Modelling with experience: construal and construction for software

Meurig Beynon

Abstract Software development presents exceptionally demanding conceptual challenges for model-building. It has such diverse phases, may touch so many disciplines, may address personal and public applications, can involve collaboration of experts from many fields, user participation, and an essential need for ongoing revision. Software modellers must see and think like designers, logicians, engineers, programmers, business analysts, artists, sociologists. This chapter reviews thinking about software development with particular reference to: the limitations of adopting the formal representations that classical computer science commends; different approaches to rationalising the use of models in software development; and the problems of conceptualising software development as theory-building. It concludes by sketching an embryonic approach to software development based on "Empirical Modelling (EM)" that draws on William James's pluralist philosophy of 'radical empiricism', the historian of science David Gooding's account of the role of 'construals' in experimental science, and the sociologist Bruno Latour's vexing notion of 'construction'. The products of EM development are interactive artefacts whose potential meanings are mediated by the patterns of observables, dependencies and agencies that they embody, as elicited by the actions of the participants in the model-building.

1 Introduction

In many disciplines the use of models is widespread and essential. The ubiquity, variety and usefulness of models is undeniable. Sometimes the model is more concrete than what is being modelled (e.g. the moving beads of an abacus corresponding to steps in a calculation), sometimes it is less concrete (e.g. a differential equation referring to fish stocks), but typically the model is a simplification – with only selected features – of the object or phenomenon being modelled. Sometimes the model refers to something 'given' (e.g. the solar system), sometimes the referent is as yet only imagined (e.g. a building or a design for a building). In each of the examples just offered the model is (or could easily be) a formal representation. At least in the sciences the everyday use of such models has become second nature to practitioners and their value and validity are, for the most part, taken for granted.

Many applications of modelling in science and engineering exploit computing as a way of implementing pre-existing models derived from theory or established empirical knowledge. Creating software in such a context is a routine design exercise. The focus in this chapter is on the challenges encountered in developing software where such methods either do not apply or do not deliver a suitably engaging product - for instance, in complex systems that require more open-ended radical design (cf. [37,64]) or in applications in humanities computing where software is developed to provoke the scholar's imagination (cf. [52]). Software development of this nature, drawing on its strong links with engineering, mathematics and human interaction, makes pervasive but problematic use of a wide assortment of models both informal and formal in character. This chapter draws attention to some of the challenging problems that are highlighted by such software development, with particular reference to the nature of the modelling it entails (cf. [50]), the semantic range of the models that it invokes, and the need to make formal and informal representations coherent. It then sketches a new approach, Empirical Modelling (EM) [70], in which the modelling processes are based firmly on direct, ‘living’ experience rather than formal representations. The principal activity in EM is the development of provisional, personal sense-making entities ("construals") that, in an appropriately engineered context, can be interpreted as public entities amenable to formal description.

In order to explain the approach of EM, three fundamental questions concerning modelling have to be addressed:

- how can direct, personal experience serve as a ‘grounding’ for modelling?

- how can we exploit and justify ‘construals’ as a means of building models?

- how can the constructed, mediated character of the ‘public entities’ developed by modelling be reconciled with the 'real-world' and 'formal logical' entities that play such essential roles in conceiving and implementing complex systems?

The three main themes of the chapter are framed by these questions. These themes are not treated here systematically or in depth; they recur and are interwoven as necessary to elaborate our thesis about modelling (in section 9). Each theme is explored in some detail by, respectively, the philosopher William James (cf. [40]), the historian of science David Gooding (cf. [30,31]) and the sociologist Bruno Latour (cf. [46]). A deeper appreciation of how these themes relate to each other and to EM can be gained from these references and other EM publications (see e.g. [8,9,11]).

It is widely taken for granted that in order to produce a computer solution, or a programmed solution, a given problem must first be replaced by an abstract version. For example, the physical symbols or pieces and their locations in a game, or the liquid levels in an experiment, are replaced by values in a mathematical, or a programming language, type. Then it is this abstract version that is actually solved by the program, or investigated by a model or simulation. Simplistic as this formulation undoubtedly is, the fundamental assumption is pervasive and deep: a computer-based solution primarily engages with an abstract version of a problem or task, in contrast to the problem or task as we experience it. In the latter case – but not in the former – there are typically physical components such as symbols, traffic lights, documents, measurements, etc, and with all such things there are borderline cases, unclear or partial cases, mistaken cases etc, arising from human psychology and context. This chapter introduces an approach to modelling, and thereby an approach to computing (in particular to software construction), developed over many years at Warwick, which reverses the broad direction of the problem-solving process just described. The concrete personal experience of a problem instance is taken here as primary, while abstractions, wherever they are useful, take an auxiliary role. The approach is known as *Empirical* Modelling because of its emphasis on observation and experiment; it promotes an alternative way of using computers: one that gives priority to our direct experience of phenomena. This is made possible by regarding the computer itself (with its associated devices) as a source of new experience that is continuous with, or of the same kind as, experience of the real-world problem or phenomenon.

The EM project has been established for about 25 years. It has generated a number of practical modelling tools and several hundred models / construals. These models have unusual interactive qualities, open-endedness and semantic plasticity. Whereas conventional modelling uses abstraction to simplify complex phenomena, EM generates an interactive environment in which to explore the full range of rich meanings and possible interpretations that surround a specific - typically seemingly 'simple' - phenomenon ("playing a game of noughts-and-crosses" [10], "learning to use your digital watch" [15], "explaining a perplexing encounter with a lift" [8], "solving a Sudoku puzzle" [9]). Because of its emphasis on enriching experiences and exposing semantic possibilities, EM can be applied directly in certain areas (such as creative design, education, and the humanities), but in general is best understood as an activity that can complement and enhance software development based on traditional abstraction techniques. Though notions characteristic of EM (such as agent-orientation and dependency) also feature prominently in modern software practice, there is no coherent conceptual framework for their adoption alongside tools and techniques aligned to traditional computational thinking (cf. [60,58]). Making the case for EM as a suitable conceptual framework motivates the fundamental questions concerning modelling raised above.

The chapter considers how the challenges of software development call for the reconciliation of views, and the integration of activities, that are superficially quite divergent in character: formal and informal, personal and public, machine and human based, goal-directed and exploratory. The scene is set (in sections 2 and 3) by highlighting contrasting perspectives that are associated with the mathematical and engineering aspects of software and with the possible roles for the machine and the human within computationalist and constructivist traditions. Several perspectives on how to bring coherence to the software process are briefly reviewed (sections 4 to 6); these include high-level critiques of the predominantly "rationalistic" emphasis endorsed by theoretical computer science (e.g. Winograd and Flores [67], Cantwell-Smith [25,26], Naur [55,56]), approaches that propose new methodologies that complement conventional techniques for representing and modelling software (e.g. Harel [33,34], Jackson [36,37,38,39], Kaptelinin and Nardi [41]) and approaches that tackle the foundational issues surrounding models and their role in development (e.g. Mahr [50], McCarty [52], Gooding and Addis [31,3,4]). It is in section 5 that the key ideas of the chapter emerge as follows. A constructivist account of software calls for a conceptual framework that makes personal experience fundamental and this is supplied in the 'radical empiricism' of William James. Software development demands modelling of both kinds - experiential and formal - but integrating models of such different kinds to afford conceptual coherence has proved a serious challenge.

All the approaches mentioned in the above paragraph operate within a framework in which the software has a semantics conceived as formal (e.g. mathematical), though this may be explicitly informed by surrounding informal processes which play a significant part. This is in keeping with an established scientific tradition such as that advocated by James Clerk Maxwell, who maintained that analogy and physical illustration have a crucial role in developing abstract mathematical models. The final sections of the chapter (from section 8 onwards) introduce EM as an alternative approach to computing, and to software, that gives priority to immediate experience of the concrete and situated - without excluding formal representations, and that invokes ideas drawn from William James for its semantic foundation. EM has much in common with the provisional, personal modelling employed by Faraday in his experimental researches for which Gooding introduced the term "making construals" [30]. Developing construals is well-oriented towards the needs of software engineering where addressing human aspects, auditing the development process, and adaptation in response to changing requirements are concerned. It can also be seen as a technique with qualities appropriate for what Latour [46] conceives as "construction". The chapter concludes with a brief account of how EM addresses the problems of modelling for software development mentioned above.

2 Ways of seeing, ways of thinking

The development of complex software systems combines abstract mathematical activities relating to formal specification and inference with concrete engineering activities associated with configuring devices and building their interfaces. The range of issues to be addressed in this way is well-represented in Jackson's paper *What can we expect of program verification?* [38], which is strongly rooted in the formal computer science tradition whilst recognising the need to endorse Vincenti's concern that engineering should "do justice to the incalculable complexity of the real-world" [64].

In understanding the different ways of thinking that are represented in mathematics and engineering, it is helpful to distinguish two ways of looking at everyday objects. We can look at an object such as a clock for instance with specific attention to those aspects that serve its characteristic function of 'telling the time'. For a digital clock, the relevant observables are starkly symbolic: the digits on the LED display, perhaps with an indicator as to whether it is now "am" or "pm". For an analogue clock, the relevant observables are somewhat more subtle: the hands of the clock and their orientation relative to marks on the face and to each other. In either case, our way of seeing is directed at abstracting specific information about mathematical measurements of time elapsed in hours, minutes and seconds. Contrast the way in which we might observe a natural object such as a tree. Like the clock, a tree maintains its integrity through continuous state change: its growth, its motion in the wind, its response to the changing seasons. But a tree object has no clear function or protocol for observation. There are a whole variety of reasons why we might pay particular attention to different features of the tree, according to whether we are interested in identifying its species, its state of health, its capacity to provide shade or shelter, its potential to cause damage to property and so on. By paying attention in one specific way we do not do justice to all that a particular tree object can be.

A clock and a tree may naturally invite different 'ways of seeing', but this has as much to do with the typical context and disposition of their observer as with their intrinsic nature. A clock is built for a specific purpose and is typically viewed as it were through the filter of its function. A tree is typically a given of our natural environment that contributes to our here-and-now experience in ways that are unpredictable and unplanned. Mathematicians enjoy distilling patterns and structures (such as differential equations and groups) through circumscribed ways of observing, interacting and interpreting "clock-like" objects. Engineers characteristically engage with concrete "tree-like" objects in more holistic and open-ended ways. But fruitful work in engineering and mathematics draws on both types of observation. In designing an attractive clock, the engineer must take account of all kinds of observations whilst focusing on those that fulfil the narrow requirement. In devising a mathematical model of a tree, the mathematician adopts a constrained way of observing features relevant to a functional objective but may first need to identify suitable features and patterns of behaviour derived from exploratory experiment.

The two "ways of seeing" we have contrasted are respectively associated with two "ways of thinking": *functional abstraction* and *exploratory interpretation*. Their co-existence and mutual interplay has been prominent in the history of mathematics and engineering, and has especial relevance in modern computing. On the one hand, digitisation and computational science have made it possible to express ever more complex and subtle systems using functional abstractions. On the other hand, new computing technologies and applications have led to the development of artefacts with unprecedented scope for imaginative exploration and interpretation. The impact of these developments has been to juxtapose two quite different perspectives on computing in a radically new way, potentially setting up a tension that threatens to polarise the discipline. A key concern in this chapter is to sketch an integration and reconceptualisation that can promote both perspectives.

The potential for tension derives in part from the way in which computing now pervades all kinds of disciplines, including many that have well-established modes of enquiry into meaning unlike those of science as traditionally conceived. The accepted theoretical foundation for the digital culture is well-disposed towards rational, logical, objective, reductionist stances. It is well-suited to scientific applications that can exploit a formal account of language and semantics. Applications of computing in an instrumental role in support of human activities in everyday life, business and the arts, by contrast, have more affinity with the notion of language as metaphor and lend themselves to hermeneutic, experiential, subjective and holistic stances. An extreme form of functional abstraction represents the universe as a machine whose behaviour can in principle - and in due course will be - comprehensively observed [66]. It may be contrasted with the extreme variety of exploratory interpretation embraced by the English literary critic I A Richards, whose vision of language as metaphor (cf. [45]) informed "[his] compelled and compelling belief in the absolute reality of the world of poetry, which was not, for him, parallel to the world of life but was coextensive with it, and the means of deepest penetration into it" [63]. If we are to make sense of the way in which computing technology is now being exploited across disciplines we need a conceptual framework in which both of these perspectives can find their place.

3 Computationalist and constructivist accounts of agency

The extent to which we can and wish to automate human activities is topical in relation to software development. Software development incorporates sense-making activities carried out in the identification of requirements and goal-oriented rule-based activities that are addressed in implementation. These two kinds of activity are respectively associated with ways of thinking represented by 'exploratory interpretation' and 'functional abstraction'. In conceiving a complex software system, it is necessary to characterise the roles of many agents, human and non-human, external and internal to the system and to engineer their interaction in accordance with their capabilities and the characteristics of the environment. The modelling approach that we advocate is one in which no absolute presumptions about agency need be imposed; at the discretion of the modeller, agents can act and interact autonomously according to law-like rules or be animated under human control. A construal for a system will typically be a hybrid between two extremes: the *computationalist* account of agency in which agent interfaces are based on 'functional abstraction' and the *constructivist* account of agency in which agent interfaces are based on 'exploratory interpretation'. These two kinds of account are introduced below. The benefits of making construals can be fully appreciated only by adopting an agnostic stance about causality (i.e. about the 'real' agencies at work) - for which Turing's stance explained below gives some motivation.

In the celebrated paper in which he introduced the so-called 'Turing Test' [61], Turing discussed the issue of whether or not humans were "regulated by laws of behaviour" and accordingly might be "some sort of machine". At that time, sixty years ago, the idea of proposing that the human mind might be machine-like was perhaps much more implausible than it is today. The very significant impact that computers based on the principles identified by Turing has subsequently had on science in general and on theories of mind has encouraged some to promote a computational conceptualisation of the universe as the next step in a sequence of historical developments in science that have displaced perspectives that privilege the human (the earth is not the centre of the universe, nor the sun; humans are biologically of the same nature as animals etc). Wolfram's "new kind of science" [66,68], Humphreys's vision for "extending ourselves" [35] and Bentley and Corne's aspirations for evolutionary design [6] invite us to reappraise the roles for the human in making judgements and performing actions. In a similar spirit, some researchers seek to demonstrate that even the core activities of the humanities can be conducted by machines rather than humans (cf. [21]).

For his part, Turing [61] acknowledged that he had "no very convincing arguments of a positive nature to support [his] views" and it is not entirely clear that - even in today's context - he would have endorsed such grand visions for computation. The argument he used in 1950 to undermine the position of those who argued against the possibility of mind-as-machine is still valid: "... we cannot readily convince ourselves of the absence of complete laws of behaviour ... The only way we know of for finding such laws is scientific observation, and we certainly know of no circumstances under which we could say, "We have searched enough. There are no such laws."" But when he further comments: "As I have explained, the problem is mainly one of programming ...", it is not clear that even Turing's visionary imagination could foresee the form that 'the problem of programming' would take by the end of the 20th century. As is widely acknowledged, the problems of complex software systems development, and the limited progress towards resolving them using formal mathematical methods, indicate the need for better ways of taking human aspects into account when exploiting automation.

The problem of finding the proper place for the human in software development can be seen in the broader perspective of controversies surrounding constructivist accounts of science. In "The Promises of Constructivism" [46], Bruno Latour spells out a vision for what adopting a constructivist perspective entails. What he declares to be its characteristic conception - that "[e]verywhere, building, creating, constructing, laboring means *to learn how to become sensitive* to the contrary requirements, to the exigencies, to the pressures of conflicting agencies where none of them is really in command" seems well-matched to the practice of software engineering. What is more, when we consider those issues that have provoked controversy (cf. [46]), a constructivist outlook seems to be much better oriented in some respects towards software engineering than it is towards science. For instance, in analysing the problems faced in promoting a constructivist account of science, Latour writes:

"The two things science studies did *not* need were to replace the fascinating site it was uncovering by an unconstructed, homogeneous, overarching, indisputable "society" and of course an unconstructed, already there, indisputable "nature." This is why science studies found itself fighting on two fronts: the first against critical sociology it wrongly appeared to descend from (as if it were merely extending social explanation coming from law and religion to science and technology); and the second against nature fundamentalists who wanted facts to pop up mysteriously from nowhere."
In the case of software engineering, it is entirely appropriate to attribute some aspects of software to laws, norms and power relations within society, and its artificial status is not in doubt. Yet, as Latour explains in [46], there is a much more fundamental concern to be

addressed in championing a constructivist viewpoint. Latour is dismayed by the way in which his constructivist vision for science has been discredited by misguided accounts from 'critical sociology' of what *construction* entails:
"... the problem of construction ... has to do ... with the inner mechanism of construction itself. The problem with constructivism is that no one could account for the building of anything ... by using this metaphor as it has been popularized in social sciences."
In his view, rehabilitation of the notion of "construction" is crucial to redressing the situation. And, by a *good* notion of construction, Latour means a notion of construction set within a conceptual framework that affords scope for debating about - if not discriminating between - 'good' and 'bad' construction.

4 Software development and constructivism

On the face of it, software development is an obvious topic to consider from a constructivist perspective. Whereas scientists may have reservations about whether scientific facts are already there to be discovered, these reservations do not apply to software engineering products (cf. Latour's discussion in which he contrasts the metaphysical status of a building with that of a scientific fact: "... everyone will agree that ... the building was not there before"). Perhaps the principal challenges facing software development are concerned with the activities that elicit requirements and with modifying software in the face of changing requirements. Both of these problems relate directly to those qualities associated with constructed objects - "history, solidity, multiplicity, uncertainty, heterogeneity, risk taking, fragility, etc." - that are identified by Latour in [46]. And having been built with sensitivity to "the contrary requirements, to the exigencies, to the pressures of conflicting agencies" is a much more holistic criterion for quality for a software product than 'meeting its functional requirement'.

There are good reasons why traditional software development is not well-aligned to constructivist thinking, however. Software development has been deeply influenced by a way of seeing that is dominated by functional abstraction and a way of thinking that favours formal representations of the kind that Turing's work inspired. These dispositions reflect the core issues that the theory of computing addresses: *how to develop ingenious algorithms to fulfil a given requirement? how to optimise these to minimise the computation involved? how to implement them on high-performance architectures? how to express them in programming languages? how to analyse them using mathematical logic?* - all of which presume pre-established requirements. Adopting this focus does not prevent us from addressing broader questions - for instance, Mahr's study of models of software [50], which takes as its central question *Does the system S comply with the requirements for its application?*, is directed at being able to modify and adapt to requirements in a flexible way, and to address non-functional requirements. But the representations that computer science has developed tend to favour situations in which the requirement has been abstracted and the methods that are to hand engage with these abstractions (cf. Kramer's characterisation of computer science as 'the study of abstraction' [44]).

Many meta-level accounts have challenged the notion that classical computer science concepts can give an adequate account of computing activity - and of software development in particular. Prominent amongst these are the writings of Winograd and Flores [67] and of Brian Cantwell-Smith [25,26]. Naur [55,56] is another fierce critic of the accepted theoretical computer science culture who has championed the idea that intuition has an essential role in software development. A key concern in these critiques is that the semantic relation that links a computer program with its application domain (which Cantwell-Smith [25] terms 'the semantics of the semantics of the program' in deference to the narrow machine-oriented interpretation of 'the semantics of the program' in computer science) is established and maintained because of human, social and contextual factors that cannot be formalised.

Such critiques highlight limitations of theory without addressing what Latour has identified as the key question for the constructivist: *What form should good construction of software take?* Critiquing the accepted theoretical framework of computer science does not necessarily give insights into a practice that has developed its own independent culture. For instance, though the traditional characterisation of models adopted in software development: "representations of a system at many levels of abstraction, and from many viewpoints" might suggest an emphasis on formal representations that characterise the product rather than its construction, such models in practice exploit informal representations to record histories and variants and to document design decisions. And whilst there may be good empirical evidence for discounting some informal but ephemeral representations that have been advocated for software development in the past, we should be sceptical of the notion that our ability to formalise is necessarily the best measure of our understanding. In this connection, Black's caveat regarding the use of mathematical models in science is pertinent: "The drastic simplifications demanded of success of the mathematical analysis entail serious risk of confusing accuracy of the mathematics with strength of empirical verification in the original field." [19]

5 Formal representations and experientially-mediated associations

Many agents have roles in the development of a software system. They may be both human and non-human, internal and external to the system. The ways in which these agents actually or metaphorically observe and act are diverse and reflect their particular viewpoints and capabilities. A fundamental challenge in developing the system is making these many viewpoints coherent by identifying some kind of 'objective reality' in which they can all be taken into account. What Winograd and Flores [67] describe as a 'rationalistic' approach to development in effect tries to use representations suited to specifying objective knowledge to address all the modes of 'knowing' that are represented in the development environment. As Brian Cantwell-Smith remarks [25], formal representations are insufficiently nuanced to reflect the rich indexical character of the communication associated with development: they encourage 'promiscuous modelling' [25] in which there is no differentiation between models that are merely logically equivalent (cf. every computer is a Turing machine when characterised as an abstract computational device). In defending such approaches, it is hard to avoid what Latour [46] terms a 'fundamentalist' position.

Latour [46] characterises *fundamentalism* as "deny[ing] the constructed and mediated characters of the entities whose public existence has nonetheless to be discussed", and suggests that "constructivism might be our only defense against fundamentalism". Even if we were to subscribe to the idea that all human behaviour could be explained by 'computational' rules, this would not necessarily be good grounds for taking a fundamentalist stance. It is quite apparent for instance that knowledge of the laws of projectile motion is neither necessary nor sufficient to guarantee that a person can intercept a ball in flight. By the same token, knowing that our actions are determined by laws, and being able to correlate our actions with these laws as they are being enacted are two quite different things.

To sustain the idea of constructivism as affirming "the constructed and mediated character of the entities whose public existence has to be discussed", it is essential to have some means of referring to personal experience. For this purpose, we shall adopt a key tenet of William James's radical empiricism [40]: that knowing something is to experience an association between one aspect of our experience and another. We shall refer to such associations as "experientially-mediated". To appreciate the highly personal and context-dependent nature of experientially-mediated associations, we might consider the different associations that would be established if a person looking at the first page of the Moonlight Sonata were to be (say) a child who has not learnt to read music *or* a professional pianist who was about to give a performance *or* a future archaeologist unfamiliar with the entire Western musical tradition.

Constructivism is centrally concerned with activities like learning and discovery that link the personal to the public: engineering the context, training the agents, discovering the boundaries so as to build bridges between the two ways of seeing identified above. The roles for formal representations and experientially-mediated associations in these activities are illustrated by considering what is involved in 'learning to read the score of the Moonlight Sonata" or might be involved far in the future in "rediscovering how to read the score".

On the one hand, the score of the Moonlight Sonata has the status of a formal representation. To interpret this representation, we have to respect public conventions about how to observe and apply preconceived rules so as to read the score as a functional abstraction. There are also experientially-mediated associations that can be invoked by the score which go far beyond what is expressed in the formal representation - the apocryphal back-story to its composition known to the child; the tempo and the shapes and movements of the hand that are prepared in the pianist's mind; the fragmentary remnants of a piano keyboard dug up by the archaeologist. The difficulty that we face in exploiting these associations effectively is that they are much more subjective in character, more contingent on specific context and have to be mediated in much more complex and obscure ways - consider for instance what is demanded of the pianist wishing to communicate her sense of preparedness to play. (Compare the challenges of communicating techniques to the user of a brain-computer interface in contemporary applications of software development.)

In general, practising computer scientists acknowledge that both formal and informal experientially-mediated representations have an essential role to play in software development, and venture to exploit both without concern for how - perhaps even *whether* - they can be coherently integrated *at the conceptual level*. Software development is a prime example of a field in which both kinds of representation come together to create intimidating conceptual challenges. The diversity of the proposals for tackling this issue is an indication of just *how* intimidating. Harel, approaching the field from the perspective of formal software development, recognises the importance of experientially-mediated representations to the developer; he devised the statechart as a "visual formalism" for this purpose [33], and has subsequently sought to exploit message sequence charts as a way of enabling end-users to participate in the formal specification of systems [34]. Jackson [38], drawing on decades of experience as a leading software consultant, identifies the vital need to engage with the environment in the spirit of the engineer when the development of a complex software system entails non-routine design; he stresses the dangers of premature commitment to specific object-based decompositions and advocates the use of 'problem frames' [36,37] as a means of critiquing decompositions. Nardi, who has drawn on her background in anthropology in her extensive studies of software development, highlights the way in which specific representations and social contexts influence the development process [54]; she favours an account of software development that is cast in the framework - similar in spirit to that advocated by Winograd and Flores [67] - of Activity Theory [41].

6 In search of principles for modelling based on experientially-mediated associations

The system development activities discussed by Harel, Jackson and Nardi all rely on the incremental construction of what are traditionally termed "models" of the system to be delivered. Such a 'model' may be quite amorphous in character, being informed by diverse elements such as UML diagrams, logical specifications, prototype code and informally expressed requirements and documentation. As Loomes has pointed out [49], a major conceptual difficulty is to justify the idea that throughout the development process the evolving models have integrity and can in some way be shown to be authentically associated with the yet-to-be-built final system to which they at all times refer. A key problem is that an abstract functional specification of a system presumes a degree of assured and objective understanding that the developer in general only acquires through the development process. In different ways, Harel, Jackson and Nardi are concerned with how experientially-mediated representations that actively call for human interpretation and judgement can be exploited in reaching a suitable specification. To address Loomes's concern, it is necessary to understand what principles - if any - legitimise the use of such informal representations in these approaches. Specifically, we need to know in what sense - notwithstanding the misconceptions, revisions, and refinements to which the model is subject - the development of a system can be regarded as a coherent and discriminating process of *construction*.

If construction is to demonstrate "the constructed and mediated character of the entities whose public existence has to be discussed", it must be based on the incremental development of a succession of models. If the construction is addressing a problem requiring 'radical design', the models developed in the initial stages will typically exploit experientially-mediated associations, and only in later stages become integrated with - or identified with - models based on formal representations. To address the key problem of identifying what constitutes 'good' and 'bad' construction (cf. Latour [46]) we need principles by which to assess the quality of human judgements involved in the construction process.

Several different strands of independent research have ventured to provide a foundation for aspects of computing (cf. Russ's notion of "human computing" [17]) that rely on experientially-mediated associations in some essential way.

Mahr, in his account of modelling in software development [50], identifies "a theory of modelling" as crucial to computer science. His observation that "every object can be a model in some context" motivates a significant conceptual move that shifts the emphasis from 'defining the notion of a model' to 'reasoning about human judgements to the effect that something is a model'. This is the basis for a "logic of models" in which "the structure of contextual relationships" has a critical role:

"The logic of models permits an abstract description of the *argument* by means of which a judgement, assumed to be indivisible, and according to which a specific object is assigned the role of being a particular model, may be justified."
"... the argument ... relies on the existing of [a] structure of contextual relationships as its condition of correctness"

McCarty [52] identifies modelling as the core activity of Humanities Computing. His key insight is that the semantic force of models in the humanities is in the modelling processes that surround their development and use: though he chooses to interpret such models within the traditional Turing framework, they derive their expressive power from a dynamic human context in which their meanings are always being interactively negotiated, and where the role of the model is characteristically provocative and problematising.

A third account of how objective knowledge is constructed is that given by Addis and Gooding (cf. [3,4,31]). This account reflects philosophical ideas from C S Peirce and proposes "a pragmatic stance ... that relates directly our actions in the world to ... our knowledge about the world" [3]. A key feature is that it draws on what Gooding, in his account of Faraday's experimental work on electromagnetism [31], calls 'construals'. Such a construal (or tentative model) directly exploits the experientially-mediated associations and "cannot be grasped independently of the exploratory behaviour that produces it or the ostensive practices whereby an observer tries to convey it." [31, p88].

Each of these approaches highlights the significance of experientially-mediated associations, and each attempts to find an effective way of conecting them with formal representations: in Mahr's case - reasoning about judgements [50]; in McCarty's case - associating meaning with the modelling process rather than model as product [52]; in Addis and Gooding's case - transforming empirical knowledge to logical form by a process of abduction [3,4]. The vexing question of defining the notion of *model* is prominent in all three treatments. If construction is to be based on a succession of models then it must involve models that have quite different ontological status. This possibility is consistent with Achinstein's conception [1] of

'a hierarchy of models based on their ontological commitments: at one end, we have models which are simply supposed to provide possible mechanisms for how natural systems might be operating, while at the other end, we have concrete claims that the real world is thoroughly like the entities and deductions in our model. In the latter case, the choice of model amounts to a metaphysical commitment regarding the contents of the universe - which things and relations really exist.'

But the versatility of models in this respect gives them a chameleon-like character that has confounded attempts to say precisely what a model *is* - it is hard to devise a definition that takes account of where a model is located along the axes personal/public, subjective/objective, provisional/assured and the ontological status we may wish to claim for it.

7 Software engineering as science?

As the previous discussions have shown, notwithstanding the challenges that software development presents to formal approaches, the idea of invoking logical and mathematical models is well-established. There are several reasons for this:

- Many of the most important developments in mainstream computing are based on exploiting pre-existing mathematical and scientific models. In this context, the theory of the application domain is already established, and the role for computer science is to optimise the use of resources through the design and implementation of efficient algorithms and computing environments. The visions of Wolfram [66,68], Humphreys [35] and Bentley and Corne [6] alluded to earlier are squarely in this tradition, and envisage ways in which theory and computation can have radical transformative impact.

- Theories, which can be mediated symbolically, are well-suited to exploitation on computers as traditionally conceived. Theoretical computer science does not give a good account of applications that cannot be framed in terms of symbolic representations - indeed, it can be seen as calling into question whether such applications are well-conceived. Though the idea that the process of software development follows a "waterfall" sequence is discredited, there is a presumption that something logically equivalent to a specification has been constructed by the time this development is completed, even if this specification is not made explicit. This presumes that the behaviour of the resulting system is comprehensively captured in terms of protocols for agents, both human and non-human, that are based on functional abstractions.

- A legitimate concern for the theoretical computer scientist is the extent to which a software system without a formal specification can be regarded as a 'public entity'. Clarity about the

> interpretation of the software by machine and human on the part of all the participating agents seemingly cannot be achieved without recourse to formalism.

A natural consequence of this outlook is that software engineering aspires to be a form of 'theory-building' (such as is envisaged by Turski and Maibaum [62]). Jackson draws attention to the problematic nature of software engineering as 'science', highlighting the need to do justice to the subtlety and complexity of the engineering concerns that underlie complex software systems [38], and to steer a proper course between formality and informality [36] by exploiting two kinds of model he attributes to Ackoff [2]: "analytic models that are simplified descriptions of the real world" and analogic models (like databases) that create a "new reality" to be kept consistent with the real world [39] .

An instructive parallel can be drawn with way in which modelling is viewed in relation to scientific theories. By way of illustration, following Kargon [42], we may consider the different ways in which two schools of scientific thought in the nineteenth century approached the emerging theory of elasticity. The "mechanico-molecular" school postulated a physical theory based on deriving the properties of bodies from the putative structure and forces governing their molecules. Their aim was to demonstrate that the predictions deduced from this theory were corroborated by experience, and regarded this as confirmation of their hypothesis. The school of "purely analytical mechanicians" by contrast sought general formulae based on abstract high-level principles that obviated the need for complicated computation of mechanical forces. A 1821 paper by Navier developed equations of motion for elastic bodies from the "mechanico-molecular" viewpoint; these equations were subsequently derived without reference to any physical theory from a mathematical function describing a elastic body under strain discovered by George Green [42].

In broad terms, these two approaches to describing an elastic body correspond to two complementary ways of specifying a computation to determine its behaviour - in terms of atomic data and rules, or via an equational constraint. Either might be the basis for approaches to computing the behaviour of an elastic body using software, and they are representative of familiar computational paradigms. As explained by Kargon [42], Maxwell appreciated the motivation for "simplification and reduction of the results of previous investigation to a form in which the mind can grasp them", but was critical of both approaches. In postulating a physical structure without beginning with the phenomenon to be explained, the molecularists "are liable to that blindness to facts and rashness in assumption which a partial explanation encourages." In framing a mathematical formula in the spirit of the analytical mechanicians "we entirely lose sight of the phenomena to be explained." In Maxwell's own treatment of elastic bodies [51], he explained how the plausible physical theory adopted by Navier and others made predictions at odds with experimental observations. He then described an alternative physical theory informed by a dependency relation that could be experimentally verified linking the compression of the solid to the pressure applied at a point, and showed that this produced results more faithful to experiment.

As described by Kargon [42], concern that abstract mathematical models should be rooted in concrete physical phenomena was central to Maxwell's concept of proper scientific practice. His methods drew upon physical analogies or illustrations such as Faraday developed for electromagnetic phenomena in his experimental researches as well as "scientific illustrations" that posited common mathematical forms associated with concepts and laws drawn from different branches of science. Such analogies, intended to aid the intuition in the development of theories but then to be discarded, had a role in ensuring that mathematical models were more than a basis for blind calculation. As Maxwell explained in his address to the British Association in 1870, here summarised by Kargon [42]:

"The human mind," he said, "is seldom satisfied, and is certainly never exercising its highest functions, when it is doing the work of a calculating machine." The man of science must, of course, sometimes *temporarily* perform the role of such a calculating machine. However, the goal of this labor ultimately is the clarification of his ideas.

Despite the strong motivations for adopting it set out above, the conceptual framework surrounding theory-building in science does not marry well with the challenges of software engineering in key respects:

- The notion of a theory associated with a simplifying mathematical model to clarify our ideas has its place in natural science, where we may be entitled to expect some uniformity and elegance. The same qualities most definitely do not apply to many of the artificial and specific environments and components with which a software system must engage.

- Scientific theories are intended to serve a foundational function and have a degree of permanence that reflects the stability of the empirical world from which they are derived. It may be appropriate to discard the intuitions that are used to scaffold the construction of such a theory once it is established. The principal challenge in software engineering, by contrast, is to build conceptual structures that typically need to be revised regularly in response to changes to the world in which they operate.

- Software systems have to take account of human agents and personal, social and cultural factors that - at the very least - cannot be readily captured in laws or studied through experiment.

In what follows, we shall sketch a radical solution to the impasse that software engineering seems to have reached through having to reconcile formal representations with the complex, dynamic and human characteristics of the world of experience. The central constituents are: a different - and broader - notion of computer-based modelling ("Empirical Modelling") that involves a radical generalisation of basic principles that underlie the spreadsheet and exploits the power of computing technology as a source of interactive experience; a semantic account in the spirit of James that favours experientially-mediated associations as the basis of 'knowing'; and a pragmatic stance to objective, or negotiated, meaning that is in line with Latour's aspirations for *construction*.

8 An account of models based on experientially-mediated associations

The key factor that distinguishes a model that relies on experientially-mediated associations from one based on formal representations is the essential role for personal judgement in the conception of the model. This is in keeping with Wartofsky's characterisation of a model [65] (as cited by Lloyd [48]) as a *triadic* relation: "*a person* takes *something* as a model of *something else*" and is central to the treatments of modelling by Mahr [50], McCarty [52] and Gooding [30]. For model-building to serve as a process of *construction*, the model itself and the contextual relationships surrounding it, being an artefact that testifies to "the constructed and mediated character of a public entity", must evolve in such a way that its integrity is respected. The complexity of this process of evolution is reflected in the many axes along which the model migrates in the modeller's understanding: from provisional to assured, subjective to objective, specific to generic, personal to public. To support this migration, the model has to be fashioned, via interactions both initiated and automated, so that the views of many agents - human and non-human - are taken into consideration. This process of construction is in general open-ended and never absolutely resolved. These features manifest in different ways according to the applications in mind. In software engineering, as Mahr observes [50], the essential need for open-endedness stems from the way in which the implementation of a system affects its environment. In McCarty's vision for humanities computing [52], modelling is the means to support and enrich the negotiation of meanings with no expectation of ultimate closure. In his account of Faraday's experimental activities, Gooding [30] traces the way in which construals are revised and developed in conjunction with new conceptions and realisations.

Empirical Modelling (EM) is a body of principles and tools for developing models that are based on experientially-mediated associations. In EM, the role for the computer is broadly parallel to the role it plays in a spreadsheet (e.g. a spreadsheet of examination marks).

In the first instance, the computer maintains dependencies, so that when a value in the spreadsheet (e.g. the mark awarded to Alice for biology) is changed, other values that are contingent upon it (e.g. Alice's overall average) are updated automatically. The nature of these dependencies is determined by the context and intended interpretation. The spreadsheet fulfills its semantic function if the user is readily able to connect each of its cells with meaningful external observables, and if the way in which changes to these observables are synchronised on interaction respect these meanings (e.g. when Alice's mark is increased, her overall average also increases).

The meaning of a formal representation is expressed in propositional statements specifying relationships between observables that pertain universally. In the experientially-mediated associations of the spreadsheet, meaning is expressed via latent relationships between observables that reflect "expectations about potential immediate change in the current state". Such a representation is highly nuanced in precisely those ways that are most relevant to model-building in a constructivist idiom. "Potential for change", "expectations" and "immediacy" are implicit references to agency and context. In interactions with a spreadsheet prototype that embrace exploratory design and development, the meanings of the spreadsheet - as mediated by different agents acting and observing in different roles - are associated with patterns of interaction and interpretation with which the model-builder becomes familiar over time.

Whereas formal representations are suited to expressing objective information about systems, experientially mediated associations are suited to tracing what might be termed 'personal' views of 'live' entities. A formal representation draws on previous experience with a view to abstracting insights to sum up this experience by eliciting the implicit constraints and rules that are deemed to govern change. Associations that are experientially mediated are oriented towards capturing the impressions we experience moment-by-moment and recording the way in which these impressions cumulatively and dynamically build up conceptions of entities. Where the use of the term "experience" in relation to formal representations is qualified by "previous" to convey the idea of reflection and rationalisation, the reference in experientially-mediated associations is to current living experience, in the spirit of Dewey's "actual focusing of the world at one point in a focus of immediate shining apparency …" [27, Ch.1 Sec.1]. And where the characteristic emphasis in formal representations is on abstract rules for interpreting symbols, the semantic connections that are invoked by experientially-mediated associations are themselves 'given in experience' as endorsed by William James's *radical empiricist* stance.

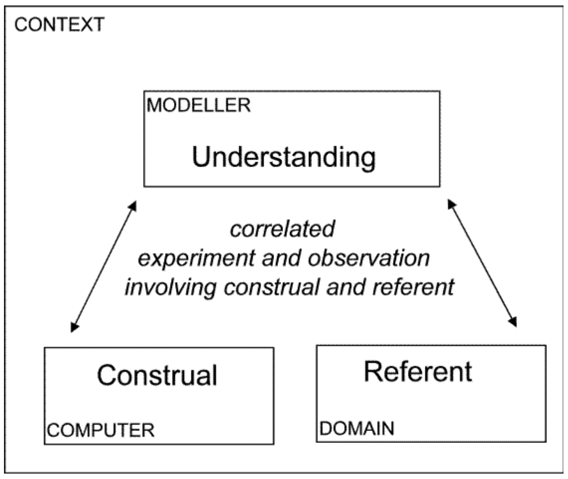
CONTEXT
MODELLER
Understanding
correlated experiment and observation involving construal and referent
Construal
COMPUTER
Referent
DOMAIN

Figure 1: The key ingredients of Empirical Modelling

In EM, four live ingredients of the modelling environment co-evolve: the computer-based construal, the external phenomenon to which the construal refers, the modeller's understanding, and the context within which interaction with the construal is being interpreted. The relationship between these ingredients is represented diagrammatically in Figure 1 - superficially an unremarkable diagram that might be drawn to express the core relationships associated with any variety of modelling, but one whose interpretation here is unusual since these relationships are themselves live. In appreciating this, it is helpful to consider how the corresponding ingredients of a spreadsheet are experienced. In deploying a spreadsheet, the values and definitions that will be associated with observables represented by cells will depend upon the current context ("When will the chemistry results be available?"), relate to agency over which we have limited control ("What mark will Alice get in chemistry?"), are subject to be modified in ways that were not preconceived ("The marks for Bill and Ben Smith were entered the wrong way round" and that might involve exceptional kinds of agency ("It would be useful to add a column to record how many first class marks candidates get in core subjects"), and prompt changes to the modeller's understanding ("I thought that Bill and Ben's marks were out of line with their overall performance"). Equally diverse are the ways in which the spreadsheet can be viewed ("How did Bill perform overall?", "How did students perform in my module?", "How transparent is the computation of each student's overall mark?", "How does the average mark for my module compare with that of other modules?"). In this respect, the range of possible observational viewpoints and affordances is well-matched to what is required of construction, as discussed above.

The fundamental EM concepts of *observable*, *agency* and *dependency* that have here been illustrated with reference to the spreadsheet are the basis for a very general variety of modelling that can be applied across many disciplines. The tabular interface that mediates state in a spreadsheet is not representative of general EM construals. The net of observables and dependencies associated with a situation is rich and open-ended, and a variety of different modes of visualisation and manipulation is appropriate if the perspectives and "experience" of the diverse agents who observe and interact are to be reflected. It might be appropriate to attach a line-drawing visualisation to the exam spreadsheet, for instance, so that the seating arrangements for an examination could be displayed, or to establish a dependency between a timeline and the spreadsheet to indicate which examinations took place in a particular period. The limits on the possible scenarios and extensions to a construal that can be investigated in this way are set only by the modeller's imagination and the quality of the supporting tools.

The scope of EM is best appreciated by considering the many different kinds of agency, human and non-human, internal and external, that are associated with a complex system that combines human, software and hardware components. Relevant agents in such a setting include systems designers, software developers, users and managers, agents automated in software, devices such as sensors and actuators, and environmental sources of state change such as weather, noise and light. The ontological status of the components an EM construal differs according to the application – as is illustrated in the EM project archive [73], a construal may have as its referent: a 'real-world' scenario in which the capabilities of individual agencies is well-understood but their corporate behaviour is to be investigated (e.g. as in a railway accident); a radical design activity in which some of the agents are pre-existing devices but others are in the process of being developed or programmed in an exploratory fashion; a personal experience (e.g. appreciating a piece of music, or working in a restaurant). The most important distinction between an EM construal and a conventional model is that a construal is a source of live interactive experience for the modeller, who is free to rehearse interactions with it and explore possible interpretations. In this respect, it is radically different in character from a model that is conceived as a simplification of its unfathomably rich real-world referent (cf. a typical "scientific model") and is derived by a process of abstraction. A parallel may be drawn with the distinction between writing prose and writing poetry to which the poet Don Paterson alludes in [57]. Paterson argues that ambiguity is an essential quality of poetic use of language, in sharp contrast to uses of language that aim at precision and clarity of expression.

9 Empirical Modelling as *construction*

The problem of understanding the nature of modelling required in software development can be seen as linked to the problem of identifying an acceptable concept of *construction* to which Latour alludes in [46]. To give a good account of how "the entities whose public existence has to be discussed" are constructed, it is essential to have a way of 'representing' what is *not yet* a public entity that can gain the approval of critics from the whole spectrum of viewpoints identified by Latour in [46]. The use of the term 'representing' in this context is slightly paradoxical, as 'representing' itself has overtones of 'making public'. But it is quite apparent that - in order to convince the "scientific" sceptic - we must be able to defend and give evidence to substantiate a claim that one entity is associated with another by invoking some device other than formal representation. And - in order to confound the deconstructionist - we must be able to demonstrate that these associations are not arbitrary products of the imagination that can have no compelling public authority.

Our thesis is that Empirical Modelling principles provide just such a means of creating associations as is required of a legitimate notion of construction. The crucial difference between the experientially-mediated associations established in EM and formal representations is that they are of-their-essence contextualised - they relate to the live experience of human observers in the presence of a customised environment for exploratory interaction. To some degree, the plausibility of this thesis can be assessed simply by seeking empirical corroboration, but it also commends - perhaps even demands - a philosophical reorientation of just such a kind as is the central theme of William James's *Essays in Radical Empiricism* [40]. Specifically, our thesis only makes sense subject to accepting certain prerequisite ideas:

- It presumes experientially-mediated associations can be empirically given in the experience of a human modeller. The force of 'empirically given' is that an association has no further explanation: an association either is or is not experienced. Both these assumptions are in line with James's radical empiricist stance [40].

- It deems such associations to be objectively real provided that interaction with other modellers indicates that they too experience the 'same' assocations. This nature of the interaction involved in justifying the claim of 'shared experience' is discussed at length by James [40]. In keeping with James's stance, his account explains how this interaction is itself rooted in experientially-mediated associations.

- It recognises a role for interactive artefacts in communication that goes beyond the use of formal representations, and in particular, mediates understanding with reference to the way in which change of state of an interactive artfact can be effected and experienced by the modeller. In this way, the most

> primitive knowledge takes the form of expectations latent in the current state to the effect that "if this feature changes, it will conceptually in one and the same transition affect other features in a certain recognisable way". These expectations (or 'dependencies' as they are described in EM) can also be viewed as experientially-mediated associations on account of the way - though they are associated with latent actions - they affect our perception of the current state. The idea that experientially-mediated associations are more expressive than formal representations is (in effect!) also endorsed by James [40].

The full development of these ideas has been the subject of extensive study relating to EM [70]. The large body of EM construals that have been developed, principally by computer science students who have studied EM at Warwick over the last twenty years, provides some of the empirical corroboration for our thesis. Particularly significant is the range of subjects that have been addressed (construals drawn from engineering, business, science, medicine, music) and the character of the most successful applications, which relate to themes in which the transition from private to public understanding is central (e.g. learning, requirements cultivation, decision support, personal sense-making).

In the context of this chapter, one of the most relevant themes that has been considered from an EM perspective (cf. [11]) is that of "supporting exploratory experiment". To illustrate how more useful links between modelling in science and modelling in software engineering might be established, it is helpful to look more closely at the way in which Faraday's experimental researches were conducted, and how their findings were related to the theory of electromagnetism. The adoption of the term 'construal' (rather than 'model') in EM was itself motivated by the way in which Gooding applied this concept in his account of Faraday's work [30]. Our speculative proposal is that Faraday's research activities can also be interpreted 'as if' within the conceptual framework of EM, with particular reference to the thesis about constructing public entities set out above.

The development of electromagnetic theory has been a particularly fruitful source of ideas about the significance of models. This is in part because of the wide range of activities represented in this development, from Ampère to Faraday to Maxwell, and the extent to which these have been documented (e.g.) through Maxwell's own reflections [19,51] and in the work of Gooding [30].

At a meeting entitled *Thinking Through Computing*, organised by the EM research group at Warwick University in November 2007, Gooding [31] reflected on how his notion of 'construal' had influenced the development of EM. He likened the traditional theoretical account of computer science, with its emphasis on characterising behaviours declaratively using functional abstractions, to the mathematical theory that was developed to account for Ampère's studies of electricity. Characteristic of Ampère's researches was the study of electrical phenomena in equilibrium. This was in strong contrast to Faraday's research, in which intervention and dynamic phenomena were key ingredients, for which accordingly pre-existing theory was inadequate. Observing the strong parallel between Faraday's construals and EM artefacts, Gooding remarked that theoretical computer science might presently be in something like the same state as electromagnetic theory prior to the work of Maxwell and Thomson. He speculated that in due course similar developments in theoretical computer science might accommodate the idea of EM construals through a broadening of the models within its mathematical framework.

Where modelling to support software engineering is concerned (especially in the context of radical design or humanities computing in the spirit of McCarty [52]), our previous discussion suggests another more radical possibility: that the primary focus of interest is shifted to construals, as complementing mathematical models in an essential way, and as better matched to the demands of the application. The spirit of Gooding's account of Faraday's work is well-expressed in the following quotation from Gorman [32]:

"Gooding believes that Faraday progressed via "a convergence of successive material arrangements (the apparatus) and successive construals (or tentative models) of the manipulation of the apparatus, and its outcomes" [30, p187] A construal corresponds roughly to what I am calling a mental model. Gooding's point is that, for Faraday, ideas, physical manipulations and objects are closely linked."

The blurring of mind-body duality that is represented here is itself an indication that the construal has neither the idealised qualities of an abstract mathematical model nor the status of a natural reality that a scientist might attribute to its referent. The resistance to accepting such a construal as a useful variety of *model* is vividly reflected in Duhem's scornful comment [28] on the lamentable judgement of English physicists, as cited by Black [19]:
"In place of a family of ideal lines, conceivable only by reason, he will have a bundle of elastic strings, visible and tangible, firmly glued at both ends to the surfaces of the two conductors, and, when stretched, trying both to connect and to expand. When the two conductors approach each other, he sees the elastic strings drawing close together; then he sees each of them bunch up and grow large. Such is the famous model of electro-static action designed by Faraday and admired as a work of genius by Maxwell and the whole English school."
Such criticism reflects certain preconceptions about the nature and purposes of 'modelling'. What is being modelled (in this case, "electro-static action") is being conceived as already part of a public reality, and the model (as defined by mathematical equations) is being understood as a simplification or idealisation derived by abstracting from this pre-existing reality. It is precisely because modelling principles and tools have traditionally been conceived in this way that they are so ill-suited for applications such as exploratory experiment and radical design.

To put Duhem's criticism of Faraday's construals in perspective, it is appropriate to view Faraday's experimental work (albeit on electro-magnetic rather than electro-static phenomena) within a constructivist frame of reference as giving insight into "the constructed and mediated character of the entities whose public existence has to be discussed". Whereas today an electromagnetic device is "an entity whose public existence has to be discussed" - most educated people have some notion of how an electric light or an electric motor works

- no such public entities were established in the initial stages of Faraday's research.

It seems plausible that Faraday conceived what Gooding characterises as his "construals" as modelling yet-to-be-discovered causal laws. This would be in accordance with Cantor's account [24] of Faraday's Sandemanian religious faith: "... he accepted that nature was governed by a set of God-given causal laws. These laws were ... the work of a wise Creator who, avoiding unnecessary complexity, constructed the world on simple, plain principles." The preliminary role that these construals served was to represent patterns of interaction and interpretation associated with as-yet-ill-understood phenomena that began to emerge over an extended period of exploratory experimentation. Identifying such patterns meant establishing contexts in which to discern within the phenomenon itself specific entities whose status might change, means to effect changes to these entities and instances of contingent changes that could be interpreted as a consequence of causal laws.

In order to identify such patterns, and potentially to communicate them to other scientists, it was essential for Faraday to find a means to represent them: the construal served this purpose as a physical construction made of familiar entities (cf. "a bundle of elastic strings, visible and tangible, firmly glued at both ends to the surfaces of the two conductors" in Faraday's construal of electro-static effects, as described by Duhem) that responded to change on the same pattern, 'as if' obeying the same causal law. As is well-documented in Gooding's account, some of the patterns that Faraday recorded in this way could be reliably reproduced and were embodied in construals, but were nonetheless later deemed to be associated with unsatisfactory experimental techniques or with misconceptions. The limitations of such construals were not to do with the quality of the experientially-mediated association between an artefact made of commonplace components and a putative 'electromagnetic phenomenon' *per se*, but with their relationship to Faraday's broader aspirations for causal laws that might be deemed to be 'God-given' and 'avoiding unnecessary complexity'. The development of new construals and experimental apparatus and protocols was then aimed at eliminating procedural complexity that required exceptional observational skill and capability on the part of the experimenter, and dependence on features of the local environment. Developments of this kind are precisely aligned with the idea of natural science as relating to universal experience, and with the "creation" or "discovery" of "entities whose public existence has to be discussed".

As Gooding observes [30], what Faraday communicated through his construals was not propositional knowledge - it was a body of interactions in the world, carried out in suitably engineered contexts through exercising certain skills, that followed a predictable and reproducible pattern. On the one hand, this body of interactions could be enacted (either physically, or as a thought-experiment) on the construal as an interactive object whose parts are constituents of our everyday human 'shared experience'. On the other hand, it had a counterpart that could be enacted in the realm of unfamiliar electromagnetic phenomena. Most crucially, the 'same' experientially-mediated association between these two modes of interaction that Faraday engineered personally and locally could also be experienced publicly and universally.

The respect that "the English physicists" had for Faraday's construals stemmed from their appreciation of the connection they established between the personal and public arena: guiding the intuitions of the learner, connecting abstract mathematical models in mind with interventions in the laboratory, and exposing the deceptive simplicity of after-the-fact accounts of experimental discoveries. As has been explained, making construals of this nature seems much more relevant to the challenges of software engineering - and realistic - than aspiring to generate "a mathematical model". As for the potential relevance of EM in this connection, the above sketchy account of Faraday's approach has been mischievously written in such a way that it refers in all but name to 'observables' ("specific entities whose status might change"), 'agencies' ("means to effect changes to these entities") and 'dependencies' ("instances of contingent changes"). To what extent this might misrepresent Faraday's experimental activities is a question beyond the scope of this chapter that merits further work for which Gooding's studies provide an excellent resource (cf. the computational model of Faraday's cognitive processes proposed by Addis, Gooding and Townsend [5]).

10 Construals and construction for software

In conventional software development, the need for what is described by Fetzer [29] as 'conceptualisation' of a domain prior to the development of use-cases, requirements and specification is well-recognised. The unusual nature of the modelling activity in EM, and its strong affinity with sense-making activities, make it possible in principle to address this conceptualisation in an alternative, constructivist, manner. The parallel between EM construals and Faraday's construals is again instructive. As described in detail by Gooding (cf. [30] and [32]), Faraday's construction of a prototype electric motor was the result of a methodical activity through which he was able to infer how forces associated with electromagnetic phenomena could be harnessed by configuring magnets and coils. In EM terms, Faraday's construals can be viewed as embodying insights about the nature of the observables, agencies and dependencies that are characteristic of electromagnetic phenomena. Their dual physical and mental nature enabled him both to experiment and to reason about how to dispose these observables, agencies and dependencies so as to achieve a desired behaviour. As Gooding suggests in [31], the characteristics of knowledge representation in traditional software engineering, which focuses on identifying structures, constraints and invariants that are intended not to be subject to change, resemble the mathematical theory developed by Ampère in connection with his research on electromagnetism "based on experiments in which nothing happens". Since the "busy phenomenology" that surrounds construals is a rich resource for the practical engineer, it is unsurprising that Faraday's research generated many more engineering products than Ampère's; building software with EM construals has a somewhat similar power to provoke experimental design and stimulate the imagination.

The potential for developing software based on EM construals has been illustrated in several practical studies (see e.g. [14,43]) and discussed in several previous publications [11, 12, 13, 16, 18, 58]. Work is in progress on tools to exploit EM principles in supporting software development on a larger scale (cf. [58]), but previous research has established proof-of-concept for many aspects of the activity. Whereas traditional software engineering has distinct phases (requirements, specification, implementation, testing) in which entirely different activities and representations are featured, development based on construals is much more homogeneous. Just as Faraday's prototype electric motor is 'simply' a way of disposing mechanisms that were embodied in pre-existing construals, so a functional piece of software can be composed from suitably engineered EM construals. The development process is associated with a seamless elaboration of patterns of interaction, engineering of contexts, acquisition of skills in observation and manipulation during which the products of the EM activity undergo conceptual transitions (cf. Figure 1).

In the initial stages, the development of a construal is primarily concerned with identifying the most primitive observables and dependencies in the domain and finding ways to express these using suitable metaphors. At this stage, the emphasis is on making an *artefact* that the modeller can learn to interact with to change observables and recognise dependencies in a predictable way (cf. devising a new scientific or musical instrument). Once such an artefact is suitably refined, so that the modeller can interact conveniently to generate reliable responses, it can be used in a more focused manner to develop counterparts of specific ensembles of observables and dependencies in the domain that can be reliably identified. At this stage, the modeller becomes familiar with particular ways of interacting with the EM artefact and its counterpart in the domain, acquires the necessary skills to support these interactions and learns how to shape the context appropriately. At this point, the artefact comes to serve the sense-making role of a *construal*. Characteristic of interactions at this stage is the open-ended exploratory character of the investigation - the modeller engages in sense-making activities because of uncertainty about the nature of the referent and its possible responses. In a software development context, such activities are appropriate for radical design.

A construal may be legitimately regarded as a *model* when its referent has, or acquires, the status of an established feature or component of the domain and all the characteristic interactions and operations associated with the referent are faithfully reflected in the construal. In a routine design exercise, such components may be known in advance, and devising construals to serve as models of them is a natural initial target for the EM development activity. A notable feature of the development of the prototype electric motor is the way in which state-changes that were first identified through what Gooding [31] calls "experiments in which things move, twist and jump" are "- with some difficulty - constrained in a dynamic, unstable equilibrium". Software development can be envisaged in an analogous way as - in general "with some difficulty" - engineering contexts in which to place construals so that they autonomously and reliably change state in appropriate ways. Within such a context, a construal has even more constrained behaviour than a model, and may be deemed to be a *program*.

It is difficult to appreciate the character of these transitions as abstractly described - practical experience of an EM exercise is ideally required. It is helpful to keep in mind the idea that an EM artefact/construal/model/program is always presented as a net of observables and dependencies that, like a spreadsheet, is to be interpreted as a live state of affairs. Its classification changes only because the human interpreter wraps it in a different context and conceives it in relation to a different family of pre-engineered interactions and interpretations. By way of illustration (as is described in detail in [13]), an artefact that resembles a heap data structure can *first* be developed into a construal that makes sense of the basic operations associated with sorting using heapsort (such as might be used in teaching), *then* to a model of the heap data structure on which the heapsorting algorithm can be rehearsed manually, *then* to a program in which the steps of the algorithm are fully automated and visually displayed. The transitions that the EM artefact undergoes in this activity are such that what the modeller can initially interact with in a very unconstrained manner (as a child might play with a set of jointed rods representing a tree data structure for instance) eventually becomes the focus of a much more disciplined interaction where the interpretation is no longer freely invoked by the modeller's imagination but is prescribed by public conventions about the nature of heapsort. Significantly, the invariants of the algorithm, which are traditionally the basis for its formal specification, here enter as sophisticated kinds of observable that are present so long as the interaction with the EM artefact is constrained to follow the correct protocol.

11 EM and some key issues in modelling for software

The above discussion highlights three key issues that are topical in modelling for software applications: the nature of models, the semantic scope that their use affords, and the need to combine formal and the informal representations coherently. In conclusion, we consider the potential impact of EM upon these issues.

EM offers a new perspective on the nature of models. Reflection on EM practice reveals the exceptional semantic subtlety of what can be embodied in an artefact by appealing to experientially-mediated associations. Appreciating this subtlety, even as it applies to such a simple algorithmic activity as heapsort, makes it quite apparent why it is so difficult to define the notion of 'model'. Though the term 'transition' has been used to describe the way in which the modeller's conception of an EM artefact can evolve, it is the modeller's conception, as defined by the attendant contexts and patterns of interaction and interpretation, that changes: the artefact itself remains no more than a net of observables and dependencies to which agencies, of which the modeller is but one, can be attached. Formal language is predicated on being able to associate a specific object or category with a word, but this is not well-matched to the way in which an EM artefact is experienced. In EM activities, object boundaries are blurred: what are initially conceived as distinct nets of observables and dependencies relating to different contexts may be composed (cf. the exam mark grid, the exam seating arrangement) and become a new net of observables and dependencies; categories are likewise blurred: it may be that, according to the modeller's current purpose and state-of-mind, a given net of observables and dependencies can be as-of-now taken as artefact, construal, model or program.

The transitions that the artefact undergoes in its relationship to the modeller help to clarify the way in which philosophers of science interpret the term 'model'. Within the frame of reference of science, the transitions can be thought of as typically moving from the personal to the public world, and it is with models that refer to public entities that such philosophers are concerned. When Duhem [28, p70] speaks disparagingly of the models of the English physicists, and Braithwaite [20] declares that "the price of the employment of models is eternal vigilance", the notion of model and the modelling context they have in mind is far from the construals that Gooding characterises as a 'tentative models'. It is not even clear that it is appropriate to interpret Achinstein's reference to "models which are simply supposed to provide possible mechanisms for how natural systems might be operating" as conceived as applying to such construals, though that interpretation suits our purpose in this chapter (cf. section 6). More ambiguous - and provocative - is the way in which Black [19] introduces the term "conceptual archetype" to refer to the kind of mathematical models akin to the 'scientific illustrations' favoured by Maxwell (of which the analogies between electromagnetic fields and fluid flow is the archetype), and then goes on to liken the use of such conceptual archetypes to "speculative instruments" in the sense of the English literary scholar I.A. Richards [59]. This draws attention to the second key issue that is relevant to our central theme.

The study of the 'artefact-construal-model-program' axis brings together two quite different cultures, according to whether we put the emphasis on the personal or the public. Taking full account of the personal realm is the aspiration of the arts and humanities, which are concerned with the universal human experience. By contrast, the universality that science seeks relates to the public sphere, and concerns what can be communicated objectively. As is more fully discussed in [17], the scope for shifting the perspective from personal to public in model-building that EM affords is potentially a basis for bridging the two cultures.

The way in which Black [19] likens Maxwell's 'scientific illustrations' to I.A. Richards's 'speculative instruments' also hints at this possibility. At the end of his chapter [19], Black writes:

"When the understanding of scientific models and archetypes comes to be regarded as a reputable part of scientific culture, the gap between the sciences and the humanities will have been partly filled."

The problem of addressing the subject of modelling in its full generality from a scientific perspective has been highlighted in the process of framing the vocabulary for this chapter. (An interesting comparison - and contrast - may be made with the problem discussed by Stephen Wolfram in his blog *The Poetry of Function Naming* [69], where words to describe a very precise functional abstraction are being sought.) The distinctions between two ways of seeing, between the computationalist and constructivist, between formal representations and experientially-mediated representations are not sharp distinctions such as science might endorse; they relate to qualitatively different but inseparable aspects of experience that are encountered at one-and-the-same time. The number in the spreadsheet cell *is* an arithmetic abstraction *and* 'Alice's mark for Geometry' and *'the reason why Alice is awarded a prize'*, and our skill as an examiner is in being able to experience all these associations at once. The notion of such - sometimes seemingly paradoxical - conjunctions in experience is at the core of James's radical empiricism [40] and is reflected in the ambiguous personal/public qualities of entities under construction [46]. More extensive and detailed discussion of the affinities between EM and radical empiricism and EM and constructivism, aimed at establishing EM as a "reputable" activity, can be found in other papers [8,9]. But - despite the impressive precedent for the use of construals in Faraday's experimental work - these affinities and the evidence of the 'science wars' suggest that it may be some time before EM is "regarded as a reputable part of *scientific culture*".

It is perhaps easier to relate the promise of EM to the arts and humanities. As explained by M.M. Lewis in a review of I.A. Richards's book "Speculative Instruments" [47,59], the problem that motivated Richards to conceive language as a speculative instrument was that "the exploration of comprehension is the task of devising a system of instruments for comparing meanings" [59]. And since Richards perceived language as the chief instrument with which we think, he concluded that "the very instruments we use if we try to say anything non-trivial about language embody in themselves the very problems we hope to use them to explore" [59]. As Faraday's experimental work illustrates so potently, artefacts can have power above and beyond language alone. The same principle is illustrated by considering the way in which different constructions on the 'artefact-construal-model-program' axis can carry nuances of meaning for which none of these words is adequate - nuances which indeed defy lexical definition (cf. the illustrations of a similar nature concerning words such as 'time', 'state' and 'agent' that are discussed in [7]). This suggests that it is more appropriate to liken construals in science - rather than conceptual archetypes, to speculative instruments in the humanities.

The blending of meanings that construals afford places the issue of dealing in a coherent fashion with formal and experiential models in a new light. As the heapsort modelling exercise outlined above illustrates, it is possible to situate a formal representation within a context moulded from experientially-mediated associations. Such a formal representation comes into being only because the engineering of the context for interaction and interpretation, and the discretion the modeller exercises in interacting and interpreting, disclose a stable pattern in experience that can be appreciated 'universally'. The acts of "engineering" and "exercising discretion" are evidence of a simplification of experience similar in nature to the "simplification and reduction of the results of previous investigation to a form in which the mind can grasp them" to which Maxwell alludes (as discussed in section 7 above). The virtue of modelling with construals is that the modeller can take account of such simplifications without needing to make absolute commitments; the patterns of observables, dependencies and agency that sustain different kinds of interaction and interpretation are always fluid and open to revision, and can be organised and recorded in such a way that the modeller can switch viewpoint at will.

As mentioned by Muldoon [53, p18], Faraday prefaced the first edition of his first book *Chemical Manipulation* (1827) with Trevoux's slogan "C'est n'est pas assez de savoir le principes, il faut savoir MANIPULER" - "It is not enough to know the principles, it is necessary to know how to MANIPULATE". For Faraday, construals were a means of representing and communicating ideas richer than formal mathematical models. According to the thesis set out in section 9 above, the expressive power of construals stems from their capacity to embody patterns of observables, dependencies and agency. In Faraday's time, the scope for this embodiment was limited by the available technology - many of the interactions with construals that Faraday conceived were enacted only in his imagination. It is in part on account of these technological limitations that such importance is attached to developing theoretical models that can to some extent obviate the need for 'manipulation' by distilling abstract 'principles'. Building software on the basis of functional abstractions is likewise a technique that was developed with the limitations of the early computing technology in mind: it permits that high degree of optimisation that remains one of the central preoccupations of computer science. But the pervasive influence of these two simplifying strategies should not seduce the computer scientist into imagining that all experience can be conceived in terms of functional abstractions. As the modes of knowledge representation for AI advocated by Brooks indicate [22,23], the theory of computing must also embrace principles that take account of what it means to "know how to manipulate". This chapter has set out to show why EM is a promising basis for such a theory. It also hints at what might be achieved by investing as much effort in developing effective tools for developing construals as has been dedicated to tools for developing programs.

Acknowledgments

I am much indebted to Steve Russ for drawing my attention to useful resources, for supplying the title and most of the introduction, for invaluable editorial help and for vital contributions he has made to the development of key ideas. I also wish to thank Russell Boyatt, Nicolas Pope and many other students who have studied EM for sharing their insights into software development.

References

1. Achinstein, P. (1968). *Concepts of Science: A Philosophical Analysis*. Baltimore: The Johns Hopkins Press.

2. Ackoff, R.L. (1962). *Scientific Method: Optimising Applied Research Decisions*. New York: Wiley.

3. Addis, T. R. (1993). Knowledge Science: a Pragmatic Approach to Research in Expert Systems. In *Proceedings BCS SGES Expert Systems '93*, (pp. 321-339). St. John's College, Cambridge, 13-15 December 1993.

4. Addis T.R., & Gooding D.C. (2008). Simulation Methods for an Abductive System in Science. *Foundations of Science,* 13(1), 37-52.

5. Addis T. R., Gooding D. C., Townsend J. J. (1991). Modelling Faraday's Discovery of the Electric Motor: An investigation of the application of a functional database language. In *Proceedings of the Fifth European Knowledge Acquisition for Knowledge-Based Systems Workshop,* Crieff Hydro, Scotland, 20-24 May.

6. Bentley, P.J., & Corne, D.W. (Eds.), (2001). *Creative Evolutionary Design Systems.* Morgan Kaufmann.

7. Beynon, W.M. (1999). Empirical Modelling and the Foundations of Artificial Intelligence. In C.L. Nehaniv (Ed.), *Computation for Metaphors, Analogy and Agents* (pp. 322-364). Lecture Notes in Artificial Intelligence 1562, Springer.

8. Beynon, W.M. (2005). Radical Empiricism, Empirical Modelling and the Nature of Knowing. *Cognitive Technologies and the Pragmatics of Cognition: Special Issue of Pragmatics and Cognition,* 13(3), 615-646.

9. Beynon, M. & Harfield, A. (2007). Lifelong Learning, Empirical Modelling and the Promises of Constructivism. *Journal of Computers*, 2(3), 43-55.

10. Beynon, W.M. & Joy, M.S. (1994). Computer Programming for Noughts-and-Crosses: New Frontiers. In *Proc. PPIG'94* (pp. 27-37). Open University, January 1994.

11. Beynon, M. & Russ, S. (2008). Experimenting with Computing. *Journal of Applied Logic* 6, 476-489.

12. Beynon, W.M. & Sun, P-H. (1998). Empirical Modelling: a New Approach to Understanding Requirements. In *Proc. 11th International Conference on Software Engineering and its Applications* Vol.3, Paris, December 1998.

13. Beynon, W.M., Rungrattanaubol, J., Sinclair, J. (2000). Formal Specification from an Observation-Oriented Perspective. *Journal of Universal Computer Science,* 6(4), 407-421.

14. Beynon, W.M., Ward, A., Maad, S., Wong, A., Rasmequan, S., Russ, S. (2000). The Temposcope: a Computer Instrument for the Idealist Timetabler. In *Proc. of the 3rd International Conference on the Practice and Theory of Automated Timetabling* (pp. 153-175). Konstanz, Germany, August 16-18.

15. Beynon, W.M., Roe, C., Ward, A, Wong, A. (2001). Interactive Situation Models for Cognitive Aspects of User-Artefact Interaction. In M. Beynon, C.L. Nehaniv, K. Dautenhahn (Eds.), *Cognitive Technology: Instruments of Mind* (pp. 356 - 372). Lecture Notes in Computer Science Vol.2117, Springer.

16. Beynon, W.M., Boyatt, R.C., Russ, S.B. (2006). Rethinking Programming. In *Proceedings IEEE Third International Conference on Information Technology: New Generations (ITNG 2006)* (pp. 149-154). April 10-12, 2006, Las Vegas, Nevada, USA 2006.

17. Beynon, W.M., Russ, S.B., McCarty, W. (2006). Human Computing: Modelling with Meaning. *Literary and Linguistic Computing,* 21(2), 141-157.

18. Beynon, M., Boyatt, R., Chan, Z.E. (2008). Intuition in Software Development Revisited. In *Proceedings of 20th Annual Psychology of Programming Interest Group Conference*. Lancaster University, UK, September 2008

19. Black, M. (1962). Models and Archetypes. In M. Black, *Models and Metaphors* (pp. 219 – 243). Cornell Univerity Press.

20. Braithwaite, R.B. (1953). *Scientific Explanation.* Cambridge: Cambridge University Press.

21. Boden, M.A. (2003). *The Creative Mind: Myths and Mechanisms.* Routledge.

22. Brooks, R. A. (1991). Intelligence without Representation. *Artificial Intelligence,* 47, 139 -159.

23. Brooks, R. A. (1991). Intelligence without Reason. In *Proc. IJCAI-91* (pp.569 – 595). San Mateo, CA: Morgan Kaufmann.

24. Cantor, G.N. (1986). Reading the Book of Nature: The Relation Between Faraday's Religion and His Science. In David Gooding & Frank A.J.L.James (Eds.) *Faraday Rediscovered. Essays on the Life and Work of Michael Faraday, 1791-1867*. London: Macmillan.

25. Cantwell-Smith, B. (1987). Two Lessons of Logic. *Computational Intelligence*, 3, 214 - 218.

26. Cantwell-Smith, B. (2002). The Foundations of Computing. In M. Scheutz (Ed.), *Computationalism: New Directions* (pp. 23–58). Cambridge, MA: MIT Press.

27. Dewey, J. (1916). *Essays in Experimental Logic.* Chicago: University of Chicago.

28. Duhem, P. (1954). *The Aim and Structure of Physical Theory*. New York: Atheneum.

29. Fetzer, J.H (1999). The Role of Models in Computer Science. *The Monist*, 82(1), 20-36.

30. Gooding, D. (1990). *Experiment and the Making of Meaning: Human Agency in Scientific Observation and Experiment*. Dordrecht: Kluwer.

31. Gooding, D. (2007). Some Historical Encouragement for TTC: Alchemy, the Calculus and Electromagnetism. At url: http://www2.warwick.ac.uk/fac/sci/dcs/research/em/thinkcomp07/gooding2.pdf (Accessed 10/5/2011)

32. Gorman, M. E. (1997). Discovery as Invention: Michael Faraday. Section 1.3 in *Invention and Discovery: A Cognitive Quest*. Online at http://cti.itc.virginia.edu/~meg3c/classes/tcc313_inuse/Resources/gorman.html (draft of a book later published as *Transforming Nature: Ethics, Invention and Discovery*, Kluwer Academic Press, 1998) (Accessed 10/5/2011)

33. Harel, D. (1988). On Visual Formalisms. *Communications of the ACM*, 31(5), 514-530.

34. Harel, D. & Marelly, T. (2003). *Come, Let's Play: Scenario-Based Programming Using LSCs and the Play-Engine*. Springer.

35. Humphreys, P. (2004). *Extending Ourselves: Computational Science, Empiricism, and Scientific Method*. Oxford: Oxford University Press.

36. Jackson, M.A. (2000). *Problem Frames: Analysing & Structuring Software Development Problems*. Addison-Wesley.

37. Jackson, M.A. (2005). Problem Frames and Software Engineering. *Information and Software Technology*, 47(14), 903-912. Online at http://mcs.open.ac.uk/mj665/PFrame10.pdf. (Accessed 12/5/2011)

38. Jackson, M.A. (2006). What can we expect from program verification? *IEEE Computer*, 39(10), 53–59.

39. Jackson, M. (2008). Automated software engineering: supporting understanding. *Automated Software Engineering*, 15(3-4), 275–281.

40. James, W. (1912/1996). *Essays in Radical Empiricism*. (Reprinted from the original 1912 edition by Longmans, Green and Co., New York) London: Bison Books.

41. Kaptelinin, V. & Nardi, B.A. (2006). *Acting with Technology: Activity Theory and Interaction Design*. The MIT Press.

42. Kargon, R. (1969). Model and Analogy in Victorian Science: Maxwell's Critique of the French Physicists. *Journal of the History of Ideas*, 30(3), 423-436.

43. Keer, D., Russ, S., Beynon, M. (2010). Computing for construal: an exploratory study of desert ant navigation. *Procedia Computer Science,* 1(1), 2207-2216.

44. Kramer, J. (2007). Is abstraction the key to computing? *Communications of the ACM*, 50(4).

45. Lakoff, G. & Johnson, M. (1980). *Metaphors We Live By*. University of Chicago Press.

46. Latour, B. (2003). The Promises of Constructivism. In Don Ihde and Evan Selinger (Eds.), *Chasing Technoscience: Matrix for Materiality* (pp. 27 – 46). Bloomington & Indianapolis: Indiana University Press.

47. Lewis, M.M. (1956). Review of *Speculative Instruments* by I.A. Richards. *British Journal of Educational Studies,* 4(2), 177-180.

48. Lloyd, E.A. (2001) Models. In *Encyclopedia of Philosophy*. London:Routledge.

49. Loomes, M. J. & Nehaniv, C. L (2001). Fact and Artifact: Reification and Drift in the History and Growth of Interactive Software Systems. In M. Beynon, C.L. Nehaniv, K. Dautenhahn (Eds.), *Cognitive Technology: Instruments of*

Mind (pp. 25-39). Lecture Notes in Computer Science Vol.2117, Springer.

50. Mahr, B. (2009). Information science and the logic of models. *Software and System Modeling* 8(3): 365-383

51. Maxwell, J.C. (1890). *The Scientific Papers of James Clerk Maxwell*, Volume 1 (ed. W.D.Niven) Cambridge University Press.

52. McCarty, W. (2005). *Humanities Computing.* Basingstoke: Palgrave Macmillan.

53. Muldoon, C.A. *Shall I Compare Thee To A Pressure Wave? Visualisation, Analogy, Insight and Communication in Physics*. PhD thesis, Department of Psychology, University of Bath, May 2006

54. Nardi, B.A. (1993). *A Small Matter of Programming*. The MIT Press.

55. Naur, P. (1985). Intuition in Software Development. *TAPSOFT*, 2, 60–79.

56. Naur, P. (1995). *Knowing and the Mystique of Logic and Rules*. Dordrecht: Kluwer Academic Publishers.

57. Paterson, D. (2004). Rhyme and Reason, T S Eliot Lecture 2004; published in abridged form in The Guardian Review, November 6, 2004, 34-5, online at http://www.poetrylibrary.org.uk/news/poetryscene/?id=20 (Accessed 11/5/2011)

58. Pope, N. and Beynon, M. (2010). Empirical Modelling as an unconventional approach to software development. In *Proc. SPLASH 2010 Workshop on Flexible Modeling Tools,* Reno/Tahoe Nevada, USA, October 2010

59. Richards, I.A. (1955). *Speculative Instruments*. London: Routledge and Kegan Paul.

60. Roe, C. & Beynon, M. (2007). Dependency by definition in Imagine-d Logo: applications and implications. In Ivan Kalaš (Ed.) *Proc. of the 11th European Logo Conference* 19-24 August 2007, Bratislava, Slovakia

61. Turing, A.M. (1950). Computing Machinery and Intelligence. *Mind*, 49, 433-460.

62. Turski, W.M. and Maibaum, T.S.E. (1987). *The Specification of Computer Programs*. Addison-Wesley.

63. Vendler, H. (1981). I.A. Richards at Harvard. *Boston Review*, originally published April 1981. Online at http://bostonreview.net/BR06.2/vendler.html (Accessed 11/5/2011)

64. Vincenti, W.C. (1993). *What Engineers Know and How They Know It: Analytical Studies from Aeronautical History*. Baltimore: The Johns Hopkins University Press.

65. Wartofsky, M.W. (1979). *Models: Representation and the Scientific Understanding*. Dordrecht: Kluwer Academic Publishers.

66. Weinberg, S. (2002). Is the Universe a Computer? *New York Review of Books*, Oct. 24, 2002 Online at http://www.nybooks.com/articles/archives/2002/oct/24/is-the-universe-a-computer/ (Accessed 11/5/2011)

67. Winograd, T. and Flores, F. (1986). *Understanding Computers and Cognition: A New Foundation for Design*. New York: Addison-Wesley.

68. Wolfram, S. (2002). *A New Kind of Science*. Wolfram Media.

69. Wolfram, S. (2010). *The Poetry of Function Naming*. Online at http://blog.stephenwolfram.com/2010/10/the-poetry-of-function-naming/ (Accessed 14/5/11)

70. The Empirical Modelling website. Online at http://www.dcs.warwick.ac.uk/modelling (Accessed 14/5/2011)

71. Wolfram S (2010) *The Poetry of Function Naming*. Online at http://blog.stephenwolfram.com/2010/10/the-poetry-of-function-naming/ (Accessed 14/5/11)

72. The Empirical Modelling website. Online at http://www.dcs.warwick.ac.uk/modelling (Accessed 14/5/2011)

73. The Empirical Modelling project archive. Online at http://empublic.dcs.warwick.ac.uk/projects (Accessed 27/5/2011)