input window could be removed at this stage as its functionality would be superfluous.

The visibility of the modelling tool could be further improved by integrating a tkeden version of Allan Wong’s Dependency Modelling Tool (DMT) [Won03], which creates a graph representation of definitive scripts, with the EASE environment. A limitation of the DMT is that graphs generated for large scripts are largely unintelligible. However, for small scripts or sections of a large script, the DMT can provide a useful means of illustrating the relationships between particular observables that are of interest. A DMT could be used in conjunction with EASE to create small graphs to illustrate the immediate relationships for the current observable of interest as indicated by the modeller selecting a particular definition through the interface. It is thought that using the DMT in this way will provide particularly useful support to the modeller when breakdown occurs during model building, and attention is drawn to the particulars of the model. Figure 6.2 depicts a possible interface for EASE\textsuperscript{1}.

\textsuperscript{1}This screenshot has been constructed for the purposes of illustration only. The author has only considered EASE in concept. The tool itself has not been implemented.
In the previous subsection, it was recommended that further research be carried out to better understand collaborative modelling in EM. Following such research, it would be possible to reconsider the current tool support for collaborative development. The key challenge to be addressed when the modelling exercise is undertaken by more than one modeller is how best to support each modeller in conveying their construal to other modellers. A central idea in EM is that the *intentional* character of computer-based artefacts is personal and subjective. Following this, the meaning of the artefact to each modeller will need to be conveyed to the other modellers if the exercise is to be effective.

One idea for supporting collaborative modelling is the development of a comment notation which could be used to introduce 'descriptive observables'. As discussed earlier, much of the secondary notation is lost in the model when it is entered into the symbol table, and yet such secondary notation can be essential in
conveying meaning among multiple developers. By introducing a simple notation to create additional observables that can be used to communicate each modeller’s construal of particular observables and to record its history, the tool might be readily modified to provide effective support for collaborative modelling.

6.6 Communicating Empirical Modelling

This thesis has considered the breadth of Empirical Modelling from its philosophical underpinnings to the current tool support. Throughout, it has been the author’s intention to uncover Empirical Modelling and the thesis has proceeded to do so by:

- Exploring connections between phenomenology and the philosophical orientation underlying EM.
- Developing a fresh account of Empirical Modelling activity and the semantics of computer-based artefacts.
- Considering areas in which EM principles and tools can be usefully applied.
- Describing work undertaken by the author in improving the current tool support and exploring alternative ways of MWDS.
- Making the connections between EM philosophy, principles, activity and tools explicit.

However, it remains an ongoing research challenge to establish the most appropriate ways of communicating the nature of Empirical Modelling.

There is an axiom attributed to Galileo that you cannot teach a man anything; you can only help him to find it for himself. The author hopes that this thesis will support all engaging with Empirical Modelling in finding it for themselves.
Bibliography


[BKR06] Meurig Beynon, Roderick R. Klein, and Steve Russ. Humanities’ com-
putings. In Digital Humanities 2006: 1st International Conference of the Alliance of Digital Humanities Organisations, Conference Ab-

[BRC⁺05] Meurig Beynon, Steve Russ, Charlie Care, Eric Chan, An-
kaleidoscopeBeynon2005/. Last accessed 24/09/06.

[BRM06] W. M. Beynon, S. Russ, and W. McCarty. Human computing: Mod-


[Nor98] Donald Norman. The Invisible Computer; Why good products can fail, the Personal Computer is so complex, and Information Appliances are the solution. MIT press, 1998.


Vita

Email address: karlking2k@gmail.com

Permanent Address: 22a St Fremund Way
Leamington Spa
CV31 1AB

This thesis was typeset with \LaTeX{}\textsuperscript{2ε}. \LaTeX{}\textsuperscript{2ε} is an extension of \LaTeX{}. \LaTeX{} is a collection of macros for \TeX{}. \TeX{} is a trademark of the American Mathematical Society. A modified version of the Warwick style file \texttt{wnewthesis} was used in typesetting the thesis.