Multi-Grounded Action Research in Method Engineering: The MMC Case

Fredrik Karlsson[1] and Pär J. Ågerfalk[2,3,4]
1 Methodology Exploration Lab, Dept. of Informatics (ESI),
Örebro University, SE-701 82 Örebro, Sweden
Email: fredrik.karlsson@esi.oru.se,
2 Dept. of Information Science, Uppsala University, Sweden
3 Jönköping International Business School,
P.O. Box 1026, SE-551 11 Jönköping, Sweden
4 Lero – The Irish Software Engineering Research Centre
University of Limerick, Limerick, Ireland
Email: agpa@jibs.hj.se

Abstract. There appears to be two schools of information systems development methods research that largely pursue their own agendas without many cross-references. On the one hand there is the method engineering research and on the other hand there is the method-in-action research. There seems to be much to be gained from integrating these two schools, developing knowledge that both has the formality (rigor) and reflects its enactment in practice. To achieve this, the research approach adopted has to embrace this duality. In this paper we explore how Multi-Grounded Action Research (MGAR) can contribute to achieving this aim. MGAR has been used in the development of a Method for Method Configuration, a research product that integrates the strengths of both schools.

1 Introduction

As noted by Ågerfalk and Fitzgerald [1], there appears to be two schools of information systems development method (ISDM) research that largely pursue their own agendas without many cross-references. On the one hand there is the method engineering (ME) research which has to a large extent concentrated on deriving situational methods from atomic method fragments or larger method chunks [2-7]. This school of ISDM research has paid limited attention to what actually happens in software development projects where the situational method is used. On the other hand, there is the method-in-action research that focuses specifically on how espoused ISDMs are enacted in practice [e.g. 8, 9, 10]. This school of ISDM research, while having contributed extensively to our understanding of method use,

Please use the following format when citing this chapter:

Karlsson, F., Ågerfalk, P.J., 2007, in IFIP International Federation for Information Processing, Volume 244, Situational Method Engineering: Fundamentals and Experiences, eds. Ralyté, J., Brinkkemper, S., Henderson-Sellers B., (Boston Springer), pp. 19-32.

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 espoused methods and methods-in-action. According to Ågerfalk and Fitzgerald [1], a basic flaw in the research of type (a) is that it often does not pay sufficient attention to actual method use. Perhaps focusing too much on what people should do, rather than on what they actually do. A basic flaw in research of type (b), on the other hand, is that it often does not pay sufficient attention to the formality (rigour) required to ensure method consistency. That is, too little focus on how to codify successful development practices into useful ISDMs. Another flaw is that (b) usually does not acknowledge the difference between what is termed base method [11] and situational methods, perhaps even confusing the latter with method-in-action (i.e. an ISDM as enacted in practice).

As pointed out by Ågerfalk and Fitzgerald [1], there seems to be much to be gained from integrating these two schools, and they even suggest method rationale could be an important link between the two. They argue that since ISDMs fundamentally are linguistic expressions as result of and basis for social action, we need to understand the complex social reality that shapes methods-in-action. On the other hand, it is imperative to use that understanding as a basis for formal construction, verification and validation of ISDMs. Subsequently, it becomes critical for the adopted research process to reflect this duality. The aim of this paper is to explore how Multi-Grounded Action Research (MGAR) can contribute to method engineering research. This is explored through reflecting on its use in the development of Method for Method Configuration (MMC), a research product where the rigor of ME research is combined with the social sensitivity of the method-in-action school.

The paper proceeds as follows. Section 2 contains an in-depth discussion of the MGAR approach. While the focus of this paper is on the MGAR approach as such, to facilitate understanding, Section 3 then provides a brief overview of the main research product, MMC. Following this, Section 4 provides empirical experiences from applying MGAR and provides an in-context perspective of the research approach. Finally, the paper ends with a concluding discussion in Section 5.

2 Multi-Grounded Action Research

MMC is the result of a collaborative project involving the Swedish research network VITS (with participants from Örebro University and University College Borås), University of Limerick, Ireland, and three Swedish software developing organizations: Volvo IT (a multi-national software and technology consultancy organization), Posten IT (the information technology division of Posten AB), and Precio (a mid-sized software consultancy company).

The research method used was that of Multi-Grounded [12] Action Research. Similar to grounded action research [13] it draws on the well-established qualitative research method Grounded Theory, particularly as it has evolved in the tradition of

Strauss and Corbin [14]. In a multi-grounded approach evolving as well as existing theory play an important part in data collection and analysis [cf. 15]. The idea is to ground theory not only in empirical data, but also internally and in other existing knowledge of theoretical character. This gives rise to three grounding processes, which were applied in this research: internal grounding, external grounding and empirical grounding.

Internal grounding means reconstructing and articulating 'background knowledge', that is, knowledge that might otherwise be taken for granted. For example, it is important to identify and explicate the basic assumptions behind MMC to understand how and when it is applicable. Internal grounding also includes defining concepts used and their interrelationships. The important contribution of this process in this particular research is a consistent conceptual model of MMC, free from ambiguities and with concepts that are anchored in explicit values and goals. That is, it ensures that the developed knowledge (MMC in our case) is logically consistent [16]. External grounding is concerned with relationships between the developed knowledge (its concepts and internal relationships) and other knowledge of a theoretical character. This is relevant for putting forward similarities and differences between the evolving knowledge and other existing knowledge. In our case, this meant ensuring that MMC builds on existing ME wisdom in a constructive way and that it does not contradict relevant previous studies. Empirical grounding emphasizes the importance of applying developed knowledge in practice to validate the concepts and their relationships in an empirical environment. In this context we use 'applying' in a broad sense, involving analysis, design and implementation, as well as test and evaluation. In our case, this involved designing parts of MMC together with qualified practitioners as well as gaining experience from using MMC in real projects when specifying situational ISDMs. Furthermore, as we learn more about the domain we research, knowledge can also be generated through classification of empirical phenomena, which results in refinement of the theoretical knowledge and thus triggers further internal and external grounding.

Our MGAR approach can be understood in terms of the traditional 'canonical' action research method with cycles of diagnosing, action planning, action taking, evaluating, and specifying learning [17]. The research project consisted of two such MGAR cycles, which are elaborated further in the sections below. Within these two cycles seven smaller 'action cases' [18] were performed, as shown in Table 1. An action case means involving competent practitioners in collaborative design and evaluation efforts. Problems and design decisions are discussed and taken together by researchers and practitioners, which means continuous feedback and interaction between the two [19]. The selection of action cases was based on finding a mixture of different organization-wide ISDMs and organizations. The choice of organizations was based on two premises: they had to use different, preferably well-known, organization-wide ISDMs, and they had to agree to put aside resources to enable the kind of collaboration envisaged. The organizations in this study ranged from quite small to very large and the ISDMs used in these organizations were the Rational Unified Process (RUP) and the Microsoft Solution Framework (MSF) – see Table 1. Furthermore, each action case served a specific purpose towards the final research product and was related to the action cycle discussed below.

Table 1. Action cases in chronological order

Action case	Business	ISDM	Case role
1. Volvo IT – pre case	Large	RUP	Method configuration diagnosis
2. Volvo IT	Large	RUP	MMC design, configuration application
3. ESI	Small	RUP	MMC validation, configuration application
4. Posten IT	Large	RUP	MMC redesign & validation, configuration application
5. Posten IT	Large	RUP	MMC redesign & validation, reconstruction of configuration
6. Precio	Medium	MSF	MMC validation, creating configuration
7. Precio	Medium	MSF	MMC validation, reconstruction of configuration

2.1 The First Multi-Grounded Action Research Cycle

The first MGAR cycle was carried out between spring 2000 and spring 2002. During this cycle the first three action cases in Table 1 were carried out, together with the research collaborators Volvo IT and ESI.

Phase 1 – Diagnosing (Action case 1): Difficulties related to tailoring an organization-wide method for a specific project was explored at Volvo IT through a series of workshops with a systems development project requiring a situational ISDM. Problems with the current way of tailoring the organization-wide ISDM were documented. Based on principles from the situational ME literature, a vision of how to improve method configuration was formulated. The data sources from this phase were: log books from three workshop sessions, organization-wide ISDM (the RUP), situational method, project deliverables. Data analysis was done using problem analysis [20] and conceptual models. Problem analysis was used to separate real problems from symptoms. Conceptual models of the organization-wide ISDM were created to facilitate understanding of experienced phenomena, using UML and other standard techniques.

Phase 2 – Action planning (Action case 2): A set of design principles for improved method configuration was developed in a series of workshops. These were anchored in the formulated vision, prioritized problems, and principles from the situational ME literature. The proposed design principles were: the principle of modularization, the principle of method rationale for selecting method parts and the principle of a multi-layered reuse model. The data sources from this phase were: log books from two planning sessions and the vision document.

Phase 3 – Action taking (Action case 2 & 3): Based on the design principles, a prototype of MMC was developed. In a series of workshops, a conceptual structure and a classification schema were chiseled, along with a number of instructions for the method engineer role. The four main concepts were: the method fragment (based on established situational ME principles [4]), the base method (the organization-wide ISDM), the configuration package, and the configuration template. The latter two concepts were introduced to facilitate modular reuse of method configurations.

Volvo IT provided a set of existing projects as input for the design sessions and emerging concepts were tried against those projects' requirements. A summary of

the work so far was presented at an international workshop [21]. When MMC had stabilized sufficiently, it was used in a small scale project at ESI, which enabled active participation throughout the project. The chosen base method was the RUP. Identified data sources were: log books from twelve design sessions and a preliminary version of MMC.

Phase 4 – Evaluation (Action case 3): The first full-scale evaluation of MMC was based on the active participation in the ESI project. The business objectives of the systems development project were to offer the situational information about personnel to internal users and external users outside the organization. The first author was project manager and method engineer during this project. Five different data sources were of interest from this phase: the situational ISDM that was used during the project, defined reusable assets, project artifacts, project results, and the project log book.

The data sources were analyzed with a focus on encountered problems and achieved design goals. Documented problems were traced to possible causes. For example, some of these causes could be traced back to the situational method and MMC, the systems developer's knowledge of the base method, or a combination. The developed software was evaluated through interviews with end users and change requests tracking. [22].

Phase 5 – Specifying learning: On the basis of the data analysis in Phase 4, lessons learnt, including practical advices on how to use the proposed meta-method, and change requests were outlined. A summary of the work so far was published in an international journal [11].

2.2 The Second Multi-Grounded Action Research Cycle

The second MGAR cycle included four action cases as shown in Table 1. These were carried out together with Posten IT and Precio between autumn 2002 and autumn 2004.

Phase 6 – Diagnosing: The diagnosing phase was based on the specified lessons learnt from the first MGAR cycle. These lessons were analysed from two different perspectives: design flaws in the MMC prototype were identified, and the need for a CAME-tool based on MMC was identified (note that the tool aspect is beyond the scope of this paper and is only mentioned here for completeness). Data sources during this phase were the lessons learnt and change requests from the first action research cycle. The data analysis, like in Diagnosing during the first MGAR cycle, was done using problem analysis [20], to separate real problems from symptoms.

Phase 7 – Action planning (Action case 4 & 5): Based on lessons and change requests from the first MGAR cycle, the set of design principles was refined. It resulted in a set of sub-principles:

- The principle of modularization: self-contained modules, internally consistent and coherent modules, support for information-hiding and implementable in a CAME-tool;

- The principle of method rationale for selecting method parts: support analysis of potential to achieve rationality resonance, and support 'method-in-action' [23] decisions;
- The principle of a multi-layered reuse model.

Phase 8 – Action taking (Action cases 4–7): MMC was redesigned based on the refined principles. A new modularization concept was introduced, based on a modification of an existing ME concept: the method component. This concept was integrated with the two concepts configuration package and configuration template. As a consequence of this redesign, the classification schema was changed as well. The method rationale and method component concepts were presented at international conferences [24, 25] and in an international journal [26]. During this phase the following data sources were produced: log books from four design sessions and MMC.

The redesigned MMC was used in live projects at Posten IT and Precio. In these projects, RUP and MSF were used as base methods. In total MMC was used in four different project settings as shown in Table 1. Two of the projects included reconstruction of existing situational methods into reusable assets and the remaining two projects focused actual tailoring of ISDMs.

Phase 9 – Evaluation (Action cases 4–7): Evaluation of MMC (Phase 8) was performed during its use at Posten IT and Precio. Through-out these projects, group interviews were performed with project members. The following data source was used during the evaluation: log books from six configuration workshops, situational methods, defined reusable assets, project artifacts, and four group interviews. Theses data sources were analyzed with a focus on encountered problems and achieved design goals.

Phase 10 – Specifying learning: On the basis of the data analysis in Phase 9, lessons learnt and change requests were outlined. As during the first MGAR cycle the first category contains practical advice and the second category contains identified design flaws that have had subsequent design implications.

3 The Method for Method Configuration – MMC

The aim of the research product MMC is to support method configuration, as a specific kind of ME, which we define as: the planned and systematic adaptation of a specific method through the use of reusable assets. The foundation of MMC is its conceptual framework, shown in Fig. 1. MMC provides the possibility to work with reusable assets of ISDMs through the three core concepts: the method component [24, 26], the configuration package [27], and the configuration template [27]. Together they constitute types of reusable assets of different magnitude.

A method component is the smallest meaningful and coherent part of an ISDM, and the organization-wide ISDM (base method) has to be represented as a set of such components in order to use MMC. Method components are used as modularization blocks that are excluded, added and exchanged from the base method. For example, an ISDM might include a method component concerned with software packaging;

involving copying the software on distributable medium, printing handbooks, and designing a cardboard box with a selling cover. In projects where the final product is delivered using the Internet such a component is often considered superfluous and can be excluded.

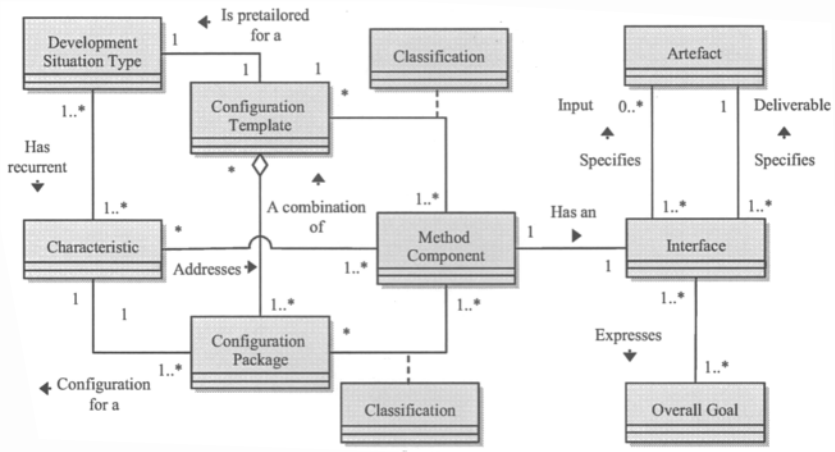

Fig. 1. A conceptual meta-model of Method for Method Configuration

Configuration packages and configuration templates are used to represent situational versions of an ISDM. The main difference between these two concepts is how much of a situational method they represent. Briefly, a configuration package can be described as a pre-made method configuration designed to fit one single specific development characteristic. If we continue on the short example in the previous paragraph it is likely that we are working with a characteristic of Internet delivery. In such a case the characteristic should affect method components aiming for product distribution. The result of how such method components are selected with respect to this characteristic is represented in the configuration package. This selection of components can, if required, include components from complementing ISDMs.

Real-life development situations obviously comprise a combination of several characteristics. For example, a single project may involve a number of diverse characteristics, such as unstable requirements, low degree of management support, a new technical platform and Internet delivery. The configuration template then represents this more complex configuration, and is a pre-configuration of the complete organization-wide ISDM for a typical project situation in the organization. A configuration template is constructed from a selection of configuration packages (each one representing a characteristic) and can be viewed as an aggregate of configuration packages. Hence, a configuration template reflects a recurring development pattern in an organization.

The remainder of this section is structured based on the three core concepts introduced above. For an extensive presentation of these concepts and content

examples see Karlsson and Wistrand [26], Wistrand and Karlsson [24] and Karlsson [27].

3.1 The Method Component Concept

In order to achieve a systematic yet straightforward way of working with method configuration a modularization concept that implements information hiding is required. Through such a concept it is possible to define the smallest coherent ISDM part that can be suppressed, added or exchanged: *A method component is a self-contained part of an ISDM expressing the transformation of one or several artifacts into a defined target artifact, and the rationale for such a transformation.*

The Method Component Content
A method component consists of two parts: its content and the rationale expressing why the content is designed as it is and what it can bring about. The content of a method component is an aggregate of method elements: *A method element is a part of an ISDM that manifests a method component's target state or facilitates the transformation from one defined state to another.*

The concept of method element can be specialized into five categories. First, there are three interrelated parts frequently mentioned in the literature: *prescribed action*, *concept*, and *notation*. Prescribed actions together with sequence restrictions guide the performance of activities and tell project members what actions to take in specific situations. In performing these actions, the concepts direct developers' attention towards specific phenomena in the problem domain. Hence, concepts are used to express an understanding of the problem domain, and also of the ISDM itself. The results of the prescribed actions are documented using a specific notation, which gives the concepts a concrete representation.

Second, based on empirical observations from MGAR Cycle 1, these categories are complemented with *artifact* and *actor role* as two further sub-types of method element. Project members tended to discuss ISDMs from an artifact perspective during method configuration and software development projects. This is also in line with previous research emphasizing the importance of 'keeping the focus on what is being produced' [28]. Artifacts act as deliverables from the transformation process as well as input to this process. Our use of the term input should not be interpreted in terms of a precondition. ISDMs are here viewed as heuristic procedures and consequently specified inputs are considered to be recommended inputs. However, a method component needs to have at least one input. Otherwise the method component will not have any meaningful support in the method. One exception is method components that initiate new activities that are later integrated with the result from other method components. The selection of actor roles are determined by the prescribed actions that need to be part of the transformation process. Actor roles are played either as drivers or as participants of the prescribed actions in the method component. Observations from MGAR Cycle 1 show that actor roles are important when mapping the situational ISDM to the actual work organization.

The rationale part of the method component concept consists of two parts: *goals* and *values*. Method elements exist for reasons, which are made explicit by means of

associating method elements to the goals. These goals are anchored in values of the method creator [1]. Taken together, goals and values are often considered important constituents of an ISDM's underlying perspective or 'philosophy' [23]. When working with method configuration according to MMC, method rationale is more important than the deliverable as such. Through the method rationale it is possible to address the goals that are essential in order to fulfill the overall goal of a specific project. Prescribed actions and artifacts are only means to achieve something and method rationale can thus help developers not to loose sight of that ultimate result, and also help them find alternative ways forward.

The Method Component Interface

The second purpose of the method component concept is to hide unnecessary details during method configuration, providing a sort of encapsulation. Thus, we draw on how the component concept is traditionally used in software engineering [29]. How a task is executed is not interesting from an external view of a component. A user of a component is primarily interested in the results offered by the component and the required inputs needed to achieve those results. The reduction of complexity is achieved through the method component interface: *A method component interface is a selection of method fragments and rationale that are relevant for the task at hand.*

The interface creates an external view of method components. The interface's content depends on the task at hand [26]. Empirical observations from MGAR Cycle 1 show that the method component's overall goals and the artifacts are central during method configuration. Therefore, they are part of the interface during method configuration as shown in Fig. 1. Artifacts are, as discussed above, designated as input and/or deliverable (output). This is necessary in order to deal with the three fundamental actions that can be performed on an artifact: create, update and delete. In cases when an artifact is only created by a method component, it is classified as a deliverable. If the artifact can be updated by the same method component it is classified as input as well. Furthermore we stipulate that a component can take one or several input artifacts, but has only one deliverable. Finally, the interface also expresses the overall goals of the method component representing the method rationale. These goals are used during method configuration and when discussing the rationality resonance possible to achieve during a project with certain characteristics.

3.2 The Configuration Package

Method configuration is about deciding whether or not method components in a base method are to be performed, and to what extent. This is done through the focus a characteristic has on the rationale of a method; rationale that is expressed through the method components' interfaces. A characteristic is viewed as a question about one aspect of the development situation type [11]. This question can have one or more possible answers that constitute the characteristic's dimension; one possible answer is termed configuration package in MMC. Each characteristic addresses one or several method components and their purpose of existence. Hence, each configuration package has a scope: the method components that are of interest for classification based on the characteristic. The scope is defined in order to reduce the

number of classification operations that have to be performed when creating a configuration package.

Thus, a configuration package is a classification of method components (see Fig. 1) with regard to how relevant their overall goals are for a specific answer in a characteristic's dimension. An example of a characteristic is 'Business processes already well understood?' which could address method components about business modeling. Two possible answers are 'We have good knowledge about existing business processes' and 'We have no knowledge about existing business processes.' In this case the dimension consists of two answers, or configuration packages. Thus component-based method configuration links to the idea of larger reusable blocks of method modules, blocks termed configuration packages: *A configuration package is a configuration of the base method suitable for a characteristic's value.*

The classification of method components is based on a two-dimensional classification schema. The vertical dimension focuses how much attention should be devoted to a particular method component: 'None,' 'Insignificant,' 'Normal' or 'Significant'. If at this stage a method component is found to be unimportant, it can be classified as 'Omit' outright. The three aspects of the horizontal dimension: 'Satisfactory,' 'Unsatisfactory' and 'Missing' cut across the vertical dimension. This dimension is referred to as the potential for achieving rationality resonance, based on the content of the base method. Together this schema provides different variants of the fundamental method configuration scenarios [24] that need to be supported: selection, exchange and addition.

3.3 The Configuration Template

The configuration package concept is used together with characteristics to simplify analysis of the base method. Still, we can conclude that software projects are not simple, nor are situational methods. Consequently, we need configurations that reflect this more complicated picture where characteristics exist in combinations. The configuration template concept is used for this purpose: *A configuration template is a combined method configuration, based on one or more configuration packages, for a set of recurrent project characteristics* (see Fig. 1). Hence, configuration templates make it possible to tailor the base method more efficiently. The concept as such allows reuse of combined configuration packages that target development situation types common within the organization. Configuration packages on the other hand are used to narrow the analysis and to reduce the complexity.

The situational method is based on a selected configuration template and is the ISDM delivered to the project team for use. This method is then turned into method-in-action when enacted by the project team members. Thus there is a difference between the tailored version of the base method and the method-in-action [1]. Experiences from the latter should be fed back to the configuration process, in order to improve configuration templates and/or configuration packages. This feedback is typically done continuously throughout the project, for example, at the end of each iteration, or during project close-out.

4 Multi-Grounded Action Research in Action

4.1 Volvo IT – Emphasis on External Theoretical Grounding

From a retrospective point of view theoretical grounding has played an important part during the research process. One clear example is the introduction of modularization concepts, a joint decision by the researchers and the practitioners.

The method fragment concept was used during the initial development of the first version of MMC, with a starting point in the process model [11]. Each process fragment has a purpose, based on the assumption that prescribed actions are prescribed for reasons. The effect on product fragments was then traced through the relationship between the process and product model. Since it was found difficult to balance precision and cost when using method fragments [11], a modified version of the method component concept was later introduced [24]. The major changes to the method component concept were the introduction of two distinctive views, the operationalization of the interface concept and viewing method components as non-hierarchic. These changes were introduced in order to reduce the number of details to present to the method engineer.

4.2 Posten IT – Emphasis on Internal Grounding

The empirical work at Posten IT included two action cases during the second MGAR cycle. The systems development aim of the first action case was to adapt an existing IS to new regulations, and in the second case to implement and host an IS in the electronic government area. The first case was a reconstruction project from a research point of view. Hence, the research aim was to test possibilities for reconstructing reusable patterns, based on how a project had been working. In the second case a configuration team was to deliver reusable patterns for an upcoming project. Both projects shared some features and parts of situational ISDMs.

During one of the joint introduction sessions for the work at Posten IT the impact of internal grounding became obvious. The tool mentor working with these project teams found an inconsistency in the activity diagram that presented MMC's overall process structure [11]. One of the decision points in the diagram was placed illogically. This error had the following consequence when searching for a configuration template as the base for a situational method: if a configuration template could not be found when trying to define a situational method it triggered an input to manage a change request for a template. But since no appropriate configuration template could be found, the result would always be defining a new configuration template. Subsequently, an improvement was to directly trigger the prescribed action for defining a configuration template.

4.3 Precio – Emphasis on Empirical Grounding

The empirical work at Precio shared similarities with the studies conducted at Posten IT, although Precio used MSF as their base method rather than RUP. Two action

cases were carried out with Precio during the second MGAR cycle: one reconstruction case and one case where method configuration was conducted in preparation for an upcoming project. The reconstruction case concerned a booking system involving two companies sharing the new IS. The second project concerned extending an existing IS with a report module. The idea was to find reusable parts during reconstruction and configuration work and hopefully share the configuration packages between the two projects.

The empirical grounding during these two action cases showed that the artifact focus of a method component together with method rationale is a natural starting point for method engineers and project team members when discussing methods. For example, the project manager of the booking system project expressed that 'it is easy to translate to the use of deliverables' and a team member of the Report module project expressed method components as 'easy to grasp.'

However, the team members of the Report module project suggested design improvements for the method component interface concerning the use of method rationale. They expressed in unison a need for more 'precise goals' to capture what a component's deliverable could be used for subsequently. At that time, the conceptual framework only allowed the interface to contain one goal. This limitation meant that it was impossible to express multiple purposes of a method component. The participating developers stressed that the resulting artifact of a method component sometimes 'is used for different purposes.' Hence, the current design thus limited their potential to discuss method components. This design restriction had forced rewriting goals, found in the base method, into one comprehensive goal for each component, which then became ambiguous. As a result the conceptual design of the method component was changed to include the possibility to express multiple purposes in the method component's interface.

5 Concluding Discussion and Lessons Learned

In this paper we have shown how Multi-Grounded Action Research (MGAR) can be used in method engineering research, in our case to devise a Method for Method Configuration (MMC). This research approach combines the need to capture knowledge of how systems development methods are used and tailored in organizations with how to formalize such knowledge, and evaluate it in use. MGAR has proved to be a relevant and valuable approach in the development of MMC. The approach provides a balanced research method that has been instrumental in combining ME rigor with the social sensitivity of the method-in-action school.

In terms of MGAR, four major lessons can be learned from this research project. First, the three grounding processes (internal, theoretical and empirical) are not to be seen as a division of work between researchers and practitioners, where the researchers do the theoretical and internal grounding and the practitioners do the empirical grounding. Neither are they a classification schema for where the work is carried out. Instead, they constitute processes carried out together; processes with a specific focus. An example is the inconsistency discussion at Posten IT, which shows

how internal grounding was carried out at the research site and involved joint efforts by practitioners and researchers.

Second, these three grounding processes are intertwined. This means that attention between the different processes can shift continuously. During a project meeting it is, for example, possible to cover each process several times. This, however, can raise difficulties when it comes to tracing the origin of data. For example: Was the inconsistency discussion generated from internal or empirical grounding? In this case it was clearly internal grounding since the discussion had a conceptual focus. It was, however, induced by empirical grounding, witheout which the problem would not have been identified.

Third, as is the case with Grounded Theory, MGAR opens a box with a vast amount of empirical data. This is certainly another reason for why it is difficult to trace data as the project moves on, sometimes in a rather fast pace. Since data is of no use if it is not documented and analysed properly, it is important to have a well functioning way of working with documentation. For this purpose, it is possible to use existing theories as filters. In our case, design decisions about evolving concepts were important together with method configuration results and judgments about the concepts in use.

Finally, working with industry may imply a trade-off between relevance and external theoretical anchoring. When solving day-to-day problems, external theoretical grounding easily falls into the background. Effectively, this means that developed knowledge may not always build on and develop existing research, which a conscious focus on external theoretical grounding then becomes a way to avoid.

References

1. P.J. Ågerfalk and B. Fitzgerald, in: In Advanced Topics in Database Research, edited by K. Siau (PA: Idea Group, Hershey, 2006), pp. 63-78.
2. S. Brinkkemper, Method engineering: engineering of information systems development methods and tools, *Information and Software Technology*. **38**(4), 275–280, (1996).
3. S. Brinkkemper, M. Saeki, and F. Harmsen, Meta-modelling based assembly techniques for situational method engineering, *Information Systems*. **24**(3), 209–228, (1999).
4. A.F. Harmsen, *Situational Method Engineering* (Moret Ernst & Young Management Consultants, Utrecht, The Netherlands, 1997).
5. J. Ralyté, R. Deneckère, and C. Rolland, Towards a Generic Model for Situational Method Engineering in: Advanced Information Systems Engineering, 15th International Conference, CAiSE 2003, LNCS 3084, Springer-Verlag, pp.202-218.
6. C. Rolland and N. Prakash, A Proposal For Context-Specific Method Engineering in: Method Engineering: Principles of method construction and tool support, edited by S. Brinkkemper, K. Lyytinen, and R. Welke (Chapman & Hall, 26–28 August 1996).
7. A.H.M. ter Hofstede and T.F. Verhoef, On the Feasibility of Situational Method Engineering, *Information Systems*. **22**(6/7), 401–422, (1997).
8. D.E. Avison and G. Fitzgerald, Where now for development methodologies? *Association for Computing Machinery. Communications of the ACM.* **46**(1), 78, (2003).
9. L.D. Introna and E.A. Whitley, Against Method-*ism*: Exploring the limits of method, *Information Technology & People*. **10**(1), 31–45, (1997).

10. N.L. Russo and E. Stolterman, Exploring the assumptions underlying information systems methodologies: their impact on past, present and future ISM research, *Information Technology & People*. **13**(4), 313–327, (2000).
11. F. Karlsson and P.J. Ågerfalk, Method Configuration: Adapting to Situational Characteristics while Creating Reusable Assets, *Information and Software Technology*. **46**(9), 619-633, (2004).
12. M. Lind and G. Goldkuhl, How to develop a Multi-Grounded Theory: The Evolution of a Business Process Theory, *Australian Journal of Information Systems*. **13**(2), 69-85, (2006).
13. R. Baskerville and J. Pries-Heje, Grounded action research: a method for understanding IT in practice, *Accounting, Management and Information Technologies*. **9** 1-23, (1999).
14. A.L. Strauss and J.M. Corbin, *Basics of qualitative research: techniques and procedures for developing grounded theory* (SAGE, Thousand Oaks, CA, 1998).
15. H.K. Klein and M.D. Myers, A Set of Principles for Conducting and Evaluating Interpretive Field Studies in Information Systems, *MIS Quarterly*. **1** 67-94, (1999).
16. A.S. Lee, A Scientific Methodology for MIS Case Studies, *MIS Quarterly*. **13**(1), 33-51, (1989).
17. R. Baskerville and A.T. Wood-Harper, Diversity in information systems action research methods, *European Journal of Information Systems*. **7** 90-107, (1998).
18. K. Braa and R. Vidgen, Interpretation, intervention, and reduction in the organizational laboratory: a framework for in-context information system research, *Accounting, Management and Information Technologies*. **9**(1), 25–47, (1999).
19. L. Mathiassen, Collaborative Practice Research, *Information Technology & People*. **15**(4), 321-345, (2002).
20. G. Goldkuhl and A. Röstlinger, *Joint elicitation of problems: An important aspect of change analysis*, in *Human, Organizational, and Social Dimensions of Information Systems Development*, D.E. Avison, J.E. Kendall, and J.I. DeGross, Editors. 1993: North-Holland. p. 107–125.
21. F. Karlsson, P.J. Ågerfalk, and A. Hjalmarsson, Process Configuration with Development Tracks and Generic Project Types in: Proceedings of the 6th CAiSE/IFIP8.1 International Workshop on Evaluation of Modelling Methods in Systems Analysis and Design (EMMSAD'01)(4–5 June 2001).
22. F. Karlsson, Method Configuration - A Systems Development Project Revisited in: The Fourteenth International Conference on Information Systems Development (ISD 2005), edited by A.G. Nilsson, et al. (Springer, Karlstad, Sweden, 14-17 August, 2005).
23. B. Fitzgerald, N.L. Russo, and E. Stolterman, *Information Systems Development - Methods in Action* (McGraw-Hill, London, 2002).
24. K. Wistrand and F. Karlsson, Method Components - Rationale Revealed in: The 16th International Conference on Advanced Information Systems Engineering (CAiSE 2004), edited by A. Persson and J. Stirna (Riga, Latvia, June 7 - 11, 2004).
25. P.J. Ågerfalk and K. Wistrand, Systems Development Method Rationale: A Conceptual Framework for Analysis in: Proceedings of the 5th International Conference on Enterprise Information Systems (ICEIS 2003)(Angers, France, 23–26 April 2003).
26. F. Karlsson and K. Wistrand, Combining method engineering with activity theory: theoretical grounding of the method component concept, *European Journal of Information Systems*. **15** 82-90, (2006).
27. F. Karlsson, *Method Configuration - Method and Computerized Tool Support* (Linköping University, Linköping, 2005).
28. J. Cameron, Configurable Development Processes, *Communications of the ACM*. **45**(3), 72-77, (2002).
29. P. Stevens and R. Pooley, *Using UML - Software Engineering with Objects and Components* (Addison Wesley, Essex, England, 2006).