A Method Assessment Framework

Tom Mcbride, Brian Henderson-Sellers

A Method Assessment Framework

Tom McBride and Brian Henderson-Sellers

School of Software, Faculty of Engineering and Information Technology,
University of Technology, Sydney, PO Box 123, Broadway, NSW 2007, Australia
{Tom.McBride, Brian.Henderson-Sellers}@uts.edu.au

Abstract. Situational method engineering is used to create methods for use on projects. It is vital that such constructed methods be of good quality and relevant to the software development project in hand. Current capability assessment approaches cannot readily be applied to such SME-constructed methods since they do not differentiate between the three "phases" of method construction and enactment: method design, method enactment and method performance. Here, we clearly differentiate the kind of quality assessment activities that need to be performed in these three different situations.

Keywords: constructed methods, method design, method enactment, method performance, quality assessment, situational method engineering

1 Introduction

Situational method engineering (SME) [1-3] is the software engineering discipline that describes the creation and use of a software development method(ology)[1] from small, atomic methodological pieces, known as "method fragments" or, at a larger scale (i.e. non-atomic), "method chunks" [4]. Each of these atomic pieces is typically conformant to the conceptual definitions in an underpinning metamodel [5, 6] and stored in a repository or methodbase (Figure 1). Fragments are selected from the repository in order that the final, constructed method (or "process model": Figure 2) will meet the many project contingencies [7, 8] in a way that an off-the-shelf method is not able to do. The constraints on fragment selection are at the heart of situational method engineering and are discussed by many authors (see, e.g. [9-11]). However, there has been almost no discussion of how one might assess the "quality" of the constructed method and little on the quality of individual fragments. A growing number of methods are currently being developed by various communities of interest to support without the aid of method engineers or method engineering[2]. We believe that such initiatives can be supported by a framework that separates the concerns of

[1] Method and methodology are taken to be synonyms for the purposes of this paper.

[2] Many methods are published process reference models in ISO standards. E.g. see ISO 20000-4, ISO 12207:2008, ISO 15288:2008.

different phases in the life cycle of a method, identifying what can reasonably be achieved at each phase. Such a framework can assist non-specialists to review their intended method as a quality check and to draw attention to different situational method engineering activities that might be necessary. The overall concern of the framework is to address the question of whether or not the method is suitable for its intended purpose in the circumstances for which it is intended.

Method quality could be defined as (a) whether the overall method is complete e.g. whether there are missing work products, work products but no way of creating them and, secondly, as (b) whether the method is actually what is needed for the particular industry application. This latter quality assessment has several aspects, one of the topics discussed in this paper. Firstly, it is possible to assess the quality of the "designed method" although many additional constraints appear when the process model (method) is enacted for a real project (Figure 2). Enactment here is taken to mean the instantiation of the process model for a particular situation or context. This will mean allocating team member names, budgets, time constraints etc. to the "slot values" of the various methodological elements. Finally, an additional quality concern relates to how well the enacted method performs in real time (the "performed process" in Fig. 2). Each of these three "phases" (method design, method enactment and method performance) are the focus of the discussion here.

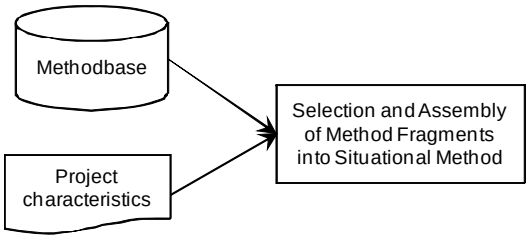

Fig. 1. Engineering a situational method from the elements in the methodbase taking into account the prevailing situation including project characteristics, overall context and other contingencies (after [2])

This paper will argue that the quality of a method can and should be assessed at the three phases in the life cycle of a method identified above: when the method is designed, when the method is enacted and when the method is being performed. We focus on the creation of a framework to identify "what" needs to be assessed and "when"; but do not attempt to answer the question regarding "how" this assessment will be undertaken in terms of what software engineering metrics might be useful in each phase.

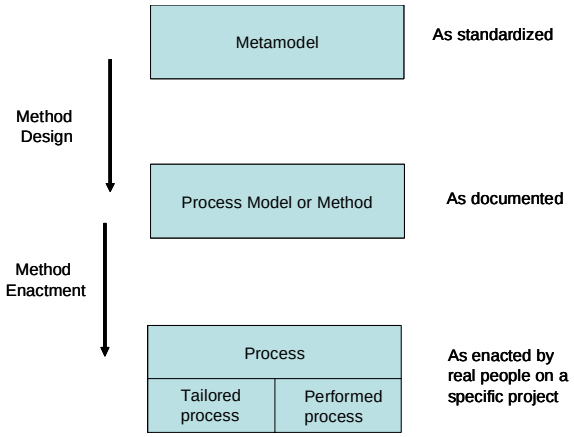

Fig. 2. The "layers" of process and method terminology revised by adding temporality (modified from [12])

Method design (Section 2) is intended to designate the time when a new method is created, in order to achieve a specific purpose but in a limited range of circumstances. For example, a method engineer might design a new method to develop life critical software using medium to large teams under tight schedule and budget constraints. Method design also includes the case of an organization adopting and possibly adapting a "standard method" for use on all of their projects. Method enactment (Section 3) designates when a method is tailored to the specific circumstances of a project. At that time, decisions can be made to address the project constraints and contingencies. Method performance (Section 4) designates the period over which the method is being used (calendar time) in order to carry out the project and actual performance can be evaluated. In Section 5, we outline the several advantages of this framework over traditional process assessment approaches, such as ISO/IEC 15504 [13] and CMMI [14]. We then, in Section 6, comment on other work (although limited) in this area of method assessment and then conclude in Section 7.

2 Method Design

The focus of this part of the assessment is on the quality of the method fragments/chunks as well as on how well they fit together in the designing of the method.

Software development methods are created to address a range of circumstances. Waterfall development [15], spiral development [16], agile [17] and its related methods of extreme programming [18], adaptive [19] and Scrum [20] are some that have been proposed to deal with particular circumstances or to achieve particular goals. These and others are not intended to be used without modification (although often perceived as being applicable "off-the-shelf") but are offered as a general method suitable as a base from which to tailor a project-specific method [21]. Even

though these methods are necessarily general, their usefulness and ability to achieve their primary purpose of developing software in the claimed range of circumstances needs to be assessed. To date, such assessment has been through peer review and commercial acceptance. But just how well any particular software development method is suited to a particular range of circumstances remains very much a matter of personal judgment of the assessor. This introduces a degree of subjectivity into the assessment (see further discussion below).

The earliest attempt to document a method is usually taken to be that of Royce [15] who set out his conclusions from his experiences in developing large software systems. Royce did not describe a software development method in any detail, simply the steps, their interaction, the problems arising within the steps and some ways of overcoming them. Royce addressed concerns arising from managing large software development projects in general. He did not address specific contingencies such as uncertainty or criticality of the system. JAD and RAD were two of the early attempts to reduce the amount of uncertainty through some initial development to test possible solutions (see e.g. [22] p473). Boehm [16] proposed the spiral method specifically to address project risk while retaining the general collection of software development activities and their inter-relationships. Since Boehm, different approaches have been proposed to address the problem of uncertainty of problem understanding and uncertainty of solution construction. Several software development methods that made use of, and depended on, experienced developers became labelled as "agile" in reference to their claims of being able to respond quickly to the growing understanding of the problem and its implemented solution as all stakeholders learned more about both. Indeed, it is well known (but poorly documented or researched) that organizations adopt and refine their chosen method in some way without making the method specific to a particular project. Assessment of these, and other, software development methods has been largely through commercial acceptance and not through any theoretical or framework based assessment.

In general, some assessment of the method is indeed possible in order to determine if it can address its known or implied range of contingencies. While there are limits to the assessment, it is possible to determine whether or not the method has the means to set and manage performance, to coordinate the work and to develop, deploy and possibly maintain the software, system or service.

The primary purpose of a method is to achieve some purpose, in this case to develop and, possibly, deploy and maintain software, a system or a service. Thus, for assessment purposes the method needs to contain the processes or activities that are capable of achieving the primary purpose. The activities to achieve the primary purpose seem to have been the focus of many proposed situational method engineering methods [1, 23-25].

Processes can, of course, be examined to determine if they contain the expected activities [26]. However, a method is not a random collection of processes nor does it consist only of the processes that support the primary activities. The interaction between processes required to support project management or other forms of coordination can also be examined, as can the process interactions necessary for performance management. For the purposes of this paper, performance management will include those activities involved in setting performance expectations and

constraints such as delivery dates, budget and project scope, then managing those expectations and constraints throughout the life of the project.

The software development work must be coordinated, so the method must contain the means to achieve or impose coordination. Coordination is usually achieved through developing a work breakdown structure, a schedule for its completion and re-integration, plans of how the work is to be completed, standards of how the work is to be done and standards relating to what work needs to be produced [27-29]. Since software development is inherently uncertain, there is usually a need for dynamic coordination, achieved through meetings, reviews and other exchanges. As with performance, the method needs to provide for coordination and, furthermore, it needs should be adequate for the range of circumstances encountered.

3 Method Enactment

The enactment assessment focusses on the SME-*constructed* process but may identify the need to replace or augment the method fragments. The artifact under assessment is the enacted method.

When projects are being planned, it is with considerable knowledge of the expected project contingencies and constraints. Some constraints are fixed, like the delivery date of software required for a specific event. Some are negotiable, like the number of people in the development team or the actual scope of the project. While such constraints are normally considered as part of project planning, they also contribute to method tailoring during that planning. Method enactment describes the association of parameters in the designed method to actual project-specific resources, such as actual team members, real deadlines, available funding etc. and the subsequent method tailoring.

There is a considerable body of information on contingency theory relating to software development of which some is discussed in the paragraphs that follow. However, there is little consensus on the contingencies of importance in different circumstances. Additionally, other fields such as product development [30] have the potential to open discussion on method enactment to a much wider treatment than the current concerns of software project planning. For these reasons, this paper will not attempt a rigorous examination of specific contingency factors that may impact software development projects.

Project contingencies such as the size of the project are generally recognized as influencing the choice of method [31, 32]. Because larger projects generally involve more people, coordination tends to be more mechanistic [28, 33] than the more organic methods typical of smaller projects. Expressed differently, larger projects tend to use plans, standards and formal exchanges to coordinate their work whilst smaller projects tend to use stand-up meetings or co-location.

Many software development methods do not yet address the effect of a distributed development team. An exception are some of the practices incorporated into the Crystal family of communicating between team members. These are significant in agile development methods [17] where the problems of communicating between team members is acknowledged and different techniques are proposed to overcome barriers

to communication. Other less agile development methods seem to be unaffected by team distribution implying that the projects concerned were already using techniques that were less affected by distance [34]. It is also possible that some methods are more susceptible to communication barriers than others and that compensation is sometimes possible.

Various authors have identified sources of uncertainty including platform and market uncertainty [35], requirements uncertainty [28], outcome uncertainty and task programmability [36] as significantly influencing choices of method and its tailoring. While specific relationships between different types of uncertainty and method may need some investigation, the general connection seems to be well accepted.

Safety critical or security critical applications may demand extra processes (in the ISO sense i.e. a second meaning to the one depicted in Figure 2), intended to augment an otherwise less critical method [37, 38]. In this case, the contingency is addressed by extra processes and activities rather than a selection among equivalent activities in a process already included in the method. A project with stringent quality requirements will need to meet those requirements through more rigorous verification activities which may, in turn, affect the selection of personnel on the verification team and the distribution of other tasks. Method enactment involves more than mere adjustment to parts of the method. It may require wholesale redistribution of activities within the method.

Although the project management literature does not specifically identify such project planning activities as method tailoring, it is achieved nonetheless. For example, a project manager may be faced with a demand to outsource some of the development, leading to a question of where verification of the outsourced development is to be performed. Verification of the work could be performed by the developers, if they were known and trusted, or by the acquirer after delivery. Such decisions will be reflected in the method as different ways in which activities are grouped together to form processes and the allocation of those processes to organizations.

The main contingencies of software development projects, described above, are those that are normally considered when a project is planned. In the parlance of method engineering, the method is configured. Method configuration through the selection of specific activities suited to the circumstances has been discussed in [9, 10].

In contrast to assessment at method design, method assessment at enactment has specific information about the project constraints and contingencies so can be expected to determine the utility of the method with greater certainty than previously possible. Assessment at enactment is unlikely to be significantly different from assessment at design but has greater immediacy. An assessment would be expected to determine whether or not the proposed means of monitoring and managing performance is likely to work in the specific circumstances. For example, if the project is large and distributed, oversight of the distributed organizations cannot rely on the same oversight processes as would be employed if all parties belonged to the same organization. Similarly, coordination processes at the low range of project size, where co-location may be possible, would be different from coordination processes needed at the high end of project size.

A method assessment at enactment would help avoid a tendency to use a familiar but possibly inappropriate method. It would help direct attention to parts of the method affected by the constraints and contingencies and help remove subject judgement about whether or not the method "feels" right.

4 Method Performance

The whole point about tailoring a method for specific constraints and contingencies is to achieve the best possible outcome in practice. A tailored method represents a hypothesis that the best outcome possible will be achieved under the specific constraints and contingencies. Like all hypotheses, it should be possible to gather evidence to prove or disprove it. Yet so far there seems to have been little attempt to do so. An assessment should be able to identify two distinct issues: the right method but the wrong performance and the wrong method with the right performance. An assumption that the first case is, by default, true seems to dominate existing process assessment methods [39, 40] and also seems to underlie quality management methods [41]. However, the argument of situational method engineering is that the method may possibly be unsuited to the circumstances leading to poor outcomes no matter how the method is performed.

Performance is usually negotiated during project planning as the scope of the project, the available personnel, quality and other requirements, budget and delivery milestones are considered and resolved [42-44]. Performance monitoring and management would normally be achieved through some unspecified means, e.g. project review meetings, to inform the project of changes in the constraints and a means to collect and report performance data. A method assessment would need to examine whether these activities were present and adequate for the expected range of circumstances.

Available process assessment methods such as SPICE [13] and CMMI [14] are assessments of process capability, not of performance. Moreover, they assess processes and not the overall performance of the method. Process capability attempts to measure the degree to which a given process is likely to achieve its stated purpose [45]. By itself, that says nothing about how well suited the process is to the circumstances. That determination relies on the knowledge and experience of the process assessor to decide whether the process itself is flawed and needs improvement or whether the process is being incorrectly performed for one reason or another. Other forms of assessment, such as a process audit, generally rely on the judgement of the auditor. This dependence on the well-intended judgement of an experienced auditor or assessor is unsatisfactory for a number of reasons. The first is that the assessor may not be sufficiently knowledgeable about software development or the particular circumstances, leading to well intended but harmful findings. The second is that such assessors tend to be expensive, out of reach for any but large software developers with project budgets able to absorb the high cost of a rigorous assessment.

Informal assessments are done all the time. People learn and adjust what they do. For example, Scrum practice includes a retrospective at the end of each sprint [20, 46] and most methods have some sort of post-mortem or other form of audit; the need for

continual improvement is built into ISO 9001. However, these general imperatives to improve don't provide guidance on what to look for or how to recognize the need for improvements.

During method performance, information is available about how well the method is achieving its intended purpose. It is necessary to compare actual performance to expected performance in order to determine where changes to the method (e.g. in the case of the wrong process performed correctly) are required or where changes to method performance (the right process performed incorrectly) are needed.

However, it is apparent that most of the method attributes of method performance, the quality of both the fragments and the constructed method are evaluated in real time. Feedback may well suggest a dynamic replacement or the addition of new method fragments in order to ensure that the project is successfully completed. Of especial interest are compound attributes for which there are no direct measures. Nor do there seem to be generally agreed-upon models of effectiveness, efficiency, coordination or governance. Like project success, it may be difficult to say what is required to achieve it, but relatively easy to determine if it is not being achieved. Rather than try to show that these are present and being achieved, it should be possible to detect their absence through errors or other symptoms of failure.

Method performance assessment would prompt a review of the project constraints and contingencies. Conversely, a change in project contingencies or constraints may prompt a method performance assessment. Additionally, a method performance assessment would provide an additional means to detect and justify changes to the method or to the manner of performing the method. It is not proposed that method performance assessment is equivalent to or should replace other, more rigorous, process assessments. The proposed method performance assessment is intended to directly and objectively assess the appropriateness of the method rather than imply it through the subjective knowledge and experience of a process assessor.

During assessment of method performance, the quality of both the fragments and the constructed method are evaluated in real time. Feedback may well suggest a dynamic replacement or the addition of new method fragments in order to ensure that the project is successfully completed.

5 Advantages of this Framework

This proposed framework identifies method design, method enactment and method performance separating the concerns of each phase and identifying what can be reasonably achieved at each phase. The proposed framework also links the typical activities of projects, particularly software development projects, to necessary changes in methods, drawing attention to different situational method engineering activities that might be necessary.

The framework provides guidance about what could be assessed at each phase as well as the desirability of doing so. This may assist method engineers to assess their methods more rigorously than through the more basic and more subjective means of expert opinion – as is currently the case. Method assessment techniques may be developed that can assist non-specialists review their intended method or method

performance without needing to resort to expensive audits or formal process assessments. Given that most of the world's software developers claim that they do not use a formal software development method [47, 48] and certainly don't review or assess what method they do use, a more readily usable method assessment technique would seem to offer some advantages.

The proposed framework provides a guide to development of assessment techniques and tools. Rather than try to develop a general assessment technique that attempts to require certainty where none is possible (method design) or fails to require rigor where some is possible (method enactment), tools can be developed to assess methods appropriately and as rigorously as possible, but no more. Similarly, existing assessment techniques can be positioned in relation to the type of method assessment they accomplish.

5 Discussion and Related Work

There has been very little direct research on this topic, However, Perez et al. [49] discuss congruence evaluation, arguing that this is the most important measure to be assessed. Congruence is proposed as a measure of how well the process model fits the intended usage/domain. They suggest the use of a contingency model as a precursor to defining the relationships between the process model and its intended usage context. They introduce three variables: a dependent variable (the effectiveness of the process model), an independent variable (a characteristic of the process model) and a contingency variable (either a characteristic of the context or the process model). They recommend that attribute values should first be assigned, often subjectively by someone familiar with the situation, and then a congruence measure calculated based on the inter-relationships established between the process model and the attributes of the process context, the derived congruence index being a real number in the closed interval (-1,1). Here, 1 indicates a perfect match and -1 the worst possible match. Low congruence values can thus be identified as suggesting that improvement is needed. A similar, contingency approach is also advocated by [50] (based on work of [51]). Using a banking example, these authors recommend the following list of contingency factors:

- Management commitment,
- Importance,
- Impact,
- Resistance and conflict,
- Time pressure,
- Shortage of human resources,
- Shortage of means,
- Formality,
- Knowledge and experience,
- Skills,
- Size,
- Relationships,
- Dependency,

- Clarity,
- Stability,
- Complexity,
- Level of innovation.

Adaptation of processes is also discussed by [52] in the context of agile methodologies.

In a much broader software development context, Kherrai et al. [53] discuss the need to use a set of quality attributes from ISO 9126 [54] stressing the need to do this for the strategy and tactics of the software project plan. A different focus is taken by [55] in which the concerns of governance, risk and compliance of the to-be-constructed method are highlighted.

Although not discussing assessment per se, Niknafs and Asadi [56], in the context of CAME (computer-aided method engineering), do split the process into four stages: enactment, elicitation, evolution and evaluation and focus on static descriptors (that they call aspects) rather than a time-based characterization as we discuss here. Key notions for method adaptation are examined in [57] although, once again, there is no discussion regarding how this might be incorporated into process assessment.

6 Conclusion

We have proposed here a framework to enable assessment of methods at different times in their life cycle. The purpose of the framework is to enable method engineers to assess the congruence of a developed, enacted or performed method according to the claimed range of known constraints and contingencies. In describing the proposed framework, we have shown that there is much to be gained from separating method assessment into the different types so that the method can be assessed appropriately and useful information provided to stakeholders. The framework and assessment techniques and metrics that may be developed in the future to support it offers the possibility that tailoring methods appropriately might be within the reach of more and smaller organizations.

The framework highlights some fruitful areas for research. In particular, it underlines the need to better link actual software development practices and their tailoring to situational method engineering. The need to coordinate work cuts across processes, as does process performance monitoring and management, and their interaction with the primary production processes could be explored. Measures of method attributes and techniques of method assessment are also identified by the framework. Research arising from the framework promises to be highly relevant to industry.

Acknowledgements

This is paper number 10/10 of the Centre for Object Technology Applications and Research within the Centre for Human Centred Technology Design of the University of Technology, Sydney.

References

1. Brinkkemper, S.: Method engineering: engineering of information systems development methods and tools. Information and Software Technology 38(4), 275-280 (1996)
2. Henderson-Sellers, B., Ralyte, J.: Situational Method Engineering: A state of the art review. Journal of Universal Computer Science 16(3), (2010)
3. Kumar, K., Welke, R.J.: Methodology Engineering: a proposal for situation-specific methodology construction. Challenges and strategies for research in systems development, pp. 257-269. John Wiley& Sons, Inc. (1992)
4. Ågerfalk, P., Brinkkemper, S., Gonzalez-Perez, C., Henderson-Sellers, B., Karlsson, F., Kelly, S., Ralyté, J.: Modularization Constructs in Method Engineering: Towards Common Ground? In: Ralyté, J., Brinkkemper, S., Henderson-Sellers, B. (eds.) Situational Method Engineering: Fundamentals and Experiences, vol. 244, pp. 359-368. Springer Boston (2007)
5. Gonzalez-Perez, C., Henderson-Sellers, B.: Metamodelling for Software Engineering. Wiley Publishing, Chichester, UK (2008)
6. Henderson-Sellers, B.: Method engineering for OO systems development. vol. 46, pp. 73-78. ACM (2003)
7. Karlsson, F., Ågerfalk, P.J.: Method configuration: adapting to situational characteristics while creating reusable assets. Information and Software Technology 46(9), 619-633 (2004)
8. Kornyshova, E., Deneckère, R., Salinesi, C.: Method Chunks Selection by Multicriteria Techniques: an Extension of the Assembly-based Approach. In: Ralyté, J., Brinkkemper, S., Henderson-Sellers, B. (eds.) Situational Method Engineering: Fundamentals and Experiences, vol. 244, pp. 64-78. Springer Boston (2007)
9. Henderson-Sellers, B., Nguyen, V.P.: Un outil d'aide à l'ingénierie de méthodes reposant sur l'approche OPEN. Génie Logiciel (70), 12 (2004)
10. Ralyté, J., Rolland, C.: An Approach for Method Reengineering. In: S.Kunii, H., Jajodia, S., Sølvberg, A. (eds.) Conceptual Modeling — ER 2001, vol. 2224, pp. 471-484. Springer Berlin / Heidelberg (2001)
11. DeLoach, S., Garcia-Ojeda: O-MaSE: An Customizable Approach to Designing and Building Complex, Adaptive Multiagent Systems. International Journal of Agent-Oriented Software Engineering 4(3), (In Press)
12. Henderson-Sellers, B.: Method Engineering: Theory and Practice. In: Information Systems Technology and Its Applications. 5th International Conference ISTA pp. 84. Gesellschaft Für Informatik, (2006)
13. ISO/IEC 15504-1:2004 - Information Technology - Process Assessment - Part 1: Concepts and Vocabulary
14. SEI. CMMI® for Development, Version 1.2. CMU/SEI-2006-TR-008. (2006)
15. Royce, W.W.: Managing the development of large software systems: concepts and techniques. Proceedings of IEEE WESCON. IEEE Computer Society Press, Monterey (1970)
16. Boehm, B.W.: A spiral model of software development and enhancement. Computer 21(5), 61-72 (1988)

17. Cockburn, A.: Agile Software Development: The Cooperative Game (2nd Edition). Addison-Wesley Professional (2006)
18. Beck, K.: Extreme Programming Explained. Addison-Wesley, Boston (2000)
19. Highsmith, J., Cockburn, A.: Agile software development: the business of innovation. Computer 34(9), 120-127 (2001)
20. Sutherland, J.: Scrum software development process. In: OOPSLA. (1995)
21. Bajec, M., Vavpotic, D., Krisper, M.: Practice-driven approach for creating project-specific software development methods. Information and Software Technology 49(4), 345-365 (2007)
22. Graham, I.: Object-Oriented Methods: Principles and Practice (3rd Edition). Addison-Wesley Professional (2000)
23. Rolland, C.: Method engineering: Towards Methods as Services. Software Process: Improvement and Practice 14, 143-164 (2009)
24. Ralyté, J., Rolland, C.: An Assembly Process Model for Method Engineering. In: Dittrich, K., Geppert, A., Norrie, M. (eds.) Advanced Information Systems Engineering, vol. 2068, pp. 267-283. Springer Berlin / Heidelberg (2001)
25. Ralyté, J., Deneckère, R., Rolland, C.: Towards a Generic Model for Situational Method Engineering. In: Goos, G., Hartmanis, J., Leeuwen, J.v. (eds.) Advanced Information Systems Engineering, vol. 2681, pp. 1029-1029. Springer Berlin / Heidelberg (2003)
26. Goldkuhl, G., Lind, M.: Coordination and transformation in business processes: Towards an integrated view. Business Process Management Journal (14), 761-777 (2008)
27. Kraut, R.E., Streeter, L.A.: Coordination in software development. Communications of the ACM 38(3), 69-81 (1995)
28. Nidumolu, S.R.: A comparison of the structural contingency and risk-based perspectives on coordination in software development projects. Journal of Management Information Systems 13(2), 77-113 (1996)
29. McBride, T.: The mechanisms of project management of software development. Journal of Systems and Software 81(12), 2386-2395 (2008)
30. Reinertsen, D.G.: The Principles of Product Development Flow: Second Generation Lean Product Development. Celeritas Publishing, Redondo Beach (2009)
31. Cockburn, A. http://alistair.cockburn.us/crystal/crystal.html (2004)
32. Elssamadisy, A., Schalliol, G.: Recognizing and responding to "bad smells" in extreme programming. In: 24rd International Conference on Software Engineering, pp. 617-622. (2002)
33. Andres, H.P., Zmud, R.W.: A Contingency Approach to Software Project Coordination. Journal of Management Information Systems 18(3), 41-70 (2002)
34. McBride, T., Henderson-Sellers, B., Zowghi, D.: Managing outsourced software development: Does organisational distance demand different project management? In: UKAIS 2006. (2006)
35. MacCormack, A., Verganti, R.: Managing the Sources of Uncertainty: Matching Process and Context in Software Development. Journal of Product Innovation Management 20(3), 217-232 (2003)
36. Eisenhardt, K.M.: Agency Theory: An Assessment and Review. Academy of Management Review 14(1), 57-74 (1989)
37. Davis, N. Secure Software Development Life Cycle Processes: A Technology Scouting Report. CMU/SEI-2005-TN-024. (2005)
38. ISO/IEC 15504-10:2010 - Information technology — Process assessment — Part 10: Safety extension
39. SEI. Standard CMMI Appraisal Method for Process Improvement (SCAMPI). CMU/SEI-2001-HB-001. (2001)
40. ISO/IEC 15504-3:2004 - Information technology - Process assessment - Part 3: Guidance on performing an assessment

41.ISO/IEC 9001:2000 - Quality Management Systems - Requirements
42.Burke, R.: Project Management: Planning and Control Techniques. Burke Publishing, Tokai, South Africa (2003)
43.Cleland, D.I., Ireland, L.R.: Project management: Strategic Design and Implementation. McGraw-Hill (2002)
44.Hughes, B., Cotterell, M.: Software Project Management. McGraw-Hill, London (1999)
45.ISO/IEC 15504-2:2004 - Information technology - Software process assessment - A reference model for processes and process capability
46.Rising, L., Janoff, N.S.: The Scrum software development process for small teams. Software, IEEE 17(4), 26-32 (2000)
47.Fitzgerald, B.: The use of systems development methodologies in practice: a field study. Information Systems Journal 7(3), 201-212 (1997)
48.SEI. Process Maturity Profile - CMMI 2005 Year-end Update. (2005)
49.Pérez, G., El Emam, K., Madhavji, N.: Customising software process models. In: Schäfer, W. (ed.) Software Process Technology, vol. 913, pp. 70-78. Springer Berlin / Heidelberg (1995)
50.van Slooten, K., Hodes, B.: In Proceedings of IFIP TC8 Working Conf. on Method Engineering. In: Proceedings of IFIP TC8 Working Conference on Method Engineering: Principles of Method Construction and Tool Support, pp. 29-44. Chapman & Hall, (1996)
51.van de Hoef, R., Harmsen, A.F., Wijers, G.M. Situatie, Scenario & Succes. (1995)
52.Henninger, S., Ivaturi, A., Nuli, K., Thirunavukkaras, A.: Supporting Adaptable Methodologies to Meet Evolving Project Needs. In: Wells, D., Williams, L. (eds.) Extreme Programming and Agile Methods — XP/Agile Universe 2002, vol. 2418, pp. 15-51. Springer Berlin / Heidelberg (2002)
53.Kherraf, S., Cheikhi, L., Abran, A., Suryn, W., Lefebvre, E.: The Need to Evaluate Strategy and Tactics before the Software Development Process Begins. Journal of Software Engineering and Applications 3(7), 644 (2010)
54.ISO/IEC 9126:2001 - Software engineering - Product quality - Part 1: Quality model
55.Gericke, A., Fill, H.-G., Karagiannis, D., Winter, R.: Situational method engineering for governance, risk and compliance information systems. Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology. ACM, Philadelphia, Pennsylvania (2009)
56.Niknafs, A., Asadi, M.: Towards a Process Modeling Language for Method Engineering Support. In: Computer Science and Information Engineering, 2009 WRI World Congress on, pp. 674-681. (2009)
57.Aydin, M.: Examining Key Notions for Method Adaptation. In: Ralyté, J., Brinkkemper, S., Henderson-Sellers, B. (eds.) Situational Method Engineering: Fundamentals and Experiences, vol. 244, pp. 49-63. Springer Boston (2007)