Method Engineering: Theory and Practice

B. Henderson-Sellers

Faculty of Information Technology
University of Technology, Sydney
Australia
brian@it.uts.edu.au

Abstract: Method engineering promotes the idea of constructing methodologies for information systems development by selecting and assembling method fragments from a repository. This repository needs first to be populated with self-contained fragments derived from industry best practice and compliant with a given metamodel. A situational method is then constructed (by a method engineer) to exactly match the requirements of the individual organization and/or project. This paper briefly outlines both the theory of situational method engineering and its application in terms of industry case studies carried out over the last 5 years in Sydney, Australia in helping organizations to create agile and flexible methodologies, capable of maturing and improving over the years.

1 Introduction to Methods and Method Engineering

Method engineering is a solution offered to the problem of the identification of the "most appropriate" methodology for an organization and/or its projects. [Ly87] notes that current methodologies are not well suited to practice, while [Av96] notes a backlash against formal methodologies. Others see process adoption as a "waste of time" [BH03], although [Co00] argues that it is both appropriate and necessary for an organization to have available to it a suite of methodologies. Since the one-size-fits-all methodology is now generally regarded as unattainable e.g. [Br87] [AW91] [KW92] [SB93] [VG94] [SH96] [[FTF96] [HV97] [Gl00] [FRO03] [WK04], alternatives have to be sought, particularly ones that take into account the human and organizational elements [CL94]. Method engineering, perhaps accompanied by method tailoring/customization, is the current most optimistic route and forms the topic of this paper.

1.1 Terminology

In the field of process and method engineering, the terminology is often differently used between authors. There are three "key" high-level terms: method, methodology and process. While the etymology of "methodology" gives its definition as the study of methods, its widespread and common usage to mean "method" [Ja94] [BGH04] gives it credence in this second meaning (a meaning also given in many modern dictionaries e.g.

the American Merriam-Webster Dictionary: [Co00]). For the purposes of this short paper, we will indeed take the words method and methodology as synonyms.

The difference in meaning between the words “process” and “method/methodology” is harder to pin down. In general terms, a process is a way of acting, of doing something. Thus the way you relocate yourself from home to the work environment follows some (usually predefined – or least practised and often repeated) process. Thus, process is intangible. However, to complement a process, there are other things that a software developer must be cognizant of - in particular, the work products produced and consumed and the people and tools involved in that production and consumption. Time sequencing is also of significant interest and concern. The word we will use here for this overall combination will be methodology or method. In other words, a methodology encompasses absolutely everything needed for software development – [Co00] calls this a “Big-M methodology”, [Gl00] the “capital-M Methodology”. Many authors use the description of an overall methodology as having two (often intertwined – or at least interdependent) aspects: product and process e.g. [RPB99].

Another viewpoint is that the process describes what is actually done in real time with a real team on a real project. This is particularly the case in the capability assessment field where the focus of a capability assessment is *the process as it is performed* e.g. [Pa93] [Do93] [II98] – although often, for example in ISO12207 [II95], processes are at a smaller granularity and are defined solely in terms of purpose and outcomes. Each process focusses on what is input to the process and what is output. It should be noted that there are both a static and dynamic aspect to such a notion of process i.e. the process (static enactment) whereby real developer’s names, deadlines, deliverables replace the generic placeholders in the process model and the dynamics of the process as it is actually enacted. [Gr01] call the static enactment a “process model instance” and the dynamic enactment “process performance”. In this approach, the formal description of the “process” to be followed is usefully and frequently described as a *process model* e.g. [FKN94] [Gn01] http://www.opfro.org. This is the information that is typically documented in books, in reports or on a website and alternatively labelled as a “method(ology)” e.g. [BGH04].

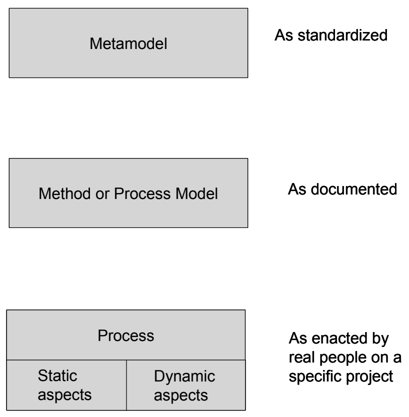

Figure 1 Thee “layers” of process and method terminology

Together, this gives a multiple-layered model in which process, process model, method(ology) and metamodel can be depicted as suggested in Figure 1.

1.2 Method Engineering

Method Engineering (ME) was introduced by [BJO85] and then, more recently, by [KW92] who named it methodology engineering; but [SB93] and [Br96] strongly recommend changing this to method engineering, a term that has been generally accepted. Brinkkemper's [BR96] definition of method engineering is useful here: "Method engineering is the engineering discipline to design, construct and adapt methods, techniques and tools for the development of information systems." When applied to a particular situational context, it is often referred to as "situational method engineering" or SME. Interestingly, [Gl00] equates the ME approach to an "ad hoc" approach in that the correct meaning of ad hoc is "suited to purpose" or "tailored to the problem at hand".

Method engineering focusses not on the acquisition of a ready-made method from some supplier (vendor or book-writing methodologist) but on the in-house construction of an organization-specific or project-specific methodological approach. This construction is accomplished by selecting pieces of method (method fragments or method chunks) that have been already created and stored in a repository or methodbase. The source of these stored fragments is not critical to the use of SME by practising software developers. They may be "carved out" of other pre-existing methods e.g. [RR01] or instantiated from a standardized metamodel e.g. [He02a].

2 The Role of an Underpinning Metamodel

The use of metamodels in general is recommended in [Ma91] [RSM95] [RP96] [TRL96] [Ja98] [KBS00]; and was always the core underpinning the OPEN Process Framework e.g. [He96] [GHY97] [FH02]. Indeed, [RDR03] refer to it as the "core technique in SME". Metamodels provide a means of defining the rules (for a modelling language or a methodology) at a higher level of abstraction. Metamodels are created by a metamodelling activity, supporting the formalism used by developers in their own modelling of a system.

There are multiple dimensions to modelling. In particular, models can be "stacked" in terms of their abstraction level. This is readily seen in the 4-layer metalevel hierarchy of the OMG (see e.g. [OM01]) – although it should be noted that the inter-level relationship of "instance-of" has recently been subject to some criticism e.g. [Se03a] [GH05]. Metamodels then became, within the OMG, the underpinning rationale not only for the UML but also for the MOF, SPEM and MDA standards initiatives. More recent standardization efforts in the use of metamodels for underpinning methodologies (and hence ME/SME) have been seen in Australian Standard 4651 [SA04] and the embryonic ISO 24744 standard. Both of these standards eschew the strict metamodelling approach of the OMG because of the serious non-transitivity problems with "instance-of" relationships e.g. [AK01], replacing it with powertype patterns [Od94] [GH05]. Although not fitting into a strict metamodelling mindset, powertypes (and their associated set representation – Figure 2) provide a solution more aligned to people and their endeavours (Figure 3). With these three layers (Endeavour, Method and (formal) Metamodel – in which all conceptual, powertype models exist), attributes can be assigned either to (i) the xxx element in the metamodel from which a method level entity[1] can inherit or (ii) the xxxKind element in the metamodel from which a slot value can be instantiated in the method level entity. In the first case, the inherited attribute is then given a slot value at the next level i.e. that of the Endeavour. In the second case, the value given at the method level acts as a kind of Class Attibute. These ideas are illustrated in Figure 4 where xxx ≡ Document.

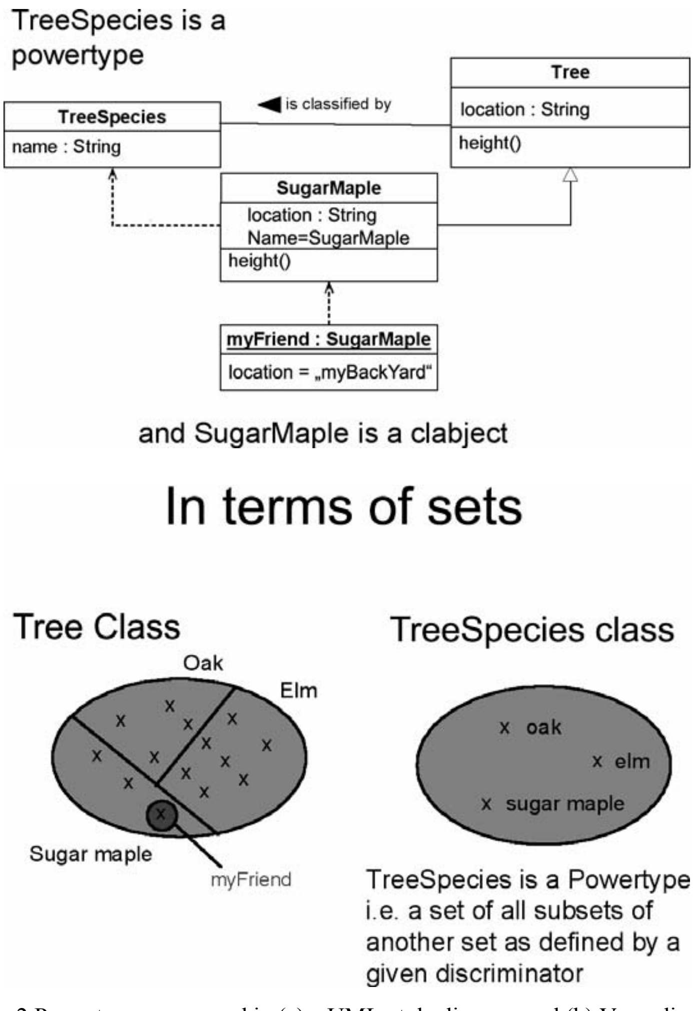

Figure 2 Powertypes expressed in (a) a UML-style diagram and (b) Venn diagrams

[1] Actually this is a clabject [At98] [AK00], which is defined as having both an object facet and a class facet.

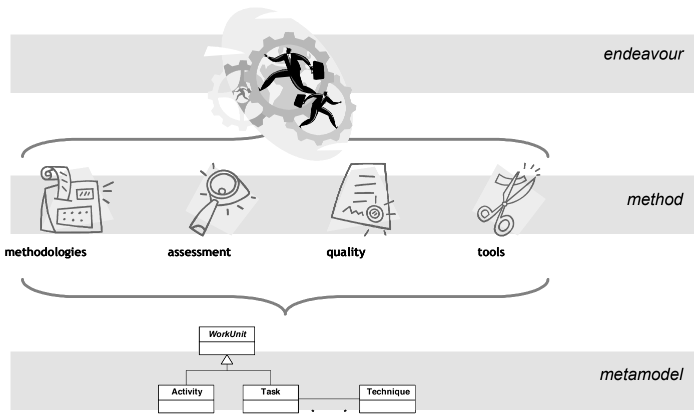

Figure 3 Revised metalevel hierarchy based on practice rather than theory

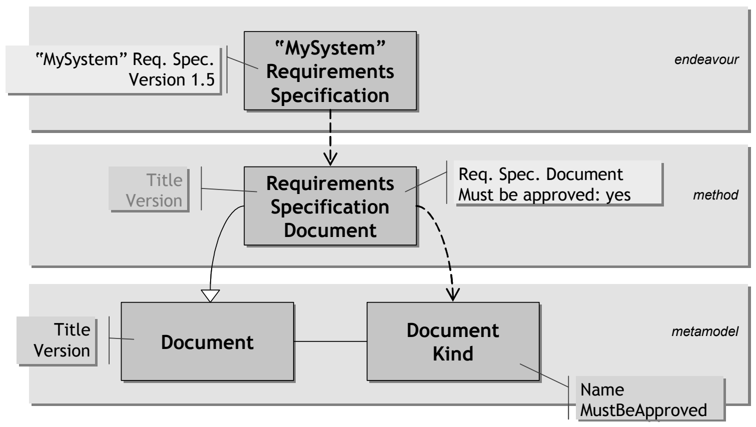

Figure 4 Application of the powertype pattern for Document/Kind in the context of the three levels of Figure 3.

3 Method Engineering with the OPEN Process Framework Approach

The OPEN Process Framework (OPF) [GHY97] [FH02] is defined by a metamodel that supports the concepts of method engineering. It provides a rich repository of method fragments, which can be used in different software projects, together with a set of guidelines (Figure 5) offering advice on the fragment selection based on the notion of possibility matrices linking each pair of method fragments [HLH02b].

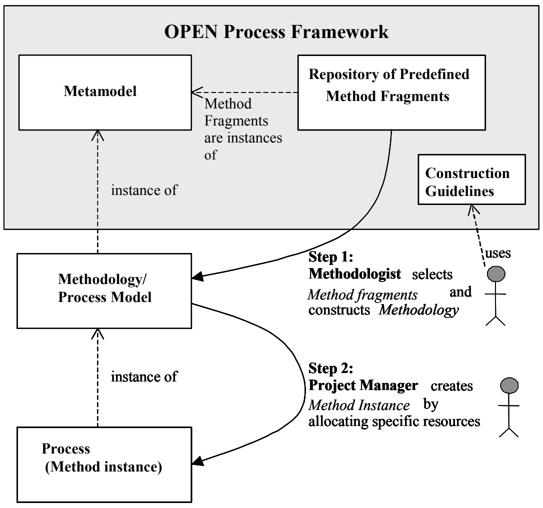
OPEN Process Framework
Metamodel
Repository of Predefined Method Fragments
Method Fragments are instances of
Construction Guidelines
instance of
uses
Step 1:
Methodologist selects Method fragments and constructs Methodology
Methodology/ Process Model
instance of
Step 2:
Project Manager creates Method Instance by allocating specific resources
Process (Method instance)

Figure 5 Method engineering using the OPF (after [HH03])

In OPF's process metamodel, there are elements to describe process fragments such as Activities, Tasks and Techniques; people components such as Producers and Roles; organizational components, such as Enterprise, Programme and Project, and product fragments in the form of a whole range of Work Products including diagrams and documents, supported by various kinds of languages (natural language, modelling language and coding language). Method construction may be top down or bottom up. Using the former as an example, the method engineering would select appropriate activities from the OPF fragment repository and then, using the possibility (or deontic) matrix approach, choose appropriate tasks, techniques, producers, work products etc.

Exemplar processes have been published to support, for example, web development [HLH02a] and agility [He02b] and, more recently, for agent-oriented software projects e.g. [He05].

4 Field trials

In order to evaluate empirically the value of the (situational) method engineering approach to industry efficacy, two major action research (AR) studies e.g. [Av99] were undertaken with Sydney-based organizations. Initially, a transition process was engineered from fragments in the OPF repository [HS00]. Using this, an action research project with a legal publisher identified both cultural (organizational and personal) as well as technical issues that might inhibit the successful use of SME [Se02] [SH04a] [HS05]. A second industry AR study was later conducted within the IT department of a large state government department [SH04b] during their transformation to e-government. This latter project successfully empowered the IT section within the governmental department with a new constructed agile method, the Usage-Centered Design (UCD)

[CL99] approach and a core set of the UML for their e-government web development initiative [SH04c] [SH04d].

The main objective of these empirical evaluations was to evaluate the introduction of a method engineering approach using the OPEN Process Framework (OPF) in order to test the construction of an agile methodology for these two organizations, a construction done in such a way that the methodology can be fully customized to suit individual projects and can later be flexibly adapted as the organization's process capability matures – so-called dual agility [HS05]. Both organizations benefited from small incremental process improvements – called a "small wins" strategy [Se02] [Se03b].

The method engineering and dual-agility approaches were strongly recommended to both study organizations for the following reasons [HS05]:

- Each organization must develop its own way of working contrary to the adaptation to an existing way [Co02].
- Based on previous experience, the idea of adopting an existing method, even agile, was rejected due to people's belief that they must adapt themselves to suit the adopted method.
- The need for a method that provides manoeuverability to deal with requirement changes.
- The need for a method that enables teams to deliver software products faster to their customers.
- Dual-agility not only provides a better way of developing software but can be enhanced by the use of SME to support method customization to best suit people's need.
- For both organizations, IT customers are internal and available most of the time for immediate review and providing feedback.
- Getting customer involvement through the entire development lifecycle is a top priority – as in the Agile Manifesto [Ag01].

5 Summary

This paper has described the "bare bones" of method engineering in both theory and practice. Experience gained during the two action research projects showed that the process of method engineering, with a rich repository of method fragments such as those provided by the OPF, can change people's perception of the main role of a software development method that, in turn, can become a driving force for culture change and resistance management.

6 Acknowledgements

The work described here was supported by several grants from the Australian Research Council, whom we thank for financial support.

References

[Ag01] AgileManifesto, 2001, Manifesto for Agile Software Development.

[At98] Atkinson, C., 1998, Supporting and applying the UML conceptual framework. In «UML» 1998: Beyond the Notation, Vol. 1618. Bézivin, J. and Muller, P.-A. (eds). Springer-Verlag, Berlin, 21-36.

[AK00] Atkinson, C. and Kühne, T., 2000, Meta-level independent modelling. In International Workshop on Model Engineering at 14th European Conference on Object-Oriented Programming.

[AK01] Atkinson, C. and Kühne, T., 2001. Processes and Products in a Multi-level Metamodeling Architecture. Int. J. Software Eng. and Knowledge Eng. 11(6), 761-783.

[Av96] Avison, D.E., 1996, Information systems development methodologies: a broader perspective, in Method Engineering. Principles of Method Construction and Too Support. Procs. IFIP TC8, WG8.1/8.2 Working Conference on Method Engineering, 26-28 August 1996, Atlanta, USA (eds. S. Brinkkemper, K. Lyytinen and R.J. Welke), Chapman & Hall, London, 263-277

[Av99] Avison, D.E., Lau, F., Myers, M. and Nielsen, P.A., 1999, Making academic research more relevant, Communications of the ACM, 42(1), 94-97

[AW91] Avison, D.E. and Wood-Harper, A.T., 1991, Information systems development research: an exploration of ideas in practice, The Computer Journal, 34(2), 98-112

[BH03] Baddoo, N. and Hall, T., 2003, De-motivators for software process improvement: an analysis of practitioners' views, Journal of Systems and Software, 66, 23-33.

[BJO85] Bergstra, J., Jonkers, H. and Obbink, J., 1985, A software development model for method engineering, in ESPRIT'84: Status report of ongoing work (eds. J. Roukens and J. Renuart), Elsevier Science Publishers .V., North-Holland

[BGH04] Berki, E., Georgiadou, E. and Holcombe, M., 2004, Requirements engineering and process modelling in software quality management – towards a generic process metamodel, Software Quality Journal, 12, 265-283

[Br96] Brinkkemper, S., 1996, Method Engineering: Engineering of Information Systems Development Methods and Tools. Inf. Software Technol., 38(4), 275-280.

[Br87] Brooks, F.P. jr., 1987, No silver bullet: essence and accidents of software engineering, IEEE Computer, 20(4), 10-19

[Co00] Cockburn, A., 2000, Selecting a project's methodology, IEEE Software, 17(4), 64-71.

[Co02] Cockburn, A., 2002, 'An interview with Alistair', The Cutter Consortium, http://www.cutter.com/consultants/cockburna.html accessed on the 23rd of February, 2004

[CL94] Constantine, L.L. and Lockwood, L.A.D., 1994, One size does not fit all: fitting practices to people, American Programmer, 7(12), 30-38

[CL99] Constantine, L.L. and Lockwood, L.A.D., 1999, Software for Use, Addison-Wesley

Do93] Dorling, A., 1993, SPICE: Software process improvement and capability determination, Information and Software Technology, 35(6/7), 404-406

[FTF96] Fayad, M.E., Tsai, W.T. and Fulghum, M.L., 1996, Transition to object-oriented software development, Communications of the ACM, 39(2), 108-121

[FKN94] Finkelstein, A., Kramer, J. and Nuseibeh, B., 1994, Software Process Modelling and Technology, Research Studies Press Ltd., John Wiley & Sons Inc., Taunton, England

[FH02] Firesmith, D.G. and Henderson-Sellers, B., 2002, The OPEN Process Framework. An Introduction, Addison-Wesley, 330pp 2002

[FRO03] Fitzgerald, B., Russo, N.L. and O'Kane, T., 2003, Software development method tailoring at Motorola, CACM, 46(4), 65-70

[Gl00] Glass, R.L., 2000, Process diversity and a computing old wives'/husbands' tale, IEEE Software, 17(4), 128-127

[Gn01] Gnatz, M., Marschall, F., Popp, G., Rausch, A. and Schwerin, W., 2001, Towards a living software process development process based on process patterns, Software Process Technology, Proceedings of the 8th European Workshop, EWSPT 2001 (ed. V. Ambriola), LNCS 2077, Springer-Verlag, Berlin, 182–202

[GH05] Gonzalez-Perez, C. and Henderson-Sellers, B., 2005, A Powertype-Based Metamodelling Framework. Software and Systems Modelling, 4(4), DOI 10.1007/210270-005-0099-9

[GHY97] Graham, I., Henderson-Sellers, B. and Younessi, H., 1997, The OPEN Process Specification, Addison-Wesley, UK, 314pp

[Gr01] Greenwood, R.M., Balasubramaniam, D., Kirby, G., Mayes, K., Morrison, R., Seet, W., Warboys, B. and Zirintsis, E., 2001, Reflection and reification in process system evolution: experience and opportunity, Software Process Technology, Proceedings of the 8th European Workshop, EWSPT 2001 (ed. V. Ambriola), LNCS 2077, Springer-Verlag, Berlin, 27–38

[He02a] Henderson-Sellers, B., 2002, Process metamodelling and process construction: examples using the OPEN Process Framework (OPF)}, Annals of Software Engineering, 14, 341-362

[He02b] Henderson-Sellers, B., 2002, Agile or rigorous OO methodologies – getting the best of both worlds, Cutter IT Journal, 15(1), 25-33

[He05] Henderson-Sellers, B., 2005, Creating a comprehensive agent-oriented methodology - using method engineering and the OPEN metamodel, Chapter 13 in Agent-Oriented Methodologies (eds. B. Henderson-Sellers and P. Giorgini), Idea Group, 368-397

[HH03] Henderson-Sellers, B. and Hutchison, J., 2003, Usage-Centered Design (UCD) and the OPEN Process Framework (OPF), Performance by Design. Procs. forUSE2003, Second International Conference on Usage-Centered Design (ed. L.L. Constantine), Ampersand Press, Rowley, MA, USA, 171-196

[HS00] Henderson-Sellers, B. and Serour. M.K., 2000, Creating a process for transitioning to object technology, Procs. Seventh Asia-Pacific Software Engineering Conference, APSEC2000, IEEE Computer Society Press, Los Alamitos, CA, USA, 436-440

[HS05] Henderson-Sellers, B. and Serour, M.K., 2005, Creating a dual agility method - the value of method engineering, J. Database Management, 16(4), 1-24

[He96] Henderson-Sellers, B. and Graham, I.M. with additional input from C. Atkinson, J. Bézivin, L.L. Constantine, R. Dué, R. Duke, D. Firesmith, G. Low, J. McKim, D. Mehandjiska-Stavrova, B. Meyer, J.J. Odell, M. Page-Jones, T. Reenskaug, B. Selic, A.J.H. Simons, P. Swatman and R. Winder, 1996, OPEN: toward method convergence? IEEE Computer, 29(4), 86-89

[HLH02a] Henderson-Sellers, B., Lowe, D. and B. Haire, 2002, OPEN process support for web development, Annals of Software Engineering, 13, 163-201

[HLH02b] Henderson-Sellers, B., Haire, B. and Lowe, D., 2002, Using OPEN's deontic matrices for e-business}, Engineering Information Systems in the Internet Context (eds. C. Rolland, S. Brinkkemper and M. Saeki), Kluwer Academic Publishers, Boston, USA, 9-30

[II95] ISO/IEC, 1995, Software Life Cycle Processes. ISO/IEC 12207: International Standards Organization / International Electrotechnical Commission.

[II98] ISO/IEC, 1998, TR15504 – Information Technology: Software Process Assessment, Technical Report, International Standards Organization / International Electrotechnical Commission.

[Ja98] Jarke, M., Pohl, K., Weidenhaupt, K., Lyytinen, K., Marttiin, P., Tolvanen, J.-P. and Papazoglou, M., 1998, Meta modelling: a formal basis for interoperability and adaptability, Chapter 9 in Information Systems Interoperability (eds. B. Krämer and H.-W. Schmidt), Research Studies Press Ltd./John Wiley & Sons Inc., 229-263

[Ja94] Jayaratna, N., 1994, Understanding and Evaluating Methodologies, NIMSAD: A Systemic Approach, McGraw-Hill

[KBS00] Kraiem, N., Bourguiba, I. and Selmi, S., 2000, Situational method for information system project, presented at SSGRR 2000, L'Aquila, Jul 31 - Aug 06 2000 (http://www.ssgrr.it/en/ssgrr2000/papers/283.pdf)

[KW92] Kumar, K. and Welke, R.J., 1992, Methodology Engineering: a Proposal for Situation-Specific Methodology Construction. In (Cotterman, W.W.; Senn, J.A. Eds.) Challenges and Strategies for Research in Systems Development. John Wiley & Sons: Chichester, UK, 257-269

[Ly87] Lyytinen, K., 1987, Different perspectives on information systems: problems and solutions, ACM Computer Surveys, 19(1), 5-46

[Ma91] Madhavji, N.H., 1991, The process cycle, Software Eng. J., 6(5), 234-242

[Od94] Odell, J.J., 1994, Power types. Journal of Object- Oriented Programming, 7(2), 8-12.

[Om01] OMG, 2001, OMG Unified Modelling Language Specification, version 1.4. OMG documents formal/01-09-68 through 80 (13 documents). http://www.omg.org, accessed 12th July 2002.

[Pa93] Paulk, M.C., Curtis, B., Chrissis, M.B. and Weber, C.V., 1993, The capability maturity model: Version 1.1, IEEE Software, 10(4), 18-27

[RR01] Ralyté, J. and Rolland, C., 2001, An approach for method engineering, Procs. 20th Int. Conf on Conceptual Modelling (ER2001), LNCS 2224, Springer-Verlag, Berlin, 471-484

[RDR03] Ralyté, J., Deneckère, R. and Rolland, C., 2003, Towards a generic method for situational method engineering, Advanced Information Systems Engineering. 15th International Conference, CAiSE 2003, Klagenfurt, Austria, June 16-18, 2003, Proceedings (ed. J. Eder and M. Missikoff), Springer-Verlag, LNCS 2681, 95-110

[RP96] Rolland, C. and Prakash, N., 1996, A proposal for context-specific method engineering. In (Brinkkemper, S.; Lyytinen, K.; Welke, R.J. Eds.) Method Engineering. Principles of Method Construction and Too Support. Procs. IFIP TC8, WG8.1/8.2 Working Conference on Method Engineering, 26-28 August 1996, Atlanta, USA, Chapman & Hall, London, 191-208.

[RPB99] Rolland, C., Prakash, N. and Benjamen, A., 1999, A multi-model view of process modelling, Requirements Eng. J., 4(4), 169-187

[RSM95] Rolland, C., Souveyet, C. and Moreno, M., 1995, An approach for defining ways-of-working, Information Systems, 20(4), 295-305

[Se03a] Seidewitz, E. 2003, What models mean, IEEE Software, 20(5), 26-31.

[Se03b] Serour, M.K., 2003, The Effect of Intra-Organisational Factors on the Organisational Transition to Object Technology, a PhD Thesis, University Technology, Sydney.

[SH04a] Serour, M.K. and Henderson-Sellers, B., 2004, Introducing agility: a case study of situational method engineering using the OPEN Process Framework, Procs. 28th Annual International Computer Software and Applications Conference. COMPSAC 2004, IEEE Computer Society Press, Los Alamitos, CA, USA, 50-5

[SH04b] Serour, M.K. and Henderson-Sellers, B., 2004, Empowering a software development team with a new methodology: a case study of e-government in Australia, Proceedings. IBIMA, Amman, Jordan, 4-6 July 2004

[SH04c] Serour, M.K. and Henderson-Sellers, B., 2004, Organizational aspects of transformation to e-business: a case study, Procs. IADIS International Conference e-Society 2004 (eds. P. Isaias, P. Kommers and M. McPherson), IADIS Press, Volume 2, 751-758

[SH04d] Serour, M.K. and Henderson-Sellers, B., 2004, OPEN for agility: an action research study of introducing method engineering into a government sector, Procs. 13th Int. Conf. on Information Systems Development. Advances in Theory, Practice and Education (eds. O. Vasilecas, A. Caplinskas,W. Wojtkowski, W.G. Wojtkowski, J. Zupancic and S. Wrycza), Vilnius Gediminas Technical University, Vilnius, Lithuania, 105-116

[Se02] Serour, M., Henderson-Sellers, B., Hughes, J., Winder, D. and Chow, L., 2002, Organizational transition to object technology: theory and practice, in Object-Oriented Information Systems (eds. Z. Bellahsène, D. Patel and C. Rolland), LNCS 2425, Springer-Verlag, Berlin, 229-241

[SA04] Standards Australia, 2004, Standard Metamodel for Software Development Methodologies, AS 4651-2004, Standards Australia, Sydney.

[HV97] ter Hofstede, A.H.M. and T.F. Verhoef, 1997, On the feasibility of situational method engineering. Information Systems. 22(6/7), 401-422

[TRL96] Tolvanen, J.-P., Rossi, M. and Liu, H., 1996, Method Engineering: current research directions and implications for future research, in Method Engineering. Principles of Method Construction and Too Support. Procs. IFIP TC8, WG8.1/8.2 Working Conference on Method Engineering, 26-28 August 1996, Atlanta, USA (eds. S. Brinkkemper, K. Lyytinen and R.J. Welke), Chapman & Hall, London, 296-317

[SB93] van Slooten, K. and Brinkkemper, S., 1993, A method engineering approach to information systems development, in Information Systems Development Process Procs. IFIP WG8.1, (eds. N. Prakash, C. Rolland and B. Pernici), Elsevier Science Publishers B.V., North-Holland

[SH96] van Slooten, K., Hodes, B., 1996, Characterizing IS development projects, in Procs. IFIP TC8 Working Conf. on Method Engineering: Principles of method construction and tool support (eds. S. Brinkkemper, K. Lyytinen, R. Welke) Chapman&Hall, Great Britain, 29-44

[VG94] Vessey, I. and Glass, R.L., 1994, Application-based methodologies: development by application domain, Information Systems Management, Fall 1994

[WK04] Wistrand, K. and Karlsson, F., 2004, Method components – rationale revealed, 2004, Advanced Information Systems Engineering 16th International Conference, CAiSE 2004, Riga, Latvia, June 7-11, 2004, Proceedings (eds. A. Persson and J. Stirna), LNCS 3084, Springer-Verlag, 189-201