Can Thomas Kuhn's Paradigms Help Us Understand Software Engineering?

Paul Wernick and Tracy Hall
Systems and Software Group
School of Computer Science, University of Hertfordshire
College Lane, Hatfield, Hertfordshire AL10 9AB, England
tel. ++1707 286323/284782; fax ++1707 284303
{p.d.wernick, t.hall}@herts.ac.uk

Abstract

Recent articles in EJIS have discussed whether or not Information Systems is a 'discipline'. In *The Structure of Scientific Revolutions,* Kuhn states that a scientific discipline can be identified by reference to its underlying belief system, the 'paradigm' or 'disciplinary matrix', to which all workers in that field must commit. An important element of Kuhn's model is the notion of 'scientific communities'.

We consider here the belief system underlying Software Engineering. We examine the extent to which a belief system analogous to the disciplinary matrix of a Kuhnian science can be identified in Software Engineering.

Our preliminary fieldwork has comprised an examination of books used by Software Engineering students and practitioners and in-depth interviews with a number of practitioners. The results of this study suggest that the current status of the theory of Software Engineering parallels Kuhn's 'pre-paradigm' stage of scientific development. At this early stage, theorists and practitioners are divided into schools. These schools are based on differences in the beliefs and models forming their disciplinary matrices.

We conclude that the application by analogy of Kuhn's view of scientific activity to Software Engineering is justifiable. Our findings can assist both Software Engineering theorists and practitioners in improving the understanding of how and why software development projects succeed or fail. Our findings also provide a framework within which to place the beliefs, models and values which underlie Software Engineering. Such a framework can contribute to the discussion as to whether the software development-related aspects of Information Systems can be considered to be a discipline, and if so how that discipline is structured.

Background and Rationale

Software Engineering (SE) has not yet fulfilled the expectations raised by its proponents both in the public community and amongst many of its practitioners. Neither has a consensus been achieved across all software practitioners behind a single mechanism or set of mechanisms for the development of software. There is currently little sign of such a consensus emerging. The rate of introduction throughout the history of software development of new mechanisms and techniques reflecting differing viewpoints of how to develop software, or of how to represent the real world in suitable model form, suggests a future of continuing division rather than convergence in SE practice.

We suggest that the underlying reasons for the differences in approach to Software Engineering, and the real-world benefits which might be obtained by changing from one development mechanism to another, are poorly understood. This diversity may be seen by practitioners and/or their managers as being valuable in that choices can be made between many different methods and tools. However a lack of understanding of the reasons for and impact of these differences weakens the effectiveness of research into Software Engineering, since work continues on divergent paths with a corresponding loss of synergy. This lack of understanding may be due in part to a failure to investigate, understand and document the nature and effect of the belief system underlying current SE theory and practice. The arguments made and conclusions drawn here are directed at Software Engineering. Our definition of SE relates to the development of all software-based systems in the real-world. We assume software-based Information Systems to be a sub-set of all software systems, even though Information Systems may require additional considerations. The development of software-based Information Systems requires a rigorous approach analogous to the development of other software systems. Consequently the work presented is relevant to the development of software-based Information Systems.

It is reasonable to assume that the use of any predefined mechanism within a process has an impact on the result of that process. In the Software Engineering process many mechanisms are used including: notations, computational or process models, software tools, or complete methods. Some researchers claim that these mechanisms induce a mindset in their users which affects the results of SE practice (Episkopou and Wood-Harper, 1986; Hirschheim and Klein, 1989). However, the identification in detail of these influences has not been explored previously. Kuhn's (1970) concept of the Disciplinary Matrix (DM) is employed here as the basis for an analysis of the belief system controlling this mindset. The DM is described in detail below.

The application of the analogy described in this paper and the analysis of the results obtained is capable of changing the nature of the discourse within SE, in particular that relating to the relative merits of methods and tools. The discourse may be moved to a higher level of abstraction than it is at present, by a consideration of the underlying, possibly implicit, belief system which is inevitably taken on board with the adoption of a particular method or tool. Analysis from the analogy may provide the basis for improved SE education, by enabling the teaching of a deeper understanding of current and past fundamental underlying principles, and how these are operationalised in specific tools. When examining the practice of SE, the mindset of a developer when making a decision can be considered in the context of the influences arising from the tools and techniques in use at that time. This understanding should provide a better-supported approach to comparing and selecting methods and tools. The analogy may also advance the understanding of SE in the context of other design disciplines, by allowing the drawing of more meaningful parallels between the DMs of SE and those disciplines.

Our work may also contribute to the recent discussion as to whether or not Information Systems (IS) is a 'discipline' in response to Paul (2002). We believe that the work presented here may provide a framework for the development of a "body of knowledge" for that part of IS which deals with the development and evolution of computer-based systems, and an understanding to form the basis of "less dubious research methods" (*op cit*, p.176) in this area. We emphatically agree with Fitzgerald (2003, p.226) that "it is clear that [IS] is not a science", and that its body of knowledge will contain much 'softer' rules than the DM of a science. Nevertheless we hope that our work will demonstrate over time that computer-based IS, with its fundamental beliefs operationalised in its tools techniques, methods and models is more than merely a "perspective" (*op cit*, p.227).

Why Kuhn?

The work reported here is based on an analogy between Kuhn's account of scientific disciplines, and the current state of SE. This immediately raises two questions: why use the philosophy of science as a starting point, and why use Kuhn's version of it? These questions are addressed in this section.

Why Use the Philosophy of Science?

The philosophy of science is a discipline which looks at another discipline's practices to understand and improve the latter's theory and practice. Since we intend to do the same for SE, we have taken the philosophy of science as a starting point. As we show later, a model drawn by analogy from the philosophy of science works well both in describing the current state of SE and in providing new ways of approaching its perceived problems.

We are not the first to consider SE from a philosophical standpoint using science as an analogy. Hirschheim and Klein's (1989) 'functionalist paradigm' underlies methodologies such as Structured Analysis and Information Engineering; in their view, many successful software system developments attempt "… to follow the scientific method." (*op cit*, p.1203). Wood-Harper and Fitzgerald (1982) refer to two different types of approach to software development, which are designated 'science' and 'systems' and used to classify a set of six lower-level approaches. The idea of SE as being a discipline which takes some aspects of its approach from the physical sciences is thus well-established.

Why use Kuhn?

The philosophy of science provides many different viewpoints as to how to identify a science and how to describe its practice. Some of these may be seen to be directly relevant to Software Engineering. For example, Popper's (1969) concept of falsification can be applied to the testing of software.

Deciding on which of the high-level theories of science to adopt for an examination of SE is less straightforward. We have employed the work of Kuhn (1970) for the following reasons:

- The similarity between the current status of SE and Kuhn's 'pre-paradigm' stage of scientific development. When theorists and practitioners are divided into schools based on differences in the beliefs and models forming their disciplinary matrices they argue over these fundamental beliefs from mutually incomprehensible starting points. Such a situation is observable in SE.
- A parallel between the application of SE tools and techniques to develop a software product and Kuhnian 'normal science'. We will explain below that in 'normal science' one disciplinary matrix is applied to a specific problem without being questioned.

- Curiosity as to whether the frequent claims of ‘paradigm shifts’ in the practice of software development, most frequently in sales literature but also in the literature (see Wernick, 1996, p.195 *et seq.*), can be justified by reference to Kuhn’s original formulation of the term.

A benefit of examining the current state of SE from this perspective is that it encourages us to consider the underlying beliefs, models and values of SE theory and practice. This research provides a firmer basis from which to consider how different SE tools and techniques, operationalising differing sets of beliefs, may produce different results when applied in practice

Kuhn’s Work Described

The following description of Kuhn’s model of science, based on Kuhn (1970, 1977) and Chalmers (1992), is intended to serve as an introduction to Kuhn’s work to support its subsequent application to SE.

In this section we describe two of the fundamental concepts in the Kuhnian view of a science: the *scientific community* and the *disciplinary matrix*. We then consider the life cycle of a scientific discipline, and how and when disciplinary matrices can and must be compared with each other.

The Scientific Community and the Disciplinary Matrix

A *scientific community* is a separately recognisable group of people, comprising the practitioners in a scientific speciality (Kuhn, 1970, p.177). Each of these communities can be seen as one of the leaf nodes of a tree of disciplines and sub-disciplines, from the most general discipline of ‘natural science’ to the individual scientific disciplines.

A scientific community is characterised by the unified thinking of its members. “To an extent unparalleled in most other fields, they have undergone similar educations and professional initiations; in the process they have absorbed the same technical literature and drawn many of the same lessons from it.” (*ibid*, p.177). Any disagreements which may divide the community are more quickly resolved than in other types of discipline, due to the open communication among the community members allowed by their shared assumptions, beliefs, models and approaches (*ibid*, p.177). Kuhn calls this shared belief system a *disciplinary matrix* (DM) or ‘paradigm’. The effect of the DM on a scientific discipline is to define what can and cannot be done within that discipline. The DM sets the problems which can be solved, identifies which are the most urgent, and indicates what constitutes a valid solution to a problem; for example, a specific underlying model of light as a phenomenon will guide scientists in their design of experiments in optics and their analysis of the results which they obtain.

Kuhn’s Characterisation of the Life Cycle of a Scientific Discipline

According to Kuhn, the life cycle of a scientific discipline starts with a *pre-paradigm* stage. A pre-paradigm discipline is recognisable by the existence of many *schools* (Kuhn, 1970, p.16) within the discipline. Kuhn states that “Whatever paradigms may be, they are possessed by any scientific community, including the schools of the so-called pre-paradigm period” (1977, p.295, footnote). For the work described in this paper, we have taken the view that Kuhn refers here to the competing schools each being identified with its own DM. As with a Kuhnian scientific discipline, the DM of each school is shared by its members and defines the (different) metaphysical bases and differing exemplars which their theories explain (Kuhn, 1970, pp.2–13). However, on the basis of their different DMs, the schools disagree over the nature of the phenomena with which they are dealing, how to interpret their observations (Kuhn, 1970, p.17) and definitions of key concepts. They therefore talk at cross purposes (*ibid*, p.198 *et seq.*). The parallel between this state and the current state of SE will be discussed later in this paper.

At some point in the life of a pre-paradigm discipline, a single DM may emerge, superseding its competitors when all the discipline’s current practitioners acknowledge the superiority of the winning DM over all other candidates. At this stage, to Kuhn the discipline has become a *science*. The nature and effect of the DM now changes from those of pre-paradigm disciplines. The most common activity in a science consists of research directed at solving problems within unchallenged limits set by the science’s current DM. The DM sets the problems of the discipline, the terms in which these may be approached to give a valid solution and the means of identifying what constitutes a valid solution. It presents challenging puzzles, supplies clues to solutions and guarantees the competent practitioner success, which those of the pre-science schools did not (Kuhn, 1970, p.179). The DM gives a firm metaphysical basis for work, and a common language. No effort need be wasted in reinventing these principles or arguing over them. This activity of puzzle-solving within the (generally unquestioned) constraints of the DM is referred to by Kuhn as *normal science.*

Not all scientific work will succeed. During normal scientific work, most failures of predictions, theories or experiments will be regarded as failures on the part of the scientist rather than the DM. However, some problems may remain unsolved, and some of these may call all or part of the DM into question. These problems may relate, say, to theoretical problems which the DM cannot explain, or to observations which the DM has predicted incorrectly. Sometimes these problems become so obvious that they lead to a sense of scandal and a feeling in the community of scientists that 'something must be done', for example the failings of the phlogiston theory of combustion in the face of experiments which led to the discovery of oxygen. The workers in the discipline enter a period of "...profound professional uncertainty..." (Kuhn, 1970, pp.67–68). The crisis deepens when workers begin to lose faith in the current paradigm, and when a rival paradigm emerges the stage is set for a change in fundamental beliefs (Kuhn, 1970, p.91). On the basis of better predictive or explanatory power, and/or for a number of other reasons (Kuhn, 1970, p.155 *et seq*.), one of these – or the original DM in a revised form – emerges as the new, generally accepted DM. The final acceptance of this new, changed or restated DM is a *scientific revolution*, or *paradigm shift*.

After the revolution, the new DM becomes the basis for another period of normal science. From the changed perspective provided by the new DM, textbooks are rewritten reflecting a view of the history of the science as a path of smooth progress culminating in the current DM, and the revolution becomes invisible. Normal science can now continue until the next crisis emerges and another revolutionary period begins.

The Incommensurability of Paradigms

More than one DM may exist at one time, either in the pre-paradigm stage or in times of revolution, Kuhn's theory of scientific change requires the comparison of competing DMs. However, he states that it is impossible for two DMs, each taken as a whole, to be measured directly against one another.

When it is necessary to select between DMs, it is therefore not merely a question of comparing the two; Kuhn states that this simply *cannot be done*. The DMs embody different sets of assumptions, and use different terms, or different definitions of the same terms, to describe the world. Judged by its own values, one DM may be superior to another. However, the same may also be true in reverse, as the application of a different set of values may result in a different conclusion. As a result, Kuhn believes that adherents to different paradigms find it so difficult to communicate effectively that a complete understanding between them of the differences between their paradigms is not possible; they are *incommensurable*. As a result, the process of comparing one DM with another requires the *translation* of terms from one DM to the other (*ibid*, p.202).

Applying the Analogy

The Analogy Described

Theory of SE

The commonality of thought across a discipline which Kuhn describes for a science is conspicuous by its absence between the communities developing SE methods, tools and techniques. How might Kuhn's ideas therefore be applied by analogy to these workers? One answer to this question is to suggest that the theory-building aspect of SE is in a state analogous to that of a Kuhnian pre-paradigm discipline. In support of this contention, it is possible to identify divergent beliefs which split SE into schools on the basis of each school having its own:

- *Way of seeing the world*, embodied in the notations and modelling techniques which it promotes. Consider, for example, different computational models which can be used in analysis and system design, including *object oriented*, *data flow* and *process-based* models. Each of these can be used as the basis of different notations used to describe problems or solutions. Different computational models underlying the notations employed may cause an analyst or designer to see the world in different ways; for example, the data flow modeller will see flows of (passive) data to and from stores and being modified in processes; the object-oriented modeller will see numbers of objects, each with its own set of allowable actions, floating in the system;
- *Set of beliefs and assumptions* concerning such fundamentals of SE as the process models to be employed (waterfall, incremental, prototyping, …), and the importance attached to involving end-users and other stakeholders in software development (Hirschheim and Klein, 1989).
- *Definitions of key terms* such as 'object-oriented', on which members of differing schools of 'object-orientedness' are unable to agree, or on what constitutes 'quality' in a SE process or product.

The reasons for these divisions may be related to the complexities and constraints of real-world software development. Non-system-related pressures, such as the need for product differentiation in the commercial world and academic pressures for publication, also encourage further differentiation. As Fitzgerald (2003, p.226) states, "there [will] always be dissenters" to many of the beliefs in the DM.

The structure imposed on SE theory by the underlying belief system of each SE model, tool or method will inevitably be complex. This complexity stems from the depth of the DMs required for a detailed articulation of a Kuhn-based model of methods. Previous work in this area has typically sought to identify a small number of different approaches to software development. Hirschheim and Klein (1989) have identified four high-level approaches, Iivari *et al.* (1999) five, and Wood-Harper and Fitzgerald (1982) six. The criteria for previous subdivisions have also been less rich than would be the case for a fully populated DM. For example, Iivari *et al.* set out between two and seven underlying principles to categorise each of their approaches (*ibid.*, Table 1). More have already been identified in initial studies conducted into the DM of SE (Wernick, 1996, Appendix 2).

In addition, a Kuhn-based model is simpler to apply than other approaches. It does not require that the analysis of methods and tools be highly formalised and the structure of the analysis results rigidly defined, as is implied by more formalised approaches such as that of Song and Osterweil (1992). Indeed, an analogy with Kuhn's view of science may suggest that SE method and tool comparison and selection cannot be performed with total objectivity, and, further, the incommensurability of DMs suggests that such a programme might be fraught with difficulty.

Practice of SE

Software developers applying an SE tool or method are influenced by its underlying DM. This happens when developers use the tool or method as part of their everyday activities, typically without questioning the theory underlying it or considering alternatives. A development team working on a project is (or should be) united in a common viewpoint, and should work within the bounds, determined *inter alia* by the DM of the toolset adopted. UML users will be expected to design object-oriented systems rather than discussing how to define the term 'object'. This unified approach contrasts with the state of the theory of SE outlined above. If a Kuhn-based analogy is to be applied to the practice of SE, i.e. the development of software-based systems, a different parallel from that of a Kuhnian pre-paradigm discipline is therefore needed.

Kuhn views the mechanism of normal science as being the development and exercising of theories relying on an unchallenged metaphysical base in which confidence is held. For Kuhn, it is necessary that the metaphysical base be *assumed* to be solid in order to perform the detailed work needed to explore its possibilities. Kuhn's justification for this approach in science is that it allows rapid development of the practice of a discipline without the need constantly to re-examine the philosophical basis of the discipline (Kuhn, 1970, p.42).

This is a reasonable parallel to what is required of the software developer who uses a predefined method or tool. He or she should concentrate on using that mechanism to develop his or her product, rather than seeking to explore the weaknesses of the mechanism. Therefore any instance of SE practice based on predefined mechanisms can reasonably be seen as being similar to Kuhnian 'normal science'. Software developers who adopt a particular method or tool, i.e. who believe in it as a valid, useful method for developing systems, can be said to have joined a community associated with this method, tool, etc. They act in ways constrained by that toolset, and may over time cease to question the associated paradigmatic beliefs enunciated by those who defined the toolset. The DM elements explicit and implicit in the tools used by a software developer influence how he or she interprets the circumstances of a situation, sees the problem in that situation, and defines a valid solution to the problem. The DM of a SE method or tool can thus be viewed as a cognitive filter, modifying a software developer's view of the world (cf. Episkopou and Wood-Harper, 1986; Petre, 1989).

In considering the detailed parallels between the activities of SE and Kuhnian normal science, the following points emerge:

- ❖ The filtering effect of the underlying belief system on the framing of problems and the definition of acceptable solutions. The DM underlying the toolset employed acts as a filter over the developer's perceptions. This influences the view a developer has of the world and the system, and therefore colours his or her development decisions.
- ❖ The parallel between SE within a method- or tool-based framework which constrains and controls it, and normal science. Both operate within a paradigm which constrains visualisations of problems and definitions of acceptable solutions.

- The acceptance of a change in the set of assumptions behind a newly-adopted tool by an individual software developer; and the change in view of the subject of study of a scientist undergoing the ‘religious conversion’ of a change in paradigm (Kuhn, 1970, pp.150–151).
- The need for translation between different scientific communities because of the different underlying DMs being used (Kuhn, 1970, p.202). Related to this are the difficulties in translating the ideas of one team of software developers employing a toolset based on one DM into a set of ideas understandable and usable by a team using tools developed and employed under the influence of a different DM.

Will SE Ever be Unified?

Our work is not intended to form the basis of a DM for SE. Whether or not it will reach this state of unanimity is not relevant to this work.

The Kuhnian analogy is intended to be descriptive, not normative. It is intended not as a recipe to be followed to make SE ‘better’ by the universal adoption of one set of beliefs, but as the basis for improved understanding of the state of the discipline. A diversity of DM content in SE may in fact be desirable to support the diversity of SE practice. However, an understanding of the common core of belief underlying SE, and the rationale for and effects of the diversity which is found, are nevertheless essential for progress in SE theory and practice to be well-founded.

Fieldwork and Results

We have performed fieldwork which aimed to explore the Kuhnian analogy in SE and to articulate the DM in SE. We used a series of investigations to identify individual elements which might be included in the DM and to examine the structure of the DM. For details of these investigations and their results, see (Wernick 1996; Winder and Wernick, 1993, 1994, 1996)

The fieldwork comprised an analysis of the theory and practice of SE. Each of these investigations focused on definitions and discussions of ‘quality’ in SE processes and products. We examined the term ‘quality’ as it is an aspect of SE which the theory and practice must address in describing or performing SE. We hoped it would prove a rich source of beliefs for this study.

1. The theory of SE.

Kuhn regards textbooks as a source of existing beliefs for those joining a discipline, and as a strong influence on the behaviour and actions of these new joiners.

We therefore analysed eight SE textbooks to identify those elements of a DM which are embodied in texts used to teach SE methods and tools. Most of the sources selected were intended by their authors for general undergraduate SE education, the others being intended for practitioners. The texts used were (Birrell & Ould 1985, Sommerville 1992, Downs et al 1992, Licker 1987)

We first analysed each text book to identify beliefs and models defined by the author. Having done this for each text we established an apparent structure of beliefs. We then tested the validity of our findings by analysing a second set of text books to establish the existence of the belief structures suggested in the first set of texts. The second set of texts were (van Vliet 1993, Jackson 1983, Schach 1993, Carmichael 1994).

Our analysis of the theory of SE as represented in SE texts uncovered 31 DM elements. These elements included, for example:

- Controlling the software process produces a better product.
- Systems should be designed to be portable
- Abstraction in design is a good thing.

2. The practice of SE.

To investigate the practice of SE we used a series of in-depth interviews with eight software developers. Six of these practitioners developed software in large commercial organisations, one in a small commercial organisation and one participant developed software in a university. Participants developed a range of software from embedded software to user computing. Most participants contributed to the whole software development lifecycle, though one focused only on analysis and design activities. The interviewees were selected to reflect the use of a variety of SE mechanisms in different system and organisational contexts and varying degrees of use of formalised SE methods, tools and techniques. The interviewees were selected via personal contacts to make the process and atmosphere less formal and encourage the interviewees to speak freely.

Semi-structured questionnaires were used during the interviews to ensure focus but also allow flexibility in the data collected. Transcriptions of interviews were then analysed using a content analysis approach. Transcripts were analysed for both explicit beliefs and models, and for any implicit aspects which their responses revealed.

Interviewees were explicitly asked what comprises ‘quality’ in a software product and in a software development process. Interviewing practising software developers allowed DM elements identified in the textbook analysis to be checked. The interviews also allowed further elements, found in the practice of SE but not well documented in the textbooks, to be identified.

Of the 31 DM elements identified in the text books, 20 of these were also found in practice. Overall the structure of beliefs and models identified in practice was similar to that identified in theory.

Our investigations generated evidence to support the following conclusions:

1. Evidence for the Kuhn-based Model in General

By identifying specific beliefs which underlie the tools used in SE and how they are applied we found evidence for the Kuhn-based model in the theory and practice of SE. In our fieldwork we have found evidence of elements of a generally-accepted set of beliefs which is expressed in the SE methods and tools.

Aspects of an unquestioned theoretical base, which thus become implicit beliefs, include the implications of the theories of computability and complexity for producing practically-usable software designs (Wernick 1996, p.241). An example of a generally-held belief, identified during the fieldwork, which has been specifically developed within SE is the use of step-wise refinement in software design. We found agreement that the reductionist process of breaking a problem down into a set of sub-problems, solving these sub-problems and recombining the results into a single ‘solution’ produces a solution to the original problem (*cf.* Wernick, 1996, p.242). This belief is a good candidate for an element of that part of a DM which unites all software engineers.

2. Evidence for SE Method Development being Analogous to a Pre-paradigmatic discipline

To support the idea of SE method development being analogous to a Kuhnian pre-paradigmatic discipline, it is necessary to identify elements of that DM over which there are fundamental differences between groups of method and tool developers. These groups are analogous to the schools of a Kuhnian pre-paradigmatic discipline.

Our fieldwork revealed a number of such beliefs, such as differences over the degree to which the users of a system need to support the process of developing the system (Wernick 1996, p.258). Another example of this is the extent to which users are treated as people or as mechanistic ‘parts of the system’; *cf.* (Isomäki, 2002). Similarly we found disagreement in the definition of the key term in assessing software quality of ‘cohesion’ amongst the text books. We did not find consensus on the relative importance of elements of the development process; for example there was disagreement about whether testing was more important than analysis.

3. Evidence for SE Method Use being Analogous to Normal Science

The uncritical application by practising software developers of the beliefs explicit and implicit in the toolsets they use has been identified in our fieldwork by considering the beliefs identified in practitioner interviews on how to achieve ‘quality’ in the results of the development process. We refer specifically to beliefs which are controversial between schools of SE theorists but not amongst the practitioners who apply them.

For example, most interviewees agreed that control over the SE process produces a higher software product quality (with none dissenting), a view over which the evidence of the book trawl suggested that theorists disagree (Wernick, 1996, p.261). Similarly, although design abstraction was thought to be good by theorists and practitioners only practitioners believed that there was an optimum level of abstraction in a design, while the texts did not mention this notion. Furthermore the practitioners were clear that the scale and complexity of the problem influences the best way of addressing or solving that problem. Again there was no mention of this idea in the theoretical texts.

In addition to drawing the above conclusions, a dichotomy was identified in the fieldwork between the belief system underlying SE ‘theory’ as stated in the textbooks and that supporting ‘practice’ as described by the practitioners. This implies that some fundamental aspects of SE are the subject of controversy between theorists and practitioners (Winder and Wernick, 1996). The controversy may be due to the need to trade off between different contradictory or divergent beliefs operationalised in sets of SE tools in order to achieve high-quality results in real-world software development projects. Differences of opinion as to the weightings to be assigned to each belief during this trading off also emerged during the interviews, which further divide SE.

The results of the fieldwork support the existence of an underlying belief system for SE, divided into those aspects adopted throughout the SE community and those over which there is disagreement. This division supports the contention that an analogy between SE theory and a Kuhnian pre-paradigm discipline can be supported. On the basis of the investigative work, it is now possible to expand the Kuhn-based model of SE into more detail, especially with regard to the structure and content of the DM of SE. An examination of elements in the DM of SE identified in the investigations (Wernick, 1996) shows that SE theorists and practitioners can be divided into schools according to their differing beliefs, and thus provides a rationale for this division of the discipline. The individual schools, the DM content for each school, and the implications for theory and practice of these differences, remain to be identified.

The beliefs identified as being common to all software developers may be considered to form a definition of who is a 'proper' software-based systems developer, as against a 'hacker', who will work under a different belief system producing different sorts of results. This parallels Kuhn's definition of who is a 'scientist', based on a cross-disciplinary set of beliefs, values and so on, common to and shared between all scientists (Kuhn, 1970, p.177). A definition of who is a proper software engineer could thus be based on the acceptance and operationalisation in his or her work of a particular DM.

Conclusions

In this paper we have investigated the current state of Software Engineering in relation to Kuhn's model of a scientific discipline. Our aim was to provide support for the existence of an underlying belief system for Software Engineering, some at least of whose content can be identified. In pursuit of this we examined SE textbooks as the repository of SE theory, and interviewed practitioners to investigate Software Engineering practice.

Our findings suggest that SE theory, viz. the development of methods and tools to support the development of software, is currently in a state analogous to a Kuhnian pre-paradigm discipline. A core disciplinary matrix does exist, comprising a set of beliefs to which all software engineers subscribe, but other beliefs divide this community into the competing schools which would be expected in such a situation. In contrast, SE practice takes on board one set of beliefs, derived from the set of tools adopted for that project, which support its work developing a software system. Despite our adopting an analogy with Kuhn's work, our results must not be taken to mean that we expect or support the unification of all of SE in a single DM, as is implicit in the claims by tool or method proponents of a 'paradigm shift' in SE.

The work we present can support progress in both the theory and practice of SE. We have shown that Kuhn's work on scientific disciplines can be applied successfully to an analysis of SE. Such an analysis, progressed to a greater extent than we have so far undertaken to identify the beliefs which make up the DM of SE, will contribute to an increased understanding of the underlying theoretical structure of SE. This structure will enable researchers to position their work within a coherent framework of understanding, and allow their research results to contribute to an incremental development of SE. A Kuhn-based approach may provide a mechanism for interpreting and integrating research carried out to identify new approaches to software and its development, such as those arising from the Feyerabend Project (Gabriel, 2002), which is dedicated to questioning the current rules and assumptions under which software and its development are considered, as well as reconsidering repositories of SE knowledge such as the Software Engineering Body of Knowledge (SWEBOK, 2003).

The long-term vision of a discipline informed by extending the work described here could be called 'software-based systems *engineering*'. This would be based on mechanisms whose effects could be predicted with confidence. A well-designed series of investigations should illuminate many of the underlying influences on SE from the DM content explicit and implicit in its tools and situations, and the effects on practice of each of these influences. The results of these investigations could be used to design methods and tools with some ability to predict the effects of each method or tool on the way in which software analysts and designers would perform their work. It would also be possible to teach students the fundamentals of SE in terms of its underlying belief system, and the effects of its DM on its practice. Such an education would afford a firmer basis for the updating which practitioners will need throughout their careers as software-based systems engineers.

References

AVISON DE and WOOD-HARPER AT (1993) *Multiview: An Exploration in Information Systems Development*; Alfred Waller; Henley-on-Thames (1990; 1993 reprint).

BIRRELL ND and OULD MA (1985) *A Practical Handbook for Software Development*; Cambridge University Press; Cambridge

CHALMERS AF (1992) *What is This Thing Called Science?*; Second Edition; Open University Press; Buckingham (1982; 1992 reprint).

CARMICHAEL A (Ed.) (1994) *Object Development Methods*; SIGS Books; New York

DOWNS E, CLARE P AND COE I (1992) *Structured Systems Analysis and Design Method*; Second Edition; Prentice Hall; New York.

EPISKOPOU DM and WOOD-HARPER AT (1986) Towards a framework to choose appropriate IS approaches. *Computer Journal* **29** (3), 222–228.

FITZGERALD G (2003) Information systems: a subject with a particular perspective, no more, no less; *European Journal of Information Systems* **12**; 225–228.

GABRIEL RP (2002) The Feyerabend Project; http://www.dreamsongs.com/Feyerabend/Feyerabend.html; accessed 14 June 2004.

HIRSCHHEIM R and KLEIN HK (1989) Four paradigms of information systems Development. *Communications of the ACM* **32** (10), 1199–1216.

ISOMÄKI H (2002) The Prevailing Conceptions of the Human Being in Information Systems Development : Systems Designers' Reflections. PhD Thesis, Department of Computer and Information Sciences, University of Tampere, Finland.

IIVARI J, HIRSCHHEIM R and KLEIN HK (1999) Beyond Methodologies: Keeping up with Information Systems Development Approaches Through Dynamic Classification. In *Proceedings of the 32nd Hawaii International Conference on System Sciences*.

JACKSON MA(1983) *System Development*; Prentice Hall; Englewood Cliffs, NJ.

KUHN TS (1970) *The Structure of Scientific Revolutions*; Second Edition; University of Chicago Press, Chicago.

KUHN TS (1977) Second thoughts on paradigms. In TS Kuhn, *The Essential Tension*; University of Chicago Press, Chicago, 293–319.

LICKER PS (1987) *Fundamentals of Systems Analysis with Application Design*; Boyd & Fraser; Boston.

PAUL R (2002) Is Information Systems an intellectual subject?; *European Journal of Information Systems* **11**, 174–177.

PETRE M (1989) Finding a Basis for Matching Programming Languages to Programming Tasks. PhD Thesis, University College London, London.

POPPER K (1969) *Conjectures and Refutations*; Routledge and Kegan Paul; London.

SCHACH SR (1993) *Software Engineering*; Richard D. Irwin, Inc., and Aksen Associates, Inc.; Homewood, Ill.

SOMMERVILLE I (1992) *Software Engineering*; Fourth Edition; Addison-Wesley; Wokingham.

SONG X and OSTERWEIL LJ (1992) Towards objective, systematic, design-method comparisons. *IEEE Software* **9** (3), 43–53.

SWEBOK (2003) *Guide to the Software Engineering Body of Knowledge*, http://www.swebok.org/home.html, accessed 27 January 2003.

VAN VLIET H (1993) *Software Engineering: Principles and Practice*; Wiley; Chichester.

WERNICK P (1996) A Belief System Model for Software Development: A framework by analogy. PhD Thesis, University College London, London.

WINDER R and WERNICK P (1996) Building a model of computer-based systems development: testing elements of the disciplinary matrix. In *Information Systems Methodologies* 1996 – *Proceedings of the 4th Annual Conference on Information Systems Methodologies* (JAYARATNA N and FITZGERALD B, eds.), British Computer Society Information Systems Methodologies Specialist Group, Cork, Ireland, 12–14 September 1996, 129–141.

WINDER R and WERNICK P (1994) Refining a philosophical model of software development: tracing elements of the disciplinary matrix. In *Information Systems Methodologies 1994: Second Conference on Information Systems Methodologies* (JAYARATNA N *et al*, eds.)*;* British Computer Society Information Systems Methodologies Specialist Group, 31 August–2 September 1994, 203–216.

WINDER R and WERNICK P (1993) The inductive nature of software engineering and its consequences. In *Proceedings of the Conference on the Theory Use and Integrative Aspects of IS Methodologies* (JAYARATNA N *et al*., eds.), British Computer Society Information Systems Methodologies Specialist Group, 1–3 September 1993, 431–443.

WOOD-HARPER AT and FITZGERALD G (1982) A taxonomy of approaches to systems analysis. *Computer Journal* **25** (1); 12–16.