METHOD-IN-ACTION AND METHOD-IN-TOOL
Some implications for CASE

Brian Lings
School of Engineering, Computer Science and Matehmatics, University of Exeter, Exeter EX4 4PT, England

Björn Lundell
Department of Computer Science, University of Skövde, P.O. Box 408, SE-521 28 Skövde, Sweden

Keywords: IS development method, Method use, CASE tools, Automated support for methods, Stakeholder triangle.

Abstract: Tool support for Information Systems development can be considered from many perspectives, and it is not surprising that different stakeholders perceive such tools very differently. This can contribute on one side to poor selection processes and ineffective deployment of CASE, and on another to inappropriate tool development. In this paper we consider the relationship between CASE tools and Information Systems development methods from three stakeholder perspectives: concept developer, Information Systems developer and product developer. These perspectives, and the tensions between them, are represented within a 'stakeholder triangle', which we use to consider how the concept of method-in-action affects and is affected by the concept of method-in-tool. We believe that the triangle helps when interpreting seemingly conflicting views about CASE adoption and development.

1 INTRODUCTION

Much effort has been expended in the development and analysis of methods within the Information Systems (IS) community. Whatever stance is taken within the ongoing debate about the value and nature of methods (see Avison and Fitzgerald, 2003; Tolvanen *et al.*, 1997; Wastell, 1996), there remains a broad consensus on the need for methods *at least* as a guide to assist thinking and acting within IS development (ISD). This has important consequences when considering ISD tools, particularly if, as stated by Iivari and Lyytinen (1999), one accepts that "ISD without CASE is not a realistic option." (p. 68)

Fitzgerald *et al.* (2002) claim that ISD "is always situated" (p. 178) and "every project is unique" (p. 178). Their interest has been in methods-in-action, reflecting ISD contextual factors, and how these differ from ISD methods as documented (or methods-in-concept).

At the same time, we are faced with the complex relationship between CASE tools and ISD methods. For example, Jankowski (1997) concludes that one of the primary purposes of CASE "is to serve as a companion to the systems development methodology used during the development process." (p. 35) In fact, Hickman and Longman (1994, p. 206) see a CASE Method as a structured approach that will "lead you through all steps in the life cycle of a system", and that use of such a CASE Method "can be automated by a combination of CASE tools."

However, evidence of the value of CASE tools in supporting methods has been variable (Kollman *et al.*, 2002). This has been put down to a lack of fit between tool support actually provided and the systems method employed by individual organisations (Post and Kagan, 2001) – in other words, there is a mismatch between method-in-action and method-in-tool.

The potential value of a CASE tool to an ISD organisation is not only an issue of the technical quality of the product itself (Lundell and Lings, 2003). A prerequisite for a CASE tool to become effectively used in an IS lifecycle process is that it fulfils the specific needs, expectations, value systems and working practices of the relevant stakeholders in the organisation involved.

It is our purpose in this paper to reflect on method-in-tool from the perspective of method-in-action, and thereby contribute to the debate on how best to support information systems development. We therefore introduce a framework through which method-in-tool and method-in-action can be considered.

2 ON ISD METHODS

By method-in-concept, we refer to a method as understood by its stakeholders. In this sense, method-in-concept is a social construction; a shared set of values and assumptions identified with a method within a professional community of method developers and users.

Successful method-in-action may not be achieved simply by adapting a method-in-concept. Viller and Sommerville (2000, p. 169) comment that methods "are unlikely to be adopted in industry unless they can be integrated with existing practice."

Of course, method-in-concept is not itself an objective phenomenon, but is open to interpretation. In this respect, Floyd (1986) distinguishes between methods as perceived by method users (which we will refer to as an *IS developer perspective*) and as perceived by method developers (*concept developer perspective*):

"We consider methods not so much as static, well-defined objects, but as dynamic sources of ideas to be tailored to a given situation and transformed by use ... there is a subtle interplay between the system development process as it is (in our view), as it should be (in our view), and as it should be (according to the method's view)." (p. 30, 31)

It is apparent from the literature that there is a range of views on method-in-concept, and one can detect an even broader range of views on how methods-in-concept might relate to methods-in-action. However, there is a lack of relevant research into this issue, despite its importance to the whole debate on support for ISD. As observed by Moody (2002, p. 393), one explanation may be that it is very difficult to get new methods used in practice, so researching different versions of a method is even more problematic because of the number of people and contexts involved.

3 THE STAKEHOLDER TRIANGLE

The stakeholder triangle (Figure 1) is an analytic device to aid in interpreting a diversity of stakeholder perspectives on CASE support for ISD. It has one of three identified perspectives at each apex. The first apex is that of an ISD organisation and represents an *IS developer perspective*. The second is that of a CASE product developer, and represents a *product developer perspective*. The third is that of a researcher and developer of the underlying methods, techniques, notations, languages, etc., and represents the *concept developer perspective.*

Stakeholders in an *ISD role* have existing practice at least informed by IS methods. There will be at least a tacit concept of method-in-action, for which CASE support will be expected. Different stakeholders will place different organisational requirements on CASE technology.

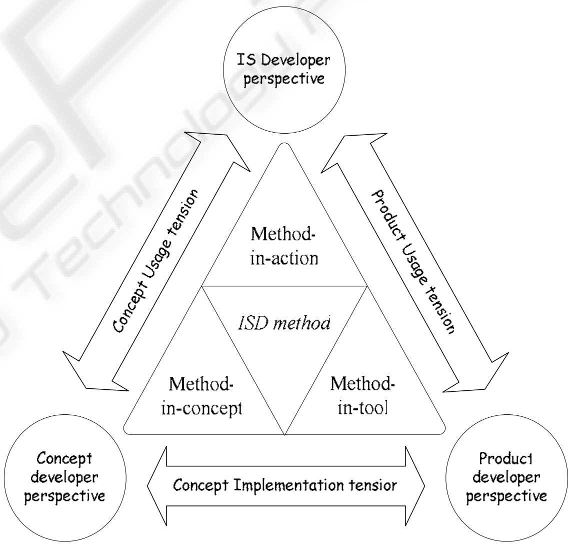

Figure 1: The Stakeholder Triangle (stakeholder perspectives are at each apex).

Stakeholders in a *product development role* are likely to view their role as supporting IS developers in their use of a method. The developers of a product compete for market share in a fluid market by offering attractive solutions which fulfil their customers' demands. This involves offering functionality within supported CASE tools to support users of a given method (or methods). This will necessarily involve interpretation of one or more methods, and this interpretation will implicitly be reflected in supported tool usage. This interpretation of a method we refer to as method-in-tool.

Stakeholders in a *concept development role* will take responsibility for developing and expounding IS methods according to perceived best practice. The outcome from stakeholders taking this role will be proposals in the form of aspects of ISD and software engineering practice, for which it would be potentially useful to provide support in CASE tools – namely, methods-in-concept.

Of equal importance is an understanding of the tensions which exist between the different roles.

The first tension is that between the concept developer perspective and the IS developer perspective, which we refer to as the *concept usage tension*. This is characterised by a discrepancy between methods as prescribed and methods as used.

The second tension is between the concept developer perspective and the product developer perspective, which we refer to as the *concept implementation tension*. This is characterised by a discrepancy between proposals for new concepts (including methods) and their embodiment in tools as delivered (e.g. Jankowski, 1997; Wieringa, 1998). For example, Wieringa (1998) cites the problem of different vendors offering inconsistent execution algorithms but each claiming correct semantics. These are issues related to method-in-tool.

The third tension is the main focus of interest in this paper. It exists between the product developer and IS developer perspectives, and we refer to it is as the *product usage tension*. It is characterised by a discrepancy between tools as delivered and user expectations of those tools, and may result in mixed experience of tool effectiveness (e.g. Senn and Wynekoop, 1995; Maccaria and Riva, 2000). It relates strongly to the first two tensions, which in effect are transitively included in it.

4 ON METHOD-IN-ACTION AND METHOD-IN-TOOL

The basic problem for IS researchers has been well put by Brinkkemper *et al.* (1996) as

"How can proper methodical guidance and corresponding tool support be provided to system developers?" (p. 277)

The stakeholder triangle is an attempt to make explicit the different dimensions of this problem, in order to throw more light on the different ways in which IS researchers have attempted to address this problem. In effect, we would characterise these approaches as focussing ones effort on reducing one of the identified tensions – perhaps only implicitly acknowledging the intrinsic presence of the other two.

For example, Lyytinen *et al.* (1998) conjecture the need in current CASE environments for a user

"to choose between efficient computerized support using a fixed methodical framework which may not fit his situation, or the freedom to do what seems appropriate in the given circumstance, but at the cost of losing efficient technological support."

This is a characterisation of the problem of addressing the *product usage tension*; in the first case by adopting method-in-tool and thereby equating method-in-tool with method-in action; in the second case by not using an automated tool, thereby removing method-in-tool and its associated tension. As acknowledged in the paper, neither approach is effective.

Method Engineering (ME) is central to many researchers' approaches to addressing the *concept usage tension*. Truex and Avison (2003, p. 508-510) characterise method engineering as effectively taking a *concept developer* perspective, aiming to address the *concept usage tension* by widening the scope of methods. Early approaches (which they categorise as Types I and II) acknowledge method-in-action through abstracting "best practice" – but remain "perhaps bureaucratic". Most later approaches increase flexibility but retain "a very technical view of the development of IS", still addressing method-in-action through method-in-concept. These they categorise as Type III approaches. This type of Method Engineering has been labelled as "methodism" by Introna and Whitely (1997, p. 31). They argue that the belief amongst method engineers that "it is possible to incorporate all necessary knowledge" is wrong, instead arguing that successful use of a method "depends on a broader, already present, tacit understanding of the world" which is not to be found in any one particular method. Following this line, Truex and Avison (2003) stress the advantages of what they categorise as Type IV approaches, characterised by recognising organisations as social constructs and resulting in highly situated methods-in-action. However, they highlight the complexities of such a contingent approach to method development. In particular, they highlight potential problems with respect to method congruence and selection, and "the control and application of standards". Such complexity can be seen to extend in the product dimension, where they see CASE support for ME as enforcing method-in-tool – "the sequence and the description of design" – on developers. Finally, Rossi *et al.* (2000, p. 3) note that where all ME approaches "fall short" is in maintenance, as "only the products of the design process are 'documented', not the process of arriving at design solutions."

4.1 On expectations of CASE support for Method-in-Action

In offering a framework for classifying CASE method support, Jankowski (1997) notes that support does not have to mean restrictive enforcement of process.

It is not uncommon for ISD organisations to adopt a variety of different types of CASE tool throughout a project life-cycle, putting demands on the integration and exchange of development data between tools. Unfortunately, as different tools are usually equipped with proprietary solutions, such practice makes it very difficult to facilitate seamless integration throughout the life-cycle. In fact, Kollmann *et al.* (2002), reporting on experiences of CASE tool support for reverse engineering, claim that "it is difficult, or even impossible, to ensure that model semantics remains unambiguous when working with different tools at the same time." (p. 22). Such issues can undermine expectations of "increased productivity, improved product quality, easier maintenance, and making software development a more enjoyable task." (Jarzabek and Huang, 1998, p. 93)

The situation is further complicated by the fact that many versions of the same documented method can be used at the same time within an organisation through (perhaps dynamic) tailoring (Fitzgerald *et al.*, 2003). Rossi *et al.* (2000, p. 3) see the maintenance and analysis of different method versions as a major weakness in current ME approaches. Further, because a method-in-tool may differ from both the method-in-concept (which will itself evolve) and the method's variants as used within the organisation (methods-in-action), later phases of development may therefore be within very changed environments. Hence, stakeholders may well be using different sets of tools within the lifecycle of a single IS, and it is important to managers not to be locked in to any integrated environment. Such environments are unlikely to outlive the IS under development; there are many systems still being maintained after 30 years, but the same cannot be said yet of any CASE product on the market (Lundell and Lings, 2004).

4.2 On developing CASE support for Method-in-Action

With recent observations amongst researchers (e.g. Glass, 1999) that CASE tools become shelfware and "unused by practitioners" (p. 76), it seems necessary for vendors to adopt a critical perspective when analysing potential reasons for limited usage of CASE tools. One explanation may be that current tools do not adequately support preferred work practices amongst IS developers. For example, Van Der Straete *et al.* (2003, p. 326) claim that "State-of-the-art UML CASE tools provide poor support for consistency maintenance", and propose an extension to the UML metamodel to resolve this. In other words, a 'better' method-in-concept (which is to be implemented in a tool) is proposed as a way to better support an existing method-in-action.

More fundamental are the opposing views on whether tools should in fact be primarily method centred. Jarzabek and Huang (1998) observe that "method-centred CASE tools are not attractive enough to users." (p. 93) Rather paradoxically, however, the realisation of methods in tools can be "seen as beneficial to the wider acceptance of" the method (Gray, 1997, p. 235). These are clear symptoms of the tensions resulting from discrepancies between method-in-action and method-in-tool, tensions which some see as irresolvable without a more radical approach to tools which places "the stakeholder centre stage" (King, 1997, p. 329)

A good method-in-tool should ideally support a designer's creative activities in ISD. One aspect of this is the way by which tools limit the design space by enforcing constraints in CASE tools. As noted by Scott *et al.* (2000), constraint enforcement in tools may have both positive and negative implications. On the positive side, constraint enforcement may "guide designers toward good solutions" (p. 232), but such enforcement "may also frustrate designers, who may relegate CASE tool to simple design capture rather than creative design development" (p. 232). As observed by Lending and Chervary (2002, p. 81), systems developers who "perceive their tool as restrictive" think their tool is less useful than those who find their tool flexible.

Clearly it is important for vendors to find an appropriate balance, as the way by which they design their tools will have consequences for how users may perceive the tools. For example, Brooks and Scott (2001) report on individual variation amongst different developers, and note that vendor's strategies for implementation of constraints on methods in their tools often seems "quite arbitrary" (p. 285). Similarly, Fowler (2003, p. 325) observes that "the priorities of those who develop the UML are not the same as all the UML's users." In other words, there is a different between method-in-concept and method-in-action which, of course, has a ripple effect on vendors' decisions for incorporation of UML into their tools (method-in-tool), if aiming to support a range of different preferences amongst users with respect to method-in-action.

Commercial CASE vendors have addressed complexity in the CASE environment by opening their proprietary products in a limited way, through export and import facilities. In such a CASE environment, a set of CASE products being supplied from different vendors can inter-operate, and each vendor can thereby focus on their "speciality", leading to decreased complexity in each individual product. However, the CASE vendors may well not utilise standard interchange formats as they can be seen as a threat to market share – allowing migration to other tools. It is an open issue whether interoperability will raise related issues in the open source community. Lyytinen *et al*. (1998) go further than simple CASE data interchange, arguing for high levels of adaptability based on metamodeling of ontology, notations and process.

In fact, the problem of tool integration appears to be a growing one in that "more and more production software organizations are [again] using old-fashioned stand-alone development tools and struggling to match up tools and their outputs and inputs to do software engineering" (Hart *et al.*, 1999, p. 225).

5 SUMMARY AND REFLECTIONS

In this paper we have presented the stakeholder triangle as representing the associated tensions apparent when considering tool support for ISD activities. It is clear that two of the tensions identified have already been explicitly but independently explored in the literature; a third tension is apparent in product usage. It concerns differences in perceptions of the utility of tools amongst stakeholders in ISD organisations, and has previously only been addressed rather implicitly. This tension arises because tools are not used in practice as envisaged and planned for by their manufacturers. This is unsurprising if one considers the mismatch apparent between method-in-action and method-in-tool. We elaborate on some of the possible implications of recognising this mismatch.

From the perspective of an IS developer, tool evaluation needs to be grounded not only in the culture and technical demands of an organisation, but must also allow systematic review of the implications and limitations of adopting specific tools. In particular, prescriptive tool use is unlikely to lead to successful adoption, so each manufacturer's design philosophy needs to be scrutinised. For example, does a tool offer flexible method support, such as notification of integrity violations, rather than enforcing a particular methodological approach designed to lessen such violations? And if enforcement is offered, is it (or can it be made) consistent with organisational practice?

From the perspective of product developers, it is important not to be prescriptive with respect to method-in-tool. This would inhibit tool take up because of the diversity of method-in-action. Flexibility in tailoring is clearly an important property to aim for; it is in fact important to acknowledge that the underlying philosophy of a tool's designers is likely to be shared by few development groups. Further, these groups are likely to be multi-tool based, and look for interoperability features to allow them greater scope in tailoring their environments to ones suited to their own value systems.

As a consequence, from the perspective of method developers, method tailoring must be accepted as a legitimate and expected goal both of method users and of tool developers. It is important to be explicit about the underlying assumptions behind methods, and to stress the important constraints on consistency within a method rather than attempting to be prescriptive through process. Such an approach would give much greater input to other stakeholders' decision processes, whether towards tool design or systems development.

REFERENCES

Avison, D. & Fitzgerald, G., 2003. Where Now for Development Methodologies?. *Communications of the ACM*, 46(1), 79-82.

Brinkkemper, S., 1996. Method engineering: engineering of information systems development methods and tools. *Information and Software Technology*, 38(4), 275-280.

Brooks, A. & Scott, L., 2001. Constraints in CASE tools: results from curiosity driven research. In *Proceedings 2001 Australian Software Engineering Conference*, IEEE Computer Society, pp. 285-293.

Fitzgerald, B., Russo, N.L. & Stolterman, E., 2002. *Information Systems Development: Methods in Action*. McGraw-Hill, London.

Fitzgerald, B., Russo, N.L. & O'Kane, T., 2003. Software Development Method Tailoring At Motorola. *Communications of the ACM*, 46(4), 65-70.

Floyd, C., 1986. A Comparative Evaluation of System Development Methods. In *Information Systems Design Methodologies: Improving the Practice*, North-Holland, Amsterdam, pp. 19-54.

Fowler, M., 2003. What Is the Point of UML?, In *UML 2003 - The Unified Modeling Language: Modeling Languages and Applications*, Springer, Heidelberg, p. 325.

Glass, R.L., 1999. The Loyal Opposition: Of Open Source, Linux, and Hype. *IEEE Software*, 16(1), 126-128.

Gray, J.P., 1997. CASE tool construction for a parallel software development methodology. *Information and Software Technology*, 39(4), 235-252.

Hart, H., Boehm, B., Taft, S.T. & Wasserman, T., 1999. PANEL: What Happened to Integrated Environments?. In *Proceedings of the ACM SIGAda Annual International Conference on Ada*, pp. 225-226.

Hickman, L. and Longman, C., 1994. *CASE Method: Business Interviewing*, Addison-Wesley, Wokingham.

Iivari, J. & Lyytinen, K., 1999. Research on Information Systems Development in Scandinavia: Unity in Plurality. In *Rethinking Management Information Systems: An Interdisciplinary Perspective*, Oxford University Press, Oxford, pp. 57-102.

Introna, L.D. & Whitley, E.A., 1997. Against Method-*ism*: Exploring the limits of method. *Information Technology & People*, 10(1), 31-45.

Jankowski, D., 1997. Computer-Aided Systems Engineering Methodology Support and Its Effect on the Output of Structured Analysis. *Empirical Software Engineering*, 2(1), 11-38.

Jarzabek, S. & Huang, R., 1998. The Case for User-Centred CASE Tools. *Communications of the ACM*, 41(8), 93-99.

King, S. (1997) Tool support for systems emergence: A multimedia CASE tool. *Information and Software Technology*, 39(5), 323-330.

Kollman, R., Selonen, P., Stroulia, E., Systä, T. & Zundorf, A., 2002. A Study of the Current State of the Art in Tool-Supported UML-Based Static Reverse Engineering. In *Proceedings of the Ninth Working Conference on Reverse Engineering (WCRE'02)*, IEEE Computer Society, pp. 22-32.

Lending, D & Chervany, N.L., 2002. CASE tool use and job design: a restrictiveness/flexibility explanation. *Journal of Computer Information Systems*, 43(1), 81-90.

Lundell, B. & Lings, B., 2003. The *2G* method for doubly grounding evaluation frameworks. *Information Systems Journal*, 13(4), 375-398.

Lundell, B. & Lings, B., 2004, Changing perceptions of CASE-technology. *Journal of Systems and Software, (to appear)*.

Lyytinen, K., Martiin, P., Tolvanen, J.-P., Jarke, M., Pohl, K. & Weidenhaupt, K. (1998) CASE Environment Adaptability: Bridging the Islands of Automation, In *Proceedings of the Eight annual Workshop on Information Technologies and Systems WITS'98*, University of Jyväskylä.

Maccari, A. & Riva C., 2000. Empirical Evaluation of CASE Tools Usage at Nokia. *Empirical Software Engineering*, 5(3), 287-299.

Moody, D.L. 2002. Validation of a method for representing large entity relationship models: an action research study. In *European Conference on Information Systems*, Gdansk, pp. 391-405.

Post, G. & Kagan, A., 2001. User requirements for OO CASE tools. *Information and Software Technology*, 43(8), 509-517.

Rossi, M., Tolvanen, J.-P., Lyytinen, K. & Kaipala, J., 2000. Method Rationale in Method Engineering. In *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, IEEE Computer Society, pp. 1-10, Vol. 2.

Scott, L., Horvath, L. & Day, D., 2000. Characterizing CASE Constraints. *Communications of the ACM*, 43(11), 232-238.

Senn, J.A. & Wynekoop, J.L., 1995. The other side of CASE implementation: Best Practices for Success. *Information Systems Management*, 12(4), 7-14.

Tolvanen, J.-P., Rossi, M. & Liu, H., 1997. Method Engineering: Current research directions and implications for future research, In *Method Engineering: Principles of method construction and tool support*, Chapman & Hall, London, pp. 296-317.

Truex, D. & Avison, D., 2003. Method Engineering: Reflections on the Past and Ways Forward, In *Ninth Americas Conference on Information Systems*, pp. 508-514.

Van Der Straeten, R., Mens, T., Simmonds, J. & Jonckers, V., 2003. Using Description Logic to Maintain Consistency between UML Models. In *UML 2003 - The Unified Modeling Language: Modeling Languages and Applications*, Springer, Heidelberg, pp. 326-340.

Wastell, D.G., 1996. The fetish of technique: methodology as a social defence. *Information Systems Journal*, 6(1), 25-49.

Wieringa, R. 1998. A Survey of Structured and Object-Oriented Software Specification Methods and Techniques. *ACM Computing Surveys*, 30(4), 459-527.

Viller, S. & Sommerville, I., 2000. Ethnographicaly informed analysis for software engineers. *International Journal of Human-Computer Studies*, 53(1), 169-196.