Methods as Action Knowledge: Exploring the Concept of Method Rationale in Method Construction, Tailoring and Use

Pär J Ågerfalk and Brian Fitzgerald

Dept of Computer Science and Information Systems
University of Limerick, Limerick, Ireland
{par.agerfalk, bf}@ul.ie

Abstract. Systems development methods are used to express and communicate knowledge about systems and software development processes; i.e. methods encapsulate knowledge. Since methods encapsulate knowledge, they also encapsulate rationale. Rationale can in this context be understood as the reasons and arguments for particular method prescriptions. In this paper we show how the combination of two different aspects of method rationale can be used to shed some light on the communication and apprehension of methods in systems development. This is done by way of clarifying how method rationale is present at three different levels of method existence. By mapping existing research on methods onto this model, we conclude the paper by pointing at some research areas that deserve attention and where method rationale could be used as an important analytic tool.

1 Introduction

Systems development methods are used in systems development as a means to express and communicate knowledge about the systems/software development process. The idea is that methods encapsulate knowledge of good design practice so that developers, presumably less experienced than the method developer, can be more effective, efficient and confident in their work. Despite this, it is a well known fact that many software organizations do not use methods at all [1, 2], and when methods are used they are not used literally, 'out of the box', but adapted to suit the particular development situation [3]. This tension between the method 'as documented' (or as inter-subjectively agreed upon) and the method 'in use' has been described as a 'method usage tension' between 'method-in-concept' and 'method-in-action' [4]. This tension has given rise to an array of different approaches, ranging from contingency factor driven method engineering [5] through method tailoring and configuration [3, 6, 7] to the various agile methods, such as XP [8] and SCRUM [9].

A basic condition for a method to be accepted and used is that method users perceive it to be useful in their development practice [10]. For someone to regard a piece of knowledge as valid and useful the knowledge must be possible to rationalize, i.e. the person needs to be able to make sense of it and incorporate it into their view of

the world. Ethnomethodologists refer to this property of human behaviour as 'accountability' [11-13]; people require an account of the truth or usefulness of something in order to accept it as valid[1]. This is probably particularly true in the case of method prescriptions since method users are supposed to use these as a basis for future actions, and thus use the method description as a partial account of their own actions. Hence, we use the term 'action knowledge' to refer to the type of knowledge that is codified as method descriptions.

In order to understand better the rationalization of system development methods, the concept of method rationale has been suggested [14-17]. Method rationale concerns the reasons and arguments behind method prescriptions, and why method users (e.g. systems developers) choose to follow or adapt a method in a particular way. This argumentative dimension is an important but often neglected aspect of systems development methods [15-17]. One way of approaching method rationale is to think of it as an instance of 'design rationale' [18] that concerns the design of methods, rather than the design of computer systems [17]. This aspect of method rationale captures how a method evolved and what options were considered during the design process, together with the argumentation leading to the final design [17], and provides insights into the process dimension of method development. A complementary view on method rationale is based on the notion of purposeful-rational action. This aspect of method rationale focuses the underlying goals and values that make people chose options rationally [15, 16] and provides an understanding of the overarching conceptual structure of a method's underlying philosophy.

In this paper we show how the combination of these two aspects of method rationale can be used to shed some light on the communication, apprehension and rationalization of methods in software and systems development. This will be done by clarifying how method rationale is present at three different levels of method existence. By mapping existing research on methods onto this three-level model, we conclude the paper by pointing at some areas that deserve attention and where method rationale could be an important analytic tool.

The paper proceeds as follows. Section 2 elaborates the concept of action knowledge and how methods represent an important instance of such knowledge. Section 3 looks at how methods as action knowledge exist at different levels of abstraction in systems/software development. It also relates these levels to the corresponding actor roles taking part in the communication, interpretation, and refinement of this knowledge. Sections 4 and 5 elaborate the concept of method rationale as a way of representing the rationality dimension of methods as action knowledge. Sections 6 and 7 reflect upon the existing research in systems/software development methodology and discuss how method rationale can be used as a tool in creating a more integrated understanding of methods, method configuration/tailoring and agile development practices.

[1] According to ethnomethodologist Harold Garfinkel, actions that are accountable are 'visibly-rational-and-reportable-for-all-practical-purposes' [11].

2 Methods as Action Knowledge

When we think of software and systems development methods, what usually spring to mind are descriptions of ideal typical software processes. Such descriptions are used by developers in practical situations to form what can be referred to as methods-in-action [19]. A method description is a linguistic entity and an instance of what can be referred to as action knowledge [20, 21]. The term 'action knowledge' refers to theories, strategies and methods that govern people's action in social practices [20]. The method description is a result of a social action[2] performed by the method creator directed towards intended users of the method. A method description should thus be understood as a suggestion by the method creator for how to perform a particular development task. This 'message' is received and interpreted by the method user, and acted upon by following or not following the suggestion (see Fig. 1); i.e. by transforming the method description (or 'formalized method' [19] or 'method-in-concept' [4]) into a method-in-action. The 'method as message' is formulated based on the method creator's understanding of the development domain and on their fundamental values and beliefs. Similarly, the interpretation of a method by a method user is based on their understanding, beliefs and values.

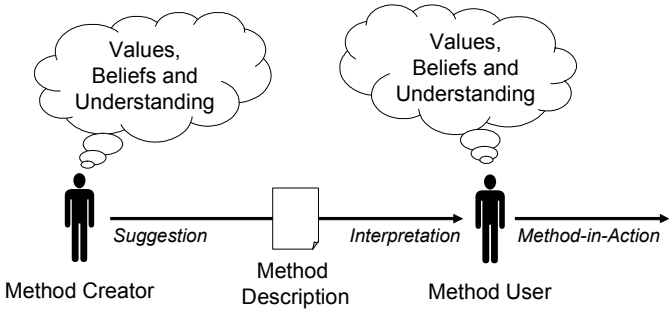

Fig. 1. Method descriptions in a communication context.

It is possible to distinguish between five different aspects of action knowledge: a *subjective*, an *inter-subjective*, a *linguistic*, an *action* and a *consequence* [20, 21]. Subjective knowledge is part of a human's 'subjective world' and is related to the notion of 'tacit knowledge' [23]. This would correspond to the two 'clouds' in Fig. 1. This would also correspond to someone's personal interpretation and understanding of a method. Inter-subjective knowledge is 'shared' by several people in the sense that they attach the same meaning to it. This could imply that some of the elements of the 'clouds' in Fig. 1 are agreed upon by the communicator (method creator) and interpreter (method user), and that they thus attach the same meaning to, at least parts of, a particular method. Linguistic knowledge is expressed as communicative signs,

[2] According to sociologist Max Weber, a social action is that human behaviour to which the actor attaches meaning and which takes into account the behaviour of others, and thereby is oriented in its course [22].

for example, as the written method description in Fig. 1. As the name suggests, action knowledge is expressed, or manifested, in action. This is the action aspect of knowledge, or 'method-in-action'. Finally, traces of the action knowledge might be found in materialized artefacts, which constitute a consequence aspect of the knowledge. This would correspond to, for example, produced models and documentation as well as the actual software.

3 Abstraction Levels of Methods

As stated above, it is a well-known fact that a method-in-action usually deviates significantly from the ideal typical process described in method handbooks and manuals [1-3]. Such adaptations of methods can be made more or less explicit and be based on more or less well-grounded decisions.

Methods need to be adapted to particular development situations since a method, as described in a method handbook, is a general description of an ideal process. Such an ideal type[3] needs to be aligned with a number of situation-specific characteristics or 'contingency factors' [5, 7]. The process of adapting a method to suit a particular development situation has been referred to as *method configuration*[4] [7]. Method configuration can be understood as a particular form of situational method engineering taking one specific method as a base for configuration. This is in contrast to most method engineering approaches, which assume that a situational method is to be arrived at by assembling a (usually quite large) number of 'atomic' method fragments [24, 25]. This latter form of method engineering allows for construction of situational methods based on a coherent integration of fragments from different methods. In many situations, a more relevant question to ask is: 'what parts of the method can be omitted?' [3], bearing in mind that omitting a particular part of a method may lead to undesired consequences later in the process. A typical example would be if a particular artefact is not produced when it is needed to proceed successfully with a subsequent activity.

When a situational method has been 'configured' or 'engineered' and is used by developers in a practical situation, it is likely that different developers disagree with the method description and adapt the method further to suit their particular hands-on situational needs (as indicated above, it is actually impossible that a method-following action is identical to the action prescribed and linguistically expressed by the method – they represent different aspects of the same knowledge). As a consequence, the method-in-action will deviate not only from the ideal typical method but also from the situational method.

---

[3] Max Weber introduced the notion of an 'ideal type' as an analytic abstraction. Ideal types do not exist as such in real life, but are created to facilitate discussion. We use the term here to emphasize that a formalized method, expressed in a method description, never exists as such as a method-in-action. Rather, the method-in-action is derived from an ideal typical formalized method. At the same time, a formalized method is usually an ideal type created as an abstraction of existing 'good practice' [15].

[4] Process configuration [6] and method tailoring [3] are other terms used to describe this.

Altogether this gives us three 'abstraction levels' of method: (a) the ideal typical method that abstracts details and addresses a generic problem space, (b) the situational method that takes project specifics into account and thus addresses a more concrete problem space, and (c) the method-in-action, which is the 'physical' manifestation of developers' actual behaviour 'following' the method in a concrete situation. It follows from this that both the ideal typical method (a) and the situational method (b) exist as linguistic expressions of knowledge about the software development process. On the contrary, the method-in-action represents an action aspect of that knowledge, which may of course be reconstructed and documented post facto (in addition to the way it is manifested in different developed artefacts along the way).

Fig. 2 depicts these three abstraction levels of method and corresponding actions and communication between the actors involved. In Fig. 2, the Method User of Fig. 1 has been specialized into the Method Configurator (or process engineer) and the Developer. Method configurators use the externalized knowledge expressed by the method creator in the ideal typical method as one basis for method configuration and subsequently communicates a situational method to developers. What is not shown in Fig. 2 is that method construction, method configuration and method-in-action rely on the actors' interpretation of and assumptions about the development context. The developer 'lives' directly with this context and thus focuses their tailoring efforts on a specific problem space. The method creator, on the other hand, has to rely on an abstraction of an assumed development context and thus focuses on a generic problem space. Finally the method configurator supposedly has some interaction with the actual development context, which provides a more concrete basis for configuring a situational method.

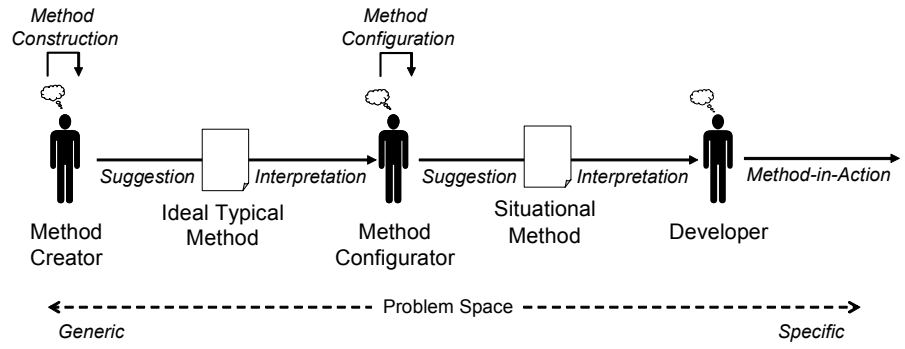

Fig. 2. Levels of method abstraction in methods as action knowledge.

In both method construction and method configuration, the method communicated is a result of social action aimed towards other actors as a basis for their subsequent actions. This means that method adaptation, both in construction and in-action, relies on the values, beliefs and understanding of the different actors involved – and this is where method rationale comes into play.

4 The Concept of Method Rationale

Since methods represent knowledge they also represent rationale. Therefore, a method user 'inherits' both the knowledge expressed by the method and the rationale of the method constructor [15]. It can be argued that regardless of the grounds, method tailoring (both during configuration and in-action) are rational from the point-of-view of the method user [26]; they are based on some sort of argument for whether to follow, adapt or omit a certain method, or part thereof. Such adaptations are driven by the possibility of reaching 'rationality resonance' between the method constructor and method user [27]. That is, they are based on method users' efforts to understand and ultimately internalize the rationale expressed by a method description.

From a process perspective, method rationale can be though of as having to do with the choices one makes in a process of design [17]. Thus, we can capture this kind of method rationale by paying attention to the questions or problematic situations that arise during method construction. For each question, we may find one or more options, i.e. 'solutions' to that question.

As an example, consider the construction of a method for analysing business processes. In order to graphically represent flows of activities in business processes, we may consider the option of modelling flows as links between activities, as in UML Activity Diagrams [28]. Another option would be to use a modelling language that allows for explicitly showing results of each action and how those results are used as a basis for subsequent actions, as in VIBA[5] Action Diagrams [29]. To help us arguing for and against each option we may specify a number of criteria as guiding principles. Then, for each of the options, we can asses whether it contributes positively or negatively with respect to each criterion. Let us, for example, assume that one criterion (a) is that we want to create a visual modelling language (notation) with as few elements as possible in order to simplify models (a minimalist language). Another criterion (b) might be that we want a process model that is explicit on the difference between material actions and communicative actions[6] in order to focus developers' attention on social aspects and material/instrumental aspects respectively (thus a more expressive language). Finally, a third criterion (c) might be that we would favour a well-known modelling formalism. The UML Activity Diagram option would have a positive impact on criteria a and c, and a negative impact on criterion b, while the VIBA Activity Diagram option would have a positive impact on criterion b, and a negative impact on criteria a and c. Thus, given that we do not regard any of the criteria more important than any other, we would likely choose the UML Activity Diagram option.

Fig. 3 depicts this notion of method rationale as based on explicating the choices made throughout method construction. The specific example shown is the choice between VIBA Action Digram versus UML Activity Diagram. This model of method rationale is explicitly based on the Question, Option, Criteria Model of Design Space

---

[5] Versatile Information and Business Analysis is a requirements analysis method based on language/action theory [29].

[6] Material actions are actions that produce material results, such as painting a wall, while communicative actions result in social obligations, such as a promise to paint a wall in the future. The latter thus corresponds to what Searle [30] termed 'speech acts'.

Analysis [18]. Other approaches to capture method rationale in terms of design decisions are, for example, IBIS/gIBIS [31, 32] and REMAP [33]. The process-oriented view of method rationale captured by these approaches is important, especially when acknowledging method engineering as a continuous evolutionary process [17]. However, another, and as we shall see below, complementary approach to method rationale, primarily based on Max Webers' notion of practical rationality, has been put forth as means to understand why methods prescribe the things they do [15, 16].

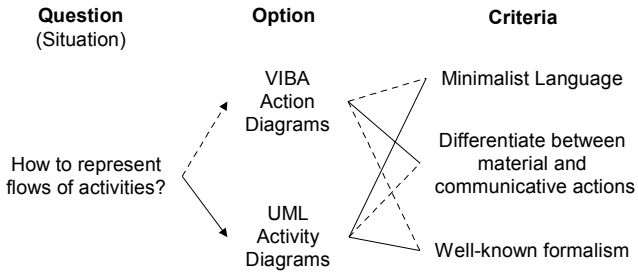

Fig. 3. Method rationale as choosing between options VIBA Action Diagrams and UML Activity Diagrams for modelling activity flows (based on the Question, Option, Criteria Model of Design Space Analysis [18]). The solid arrow between 'situation' and 'option' indicates the preferred choice; a solid line between an option and a criterion indicates a positive impact, while a dashed line indicates a negative impact.

According to Weber [22], rationality can be understood as a combination of means in relation to ends, ends in relation to values, and ethical principles in relation to action. This means that rational social action is always possible to relate to the means (instruments) used to achieve goals, and to values and ethical principles to which the action conforms. Weber's message is that we cannot judge whether or not an end is instrumentally optimal without considering the value-base upon which we judge the possibilities.

In this view of method rationale, all fragments of a method (prescribed concepts, notations and actions) are related to one or more goals. This means that if a fragment is proposed as part of a method, it should have at least one reason to be there. This idea, which is based on Weber's concept of 'instrumental rationality', is referred to as goal rationale. Each goal is, in turn, related to one or more values. This means that if a goal is proposed as the argument for a method fragment, it should have at least one reason to be there. The reason in this latter case is the goal's connection to a 'value base' underpinning the method. This idea, which is based on Weber's concept of 'rationality of choice', is referred to as value rationale.

Fig. 4 depicts this notion of method rationale, which also includes the idea that goals and values are related to other goals and values in networks of achievements and contradictions.

To illustrate how these two concepts of method rationale fit together, we will return to the example introduced above. Assume we have a model following Fig. 4 populated as follows (assuming that the classes in the model can be represented as

sets and associations as relations between sets, i.e. as sets of pairs with elements from the two related sets):

A set of method fragments F = {f_1: Representation of the class concept; f_2: Representation of the activity link concept; f_3: Representation of the action result concept}; A set of goals G = {g_1: Classes are represented in the model; g_2: Activity links are represented in the model; g_3: Activity results are represented in the model}; A set of values V = {v_1: Model only information aspects; v_2: Minimalist design of modelling language; v_3: Focus on instrumental v. communicative; v_4: Use well-known formalisms}; Goal rationale R_G = {(f_1, g_1), (f_2, g_2), (f_3, g_3)}; Value rationale R_V = {(g_1, v_2), (g_1, v_3), (g_1, v_4), (g_2, v_1), (g_2, v_2), (g_2, v_4), (g_3, v_3)}; Goal achievement GA = {(g_3, g_2)}; Value contradiction VC = {(v_1, v_3)}; VA = GC = Ø.

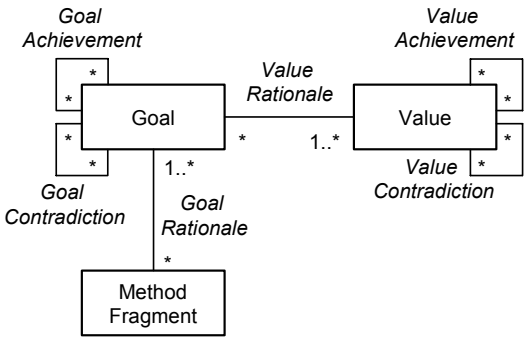

Fig. 4. Method rationale as consisting of interrelated goals and values as arguments for method fragments [16].

A perhaps more illustrative graphical representation of the model is shown in Fig. 5. If we view each method fragment in the model as possible options to consider, then the goals and values can be used to compare with the criteria in a structured way. Given that we know that what we want to describe in our notation is a flow of activities (or more precisely the link between activities) we can disregard f_1 outright, since its only goal is not related to what we are trying to achieve. When considering f_2 and f_3 we notice that each is related to a separate goal. However, since there is a goal achievement link from g_3 to g_2, we understand that both f_2 and f_3 would help satisfy the goal of representing visually a link between two activities (if we model results as output from one activity and input to another, we also model a link between the two). Since these two goals are based on different underlying and contradictory values. Since g_2 is related to v_1, and g_3 to v_3, we must choose the goal that best matches or own value base. This could, and should be expressed by the criteria we use. If we, for example, believe that it is important to direct attention to instrumental versus communicative aspects (v_3), then we should choose g_3 and consequently f_3. If, on the other hand, we are only concerned with modelling information flows, then g_2 and consequently f_2 would be the option to choose.

The concept of method rationale described above applies to both construction of methods and refinement of methods-in-action [17]. Since method descriptions are

means of communicating knowledge between method creators and method users, it could be used as a bridge between the two and thus as an important tool in achieving rationality resonance, as discussed above.

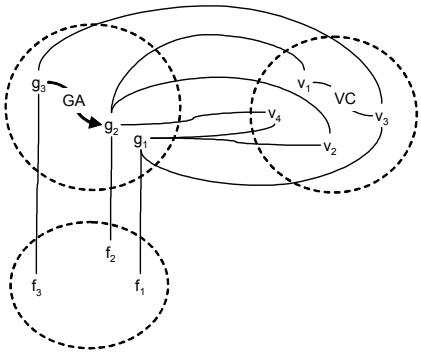

Fig. 5. Graphical representation of the method rationale mode showing the tree method fragments, the three goals, the three values, and their relationships. The goal achievement relation is represented by an arrow to indicate the direction of the 'goal contribution'. All other relationships are represented by non-directed edges since the direction of reading is arbitrary.

5 Using Method Rationale

From the example in the previous section we can see that method rationale is related to both the choices we make during method construction and to the goals and values that underpin the method constructs we choose among. In the theory of method fragments [24, 25], method fragments are thought of as existing on different layers of granularity, from the atomic 'concept level', through 'diagram', 'model' and 'stage', to the complete 'method'. The example used above was at a very detailed level, focusing on rationale in relation to method fragments at the concept layer of granularity. The same kind of analysis could be performed at any layer of granularity, and may consider both process and product fragments (i.e. both activities and deliverables).

In order to clarify, let us use a brief example from an ongoing case study. In this project we consider leveraging the use of agile methods for globally-distributed software development. This may seem counter-intuitive in many ways. One example is that agile methods usually stress the importance of having the development team co-located, even with an always present on-site customer [8]. This would obviously be impossible were the team geographically distributed across the globe. However, by analysing the reasons behind this method prescription (i.e. the suggestion by the method creator) we may find that we can operationalize the intended goals of co-location (such as increased informal communication) into other method prescriptions, say utilizing more advanced communication technologies. This way we could make sure that the method rationale of this particular aspect of an agile method is transferred into the rationale of a method tailored for globally-distributed

development. Thus we may be able to adhere to agile values even if the final method does look quite different from the original method.

It is important to see that method rationale is present at all three levels of method abstraction: ideal typical, situational, and in-action. At the ideal typical level, method rationale can be used to express the method creator's intentions, goals, values, and choices made. This would serve as a basis for method configurators (i.e. those who perform method configuration) and developers in understanding the method and tailoring it properly. In the communication between configurator and developer, method rationale would also express why certain adaptations were made when configuring the situational method. Finally, if we understand different developers' personal rationale, we might be able to better configure or assemble situational methods.

Combining the two aspects of method rationale gives us a structured approach to using method rationale both as a tool to express and document a method's rationale, and as a tool to analyse method rationale as basis for method construction, assembly, configuration and use.

6 Method Rationale Research

Method rationale has not received much attention in the literature so far, except for a few studies on why methods-in-action deviate from ideal typical and situational methods (although the latter distinction is not maintained). Obvious exceptions are the sources cited above, but the uptake by other researchers has so far been limited. It is interesting to note that there seems to be two strands of method research that largely pursue their own agendas without many cross-references. (Note that we intentionally construct two ideal types here.)

On the one hand we have the method engineering research which, as stated above, has to a large extent concentrated on the engineering of situational methods from 'atomic' method fragments forming larger 'method chunks' [24, 25, 34-38]. This strand of method research has not paid much attention to what actually happens in systems and software development projects where the situational method is used.

On the other hand, we have the method-in-action research that focuses on the relationship between linguistically expressed methods and methods-in-action [2, 39-41]. This research, while having contributed extensively to our understanding of method use and rationality resonance, seems to neglect the intricate task of defining and validating consistent method constructs.

Another way to put it is that there has been a lot of research on (a) the construction of situational methods out of existing method parts, and (b) the relationship between linguistically expressed methods (ideal typical methods and situational methods) on the one hand and methods-in-action on the other. The basic flaw in the research of type (a) is that it does not pay sufficient attention to actual method use. The focus is perhaps too much on what people should do, rather than on what they actually do. The basic flaw in research of type (b) is that it does not pay sufficient attention to the formality (rigour) required to ensure method consistency. That is, too little focus on how to codify successful practice into useful methods. Another flaw is that (b) does

not acknowledge the two different forms of linguistically expressed method abstraction levels.

There seems to be much to be gained from a systematic effort of integrating these research interests, and method rationale could be an important link between the two. It is not enough simply to state that a purported objectivistic and instrumental perspective inherent in the Method Engineering approach (sometimes somewhat incisively referred to as method-*ism* [39]) is fundamentally flawed if we are to understand methods-in-action properly. Methods are linguistic expressions as result of and basis for social action. Therefore we need to understand the complex social reality that shapes methods-in-action. Equally important, though, is to find ways to use that understanding as a basis for being able to cope better with the formal construction, verification and validation of methods at all three levels of method abstraction. The concept of method rationale can be used as an important tool in such a research effort. The reason is that it gives us one construct that can be used to understand method construction and use as social activity. At the same time it can be used to create a frame of reference for method engineering in terms of analysing, validating and communicating methods.

7 Conclusion

In this paper we have presented a communicative view on systems/software development methods. From this perspective, method descriptions are conceived of as linguistic expressions. As such, they are not just descriptions of ideal typical development processes, but expressions of method creators' suggestions as to how system development should be performed. Such descriptions are subsequently interpreted and (possibly) rationalized by method users. This is also a way of clarifying the distinction between method-in-concept and method-in-action [4] by highlighting that there are in fact several methods-in-concept (at least one per actor) involved in method formulation, communication and use. A method description is here seen as the linguistic expression of the method creator's method-in-concept. This description is then interpreted by method users when forming their own method-in-concept, which is a basis for their method-in-action.

With this foundation, we have also presented a comprehensive concept of method rationale by integrating two different method rationale aspects. Our conclusion is that method rationale exists as the goals and values upon which we choose what method fragments should belong to a particular method, method configuration or method assembly. Method rationale exists as an expression of the method creator's values, beliefs and understanding of the development context. This 'intrinsic' method rationale is then compared with method user's values, beliefs and understanding in method configuration and systems development.

This method rationale existence maps directly to three abstraction levels of methods: the ideal typical method (as expressed by the method creator), the situational method (as adapted by a process engineer/method configurator), and the method-in-action (as manifested by actual method-following actions). The first two levels constitute a linguistic aspect of method, and the last an action aspect.

A method, at any of the three levels, represents knowledge about software and systems development processes. Therefore, method rationale is present at all three levels. Method rationale can be made explicit, which may aid in communication between method creators and method users; a communication that is usually performed through method handbooks and modelling tools.

Finally, we have discussed how method rationale may be an important tool in understanding better the relationships between the three method levels, and in synthesising important (past, current and future) research on method engineering and method-in-action.

Acknowledgements

This work has been financially supported by the Science Foundation Ireland Investigator Programme, B4-STEP (Building a Bi-Directional Bridge Between Software ThEory and Practice).

References

1. Iivari, J., Maansaari, J.: The Usage of Systems Development Methods: Are We Stuck to Old Practice? Information and Software Technology 40 (1998) 501–510
2. Nandhakumar, J., Avison, D. E.: The Fiction of Methodological Development: A Field Study of Information Systems Development. Information Technology & People 12 (1999) 176–191
3. Fitzgerald, B., Russo, N. L., O'Kane, T.: Software Development Method Tailoring at Motorola. Communications of the ACM 46 (2003) 65–70
4. Lings, B., Lundell, B.: Method-in-Action and Method-in-Tool: Some Implications for CASE. In: Proc. 6th International Conference on Enterprise Information Systems (ICEIS 2004) (2004)
5. van Slooten, K., Hodes, B.: Characterizing IS Development Projects. In: S. Brinkkemper, K. Lyytinen, and R. Welke, (eds.): Method Engineering: Principles of Method Construction and Tool Support. Chapman & Hall, London (1996) 29–44
6. Cameron, J.: Configurable Development Processes: Keeping the Focus on What Is Being Produced. Communications of the ACM 45 (2002) 72–77
7. Karlsson, F., Ågerfalk, P. J.: Method Configuration: Adapting to Situational Characteristics While Creating Reusable Assets. Information and Software Technology 46 (2004) 619–633
8. Beck, K.: Extreme Programming Explained: Embrace Change. Addison-Wesley, Reading, MA (2000)
9. Schwaber, K., Beedle, M.: Agile Software Development with Scrum. Prentice-Hall, Upper Saddle River, NJ (2002)
10. Riemenschneider, C. K., Hardgrave, B. C., Davis, F. D.: Explaining Software Developer Acceptance of Methodologies: A Comparison of Five Theoretical Models. IEEE Transactions on Software Engineering 28 (2002) 1135–1145

11. Garfinkel, H.: Studies in Ethnomethodology. Polity Press, Cambridge, UK (1967)
12. Dourish, P.: Where the Action Is: The Foundations of Embodied Interaction. MIT Press (2001)
13. Eriksén, S.: Designing for Accountability. In: Proceedings of the Second Nordic Conference on Human-Computer Interaction (NordiCHI 2002). ACM Press (2002) 177–186
14. Oinas-Kukkonen, H.: Method Rationale in Method Engineering and Use. In: S. Brinkkemper, K. Lyytinen, and R. Welke, (eds.): Method Engineering: Principles of Method Construction and Support. Chapman & Hall, London (1996) 87–93
15. Ågerfalk, P. J., Åhlgren, K.: Modelling the Rationale of Methods. In: M. Khosrowpour, (ed.) Managing Information Technology Resources in Organizations in the Next Millennium. Proceedings of the 10th Information Resources Management Association International Conference. IDEA Group Publishing (1999) 184–190
16. Ågerfalk, P. J., Wistrand, K.: Systems Development Method Rationale: A Conceptual Framework for Analysis. In: Proc. 5th International Conference on Enterprise Information Systems (ICEIS 2003) (2003) 185–190
17. Rossi, M., Ramesh, B., Lyytinen, K., Tolvanen, J.-P.: Managing Evolutionary Method Engineering by Method Rationale. Journal of the Association for Information Systems 5 (2004) 356–391
18. MacLean, A., Young, R. M., Bellotti, V. M. E., Moran, T. P.: Questions, Options, and Criteria: Elements of Design Space Analysis. Human-Computer Interaction 6 (1991) 201–250
19. Fitzgerald, B., Russo, N. L., Stolterman, E.: Information Systems Development: Methods in Action. McGraw-Hill, Berkshire, UK (2002)
20. Goldkuhl, G.: The Grounding of Usable Knowledge: An Inquiry in the Epistemology of Action Knowledge. Linköping University, CMTO Research Papers 1999:03, Linköping, Sweden (1999)
21. Ågerfalk, P. J.: Grounding through Operationalization: Constructing Tangible Theory in IS Research. In: Proc. 12th European Conference on Information Systems (ECIS 2004) (2004)
22. Weber, M.: Economy and Society. University of California Press, Berkeley, CA (1978)
23. Polanyi, M.: Personal Knowledge: Towards a Post-Critical Philosophy. Routledge & K. Paul, Chicago (1958)
24. Harmsen, A. F.: Situational Method Engineering. Moret Ernst & Young Management Consultants (1997)
25. Brinkkemper, S., Saeki, M., Harmsen, F.: Meta-Modelling Based Assembly Techniques for Situational Method Engineering. Information Systems 24 (1999) 209–228
26. Parnas, D. L., Clements, P. C.: A Rational Design Process: How and Why to Fake It. IEEE Transactions on Software Engineering 12 (1986) 251–257
27. Stolterman, E.: The Paradox of Information Systems Methods: Public and Private Rationality. In: Proc. British Computer Society 5th Annual Conference on Methodologies (1997)
28. Booch, G., Rumbaugh, J., Jacobson, I.: The Unified Modeling Language User Guide. Addison-Wesley, Harlow, UK (1999)

29. Ågerfalk, P. J., Goldkuhl, G.: Business Action and Information Modelling: The Task of the New Millennium. In: M. Rossi and K. Siau, (eds.): Information Modeling in the New Millennium. Idea Group Publishing (2001) 110–136
30. Searle, J. R.: Speech Acts: An Essay in the Philosophy of Language. Cambridge University Press, Cambridge (1969)
31. Conklin, J., Begeman, M. L.: gIBIS: A Hypertext Tool for Exploratory Policy Discussion. ACM Transactions on Office Information Systems 6 (1988) 303–331
32. Conklin, J., Selvin, A., Shum, S. B., Sierhuis, M.: Facilitated Hypertext for Collective Sensemaking: 15 Years on from gIBIS. In: H. Weigand, G. Goldkuhl, and A. de Moor, (eds.): Proceedings of the 8th International Working Conference on the Language-Action Perspective on Communication Modelling (LAP 2003). Tilburg University (2003) 1–22
33. Ramesh, B., Dhar, V.: Supporting Systems Development by Capturing Deliberations During Requirements Engineering. IEEE Transactions on Software Engineering 18 (1992) 498-510
34. Brinkkemper, S.: Method Engineering: Engineering of Information Systems Development Methods and Tools. Information and Software Technology 38 (1996) 275–280
35. Rolland, C., Prakash, N.: A Proposal for Context-Specific Method Engineering. In: S. Brinkkemper, K. Lyytinen, and R. Welke, (eds.): Method Engineering: Principles of Method Construction and Tool Support. Chapman & Hall (1996)
36. Ralyté, J., Deneckère, R., Rolland, C.: Towards a Generic Model for Situational Method Engineering. In: J. Eder, n. nn, and n. nn, (eds.): Proceedings of 15th International Conference on Advanced Information Systems Engineering (CAiSE 2003), Klagenfurt, Austria, June 16–18, 2003. Springer-Verlag (2003) 95–110
37. ter Hofstede, A. H. M., Verhoef, T. F.: On the Feasibility of Situational Method Engineering. Information Systems 22 (1997) 401–422
38. Rolland, C., Prakash, N., Benjamen, A.: A Multi-Model View of Process Modelling. Requirements Engineering 4 (1999) 169–187
39. Introna, L. D., Whitley, E. A.: Against Method-*Ism*: Exploring the Limits of Method. Information Technology & People 10 (1997) 31–45
40. Russo, N. L., Stolterman, E.: Exploring the Assumptions Underlying Information Systems Methodologies: Their Impact on Past, Present and Future Ism Research. Information Technology & People 13 (2000) 313–327
41. Avison, D. E., Fitzgerald, G.: Where Now for Development Methodologies. Communications of the ACM 46 (2003) 79–82