Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 2000 Proceedings

Americas Conference on Information Systems (AMCIS)

2000

A Dialectical Basis for Software Development Tool Building

P Wernick
University of Hertfordshire, p.d.wernick@herts.ac.uk

B Christianson
University of Hertfordshire

M J. Loomes
University of Hertfordshire

D W. Shearer
University of Hertfordshire

A Dialectical Basis for Software Development Tool Building

P Wernick, B Christianson, MJ Loomes and DW Shearer

Department of Computer Science, University of Hertfordshire, Hatfield, Hertfordshire, England

tel. ++44 1707 286323 fax ++44 1707 284303

p.d.wernick@herts.ac.uk

Abstract

We identify typical problems in the interactions of people with current software-based systems. In particular we observe the need to expend significant on-going effort to adapt these systems to reflect changes in the world about them, the need for people to adapt their working practices to fit in with these systems, and the inflexibility of these systems when faced with unusual circumstances or the need for change. We believe that these problems follow, at least in part, from these systems being developed and evolved using mechanisms each based on one Inquiry System only. This basis leads to assumptions being embedded in the mechanisms' analysis outputs, and in system designs and implementations. We suggest that the problems noted may be mitigated by the use of a dialectical approach to Inquiry System selection for software development, based on the work of Hegel, which places in opposition different models of a situation based on different Inquiry Systems. We claim that such a mechanism has the potential to make explicit some of the assumptions which would otherwise be embedded implicitly in the delivered system without being questioned. We outline a research programme intended to test this hypothesis, and suggest other research directions.

Introduction

The problems arising from the need for constant evolution of software systems (Lehman 1998) and the effects of their inflexibility on their users, as encountered in everyday life (Chancellor 2000), have previously been noted. In response to these issues we propose an alternative to the philosophical basis of current software development methods. The long-term objective of this research is to identify and use tools developed for philosophical analysis to inform and improve the practice of software development. We set out below an initial analysis of the current state of use of Inquiry Systems in software development, note its weaknesses, and suggest an improved approach. We also describe additional work which we believe is required.

A Variety of Inquiry Systems

A fundamental element of any mechanism intended to analyse a situation, or to hypothesise about how to change it, is the way in which information is to be gathered and organised; the 'Inquiry System'. Mitroff (1973) describes five types of Inquiry Systems (ISs), differing in their philosophical bases. He states that the differences between them result in different views of what constitutes information, as well as how to obtain it. Thus, a model built using one IS will differ from those developed using others, in its view of the world and in the nature of the elements forming its content.

Mitroff's characterisation of the ISs can be summarised as follows:

- Leibnitzian, in which innate ideas (primitive variables, analytic truths) are combined into more complex arrangements, using logic-based mechanisms. A further mechanism determines whether or not the process is converging to an optimal set. The logical systems which result from such an inquiry are claimed to display rigour, logical coherence, precision, and unambiguity in the use of terms. However, no defence of the selected set of terms can be made from within the logical system, and this IS, whilst suited to the analysis of sets of symbols, is poor at representing experience/empirical content;
- Lockean, inductive and consensual in its basis, emphasising sensory data and building up from these to its conclusions. The result is a combination of human judgement (in contrast to Leibnitz's logical basis) and agreement between people. This IS can take as its inputs much richer experiential data than Leibnitzian systems. However, the simple 'sensations', 'facts' or 'observables' which form Locke's starting point have usually proved on investigation to be more complex than he has suggested, and have thus themselves required further investigation. In addition, the effort required to obtain the necessary consensus may be considerable;
- Kantian, which relies on building more than one Leibnitzian model, each intended to explain the phenomena under investigation, and collecting data for each based on the demands of the model. The inquirer is thus presented with models of the same problem from multiple viewpoints, and is able to select one model from those available as his or her

chosen result. However, the creation of too many models can overwhelm the recipient, the cost is greater due to the necessity to build and collect data for multiple models, and there is no guarantee that the 'right' model will be amongst those constructed;

- Hegelian, based on bringing conflicting models into contact and learning from the result. The mechanism requires the construction of two antagonistic, strongly conflicting Leibnitzian models, embodying conflicting sets of assumptions about the problem under consideration. These models are then applied to the same Lockean data set, with the aim of demonstrating that both models can be supported by the data, and that the diverging conclusions to be drawn differ due to assumptions made in constructing the models. These assumptions are thus opened up for examination. The decision-maker builds his/her own view of the situation, informed by witnessing the conflict and examining the assumptions brought to light. The advantages of this approach include the active involvement of the decision-maker in information interpretation and synthesis into the final model, and the prevention of assumptions from being hidden by agreement amongst all the other participants. Therefore it is seen as being particularly suitable for investigations into ill-structured problems. However, it is less successful in examining well-structured problems. A major problem is that of selecting opposing *Weltanschauungen*;[1] and
- the IS proposed by Singer, and extended by Churchman, which emphasises a perceived need to take an interdisciplinary approach to inquiry, and an approach which encompasses scientific, ethical and aesthetic viewpoints. The aim is to integrate all of these within one frame of reference. As a result, the primitive elements of information which are taken as given and thus unquestioned by the other ISs described, are regarded by a Singerian IS as aspects to be opened up and studied. Unlike the other ISs, it does not regard its primitives as representations of reality, but suggests instead that *any* description becomes become 'real' if its proponents can convince enough people of its validity. The body of information to be considered is no longer purely scientific, but incorporates ethical aspects as well. This IS gives the most comprehensive modelling of the ISs outlined, but problems arise both in making ethical content explicit and in the very complex model specifications necessitated.

A Critique of the Current State of Software Systems Development

An examination of the current state of the development and use of software systems reveals a number of problems. These include the following:

- the embedding, explicitly or implicitly, of assumptions about the system, its environment and the world in software products, necessitating sometimes difficult 'evolutionary' changes or even system replacement when these assumptions are invalidated by time or change (Lehman 1998; Loomes and Jones 1998). The need to evolve software systems results in the evolutionary costs of a software system exceeding the initial development costs, sometimes comprising up to 60%[2] of the total system lifetime costs. This requirement also means that part or all of the system in use at any particular time is not well-suited to its environment;
- a lack of true user involvement in systems development, in which the stakeholders are often presented with a single option and not allowed to make an informed choice between alternatives. For example, a prototyping approach to software development typically consists of presenting a user with one version of a prototype system at a time and asking if *this* is what they want or, if not, what is wrong with it. By not allowing stakeholders to compare and contrast different prototypes placed before them simultaneously, this mechanism fails to offer stakeholders a fully informed choice between the range of alternatives which the technologies and developers' imaginations can provide. In addition, the users can usually only respond within the framework offered by the prototype and the mechanisms which led to its development; radical criticisms are unlikely to arise since the questions leading to them will not be posed. This lack of choice may also reduce the degree of perceived flexibility in the finished systems, as fewer possible models of the problem and/or solution are explored by system developers and stakeholders. The complexity of the situation within which the system will be embedded (and to which it will need to be able to respond) is therefore potentially underestimated, and the need for users to adapt to the system when it is installed is thus increased. In effect, people have become the tools of 'the system' rather than vice versa (Chancellor 2000); and

[1] The *weltanschauung* is the world-view of the observer, or, as Checkland has described it, "the (unquestioned) image or model of the world" (1990: 319)

[2] Pfleeger (1998) reports surveys suggesting that 80% of lifetime system effort is expended on its maintenance, and that only 25% of this effort goes into preventive and corrective maintenance.

- the lack of recognition of the influence of the developers' mind-sets on software systems being developed or evolved. These mind-sets are derived both from the developers' own previous experience and from the tools, techniques, languages and notations which they employ (Loomes 1990; Wernick 1996).

An Alternative to Current Uses of Inquiry Systems in Software Development

In this section, we identify the Inquiry Systems in use in current software development methods, and note that each method only uses one IS in its information-gathering work. We then suggest an alternative to the ways in which ISs are used in these methods, and set out the anticipated advantages of such an approach.

The Use of Inquiry Systems in Software Development

On the basis of Mitroff's interpretations given above, we note that the Inquiry Systems underlying current software development mechanisms generally appear to be either Leibnitzian in style, as is typical of hard systems design 'methodologies' such as SSADM (Downs et al. 1992) or Churchman/Singerian, such as Soft Systems Methodology (SSM: Checkland 1990).

We also observe that only one IS is used for any one stage in the process. At present, when more than one IS is employed in a single software development mechanism, the current approach is to employ them sequentially. The output from one sub-process based on one IS, typically a non-Leibnitzian IS, is used as the input to a second, usually Leibnitzian, sub-process. Consider, for example, Multiview (Avison and Wood-Harper 1993), which uses a Churchman-based SSM analysis, followed by a hard Leibnitzian methodology. In current practice, the use of multiple ISs in combination to address difficult problems is thus often reduced to the decision, which is potentially difficult to reverse, of when to stop using one IS and start using another.

We suggest that adopting a single IS in examining a situation or building a model may result in an incomplete understanding of the situation or in the development of a limited range of models. This is because paradigmatic assumptions (cf. Kuhn 1970) embedded in the IS itself will direct and influence the questions which it is necessary to ask and the interpretation of the answers obtained, and because these assumptions will not be made explicit and exposed to questioning. We question whether, for example, a purely Leibnitzian approach would capture all the nuances of a human-based situation. However, it may also be asked whether multiple models based exclusively on a Churchman or Singerian IS will capture all the aspects needed for the incorporation of a hard software-based system in the final human activity system, given the current state of such software-based systems and their underlying computational models.

We suggest that the adoption of a Hegel-like dialectically-based approach may address this problem successfully, thus alleviating some of the issues raised above. Checkland has employed a Hegelian approach in SSM, placing in opposition different (Churchman-based) SSM root definitions to reflect differing viewpoints (Checkland 1990: 261). However, we believe that a weakness remains in this mechanism, since the IS underlying all these root definitions is still the same, and any assumptions implicit in that IS are perpetuated in all its models. As a result, some of the benefit of a conflict-based mechanism in identifying assumptions may be lost.

We do not take it as being immutable that, in adopting a Hegelian IS as a framework, we are limited to the use of a single IS to build the models to be opposed. We believe that situations should be examined, models developed and systems designed from multiple viewpoints obtained from different ISs. In addition to the potential for exposing assumptions in the ISs themselves, the use of more than one IS may afford a greater opportunity to make explicit the assumptions of the analysts concerned. In any analysis task, the analyst is a part of the observation, recording and analysis system; a multi-IS approach may provide a mechanism for revealing at least partially how this inevitability affects the models produced.

The analogy drawn previously between the development of software systems and Feyerabend's view of science (Feyerabend 1993; see Wernick 1998) supports the view that we may need to oppose a number of different types of model, based on different ISs, to obtain the wider range of viewpoints which we desire. The use of different ISs here may also assist in provoking the conflict needed to bring out the underlying assumptions.

A Mechanism Extending the Hegelian Dialectic

We now outline an approach intended to place in Hegelian opposition before a system's stakeholders a number of models developed using methods based on different Inquiry Systems. The presentation of such a set of models to stakeholders is intended to allow them see the results of taking into account a wider set of viewpoints.

This Hegel-based mechanism comprises:

- the development of models based on different ISs, designed to reflect the current or projected future state of *the same part* of the system and/or the *same situations* – in this, it is unlike current mechanisms,

- the presentation of these models to the intended owners, users and other stakeholders of the system, and
- the selection by the stakeholders, informed by similarities and differences between the models, of one of the models or of a composite model based on selected features of more than one model as a basis for proceeding.

We suggest that this approach would improve software development processes by:

- making explicit and bringing to users' attention at least some of the assumptions currently embedded in software systems analysis and design development tools and people, and thus implicitly embedded in software systems;
- facilitating greater involvement for stakeholders as selectors of the solution, allowing them a greater degree of choice amongst possible solutions and providing them with greater understanding (and therefore control) over the criteria by which such decisions are to be made; and
- allowing the development of more flexible systems, or at least of systems better able to cope with the range of unusual circumstances and rare events found in the real world and often missed or ignored by analysts. This may be achieved as stakeholders' views of what is possible and necessary in the system are informed by the presentation to them of a wider range of situations, options and facilities generated by the different ISs employed, rather than the limited set of options seen from a single standpoint as at present.

Given the parallel efforts which need to be made in each part of the development, it is inevitable that the initial development of real-world software systems employing this approach would cost more than traditional mechanisms, due to the need to model each situation in two or more ways. However, the benefit may be seen be in terms of a system better able to cope with evolutionary pressures over time. As noted above, the latter result in a large proportion of system lifetime costs. The additional initial investment in a Hegelian development mechanism for a long-lived software system may therefore be justified by savings later in the life of the system.

The Nature of 'Conflict' in the Hegelian-based Inquiry System

In developing a software development mechanism based on the Hegelian dialectic, a crucial question to be addressed is that of how to conduct the process of placing the different models in conflict in the presence of the decision-makers in such a way as to optimise the learning process.

In a scientific exploration, this public conflict could be achieved by, for example, devising and conducting an experiment intended to falsify one, and only one, of the theoretical models under consideration. The result of this experiment would show that this one model was (or was not) supported by the now-augmented set of observations. The order in which such experiments were devised and conducted would not affect the results obtained from each experiment.

However, the situation is different in the development of software systems. In this case, the order in which stakeholders are presented with, for example, different prototype systems for examination may affect which prototype they prefer, or which features they might take from each to produce a combined model, as they learn from exposure to successive prototypes more about their problem and how to address it. This may even lead to a need for different groups of analysts and of stakeholder personnel to be involved in developing each model, to avoid the clear division between the different ISs from being bridged by exposure to other viewpoints. However, such an approach would also result in a loss of synergy over the entire set of people involved in the development, since opportunities would inevitably be lost for members of one IS group to learn about the situation from members of other groups.

The purpose of the proposed conflict mechanism in software development is also significant in considering how such a mechanism should operate. Its currently perceived objective is to bring to the surface the assumptions embedded in each model and allow multiple viewpoints to be considered, rather than to falsify one of a set of scientific theories. In the former, the enemy is incompleteness of the system functions or attributes, rather than the identification of inconsistency in a formalised system of scientific theory.

The mechanism would also need to emphasise the nature and quality of the choices made available to decision-makers, with the intention of informing them in their decision-making processes. Some technical system design decisions may need to remain with the software developers, but the dialectical mechanisms would need to inform decision-makers about how and why these decisions were made.

By whatever means it is to be achieved, the placing of two or more models in opposition to each other also requires that those aspects which need to be compared between models are capable of being compared meaningfully; that is, these aspects need to be commensurable (cf. Kuhn 1970). This requirement may place a limit on the diversity of the ISs to be employed in areas where comparisons need to be made, or to an expectation that models may require some reworking or

explanation in order to make the relevant aspects of each model commensurable.

Future Developments

The directions in which this line of research will take us are not yet clear. However, some work is already under way, *viz.* the extension of the philosophical analysis described here, and the design of an investigation to apply these ideas in a practical environment to test their validity and usefulness. The long-term goal of this work is to devise an improved philosophical basis for software development mechanisms in general. We also hope that, as our understanding of this basis grows, it will be possible to make concrete suggestions as to how the practice of current software development can be improved.

Practical Applications of a Dialectically-Based Inquiry System

What we have presented above is a hypothesis, based on observation of the current state of software system development. Practical investigations are needed to determine whether the hypothesis can be supported in practice, and to inform practitioners of any positive conclusions which can be drawn at this stage. The long-term objective of this research programme is neither to develop a 'super-methodology' with the intention of selling it as the solution to all problems in software development, nor to validate the use of any particular tool or set of tools used in combination. Instead, the aim is to determine the success or otherwise of the philosophical research direction, *viz.* to see whether, and if so how well, the underlying dialectic-based mechanism works.

One possible programme of practical research would start with the selection of a set of appropriate ISs and of an existing approach based on each, to save the need to develop new methods and test them individually before starting the research. For example, it may be possible to present, for comparison by stakeholders, models of current situations or proposed software-based systems developed using SSM with its Churchman-based IS, and high-level systems models developed using a hard method such as UML (Rational 2000) or SSADM (Downs et al. 1992) with their Leibnitzian basis. Given the current state of software development, it may be advantageous at present to employ at least one Leibnitzian method or tool-set, in order to reflect the computational models underlying software implementation mechanisms and thus make implementation of the finally-selected model easier.

The selected approaches would then be incorporated into a single project plan. Models developed using the different ISs would be placed before a group of stakeholders, to allow them to make comparisons and to devise a unified model. Guidance to the stakeholders on how to perform this comparison, and support during the process, would almost certainly be required.

An analysis of the lessons to be learned from this process would need to consider how well the different standpoints adopted and models created work together in bringing to the surface the assumptions implicit in the situation, in each approach and in the system functions identified by each approach. The ways in which the comparisons were performed and their degree of success would also need to examined and improved, to inform further investigations. The outputs of the synthesis of the models created could also be compared with the outputs of each approach taken individually. The results of this analysis would be fed into succeeding practical research exercises, possibly employing different sets of ISs.

Furthering the Philosophical Examination

The thinking behind the examination of the current state of software development and its potential for improvement by application of mechanisms based on Hegel's work may lead us to the examination of other related philosophical questions. Currently open questions include how to select the Inquiry Systems underlying a software development method, and how to bring models developed using those ISs into contention.

Looking from a wider perspective, how might the ISs selected interact with other elements of the Kuhnian disciplinary matrices (Wernick 1996; Wernick and Winder 1997) on which the method, and each of the tools comprising it, is based? We suggest initially that the ISs adopted will interact, for example, with the computational models underlying the notations to be used for model-building, to produce a more or less 'hard' approach, and that these in combination will in turn affect the degree of 'hardness' of the final system. The effect of this choice on other possible elements of the disciplinary matrix and how influential they are in determining the mind-set which they project on to their users, is a question for future investigation.

Summary

We have observed that the mechanisms currently in use for the analysis of human activity systems, and the devising of software-based tools intended to improve the working of those systems, are each based on the use of a single Inquiry System at any one stage. We suggest that such an approach might fail to capture a sufficiently wide range of relevant information about the situation to enable a long-lasting software-based system to be developed and installed, and may also embed assumptions in the software itself which might cause problems later in the life of the system. We therefore conclude that the feasibility of approaches incorporating more than one IS at the same time should be examined, and the results of such studies

compared with those obtained from using current mechanisms.

References

Avison, D.E. and Wood-Harper, A.T. *Multiview: An Exploration in Information Systems Development*, Alfred Waller, Henley-on-Thames, 1993, reprint of 1990 revised edition.

Chancellor, A.. untitled column, *Guardian Weekend*, 19 February 2000, p.5.

Checkland, P. *Systems Thinking, Systems Practice*, Wiley, Chichester, 1990, reprint of 1984 revised edition.

Downs, E., Clare, P. and Coe, I. *Structured Systems Analysis and Design Method*, Second Edition, Prentice Hall, New York, 1992.

Feyerabend, P. *Against Method*, third edition, Verso, London, 1993.

Kuhn, T.S. *The Structure of Scientific Revolutions*, Second Edition, Enlarged, University of Chicago Press, Chicago, 1970.

Lehman, M.M. "The Future of Software - Managing Evolution", invited contribution, *IEEE Software* Jan-Feb 1998, pp.40–44.

Loomes, M. J. "Selfconscious or Unselfconscious Software Design?", *J. Information Technology*, (5), 1990, pp.33–36.

Loomes, M.J. and Jones, S. "Requirements Engineering: a Perspective Through Theory-Building", *Proc. Third International Conference on Requirements Engineering*, Colorado Springs, CO, April 6–10 1998, pp.100–107.

Mitroff, I.A. "Systems, Inquiry and the Meanings of Falsification", *Philosophy of Science*, (40), 1973, pp.255–276.

Pfleeger, S. *Software Engineering: Theory and Practice*, Prentice Hall, 1998.

Rational Rational Software Corporation web site on UML, http://www.rational.com/uml/index.jtmpl, (current 6 March 2000).

Wernick, P. *A Belief System Model for Software Development: A Framework by Analogy*, PhD thesis, Department of Computer Science, University College London, 1996.

Wernick, P. "Feyerabend and Information Systems Development : Against Methods?", *Systemist*, (20), December 1998, pp.256-259.

Wernick, P. and Winder R. "Software engineering as a Kuhnian discipline", *in* Winder R., Probert S. and Beeson I. (eds.) *Philosophical Aspects of Information Systems*, Taylor and Francis, London, 1997, pp.117–129.