Against method-*ism*

Exploring the limits of method

Lucas D. Introna and Edgar A. Whitley
London School of Economics and Political Science, London, UK

Received August 1996
Revised January 1997

I. Introduction

This paper seeks to address an interesting paradox in current research on information systems development. On the one hand, various authors are undertaking valuable work in what they call "method engineering" – attempting to create and develop approaches that can be used to develop information systems (Brinkkemper *et al.*, 1996) – while, on the other hand, empirical research shows that methodology use in practice is rather limited (Hidding, 1996). Those systems developers who do use methodologies report that they tend to use parts of methodologies (often, parts taken from a variety of different methodologies) rather than following all the steps required by a particular methodology (Fitzgerald, 1996).

Why is it that the products of quality work, by very capable researchers, are often so underused? Is there any point to developing complete methodologies, if people tend to pick and choose parts of methodologies and discard others? Is the drive to develop methodologies itself based on unreliable assumptions? Is it possible to develop *the* methodology, or would it not be more effective to just develop particular techniques and tools which could then be crafted together, as and when required?

The answer to these questions, we believe, lies in the assumptions made about the role of method in developing information systems. We will argue that the dominant mindset in method engineering, which we will label "method-ism", believes that methodology is a necessary and sufficient requirement for successful information systems development. Hence, it is believed that it is possible to incorporate *all* necessary knowledge into a particular methodology to ensure its effective use in a variety of development situations. In contrast, we suggest that the successful use of methodology itself depends on a broader, already present, tacit understanding of the world, an understanding not to be found in any one particular methodology. It is this background knowledge

An earlier version of this paper was submitted to the joint IFIP WG 8.1/8.2 conference on method engineering, held in Atlanta, Georgia, in 1996. The paper was presented by a panel at the conference. We are grateful to the reviewers of the paper for their insightful comments. Frank Land also read a previous version of this paper and, as always, made many positive contributions and suggested some important new ideas for the paper. Finally the paper has also benefited greatly from the feedback and suggestions from presentations to Yair Wand and his colleagues at the University of British Columbia and Rudy Hirschheim and colleagues at the University of Houston, Texas.

Information Technology & People,
Vol. 10 No. 1, 1997, pp. 31-45.

against which methodology use becomes sensible. When this background understanding is not present, methodology use fails.

I.1. Methods, approaches and methodologies

At this point it is important to clarify our use of various terms relevant to our discussion. These terms have been given various interpretations in the literature and we will be using them in accordance with the emerging conventions (Avgerou and Cornford, 1993; 1995). For clarity, we define a *methodology* as a structured set of *techniques* and *tools* that are used to tackle a particular problem, in this case, developing an information system.

The term *method* has been used in two distinct ways. The first sees a method as a component of a methodology, alongside techniques and tools. The second sees method as an all encompassing meta-term: this is illustrated by its use in the term "method engineering". We will use the second meaning of the term in this paper, normally in the context of *method*-ism. Method-ism can best be characterized by the following axioms:

- *Methodology is necessary and sufficient for information systems' development success.* This is the belief that everything that is required to analyse and specify a problem situation, and all that is required to design a solution, can be found *in* a suitably designed methodology.
- *If systems developers have a suitable methodology at their disposal they will use it*. This is the belief that systems developers understand the value of methodology and will prefer to work with it rather than without it.

An *approach* exists at a higher level than that of a particular methodology and describes the underlying philosophy which guides the shape of the methodology (the methodology may, in turn, shape the particular tools and techniques). For example, the Scandinavian approach to worker empowerment has led to the development of a number of different methodologies, such as the participative design methodologies. In a similar vein, the socio-technical approach influenced the ETHICS methodology of Mumford and Weir (1979).

I.2. The role of method

It is important to emphasize that this paper is not against methodology *per se*, nor is it against particular techniques or tools. Rather, the focus of its attention is on the mindset of method-ism, with its nomothetic belief in method. The obvious analogy is to scientism and its nomothetic belief in the scientific method as necessary and sufficient for discovering the truth (Feyerabend, 1993).

Each of the authors, both individually and together, have experience of a range of structured and soft methodologies which we find useful for particular circumstances and situations. On numerous occasions we have benefited from the insights and structure given by a particular methodology and its associated tools and techniques. The point of our argument, however, is that the use of

method emerges from our understanding of and involvement *in* the problem situation, not merely because it is required by the methodology.

In arguing against method-ism, we are arguing in favour of the appropriate skilled use of methodologies, tools and techniques, use which emerges from involvement in the world as part of "getting the job done". We believe it more realistic to abandon the idea of a rational and detached developer who objectively applies a methodology, in a way analogous to the computer applying a correct algorithm, for a more pragmatic image of an involved developer who will "hack" with whatever tools are available (ready-to-hand) to get the job done (Introna, 1997). The question should not be to get *the* methodology; it should be how to help the developers to improvise in the situation in order to get the job done (Ciborra, 1996).

I.3. Outline of the paper

The purpose of this paper is to provide a critique of method-ism. This will be done in three sections. First, we will discuss the relationship between method and understanding. We will argue that the assumption implicit in method-ism – that method can replace or induce understanding – is wrong. Method flows from understanding, and not the reverse. Second, we will attempt to explain the way humans interact with the world by means of ready-to-hand tools. This discussion will show that tools are used only if available (ready-to-hand) in the world of doing. If a methodology is not ready-to-hand it will break down and be ignored in the pragmatics of getting the job done. Third, we will present a number of reasons why methodologies, by design, will tend to break down (not be ready-to-hand) and hence be discarded. Finally we will conclude with some implications and ideas for future research.

II. Method and understanding

The optimism that drives method-ism believes that by using method (in the broad sense of the word) it is possible, in principle and practice, to understand and solve the typical system development problem. The implicit metaphor that provides the impetus for this optimism is the analogy of a computer applying an algorithm to solve a problem. The real expertise and understanding is located in the designer and built into the algorithm. Once designed, the computer must merely implement the algorithm step-by-step to obtain the solution. Thus, it is implied that it is possible to create methodologies (as meta-algorithms) that can be used in situations where the comprehensive understanding is built into the method by the designer. In this scenario the developers need to understand little or nothing about the problem situation, as the methodology will bring to light all the characteristics of the situation that need to be discovered, and it will do this in a way that requires solely an understanding of the methodology itself. We can consider this conceptually in the following generic diagram, where method (in the broad sense of the term) leads to understanding:

Method → Understanding

The method-ism mindset therefore leads to the development of methodologies that will be used to understand a particular situation.

Methodology X → Understanding of situation Y

The implications of this view can be seen by considering the following imaginary scenario. Imagine an information technology consulting company that hires a number of bright graduate students as consultants and gives them an intensive six-week training course in the use of their own particular methodology. Immediately on satisfactorily completing the training course, the new consultants are sent to a client site without supervision. Simply by utilizing the tools and techniques of the methodology they have been trained to use they are able to analyse, specify and design an effective information system for the client organization. Surely such a situation would never arise in practice, and yet method-ism, in its purest form, would suggest that this should be attainable and should be the goal of all methodology designers.

We will argue that the problem arises because of the mistaken relationship between method and understanding, possibly due to the objectivist algorithmic metaphor. We propose reversing the relationship.

Understanding → Method

That is, it is only through a proper understanding of the context and situation that we can learn successfully to use the tools and techniques provided by method. Tools have their sense in a context of use (Heidegger, 1927). In doing the work, the craftsman discovers which tools are required. Tools always assume an understanding of use, unless they are fully automated machines; if this is the case, then we are back to the mechanistic computer and algorithm notion.

Requirements for tools emerge as part of getting the job done; this is the reference that makes them significant as tools. A hammer means only something as a tool in-order-to drive in a nail. It is when wanting to drive in a nail that we realize (understand) that we need a hammer. The significance is in using the tool to do something, and is not itself in the tool. To summarize: it is our understanding of what is needed in-order-to develop a system that makes the tools and techniques emerge as significant (usable). The methodology in itself cannot create or specify this sense of reference or significance. Moreover, this process is in fact cyclical, in that once we have an understanding that allows us to use method we will understand also the limits of the methods, which will allow us to use the appropriate methods, which will improve our understanding of the methods, and so forth. It is a hermeneutic process (Introna, 1993).

Understanding → Method → Understanding → Method …

In the instance of analysing a particular organizational setting, it is only when we have the necessary background understanding or tacit knowledge (Polanyi, 1966) that we can use the methodology effectively for that situation. This

improves our understanding of the situation and the methodology and hence makes further improvements possible. Our understanding of the situation and the methodology, as we will see below, might require us at points to break with the use of the methodology rather than follow it uncritically. The knowledge of how to do this cannot come from within the methodology.

> Understanding → Methodology X → Understanding of situation Y → Improved use of Methodology X and others (including possibly no methodology)

The key characteristic of our restatement of the relationship between method and understanding, however, is that we believe that method *requires* a background understanding for it to be used effectively. Understanding in doing renders method significant. Only when method is significant as part of getting the job done will it be used. An engineered methodology, of the kind outlined in the hypothetical example above, *cannot* give us this background understanding.

III. Revealing the failure of method-ism

The failure of method-ism becomes apparent when users end up focusing more on the features of a methodology than on the task at hand. How many users of methodologies become bogged down with the intricate documentation and diagramming techniques required by the methodology rather than analysing the problem situation? In Heidegger's terminology, what is happening is that the methodology is becoming unready-to-hand or present-at-hand; it is no longer invisible to the task at hand (Heidegger, 1928; Winograd and Flores, 1986). Methodologies are tools used in-order-to develop information systems. Tools, if used in-order-to get the job done, become invisible to the task. To a person hammering in nails, the hammer is normally invisible to the task, and the person's attention is focused on the task of hammering.

In order to help understand this invisibility, we propose using a simple framework developed by Ihde (1990). Ihde discusses the various technologies we use and shows how they relate to individuals and the world. In order to do this, he uses the following framework:

> I – Technology – World

Effective technologies, he argues, become combined seamlessly so that two parts of the framework are merged. For example, if we wear spectacles, I and the technology are seamlessly merged so that we have a clearer way of looking at the world. In such a case, the spectacles are invisible to us. Similarly, when we use a mouse in a windowing-based computer environment, the mouse becomes merged with ourselves so that we are simply focusing on the world (in this case pointing on the computer screen):

> (I – Technology) – World

In other scenarios, the technology is merged with the world. A simple example here would be a car speedometer. In such a situation we are not directly

"measuring" the world; instead, the driver of the car is looking at the speedometer (technology) which is seamlessly merged with the world of the road and the car's movement on it. In many cases, in fact, there is not only one technology between the individual and world, but a whole stream of mediations and inscriptions (Latour and Woolgar, 1986) which help establish and maintain our relationship with the world (Hutchins, 1995). Again, this chain of inscriptions (from the wheel turning on the road to the digital display on the dashboard) is normally invisible to us. Similarly the individual connected to the Internet via a modem passes through a series of technologies to the world.

I – (Technology – World)

In both these cases, however, when the technology is used in-order-to see clearly or to check the speed of the car, it is invisible to us. Sometimes, however, the relationship between the individual, the technology and the world breaks down. In such situations the technology ceases to be invisible and becomes ready-to-hand. For example, if our mouse breaks down and the pointer moves up when the mouse moves down, we notice the features of the mouse. Similarly, when the mouse is not present and we try to use our windowing software, we notice features of the (missing) mouse.

Finally, when we try to use the mouse for certain tasks (for example, making precise movements), we again notice features of the mouse that were invisible when we were doing tasks that the mouse was designed to do well. In the same way, we become aware of features of our modem, telephone plugs and telephone lines only when things are not working rather than when they are working. There is also a mode of being where the technology is the focus of attention as a matter of choice, for example when studying how telecommunications networks work, but this is beyond the scope of this paper.

I – (Technology) – World

Consider then, the case of the information systems' development methodology that requires completing ten pages of documentation for every task. Such a methodology is likely to become the focus of attention, rather like the faulty mouse or the modem that will not work. It becomes a technology that will not be merged with either the individual or the world. As such it sticks out, it becomes a pain to use and, as a result, is unlikely to be used further. The methodology that was meant to be all-encompassing, the meta-algorithm, has suddenly become a collection of tools and techniques, some of which are used, some of which are not.

It seems from the section on methods and understanding that we are arguing for a relationship in the form of (I – Technology) – World. However, is it not also possible to create a seamless methodology in the form of I – (Technology – World)? Is this not what information engineering and CASE methodologies are about? In the next section we will present a number of arguments that suggest that this is not a feasible strategy.

IV. Why methodologies break down and need understanding

This section will outline a number of reasons why problems reveal themselves when methods are applied without understanding. Each will illustrate ways in which the technology (in this case the methodology) can become the focus of attention rather than being an invisible component in the task-at-hand of developing information systems.

IV.1. Methodologies and variety

Ashby's (1957) work in cybernetics resulted in his "law of requisite variety". This states that a system can control another system if and only if the system to be controlled has less variety than the controlling system. Or, as Beer (1985) states it: "only variety can absorb variety". For example, in the case of application software, the controlling algorithm must be able to cope with *all* the possible situations the system can be faced with if it is to be totally automatic. All too often, however, this requirement is not satisfied and the resulting systems either fail to be accepted at all or often need to pass control to human operators who have the necessary variety to cope. This has been particularly prominent in the development of knowledge-based systems where it is often impossible to devise rules that cover all the circumstances that the human expert has to deal with. The result has tended to be either a lowering of the problem to the level of the computer (Roszak, 1994) or preventing the system from becoming operational and denoting it as a "training system" instead.

A similar situation arises if we use variety-limited systems analysis and design methodologies to structure and drive the development process. A useful analogy can be made with a painter who has only a limited number of different coloured paints. In this case the paints (the features of the methodology) limit the range of paintings (systems) that can be developed. It could be argued that in such circumstances the artist is able to mix the limited number of colours available on the palette to create the infinite range of shades and colours found in the best works of art. These artists, however, are not just doing painting-by-numbers: they are not simply following a methodology to create a painting, but rather using the skills they have developed over time to do so. They have an existing understanding of colour, texture and light that helps them make best use of the features of the methodology.

We may try to solve this problem by increasing the variety in methodologies available to us, or the variety of methodologies, but this will merely lead to an infinite regress, as we will then need methodologies to manage our methodologies. For example, to choose the best methodology for a particular situation, the so-called contingency approach to method engineering (Rolland and Prakash, 1996, will lead to an infinite regress. If we reach the extreme of (re)generating a methodology anew every time we do a particular activity, is there any point in calling it a methodology?

IV.2. Methodologies and creativity

By definition, methodologies are designed to structure the design process – to turn the undefined, unpredictable and uncontrollable into the defined,

predictable and controllable. They are therefore, by necessity, rigid. This rigidity prevents the generation of new order: creativity is stifled (Jantsch, 1980; Prigogine and Stengers, 1985; Von Foerster, 1984).

It is always possible that sophisticated users of methodologies are aware of these limitations and therefore know when to "break" with the methodology. However, under method-ism, methodologies are designed mostly as an aid, not for these mature designers who have the necessary understanding, but precisely for those who depend heavily on them in the first place and who would not know when and how to break with them.

Knowing when to break the rules requires two things. The first is an awareness of the limitations of rules; the second, the background knowledge (form-of-life or understanding) against which this rule-breaking behaviour makes sense (Wittgenstein, 1956). These two concepts can be illustrated with a simple example (Collins, 1992). Suppose you are asked to "Continue the sequence '2, 4, 6, 8' in the same way". What should your response be? "10, 12, 14"; or "2, 4, 6, 8, 2, 4, 6, 8, 2, 4, 6, 8"; or "2, 4, 6, 8, 10, 2, 4, 6, 8, 10, 12, 2, 4, 6, 8, 10, 12, 14"; or "8, 6, 4, 2, 2, 4, 6, 8, 8, 6, 4, 2"? Any rule that is devised for continuing the sequence can be misapplied in this way because *rules do not contain rules* for their own application. By resorting to a shared form-of-life you could argue that "10, 12, 14" is the most "reasonable" response (unless you are on British soccer terraces when "Who do we appreciate?" is more appropriate).

IV.3. Methodology and meaning

Meaning, as has been argued by Gadamer, Heidegger, Wittgenstein and others (see e.g. Gadamer *et al.*, 1989), is always about context. We make sense of things against the context that they arise in. Sense in the world is not the result of the application of context-free elements; rather it presupposes a context of meaning. Academic papers do not form an element that makes up the world of academia; they presuppose the academic world and make sense *only* against it.

Meaning is that which emerges from the process of hermeneutic understanding (Introna, 1993). Meaning is always incomplete. There are always new possible contexts, new possible understandings, that can emerge. Meaning is historical. My understanding today is always mediated by my current and past experiences and interpretations.

A methodology, on the other hand, is normally acontextual and the more context-independent it is, the more useful it will be. Therefore, methodology designers endeavour to decontextualize and generalize their work as a matter of principle. There are some methodologies that are intentionally contextual; they are designed to be contingent on the particular circumstances of their application (Land *et al.*, 1995). Such methodologies, however, are developed after reflection on the issues presented here and can be seen as one response to the concerns raised.

Methodologies must be ahistorical. The same methodology, under similar circumstances, must be equally valid today as it was yesterday. If not, it would not qualify as a methodology. It then becomes very difficult to see how a context-

free methodology can be applied in situations where context and meaning are foremost. This can happen only with considerable recontextualization by the analyst applying it. This raises again questions about the validity of the methodology in the first place. Nonetheless information systems must be meaningful (become ready-to-hand) otherwise they will not form part of the meaningful action of the organization: they will be seen as dead and be discarded by their potential users.

Part of the cause of meaningless systems may be the use of top-down approaches. These require the methodology to create artificial environments to ensure success. Methodologies, therefore, are prepared to "ride roughshod over the variations of naturally-occurring conditions" (Tiles and Oberdiek, 1995, p. 111) in order that they may use standardized procedures and devices.

IV.4. Methodologies and tacit knowledge

Tacit knowledge arises from our involvement in the world (Introna, 1997); or, as Polanyi (1966; see also Polanyi and Green, 1969) states it: "we know more than we can tell". Methodologies typically aim to be complete, and yet a methodology has to be written down in some form. But there are things associated with methodologies which we know but cannot tell; the use of a methodology involves the application of tacit skills which cannot be told. A methodology must therefore be incomplete, and yet its very intention is to provide a complete description of all the steps required to solve a particular task.

In some cases this vagueness is acceptable, as much of the tacit knowledge is already known and assumed by the user of the methodology (again, pre-existing understanding). Thus, a recipe may list ingredients in terms of their exact weight in grammes, but may also tell the user of the recipe to mix the ingredients until they have a "fine, runny texture". The nature of a fine, runny texture cannot be specified, but most would-be cooks have the tacit skills necessary to recognize this. The same cannot be said always of system developers, particularly those with no experience of developing systems, as can be seen by the large number of methodologies which seem to be successful only when used by their designers.

IV.5. Methodologies and instability

In his classic text *Notes on the Synthesis of Form,* Alexander (1964) illuminates the role of tradition in design. Traditional designs, he suggests, show very little variation in design over a time period measured in generations, whereas more modern designs vary far more rapidly.

Tradition, he argues, encourages rapid adjustment to minor failures in the design, while providing resistance to change in the remainder of the design. As a result the design is able to adjust, subsystem by subsystem, problem by problem. Methodologies, in contrast, do not place such restrictions on stability; and changes in one part of the design often encourage unnecessary change in independent elements of the design. Thus the design is not necessarily heading for closure, but is moving potentially further and further from equilibrium.

With a methodology, all too often the:

> firmness of tradition too, dissolves. The resistance to willful change weakens, and change for its own sake becomes acceptable. Instead of forms being held constant in all respects but one, so that correction can be immediately effective, the interplay of simultaneous changes is now uncontrolled … [and] as a result the system's drive to equilibrium is no longer irreversible; any equilibrium the system finds will not now be sustained; those aspects of the process which could sustain it have dropped away" (Alexander 1964, p. 56).

Thus a move to methodology, rather than tradition with its learned appreciation of quality of materials and workmanship, often results in unstable designs: systems with components that work well in one version and fail in the next. Object-oriented techniques may be an attempt to overcome this problem, although one can question the extent to which they are in widespread use at this time.

IV.6. Methodologies and domination

Methodologies, by their very nature, require steps to be taken. Methodologies therefore can be seen to be performing speech acts (Austin, 1962; Searle, 1969) and speech acts bring with them certain expectations of obligations (Habermas, 1984 and 1987; Whitley, 1996; Winograd and Flores, 1986).

In performing the speech act of requesting a certain action from the person applying the methodology, the designer assumes certain *validity claims*. It is expected by both speaker and hearer that the speech act (the methodology in this case) will:

- be uttering something understandably (the comprehensibility claim);
- be giving the hearer something to understand about the world (the propositional or truth claim);
- be making understandable statements about the world (the truthfulness or sincerity claim); and
- will come to an understanding with another person about the world (the normative validity claim).

Applying these to the world of methodologies, the comprehensibility claim assumes that the user understands how to apply it; the truth claim assumes that the user knows there is a world of computers, organizations, etc.; the sincerity claim assumes that the methodology is making an honest request – you need to interview this group of users for a particular purpose, not just to waste some time and generate more reports; and the normative claim suggests that it is socially acceptable for this to be requested of the user.

It is this last expectation that causes the most problems, because the methodology designer cannot be present for every single application of the methodology, since once it is designed it is severed from the designer. Thus, the person applying the methodology must accept the normative claim (and other validity claims) as is, without being able to challenge the designer, or alternatively discard it altogether (Whitley, 1991). However, the decision to

discard it, in many cases, may not be that of the person applying the methodology. Hence a methodology's power is always at the cost, to a lesser or greater degree, of some form of domination.

IV.7. Methodologies and values

Methodologies are socially constructed. They do not just appear, but are consciously designed and developed by individuals. These individuals come with technological frames (Bijker, 1995) that shape the approach and the assumptions within it. Thus Mumford and Weir's (1979) socio-technical background and belief in empowerment and job redesign shaped the ETHICS methodology, while Checkland's (1981) systems background provides some of the distinct character of his Soft Systems Methodology.

All too often, however, it is assumed that methodologies are value-free. Indeed, it is often claimed that the value-free nature of a methodology ensures that the best solution is developed. Therefore the methodology can be applied in good faith as there is no inherent interest served through the methodology and no one will be discriminated against intentionally.

An example of the value-laden nature of a methodology is given by Tiles and Oberdiek (1995, p. 133). Robert Moses, they report, designed overpasses on the parkways in New York which were:

> deliberately built with a low clearance. This effectively prevented public buses – whose passengers are predominantly poor and African-American – from using these parkways to get to Jones Beach, an acclaimed Moses-designed "public beach".

Similarly, methodologies for the development of computer-based systems are normally biased towards capital and the development of new systems. They aim also to make cost reductions a top priority, despite acknowledged deskilling effects and the downstream effects on employment (Cooley, 1987).

V. Concluding discussion

The paper began by contrasting the relationship between method and understanding found in method-ism, with our own view, which emphasizes the importance of background understanding. It went on to show how methodologies developed in the method-ism mindset reveal their limitations by becoming the visible focus of attention rather than the invisible tools for completing the task at hand. Then the paper examined a number of reasons why engineered methodologies tend to break down if they are used without a background of understanding that enables them to be used appropriately.

These reasons can be grouped into three categories, each of which emphasizes the importance of understanding for the methodology to be fully appreciated (see Table I).

Should we conclude, therefore, that methodology has no significant impact on the developmental effort? We would argue that methodology still plays an important role in the substance of development, but certain adjustments to its use are necessary.

Table I.
Summary of reasons for breakdown of engineered methodologies: the centrality of understanding

Category	Role of understanding	Example
Confines of method	Understanding of the confines of method is required to appreciate the limitations of methodologies and to know when they can be sidestepped to make progress	Variety-poor systems Lack of creativity
Contextual knowledge	Understanding is required to provide a context for the system, to provide the necessary tacit knowledge to enable the methodology to be used and to encourage stability	Meaningless systems Tacit knowledge required Instability
History	An understanding of the history of a methodology helps with the discourse of domination and provides a better appreciation of the values inherent in the methodology	Discourse of domination Value-ladenness

First, we contend that it is essential to abandon the mindset that believes that a suitable single methodology is all that is required. As we have shown, this is not the same as suggesting that we abandon method. If we no longer expect a single methodology to be sufficient for all our concerns, we should expect elements from different methodologies to be picked up and used as and when necessary. Methodologies and their associated tools and techniques should therefore be designed with this improvisation in mind.

Method engineers must expect that their techniques and tools will be used in ways that they themselves did not anticipate. It may be sensible to group together certain tools and techniques in a methodology, particularly if they share a common context of use (for example data-flow diagrams and data dictionaries) or come from a common conceptual basis. But this grouping should not prevent them from being appropriated for other uses, if that is what is required.

Our second adjustment concerns the way that the methodology is used. We have argued that methodology can be effective only when it is combined with background understanding. We would suggest that the most effective way of providing this knowledge is through formal apprenticeship. Informal apprenticeship is currently used to smooth over the limitations of methodologies, but institutionalized apprenticeship will be far more productive. That is, rather than being given only a short training course in a particular methodology, we would expect would-be systems developers to spend considerable time learning about the use of tools in the context of information systems development. In common with all apprentices, these people would come to appreciate the features of the materials and tools that they are working with and recognize quality craftsmanship.

It could be argued that long periods of apprenticeship could prove disastrous for the so-called application backlog. But, given that many current systems are abandoned, modified significantly or are not even delivered, this may not be a bad thing, especially when one takes into consideration the concerns of data

overload resulting from increasing numbers of essentially insignificant systems, highlighted by writers such as Roszak (1994) and Postman (1992).

A strong argument against the ideas presented in this paper is that methodologies *are* used in practice (however limited the coverage may be), and this would suggest that our concerns are theoretical and of limited practical consequence.

We would use the same evidence to suggest the opposite. Methodology use arises because the central tenet of method-ism, namely that the methodology is necessary and sufficient for effective systems development, is being sidestepped. All too often, organizational factors are introduced to ensure the smooth application of a methodology. This may take the form of an experienced project manager assisting inexperienced developers. The manager brings the necessary background understanding to the project and translates the steps of the methodology into practical, doable work.

Methodologies serve other useful social purposes which may result in their "being used" – purposes that are unrelated to the detail of the systems development effort. For example, it may be necessary to be seen to follow a particular methodology in order to be awarded the contract for systems development work (government agencies often require the use of a particular methodology as the non-negotiable basis for awarding work). In other cases, pages of documentation and charts may be used as a prop (Goffman, 1959) to help construct the notion of developer expertise and convince the client of the value added by the methodology used by the system developer.

The view of the rationals detached developer using an objective methodology must be replaced by that of an involved developer who pragmatically tries to get the job done. Method engineers should focus on helping developers in their job of creating usable information systems by creating methodologies, techniques and tools that enable systems developers to focus on their task and develop their understanding of what they are doing, and not end up focusing on the tools that they are using.

V.1. Future research

There is considerable scope for the ideas expressed here to be explored in a practical setting. It would be interesting to undertake a survey, using interpretivist techniques of observation and semi-structured interviews to explore the extent to which successful organizations overcome the limitations put in place by method-ism to ensure that methodology use is effective.

Thus, a researcher who has been sensitized by the ideas outlined in this paper could explore the methodology use of an information technology consultancy or a large systems development house, watching to see how the existing infrastructure, management and workplace support operate to ensure that newly appointed consultants or programmers are supported in those situations where the knowledge provided by the methodology of choice is lacking. Of particular interest would be a comparative study whereby two organizations, with different methodologies of choice (and hence different levels

of background knowledge for different parts of the task) are compared and contrasted, as this would help clarify the relative importance of the different factors outlined in the paper. It would provide also a valuable resource about the nature of best practice in systems development.

References

Alexander, C. (1964), *Notes on the Synthesis of Form*, Harvard University Press, Cambridge, MA.

Ashby, W.R. (1957), *An Introduction to Cybernetics*, Chapman and Hall, New York, NY.

Austin, J.L. (1962), *How to Do Things with Words: The William James Lectures Delivered at Harvard University in 1955*, Clarendon Press, Oxford.

Avgerou, C. and Cornford, T. (1993), "A review of the methodologies movement", *Journal of Information Technology*, Vol. 5, pp. 277-86.

Avgerou, C. and Cornford, T. (1995), "Limitations of information systems theory and practice: a case for pluralism", in Falkenberg, E., Hesse, W. and Olive, A. (Eds.), *Information Systems Concepts: Towards a Consolidation of Views*, Chapman & Hall, London, pp. 130-43.

Beer, S. (1985), *Diagnosing the System for Organizations*, John Wiley & Sons, New York, NY.

Bijker, W.E. (1995), *Of Bicycles, Bakelites, and Bulbs: Towards a Theory of Sociotechnical Change*, MIT Press, Cambridge, MA.

Brinkkemper, S., Lyytinen, K. and Welke, R.J. (1996), *Method Engineering: Principles of Method Construction and Tool Support*, Chapman and Hall, London.

Checkland, P. (1981), *Systems Thinking, Systems Practice*, John Wiley & Sons, Chichester.

Ciborra, C.U. (1996), "Improvisation and information technology in organizations", paper presented at The International Conference on Information Systems (ICIS '96), Cleveland, OH, pp. 368-80.

Collins, H.M. (1992), *Changing Order: Replication and Induction in Scientific Practice*, University of Chicago Press, Chicago, IL.

Cooley, M. (1987), *Architect or Bee? The Human Price of Technology*, Tigerstripe, London.

Feyerabend, P. (1993) *Against Method*, 3rd ed., Verso Books, London.

Fitzgerald, B. (1996), "An investigation of the use of system development methodologies in practice", in Coelho, J.D., Jelassi, T., König, W., Krcmar, H., O'Callaghan, R. and Sääksjarvi, M. (Eds), *Fourth European Conference on Information Systems*, Lisbon, Portugal, pp. 143-61.

Gadamer, H.-G., Weinsheimer, J. and Marshall, D.G. (1989), *Truth and Method*, Sheed & Ward, London.

Goffman, E. (1959), *The Presentation of Self in Everyday Life*, Penguin, London.

Habermas, J. (1984), *The Theory of Communicative Action*, Heinemann Education, London.

Habermas, J. (1987), *The Theory of Communicative Action*, Polity Press, Cambridge.

Heidegger, M. (1927), *Being and Time*, translated (1962) by Macquarrie, J. and Robinson,T., Basil Blackwell, Oxford.

Hidding, G. (1996), "Method engineering: experiences in practice", in Brinkkemper, S., Lyytinen, K. and Welke, R.J. (Eds), *Method Engineering: Principles of Method Construction and Tool Support*, Chapman & Hall, London.

Hutchins, E. (1995), *Cognition in the Wild*, MIT Press, Cambridge, MA.

Ihde, D. (1990), *Technology and the Lifeworld: From Garden to Earth*, Indiana University Press, Bloomington, IN.

Introna L D (1993) Information: A Hermeneutic Perspective. In The First European Conference on Information Systems (Whitley E A ed.) 171-179, Operational Research Society, Henly on Thames.

Introna L D (1997) Management, Information and Power. Macmillan, Basingstoke.

Hidding, G. (1996), "Method engineering: experiences in practice", in Brinkkemper, S., Lyytinen, K. and Welke, R.J. (Eds), *Method Engineering: Principles of Method Construction and Tool Support,* Chapman & Hall, London.

Hutchins, E. (1995), *Cognition in the Wild*, MIT Press, Cambridge, MA.

Ihde, D. (1990), *Technology and the Lifeworld: From Garden to Earth*, Indiana University Press, Bloomington, IN.

Introna, L.D. (1993), "Information: a hermeneutic perspective" in Whitley, E.A. (Ed), *The First European Conference on Information Systems*, Operational Research Society, Henley-on-Thames, pp. 171-9.

Introna, L.D. (1997), *Management, Information and Power*, Macmillan, Basingstoke.

Jantsch, E. (1980), *The Self-Organizing Universe: Scientific and Human Implications of the Emerging Paradigm of Evolution*, Pergamon Press, Oxford.

Land, F., Farbey, B. and Targett, D. (1995), *Hard Money, Soft Outcomes: Evaluating and Managing the Information Technology Investment,* Alfred Waller, Henley-on-Thames.

Latour, B. and Woolgar, S. (1986), *Laboratory Life: the Construction of Scientific Facts*, Princeton University Press, Princeton, NJ.

Mumford, E. and Weir, M. (1979), *Computer Systems in Work Design, the ETHICS Method: Effective Technical and Human Implementation of Computer Systems: a Work Design Exercise Book for Individuals and Groups*, Associated Business Press, London.

Polanyi, M. (1966), *The Tacit Dimension*, Peter Smith, Gloucester, MA.

Polanyi, M. and Green, M. (1969), *Knowing and Being: Essays*, Routledge & Kegan Paul, London.

Postman, N. (1992), *Technopoly: The Surrender of Culture to Technology*, Vintage Books, New York, NY.

Prigogine, I. and Stengers, I. (1985), *Order out of Chaos*, Bantam Books, New York, NY.

Rolland, C. and Prakash, N. (1996), "A proposal for context-specific method engineering", in Brinkkemper, S., Lyytinen, K. and Welke, R.J. (Eds), *Method Engineering: Principles of Method Construction and Tool Support*, Chapman & Hall, London, pp. 191-208.

Roszak, T. (1994), *The Cult of Information: A Neo-Luddite Treatise on High Tech, Artificial Intelligence, and the True Art of Thinking,* 2nd ed., University of California Press, Berkeley CA.

Searle, J.R. (1969), *Speech Acts: An Essay in the Philosophy of Language*, Cambridge University Press, Cambridge.

Tiles, M. and Oberdiek, H. (1995), *Living in a Technological Culture: Human Tools and Human Values*, Routledge, London.

Von Foerster, H. (1984), "Principles of self-organization – in a socio-managerial context", in Ulrich, H. and Probst, G.J. (Eds), *Self-organization and Management of Social Systems*, Springer-Verlag, Berlin, pp. 2-24.

Whitley, E.A. (1996), "Confusion, social knowledge and the design of intelligent machines", *Journal of Theoretical Artificial Intelligence*, Vol. 8 No. 4, pp. 365-81.

Whitley, E.A. (1991), "Two approaches to developing expert systems: a consideration of formal and semi-formal domains", *AI & Society*, Vol. 5 No. 2, pp. 110-27.

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

Wittgenstein, L. (1956), *Philosophical Investigations*, translated by Anscombe, G.E.M., Basil Blackwell, Oxford.