Situational Method Engineering: Fundamentals and Experiences

IFIP WG 8.1 Working Conference
Poster Proceedings

September 12 – 14, 2007, Geneva, Switzerland

Edited by

Jolita Ralyté
University of Geneva, Switzerland

Sjaak Brinkkemper
Utrecht University, The Netherlands

Brian Henderson-Sellers
University of Technology, Sydney, Australia
Preface

This proceedings volume contains the papers of the poster session of the IFIP Working Group 8.1 Working Conference on Situational Method Engineering: Fundamentals and Experiences – ME’07. The conference has been held from 12 to 14 September 2007 at the University of Geneva in Switzerland.

After two successful Method Engineering conferences in Atlanta in 1996 and in Kanazawa in 2002 organised by the IFIP WG8.1, we provide again a forum for the exchange of ideas in and give a state of the art overview in Method Engineering. The conference programme features three invited keynote presentations, regular paper presentations, poster session and one interactive panel session. The format of a working conference allows for extensive paper discussions featured by discussant reviews in plenary sessions.

In total 47 submissions were received and each paper has been reviewed by four members of the program committee, recruited from IFIP 8.1 members and other researchers active in method engineering domain. The overall quality of the papers was very high, and very well fitting to the scope of Method Engineering. The program committee decided to accept 23 papers for presentation in the main program and 5 papers for the poster session which are published in this volume. We take this opportunity to thank all the authors for their interest to the ME’07 conference.

We wish to thank the members of the international program committee and the additional reviewers, who assisted in making a good selection for a high quality program. A special word of thanks goes to the chairman of IFIP Working Group 8.1, Barbara Pernici of the Politecnico di Milano, and the keynote speakers Steven Kelly of the MetaCase, Colette Rolland of the University Paris 1 – Sorbonne and Cesar Gonzalez-Perez, ESI, Spain.

We wish you to enjoy the conference and have a great time in Geneva, Switzerland!

September 2007

Jolita Ralyté
Sjaak Brinkkemper
Brian Henderson-Sellers
Conference Committee

General Chair

Brian Henderson-Sellers, University of Technology, Sydney, Australia

Program Chairs

Jolita Ralyté, University of Geneva, Switzerland
Sjaak Brinkkemper, Utrecht University, The Netherlands

Organising Chair

Mehdi Snene, University of Geneva, Switzerland

Web Master

Inge van de Weerd, Utrecht University, The Netherlands

Program Committee

Pär J. ÅGERFALK, Ireland  
Jacky AKOKA, France  
David AVISON, France  
Per BACKLUND, Sweden  
Albertas ĖAPLINSKAS, Lithuania  
Massimo COSSENTINO, Italy  
Rébeccca DENECKERE, France  
Xavier FRANCH, Spain  
Cesar GONZALEZ-PEREZ, Spain

John GRUNDY, New Zealand  
Peter HAUMER, USA  
Frank HARMSSEN, The Netherlands  
Remko HELMS, The Netherlands  
Manfred JEUSFELD, The Netherlands  
Paul JOHANNESSON, Sweden  
Frederik KARLSSON, Sweden  
John KROGSTIE, Norway  
Susanne LEIST, Germany
Michel LÉONARD, Switzerland  
Mauri LEPPANEN, Finland  
Kalle LYYTINEN, USA  
Isabelle MIRBEL, France  
Haralambos MOURATIDIS, UK  
Mohan NARASIPURAM, Hong Kong  
Selmin NURCAN, France  
Antoni OLIVE, Spain  
Leon J. OSTERWEIL, USA  
Oscar PASTOR, Spain  
Juan PAVÓN, Spain  
Barbara PERNICI, Italy  
Anne PERSSON, Sweden  
Yves PIGNEUR, Switzerland  
Klaus POHL, Germany  
Naveen PRAKASH, India  
Erik PROPER, The Netherlands  
Colette ROLLAND, France  
Matti ROSSI, Finland  
Motoshi SAEKI, Japan  
Guttorm SINDRE, Norway  
Monique SNOECK, Belgium  
Il-Yeol SONG, USA  
Juha-Pekka TOLVANEN, Finland  
Richard WELKE, USA  
Robert WINTER, Switzerland

**Additional Referees**

Lena AGGESTAM, Sweden  
Tobias BUCHER, Switzerland  
Stephan KURPJUWEIT, Switzerland  
Valeria SEIDITA, Italy  
Maurice VERHELST, Belgium  
Jonas SJÖSTRÖM, Sweden
# Table of Contents

Towards a Method for Collaborative Policy Making ................................. 1  
*Josephine Nabukunya, Patrick van Bommel, and Erik Proper*

A Practitioner Based Method Tailoring Model for Information Systems  
Development ........................................................................................................ 15  
*Timothy J. Burns and Fadi P. Deek*

Utilizing Theories to Reduce the Subjectivity of Method Engineering  
Processes ........................................................................................................... 30  
*Andreas Gehlert and Daniel Pfeiffer*

The Adoption of Method Engineering Principles for the Creation of  
Organization Specific IT Processes ................................................................. 44  
*Boštjan Žvanut, Marko Bajec*

Practical Use of Method Engineering: Analysing Benefits and Challenges ..... 61  
*Susanne Leist and Josef Blasini*
Towards a Method for Collaborative Policy Making

J. (Josephine) Nabukenya, P. (Patrick) van Bommel, H.A. (Erik) Proper
Institute for Computing and Information Sciences
Radboud University Nijmegen
Toernooiveld 1, 6525 ED Nijmegen, The Netherlands
{J.Nabukenya, P.vanBommel, E.Proper}@cs.ru.nl

Abstract. This paper is concerned with the development of a method for collaborative policy making. The aim of this method is to improve the quality of policymaking processes. The creation of policies is a collaborative process. The quality of this collaboration has a profound impact on the quality of the resulting policies and the acceptance by its stakeholders. We therefore aim to integrate techniques from the field of collaboration engineering into our policy making method in order to improve the quality of the process and its outcomes. We present the results of two case studies conducted on the use of collaboration engineering in the context of policy making processes. A key element in this result involves the initial design of a method for policy making in terms of elementary constructs from collaboration engineering.

1 Introduction

The current complexity in organizational decision-making has led to a multitude of approaches. Among them is the concept of policy. A policy [1] is a guide that establishes parameters for making decisions; it provides guidelines to channel a manager’s thinking in a specific direction. The concept of policy is not limited to the world of business and government alone. In the field of IT, several forms of policies exist as well. For example, [2] discusses the notion of IT policies to govern and direct an organization’s IT portfolio, while [3, 4] have used the term architecture principle to refer to the same notion. Another form of policy playing an increasingly important role in the field of IT are business rules as a mechanism to formalize business policies [5].
Policies are created in a policy-making process, which involves an iterative and collaborative process requiring an interaction amongst three broad streams of activities: problem definition, solution proposals and a consensus based selection of the line of action to take. The core participants of a policy-making process must be involved in complex and key decision making processes themselves, if they are to be effective in representing organizational interests. Explicit policies are a key indicator for successful organizational decision-making.

The complexity of policy-making processes in organizations may be described as having to cope with large problems. Examples include: information technology, innovation, procurement, security, software testing, etc. These problems may be affected by (i) unclear and contradictory targets set for the policy goals; (ii) policy actors being involved in one or more aspects of the process, with potentially different values/interests, perceptions of the situation, and policy preferences. Policy makers and others involved in the policymaking process need information to understand the dynamics of a particular problem and develop options for action. A policy is not made in a vacuum. It is affected by social and economic conditions, prevailing political values and the public mood at any given time, as well as the local cultural norms, among other variables.

A policy-making process is a collaborative design process whose attention is devoted to the structure of the policy, to the context and constraints (concerns) of the policy and its creation process, and the actual decisions and events that occur [6]. We aim to examine, and address, those concerns that have a collaborative nature. Such concerns include the involvement of a variety of actors resulting in a situation where multiple backgrounds, incompatible interests, and diverging areas of interest all have to be brought together to produce an acceptable policy result. Due to the collaborative nature of a policy-making process, its quality is greatly determined by a well-managed collaborative process. We look towards the field of collaboration engineering to be able to deal with such concerns. Collaboration engineering is concerned with the design of recurring collaborative processes using collaboration techniques and technology [7].

The main purpose of our paper is to establish a method for the realization of “good policies” in a collaborative process and how this process can be improved by the support of collaboration engineering. This will take the form of a generic design of a policy making process in terms of constructs from collaboration engineering, which has been arrived at using the action research approach. As a next step we will further elaborate this initial method using techniques from situational method engineering [8, 9], allowing us to introduce more parameterization of the method for specific situations.

The remainder of this paper is structured as follows. Section 2 briefly explains the concepts of policy, policy making processes and collaboration engineering. Section 3 provides a discussion of two case studies we have performed. Based on these case studies, section 4 discusses the design of our current policy making method. Finally, Section 5 provides the conclusion as well as a discussion on further research.
2 Policy making processes and collaboration engineering

The concept of policy has been defined by several researchers. Rose [10], defines a policy as “a long series of more-or-less related activities” and their consequences for those concerned rather than as a discrete decision. Rose’s definition embodies the useful notion that policy is a course or pattern of activity and not simply a decision to do something. Friedrich [11], regards policy as “a proposed course of action of a person, group, or government within a given environment providing obstacles and opportunities which the policy was proposed to utilize and overcome in an effort to reach a goal or realize an objective or a purpose.” To the notion of policy as a course of action, Friedrich adds the requirement that policy is directed toward the accomplishment of some purpose or goal. Although the purpose or goal of government actions may not always be easy to discern, the idea that policy involves purposive behavior seems a necessary part of a policy definition. Policy, however, should designate what is actually done rather than what is proposed in the way of action on some matter. Anderson [12], defines policy as “a purposive course of action followed by an actor or set of actors in dealing with a problem or matter of concern”. Anderson’s concept of policy focuses attention on what is actually done as against what is proposed or intended, and it differentiates a policy from a decision, which is a “choice among competing alternatives”. Eulau and Prewitt [13], define a policy as a “standing decision characterized by behavioral consistency and repetitiveness on the part of both those who make it and those who abide by it”. Whether in the public or private sector, policies also can be thought of as the instruments through which societies regulate themselves and attempt to channel human behavior in acceptable directions [14].

Taking into account the various perspectives of policy, and to put our research into context, we offer the following definition to help integrate them: a policy is a purposive course of action followed by a set of actor(s) to guide and determine present and future decisions, with an aim of realizing goals.

According to [6], the process of policy-making includes the manner in which problems get conceptualized and are brought to a governing body in order to be resolved. The governing body then formulates alternatives and select policy solutions; and those solutions get implemented, evaluated, and revised. Policy stages are thought of as a typology that completely describes policy decisions and actions that occur around a policy. The policy-making process “connotes temporarily, an unfolding of actions, events, and decisions that may culminate in an authoritative decision, which, at least temporarily, binds all within the jurisdiction of the governing body”. In explaining policy-making process, Sabatier says that the emphasis is much more on the unfolding than it is on the authoritative decision. In examining the unfolding, attention is devoted to structure, to the context and constraints of the process, and to actual decisions and events that occur. Dunn [15] defines policy-making process as “the administrative, organizational and political activities and attitudes that shape the transformation of policy inputs into outputs and impacts”. Even with the structured definitions of policy processes given, there is, it should be stressed, no one single process by which policy is made. Variations in the subject of policy will produce variations in the manner of policy-making. For
instance, taxation, railroad regulation, aid to private schools, and professional licensing, are each characterized by distinguishable policy processes [12]. Sometimes the phrase policy cycle is used to make clear that the process is cyclical or continuous rather than a one-time set of actions. Instead of a top-down listing of each stage, it could be presented as a series of stages linked in a circle because no policy decision or solution is ever final. Changing conditions, new information, formal evaluations, and shifting opinions often stimulate reconsideration and revision of established policies. In the real world these stages can and do overlap or are sometimes skipped. In other words, policies might be formulated before they are high on the political agenda; otherwise it would be impossible to differentiate policy formulation from legitimation.

Essentially, collaboration engineering revolves around the use of information and communication technologies to enable the collaboration between people. Although organizations have tried to collaborate in their organizational processes to achieve maximum value from their efforts, achieving effective team collaboration still remains a challenge. Collaboration is the degree to which people in an organization can combine their mental efforts so as to achieve common goals [16]. Because of this challenge, organizations have resorted to using groupware technologies in order for collaboration to work for them. However, technology alone seldom is the answer. What is needed is the design of effective collaboration processes. This can be achieved by following the collaboration engineering approach which is defined by [7] as “the design of re-usable collaboration processes and technologies meant to engender predictable success among practitioners of recurring mission-critical collaborative tasks”. In other words, collaboration engineering addresses recurring collaboration processes that can be transferred to groups that can be self-sustaining in these processes, using collaboration techniques and technology [17].

In collaboration engineering research, collaboration engineers need to follow standard, repeatable procedures to achieve predictable success with group processes. These procedures should enable people to move from one activity to another during collaboration, and they accomplish the activity by moving through some combination of patterns of collaboration [7]. Collaboration engineering researchers identified five general patterns of collaboration to enable a group to complete a particular group activity [7]: i) Diverge – to move from a state of having fewer concepts to a state of having more concepts. The goal of divergence is for a group to create concepts that have not yet been considered; ii) Converge – to move from a state of having many concepts to a state of having a focus on, and understanding of, fewer concepts worthy of further attention. The goal of convergence is for a group to reduce their cognitive load by reducing the number of concepts they must address; iii) Organize – to move from less to more understanding of the relationships among the concepts. The goal of organization is to reduce the effort of a follow-on activity; iv) Evaluate – to move from less to more understanding of the benefit of concepts toward attaining a goal relative to one or more criteria. The goal of evaluation is to focus a discussion or inform a group’s choice based on a judgment of the worth of a set of concepts with respect to a set of task-relevant criteria; v) Build Consensus – to move from having less to having more agreement among stakeholders on courses of action. The goal of consensus building is to let a group of mission-critical stakeholders arrive at mutually acceptable commitments.
The patterns of collaboration do not explicitly detail how a group could conduct a recurring collaboration process, especially with teams who do not have professional facilitators at their disposal. This can be achieved by the key collaboration engineering concept: the thinkLet. A thinkLet is defined by [7] as “the smallest unit of intellectual capital required to create a single repeatable, predictable pattern of collaboration among people working toward a goal”. ThinkLets can be used as conceptual building blocks in the design of collaboration processes. Some examples of thinkLets are provided in Table 1. More examples of thinkLets can e.g. be found in [18].

Table 1. Examples of thinkLets with their respective Collaboration Patterns

<table>
<thead>
<tr>
<th>ThinkLet Name</th>
<th>Collaboration Pattern</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>DirectedBrainstorm</td>
<td>Generate</td>
<td>To generate, in parallel, a broad, diverse set of highly creative ideas in response to prompts from a moderator and the ideas contributed by team mates.</td>
</tr>
<tr>
<td>BucketSummary</td>
<td>Reduce and clarify</td>
<td>To remove redundancy and ambiguity from broad generated items.</td>
</tr>
<tr>
<td>BucketWalk</td>
<td>Evaluate</td>
<td>To review the contents of each bucket (category) to make sure that all items are appropriately placed and understood.</td>
</tr>
<tr>
<td>MoodRing</td>
<td>Build Consensus</td>
<td>To continuously track the level of consensus within the group with regard to the issue currently under discussion.</td>
</tr>
</tbody>
</table>

3 Case study and evaluation

In this section, we present how our research was conducted and evaluated. We will do so in terms of a description of the research approach and cases involved. We also present a description of the generic method for collaborative policy-making, and relate this to the results of the case studies in the sections that follow.

3.1 Research approach

To develop and evaluate our method for collaborative policy-making, we followed the action research methodology process proposed by [19] where four activities that can be carried out over several iterations (in our case two) are involved. The ‘Plan’ activity is concerned with the exploration of the research site and the preparation of the intervention. The ‘Act’ activity involves actual interventions made by the researcher. The ‘Observe’ activity is where the collection of data, enabling evaluation, is done during and after the actual intervention. Finally, the ‘Reflect’ activity involves analysis of collected data and infers conclusions regarding the intervention that may feed into the ‘Plan’ activity of a new iteration.
We used action research because it permits highly interpretive assumptions to be made about observations; also the researcher intervenes in the problem setting, and it is performed collaboratively yet enhances the competencies of the respective actors [20]. In addition, we selected action research because it is an applied research method that can be tested in the field. Better still, it addresses the “how to” research questions. Our research aimed at developing and testing a method for collaborative policy-making, that is, a method of how to realize a quality policy in a collaborative effort. More so, the continuous design and evaluation of a method for collaborative policy-making may not be easy to study in a constructed setting. Lastly, action research allowed us to evaluate and improve our problem-solving techniques or theories during a series of interventions.

Based on the action research process described above, we executed the four activities as follows: In the ‘Planning’ activity, we conducted interviews with four organizations that have policy-making functions and also performed a literature review to understand organizational policy-making. The data collected formed the initial requirements for the generic method.

The ‘Act’ activity involved actual execution of the method in the field both in an industrial setting and an inexperienced environment. We applied the method for collaborative policy-making with two policy types in two case organizations:

- Case Organization 1 – it was used to observe the performance of the method in an industrial setting. A team of five experienced Information and Technology (IT) workers and involved in making policies for the Information Technology Department of the Ministry of Finance, Planning and Economic Development (MOFPED), Uganda used the method to develop an Information Technology (IT) policy for the department.
- Case Organization 2 – it was used as an inexperienced environment. A team of sixteen people comprised of two experienced IT workers involved in IT policy-making and fourteen Master’s Students (2nd year, Computer Science) at Radboud University Nijmegen (RUN), the Netherlands, used the method to develop a policy in the form of architectural principles for the student portal information system for RUN. The two experienced participants mainly assisted the students with the appropriate content.

To evaluate the performance and perception of the method for collaborative policy-making by the participants, we collected and analyzed explorative data during the ‘Observe’ activity. Three kinds of instruments, that is, observations, interviews and questionnaires comprising of qualitative and quantitative questions, were used for data collection. The tools enabled us to collect and analyze data regarding effectiveness, efficiency and policy stakeholders’ satisfaction with the method to improve the policy process and its outcomes; perceived policy elements identification; and the degree of applicability of the method.

Finally, in the ‘Reflect’ activity, our observations were analyzed with the aim of improving the method.
3.2 Method design for Collaborative Policy-making

This section presents the design of the initial method for collaborative policymaking. The method was designed following the collaboration engineering approach described in Section 2. Even though this approach comprises several design steps, the ones relevant to our research study included decomposing the method into collaborative activities, the classification of these activities into patterns of collaboration, selection of appropriate thinkLets to guide facilitation of the group during the execution of each activity as well as making the design method more predictable and repeatable. In the subsections below we give a description of the criteria we followed to evaluate the performance of the method, and a presentation of the final design of the method, respectively.

3.3 Evaluation criteria

The design of the method for collaborative policy-making was derived from two iterations based on selected design criteria. The criteria selection was derived from the goal of our research. Our research aimed at establishing a method for the realization of good policies in a collaborative policy-making process and how this process can be improved by the support of collaboration engineering. The following four criteria were considered by us:

- **Effectiveness** – the method for collaborative policy-making should enable stakeholders to achieve their goal.
- **Efficiency** – the method for collaborative policy-making should take stakeholders less time for attainment of the policy than without the use of a collaborative approach.
- **Degree of applicability** – the extent to which the method can be applied to varying policy types.
- **Perceived policy elements identification** – the method should enable stakeholders to have a common understanding of the policy elements (and their definitions).

4 Design Method

The method for collaborative policy-making was not designed from scratch. We based our design on method requirements derived from the explorative field study with four case organizations that have policy-making functions and also in concurrence with the policy process discussed by [21]. A typical policy-making process includes six stages [21]. However, our method design only involves the development/formation phase of the policy-making process. The method (development/formation phase) has two main parts: part 1 – pre-development/meeting phase, and part 2 – development phase.

The method underwent two iterations prior to deriving the final method design. The two iterations of the earlier versions of the method were applied in the two cases described above. The final method design is shown in Figure 1 in which we present
the steps required to develop/form a policy document, and the patterns of collaboration with related thinkLets used to guide the group to execute each step.

Fig. 1 Method for Collaborative Policy-Making
The method is divided into two main phases, as mentioned earlier on. It starts with the participants familiarizing themselves with each other and agreeing on the pre-development elements gathered in several earlier pre-meetings.

The participants familiarize themselves on these elements for the actual development of the policy. The elements comprise the problem to be solved; the relevant information to be used to develop the policy; a legal framework to support the policy to be developed; the ownership of the policy; leadership positioning; who are the stakeholders (internal and external); and technical resources for facilitation.

In the activity that follows, guided by the DirectedBrainstorm thinkLet, the participants are invited to brainstorm the mission objectives that they think would be relevant for the intended policy. The result from this activity is a brainstormed list of Policy Mission Objectives waiting for cleaning up.

In the next activity, and using the FastFocus thinkLet, all the participants are asked to organize the brainstormed public list displayed by extracting only the Mission Objectives that they feel are Key to the policy. They do this by grouping ideas and eliminating any redundancies. During this discussion, participants are allowed to also crosscheck to see if there is any important issue/Mission Objective that has not yet been posted on the public list. If this arises, a quick DirectedBrainstorm thinkLet followed by FastFocus thinkLet are performed. The result from this activity is a cleaned list of Key Policy Mission Objectives.

Based on the resulting Key Policy Mission Objectives, the participants are asked to identify and agree on common policy elements definitions that suit the Key Mission Objectives. This activity is guided by the DirectedBrainstorm thinkLet and followed by the FastFocus thinkLet. The result from this activity is a brainstormed list for policy elements. Using the FastFocus thinkLet, the participants organize the resulting brainstormed list as described in activity 2 above. They then reframe the extracted Key elements in a few words, while categorizing them into sections if needed, depending on the policy structure/format chosen by the participants. During this time, participants crosscheck to see if there is any important issue/policy element that has not yet been posted on the cleaned public list. If the need arises, again a quick DirectedBrainstorm followed by FastFocus is performed. The result of this activity is a cleaned list of Key Policy Elements.

The activity that follows involves defining the Key terms for each of the policy elements defined. Using the CouldBeShouldBe thinkLet, participants are asked to brainstorm terms that they ‘could’ consider as appropriate for each of the policy elements. Based on the resulting brainstormed list of terms per each policy element, participants are then asked to propose a term that they ‘should’ take as Key to each policy element. This exercise is continued until all the Key terms for each policy element are defined.

The activities above result into a Policy document. In this activity, and using the MoodRing thinkLet, participants are required to check if the policy document meets the desired objectives for which it was intended for. They do this by voting on a YES/NO basis, where a YES is voted if the elements definitions and terms meet the desired end states and a NO if it does not meet the desired end states, and therefore certain areas need to be re-addressed. A verbal discussion to address any issues raised is conducted until all the participants have reached some sort of consensus on the final policy document.
Finally, the participants need to plan how they will communicate the policy document to its intended users/owners. In this activity, they are required to draw up a policy awareness plan. Two ways are pre-determined that can be used, i.e. communication and education. Following the LeafHopper thinkLet, participants brainstorm about ways in which each of these can be addressed. The result of this activity is brainstormed lists of each awareness category. The resulting brainstormed lists for each awareness category are evaluated to determine if there is any issue that doesn’t belong to them respectively, at the same time removing any redundancies. This is achieved by using the BucketWalk thinkLet.

The evaluation of the method design for collaborative policy-making was implemented following a manual procedure. We used the Microsoft Word (MSWord) tool, an LCD projector, removable disks and voting sheets (paperbased) to implement the method. Results from the cases are presented in the section below.

4.1 Results

We now present the results from the two cases in which the method for collaborative policy-making was applied. We collected and analyzed data regarding effectiveness, efficiency, and participants’ satisfaction with the method to improve the policy process and its outcomes; perceived policy elements identification; the degree of applicability of the method.

Efficiency – We define efficiency of the method for collaborative policy-making as the degree to which policy-making stakeholders can reduce the amount of time required to attain a policy. To measure this, we considered the execution duration of each stage of the method; also how well the participants understood the method to execute the process tasks; and on the whole also considered the time it took the participants to come up with the final policy document and the awareness plan.

Based on our observations, we concluded that the method execution time was efficient. It took about an hour and fifteen minutes for execution in each of the workshops. This duration is comparable to the traditional way of policy formation, taking place under time pressure stemming from the fact that organizing participation in a policy procedure is hard and time consuming [6]. Even though the majority of the participants felt that the process execution was efficient, not all were happy with this time length; some required that more time should have been assigned to particular activities such as policy elements identification.

Policy formation effectiveness – Policy formation effectiveness is defined as the extent to which the method for collaborative policy-making enables policy stakeholders to achieve their goal.

We measured the effectiveness of the method by how well the participants managed to come up with a policy at the end of execution.

From our observations, it was noted that the participants effectively managed to form policies with respective awareness plans. This was demonstrated during the consensus stage. In this stage, participants were required to check if the policy document met the desired objectives for which it was intended for. They did this by voting on a YES/NO basis, where a YES was voted if the elements definitions and terms met the desired end states and a NO if it did not meet the desired end states.
Based on the feedback from the voting sheets (see Table 2), it was observed that the participants achieved fairly satisfactory results, that is, they managed to form a policy based on the desired end states. For those that voted a NO, a verbal discussion was held to re-address their issues. This increased consensus among the participants.

Table 2. Voting consensus results

<table>
<thead>
<tr>
<th></th>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case 1</td>
<td>4 (80%)</td>
<td>1 (20%)</td>
</tr>
<tr>
<td>Case 2</td>
<td>12 (75%)</td>
<td>4 (25%)</td>
</tr>
</tbody>
</table>

Having arrived at a complete policy document during the consensus stage, the participants also perceived it as having a common understanding of the policy elements identification.

Degree of applicability – We define this construct as the extent to which the method for collaborative policy-making can be applied to varying policy types. To measure this, we applied the method to two cases with different policy types. These included formation of an Information Technology policy, and Architectural Principles for an Information System. It was observed that the method was flexible in terms of its applicability in formation of two different types of policies.

Policy stakeholders’ satisfaction – To measure this construct, we used the 7-point Likert scale general meeting survey questionnaire where participants can strongly disagree to strongly agree. The instrument validation and theoretical underpinnings can be seen in Ref. 22. Results in Table 3 indicate that the participants were reasonably satisfied with the method outcomes, and the method by which the policies were formed.

Table 3. Satisfaction with method and outcome

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Satisfaction with method</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Score</td>
<td>4.800</td>
<td>3.838</td>
</tr>
<tr>
<td>Standard deviation</td>
<td>1.376</td>
<td>0.995</td>
</tr>
<tr>
<td><strong>Satisfaction with outcome</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Score</td>
<td>5.160</td>
<td>4.363</td>
</tr>
<tr>
<td>Standard deviation</td>
<td>1.310</td>
<td>1.094</td>
</tr>
</tbody>
</table>

The participants indicated that the results were useful to them as they gave better understanding of what issues they find important/key to the policy. They also observed this method as an all encompassing, interactive, efficient and better method of forming policies.
5 Conclusions and further research

This paper focussed on the development of an initial method for the creation of policies, using collaboration engineering to improve the quality of policymaking processes. We presented the results of two case studies conducted, regarding the use of collaboration engineering in the context of a policy making processes. Based on the results, the quality of the initial policy making method, in terms of its effectiveness, efficiency and applicability, proved to a satisfactory. As such, the collaborative method has indeed the potential to support organizations in developing quality policies.

As a next step, we aim to more explicitly rationalize design decisions taken in policy making processes (and associated method). We aim to do so by explicitly relating the goals of the policy making process (its why), the requirements on the process following from these goals (its what), the situation in which it needs to be executed (its within), to the construction of the policy making process/method (its how). In doing so, we will draw on past results concerning modeling processes [23, 24, 25, 26] and combine these with results from situational method engineering [8, 9]. A policy making process can essentially be regarded as a collaborative modeling process, where the model being produced is the policy.

Furthermore, we also intend to further elaborate the issue of perceived policy elements identification. The applicability and longevity of a policy document is highly dependent on a shared (and committed) understanding by all stakeholders involved, including those who are to execute the policy. We are currently using techniques from conceptual modeling [27, 28, 29] to more clearly exhibit the meaning of policies by grounding the underlying concepts and semantics (see [30] for an application of this idea to architecture principles). Our next step will be to integrate this grounding process into policy making processes, in particular the CouldBeShouldBe and FastFocus thinkLets of the process depicted in 1.

References


A Practitioner Based Method Tailoring Model for Information Systems Development

Timothy J. Burns¹ and Fadi P. Deek²

¹ Assistant Professor of Information Systems
Ramapo College of New Jersey
505 Ramapo Valley Road
Mahwah, NJ 07430
tburns1@ramapo.edu

² Dean of the College of Science & Liberal Arts and
Professor of Information Systems/Information Technology/Mathematical Sciences
New Jersey Institute of Technology
Newark, NJ

Abstract. Research has shown that information systems development is a highly circumstantial process and that no one system development methodology will be optimal for every context of every project [1, 2]. Several formal techniques such as the contingency factors approach and situational method engineering have been introduced in order to tailor a system development methodology to the project. However, there is evidence that system development practitioners have largely neglected these techniques in favor of ad hoc method tailoring approaches. This paper presents a formal method tailoring approach, based on the principles of general systems theory, geared towards the practitioner. The model is tested in a laboratory experiment and the results are reported.

1 Introduction

Over the years a plethora of information systems development methodologies have emerged. While many of these methodologies have served their purpose, research shows that system development is a highly circumstantial process, and that no one methodology will be optimal for every context of every project [1, 2]. Several formal techniques such as the contingency factors approach and situational method
engineering [3] have been introduced in order to tailor a methodology to the project. However, there is evidence that system development practitioners have largely neglected these techniques in favor of ad hoc method tailoring approaches [2, 4, 5]. Research has also shown that there is a divide between the IS development practices advocated and taught in academia and those used by practitioners [6, 7]. This paper presents a formal method tailoring approach geared towards the practitioner. A model based on the principles of general systems theory is described, which is aimed at the practitioners in the field seeking to adapt existing information system development methodologies to the circumstances of the project.

2 Preliminaries

2.1 Background

A system development methodology is defined as a recommended collection of phases, procedures, rules, techniques, tools, documentation, management, and training used to develop a system [1, 8, 9]. There have been significant advances and changes to methodologies over the last 30 years. Those changes can be characterized into specific eras including the pre-methodology era, when no methodologies existed, and the methodology era, when a plethora of new methodologies were introduced [8, 10]. Many in the Information Systems field feel that in recent years we have entered a post-methodology era wherein researchers and practitioners are questioning the older methodologies [8, 10]. Most of the serious criticism of the methodologies from the methodology era suggests that they are bureaucratic and labor intensive or “heavy” methodologies [10].

In response to this, new methodologies introduced in the post-methodology period are considered as lightweight or agile methodologies [10]. These agile methodologies are considered by many in this postmodern era to be “amethodological” (i.e., a negative construct connoting not methodological) [11]. The biggest criticism of the agile methodologies has been the lack of empirical evidence supporting the claims of their benefits and their lack of theoretical foundation [12]. However, there is a growing body of literature both supporting and repudiating the claims of success of the agile methodologies [12].

2.2 Problem Description

The research has pointed out three distinct areas of concern. First, no one methodology can claim to be the best for every software project [1, 2]. Second, present day developers are becoming increasingly discouraged with the traditional methodologies and their shortcomings [8]. Finally, those developers that have continued to use the traditional methodologies have informally modified and adapted them to meet the specificities of the project [2, 4].

Several formal method tailoring approaches have been introduced [2]. The contingency factors approach suggests that specific features of the development
context should be used to select an appropriate methodology from a portfolio of methodologies. This approach requires developers to be familiar with every contingent methodology or have contingency built in as part of the methodology itself.

A suggested alternative has been a technique called “Method Engineering” (ME) [2, 3]. With this technique, a methodology is constructed from a repository of “existing discrete predefined and pre-tested method fragments” [2]. Using a method-engineering tool, software developers build a meta-method that is made up of fragments from popular development methodologies. The fragments are each designed to handle a particular contingency inherent to the software project. The fragments are categorized as either product or process. Product fragments are artifacts capturing the structure in deliverables such as diagrams, tables, or models, while process fragments project strategies and detailed procedures [3].

Method Engineering has several shortcomings. For example, it is impossible to plan for every contingency that may arise, and therefore, critical fragments will always be missing [5]. Also, the burden of selecting the correct fragment falls upon the analyst [13]. Furthermore, a tool is usually required and ME tool development has been a problematic procedure [2]. Thus, evolution of software development methodologies using fragments is problematic.

Both contingency factors and ME techniques have had little success in practical industry applications [2, 5]. However, ad hoc methodology tailoring has been an implied concept for many years in industry [4].

3 The Model

3.1 Basis

It is hypothesized that a model can be created that will provide a formal process whereby practitioners can modify methodologies to the context of the project. The goal of the model is to provide practitioner utility. It is believed that one of the keys to finding a model that creates practitioner utility is to find one that is intuitive to the practitioner. Fitzgerald [4] demonstrated that practitioners will bypass the use of methodologies simply because they do not see the utility in using them. It is important that the model simulates the ad hoc informal method tailoring techniques currently employed by practitioners in the field.

The second condition that the model must meet is that it must be based on sound academic theory. In order to accomplish this, a root theory must be found that can be used to explain the model and its concepts. Furthermore, the model must be evaluated using an accepted methodology and the results must be reported in a statistically accepted manner.
3.2 Practitioner Utility

There are several theories and models that can be used to predict the degree to which an innovation will be accepted [14]. Included in this list would be the Diffusion of Innovations Theory [15], the Theory of Reasoned Action (TRA) [16], the Theory of Planned Behavior (TPB) [17], the Technology Acceptance Model (TAM) [18] and TAM2 [19].

TAM has been proven valid in numerous studies and under a multitude of conditions [14]. TAM suggests that when users are presented with a new technology, a number of factors influence the decision about how and when they will use it. The two primary factors are perceived usefulness (i.e., the degree to which a person believes that using a particular system would enhance his or her job performance) and perceived ease-of-use (i.e., the degree to which a person believes that using a particular system would be free from effort). The TAM2 model extends the TAM model to include social factors (i.e., subjective norm, voluntariness, and image) and cognitive factors (i.e., job relevance, output quality, and results demonstrability) [19].

Based on TAM2, in order for a practitioner to utilize a method tailoring approach, they must perceive it to be useful, easy to use, and socially and cognitively acceptable. Informal, ad hoc method tailoring meets these requirements given its widespread use in industry [4]. Therefore a formal method tailoring approach that simulates the, already accepted, ad hoc practitioner method tailoring approach, would also be accepted, provided it continues to meet the conditions put forth by TAM2.

Although the literature is insufficient on the question of how practitioners informally tailor methodologies in the field, there are some things that are known. First, practitioners generally take a shorter-term view than academics and tend to emphasize the completion of tasks and the solution of problems [7]. Second, the methodologies utilized by practitioners are influenced by the universality of the method, the method introduction process, the experience level of the developer, developer confidence in the methodology, and developer participation with the methodology [20].

Based on this information, in order for a formal method tailoring model to be utilized by practitioners, it must aid in the completion of tasks and the solution of problems. Also, it must provide universal applicability, have management support, provide utility to both experienced and in-experienced developers, and encourage developer confidence and participation.

3.3 Theoretical Foundation

The theoretical foundation for the practitioner model comes from General Systems Theory. Hungarian biologist Ludwig von Bertalanffy originally proposed general systems theory in 1928 [21] as a reaction against the reductionistic and mechanistic approaches to scientific study, and in an attempt to unify the fields of science. The scientific method is based on the assumptions that an entity can be broken down into its smallest components so that each component can be analyzed independently (reductionism), and that the components can be added in a linear fashion to describe
the totality of the system (mechanism). Rather than reducing an entity to the properties of its parts or elements, general systems theory focuses on the arrangement of and relations between the parts that connect them into a whole (holism).

One of the goals of general systems theory was to find common ground upon which scientific study could be conducted across all disciplines. Von Bertalanffy felt that it was futile to try and find a unitary conception of the world by reducing all levels of reality to the level of physics. He felt that the answer to a unitary conception could be found by defining the commonalities among the fields through the discovery of the isomorphy of the laws of the different fields [22]. Von Bertalanffy thought that the systems that are present in the various fields could identify those commonalities.

Von Bertalanffy defined a system as “complexes of elements standing in interaction”. He found that conventional physics dealt only with closed systems (i.e., systems which are isolated from their environment). In particular, the laws of thermodynamics expressly stated that they were intended for closed systems. The essence of the second law of thermodynamics (law of entropy) is that entropy (i.e., the degree of disorder or uncertainty in a system [22]) will increase over time in a closed system.

General systems theory realizes that many systems, by their nature, are open systems that interact with their environment. Von Bertalanffy observed that the second law of thermodynamics does not hold true in open systems. He realized that in an open system, the degree of disorder or uncertainty decreases over time or that “negative entropy” occurs [22]. General systems theory also realizes that open systems have a tendency to self-organize. This is a process in which the internal organization of a system increases automatically without being guided or managed by an outside source [23]. This happens through a process of feedback and decision-making.

An IS development methodology can be considered a “system” [22], that is used to develop an information system. IS development is also a problem solving process [24, 25]. This suggests that methodologies are essentially problem solving systems with several common elements including the problems (i.e., the difference between a goal state and the current state of the system [26], which have a hierarchical order [27], problem solving processes (i.e., the tools, procedures, processes, etc. that are used to do define and understand problems, plan solutions to problems, implement solutions, and verify and present the results [28], solutions (i.e., the answer to or disposition of a problem) [29], feedback (i.e., part of the output is monitored back, as information on the preliminary outcome of the response, into the input) [22], and an environment which defines the context, contingencies, constraints, rules, laws, etc. of the organization, people, technology, etc. These systems employ incremental problem solving which involves using intermediate states as intermediate goals in solving problems [30].

Based on general systems theory, IS development methodologies can be characterized as collaborative, hierarchical, incremental, and problem solving systems. They are open systems that interact with their outer environment [31], which means that they have the propensity for negative entropy. Also, these systems
all have a “system state” [32], that represents the current condition of system variables (such as the current number of open, unsolved problems in the system).

3.4 Model Definition

The contribution of this research is a model (i.e., an artifact used to abstract and represent phenomena) [26, 33] that provides a formal vehicle whereby practitioners can tailor, refine, augment, and combine methodologies. The model provides practitioner utility in that it aids in the completion of tasks and the solution of problems. Also, it provides universal applicability, provides utility to both experienced and in-experienced developers, and encourages developer confidence and participation [20].

Based on general systems theory, the model seeks to unify the methodologies, not by breaking the methodologies down into fragments, but by discovering the concepts that are isomorphic across the methodologies [22] and using those concepts to combine the methodologies. Discovering those isomorphic concepts requires abstracting the methodologies to a common level. The model suggests that the commonality among all methodologies is their inherent role as problem solving systems.

It must be pointed out that the principle of equifinality [22] holds true in the model. Equifinality is a condition in which different initial conditions lead to similar effects or in which different courses of action lead to similar results. Application of this principle suggests that there are multiple methodologies and instantiations that would fit the model and still produce the desired result.

The model in figure 1 represents a process that iterates among three phases throughout the life of the project. The “Describe” phase is used to understand the current state of the project. It is a knowledge producing activity [33]. It includes analyzing the current environment and identifying circumstances that have changed since the last definition phase, analyzing feedback that was obtained from the previous iteration, analyzing and parsing the list of problems still open at the conclusion of the cycle, and adding to the list any new problems that can be identified. The list of open problems is then broken down into sub-problems, which are prioritized and arranged in a hierarchy.

The “Problem Solve” phase is used to solve the highest priority problem in the list of open problems. If the problem is something simple, for instance a task that needs to be completed, then it can immediately pass to the next phase. However, if the problem is complex, then a problem-solving technique must be applied in order to collaboratively find a solution to the problem. The solution to the problem may be a methodology fragment. For instance, it may be determined that the best solution at this phase would be to build a prototype or to create an ER diagram.

The final phase is to “Prescribe”. This is a knowledge using activity [33]. Using the knowledge gained during the previous two phases the next course of action is prescribed. The problem that is solved by completing the action is now marked as a solved problem and the solution is recorded in the knowledge base.
A methodology that fits the practitioner model is illustrated in figure 2. Generally, it has a cyclical nature that involves refinement (to improve by eliminating unnecessary elements), and tailoring (to fit by exacting adjustments).

The process begins by selecting a base methodology with core competencies, (i.e., the set of the most strategically significant and value-creating skills in any organized system or person), that most closely match the context of the project and organization. Several key factors contribute to this selection process, for instance, the knowledge and background of the developers.

Once the base methodology has been selected, the next step is to extract the fragments from it that will serve as a skeleton methodology for the project. These fragments are arranged in a temporal fashion, with intentional gaps left in the prescribed process. This is represented by the shaded base fragments in figure 2.

The methodology now progresses into a cyclical process. The process includes the phases to describe, problem solve, and prescribe as defined by the model. The methodology continues to follow this cycle throughout the course of the project. The base methodology fragments that were initially extracted as the skeleton methodology serve as anchor points which keep the project grounded. The prescribed actions must be collated within the fragments of the base methodology that were initially prescribed. The methodology can continue to be employed throughout the lifecycle of the project, even after the project has progressed into the maintenance phase.

![Fig. 1 A practitioner model for method tailoring](image-url)
3.5 Advantages of the Model

The goal of the model is to provide practitioner utility. The model reaches that goal by presenting a simple process that is intuitive to the system developer and simulates the developer’s typical procedure. This puts the model in accordance with the conditions set forth by the technology acceptance model.

Furthermore, the model is based on a sound academic theory and thus presents an anti-reductionistic and anti-mechanistic approach. It seeks to integrate by identifying...
the isomorphic characteristics of the IS development methodologies. In particular, the model capitalizes on the common inherent problem solving nature of the various methodologies.

The inadequacies of the contingency factors approach are apparent [2]. It is just not feasible or possible for all the developers in an organization to be familiar with all of the possible methodologies that would work best for a given situation [2]. Plus as the contingent factors of the project change over time, so will the optimum methodology.

If method engineering is analyzed through the lens of general systems theory, it becomes apparent that it is both a reductionistic and mechanistic solution to the problem. It is reductionistic in the sense that it attempts to solve the problem by reducing the phenomenon (the methodology) to its smallest component (method fragments) and analyzing the components. It is mechanistic because it attempts to build a whole meta-methodology from the sum of its parts, with no regard for the interrelationships of those parts.

4 Research Methodology

It was decided that a two stage approach would be used to evaluate the model. In stage one, reported in this paper, a laboratory experiment was conducted to establish the model as a viable system development process. The second stage, to be completed in future research, will be to assess the utility of the model in the field. This will be accomplished through practitioner based analysis.

As previously stated, there are currently three well known method tailoring approaches; method engineering, contingency factors, and practitioner ad-hoc. Prior research has already shown the advantages of method tailoring over individual methodologies [1, 2]. Therefore, In order to evaluate the model, it had to be compared to one of the three well known method tailoring approaches. Given the constraints of a lab experiment, the logical choice was to compare it to practitioner ad-hoc method tailoring.

The experiment was crafted in such a way as to force the subjects in all treatments to go through a method tailoring process. The subjects were given a series of tasks that represented fragments from various methodologies. They were also asked to submit a completed system design project that was a culmination of the various independent tasks.

One set of subjects acted as the experiment group and used the practitioner model to complete the tasks and one set of subjects acted as the control group and were allowed to use an ad hoc approach. Half of the subjects in each group were given a significant change to the requirements late in the process. This resulted in a “2x2” factorial design to the experiment. Subjects were randomly assigned to teams of three and then randomly assigned to a treatment.

Steps were taken to ensure that the experiment group would not have an advantage over the control group. Subjects in all treatments were given the same introduction and the same level of instruction covering information systems development techniques and their commonality with the problem solving process,
including open problems, problem analysis, and solution design. All subjects were also introduced to three commonly used problem solving methods (Polya's method, brainstorming, and SWOT analysis). The experiment group was asked to use an instrument that explicitly prompted the participants to adhere to the process of the practitioner model.

Subjects’ satisfaction with the development process, the finished design, and their group’s problem solving capabilities was measured using questionnaires. Two blinded, expert judges were used to rate the final project submitted by the subjects.

There were twelve hypotheses, stated as follows:

- **H1a**: Developers will be more satisfied with the finished system design when it is developed using the practitioner model than when it is developed using an ad hoc method tailoring approach.

- **H1b**: When system requirements remain constant throughout the development process, developers will be more satisfied with the finished system design when it is developed using the practitioner model than when it is developed using an ad hoc method tailoring approach.

- **H1c**: When system requirements change late in the development process, developers will be more satisfied with the finished system design when it is developed using the practitioner model than when it is developed using an ad hoc method tailoring approach.

- **H2a**: Developers will be more satisfied with their problem solving capabilities using the practitioner model than when they are using an ad hoc method tailoring approach.

- **H2b**: When system requirements remain constant throughout the development process, developers will be more satisfied with their problem solving capabilities using the practitioner model than when they are using an ad hoc method tailoring approach.

- **H2c**: When system requirements change late in the development process, developers will be more satisfied with their problem solving capabilities using the practitioner model than when they are using an ad hoc method tailoring approach.

- **H3a**: Developers will be more satisfied with the development process when using the practitioner model than when using an ad hoc method tailoring approach.

- **H3b**: When system requirements remain constant throughout the development process, developers will be more satisfied with the development process when using the practitioner model than when using an ad hoc method tailoring approach.

- **H3c**: When system requirements change late in the development process, developers will be more satisfied with the development process when using the practitioner model than when using an ad hoc method tailoring approach.

- **H4a**: Expert judges will rate the finished system design better when developers use the practitioner model to develop the system than when developers use an ad hoc method tailoring approach.
• H4b – Expert judges will rate the finished system design better when system requirements remain constant throughout the development process and developers use the practitioner model than when system requirements remain constant throughout the development process and developers use an ad hoc method tailoring approach.

• H4c - Expert judges will rate the finished system design better when system requirements change late in the development process and developers use the practitioner model than when system requirements change late in the development process and developers use an ad hoc method tailoring approach.

5 Results

Data on the use of the CHIPS model was gathered over a six-month period. A total of 140 subjects participated in the study. There were two primary instruments used to collect the data. The subjects acting as developers completed a questionnaire and the judges who rated the finished projects used a standardized grading sheet to compute the scores. Cronbach’s alpha was used to validate the questionnaire. The judges’ scores were validated using the paired two-sample t-test and the Pearson r calculation of the correlation coefficient between the two judges. As each hypothesis was a bi-variable statement, the t-test was selected as the appropriate test of significance to compare the means of the two variables analyzed.

The tests of significance show that the following hypotheses were supported (as summarized in table one):

Table 1. Summary of Hypotheses Supported and Not Supported

<table>
<thead>
<tr>
<th>Hypothesis</th>
<th>Description</th>
<th>Supported/Not Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>H1a</td>
<td>Overall Developer Satisfaction with Design</td>
<td>Not Supported</td>
</tr>
<tr>
<td>H1b</td>
<td>Developer Satisfaction Design / No Change</td>
<td>Supported</td>
</tr>
<tr>
<td>H1c</td>
<td>Developer Satisfaction Design / Late Stage Change</td>
<td>Not Supported</td>
</tr>
<tr>
<td>H2a</td>
<td>Overall Problem Solving Satisfaction</td>
<td>Not Supported</td>
</tr>
<tr>
<td>H2b</td>
<td>Problem Solving Satisfaction / No Change</td>
<td>Supported</td>
</tr>
<tr>
<td>H2c</td>
<td>Problem Solving Satisfaction / Late Stage Change</td>
<td>Not Supported</td>
</tr>
<tr>
<td>H3a</td>
<td>Overall Developer Satisfaction with Process</td>
<td>Not Supported</td>
</tr>
<tr>
<td>H3b</td>
<td>Developer Satisfaction Process / No Change</td>
<td>Supported</td>
</tr>
<tr>
<td>H3c</td>
<td>Developer Satisfaction Process / Late Stage Change</td>
<td>Not Supported</td>
</tr>
<tr>
<td>H4a</td>
<td>Overall Judges Rating</td>
<td>Supported</td>
</tr>
<tr>
<td>H4b</td>
<td>Overall Judges Rating / No Change</td>
<td>Not Supported</td>
</tr>
<tr>
<td>H4c</td>
<td>Overall Judges Rating / Late Stage Change</td>
<td>Supported</td>
</tr>
</tbody>
</table>
6 Conclusion

An analysis of the results obtained for the first dependent variable, developer satisfaction with the final design, shows that there were no significant differences between the practitioner model and ad-hoc method tailoring developers when their satisfaction was measured at the aggregate level. However, when the results were analyzed against the factor of whether late stage requirements changes were introduced, there were some significant differences. In particular, when late stage changes were not introduced, the practitioner model developers were more satisfied with their design than those developers who did have late stage changes introduced.

At first, this result appears to be contrary to the intent of the practitioner model. The practitioner model is meant to make developers more satisfied with their finished products, particularly when there is uncertainty and change involved. The explanation for the results demonstrated here may be found in the deficiencies of the experiment. In particular, the fact that university students were used as subjects.

The experiment required that half the subjects be assigned a change in the requirements of the project late in the project development process. The changes came at a time when the subjects were close to finishing the project and were a total surprise to the subjects. While a seasoned system developer would probably know that this is a common occurrence in a typical system development project, students would not. Students would look at the late stage change as an unmitigated burden and as an act of deception perpetrated upon them by their instructor. They had been told that they had to do x and now they also had to do y. This argument is bolstered by some of the emails the researcher received from subjects who had received the late stage change. Many of the subjects were very angry at the fact that the requirements had changed (and many took the time to express that anger).

It is believed that the anger of the subjects who received the late stage change outweighed any increased satisfaction they may have gained by using the practitioner model. This skewing of the results is reflected in all three dependent variables that measured developer satisfaction. Furthermore, it is believed that the impact of the decreased satisfaction of the late stage change subjects was so strong, that it affected the aggregate numbers that demonstrated overall satisfaction of the practitioner model versus ad-hoc method tailoring developers.

Further weight can be made to this argument when the results of not introducing the late stage change are analyzed. With no late stage change, and thus no reflective anger, the practitioner model developers are clearly more satisfied with their finished designs than ad-hoc method tailoring developers.

The second dependent variable measured, developer satisfaction with their problem solving process, returned the same results as the developers satisfaction with their finished designs. Developers were not more satisfied with their problem solving process when this variable was measured overall showing the practitioner model developers versus ad-hoc method tailoring developers and when late stage changes were introduced to the project. However, consistent with the previous variable, when no late stage changes were introduced, the practitioner model developers were significantly more satisfied with their problem solving process.
The third dependent variable, developer satisfaction with the development process, returned the same results as the previous two variables. Overall, the practitioner model developers were not more satisfied with the development process than ad-hoc method tailoring developers and were significantly less satisfied when late stage changes were introduced to the project. However, consistent with the previous variables, when no late stage changes to the requirements were introduced, the practitioner model developers were significantly more satisfied with the development process than ad-hoc method tailoring developers.

The final dependent variable that was measured was the expert judges’ rating of the finished designs. It is felt that the results obtained through the measurement of this variable provide the strongest and clearest indicator of the outcome of this research. This is because the limitations of the experiment, which were outlined previously, are mitigated by the results obtained for this variable.

The expert judges were not affected by the factors that caused the results obtained from the other variables to be somewhat skewed. They had to do the same amount of work to grade each and every project, no matter what process the developers used. Furthermore, the expert judges were blinded as to what treatment the projects they were grading had been exposed to. The judges developed the grading criteria themselves and then used that grading criteria to grade the projects.

The results obtained from the measurement of the expert judges’ ratings of the projects are clearly dissimilar to the results obtained from the other variables. The expert judges rated the finished system designs significantly higher when developers used the practitioner model to develop the system than when developers did not use the practitioner model. The judges also found that when late stage changes were introduced to the developers, the developers created statistically significantly higher rated projects when they used the practitioner model versus when they did not use the practitioner model. This would make sense as the essence of the practitioner model is to make the developers better able to adapt to change and thus provide a better product when there is change late in the development process. When there were no late stage changes for the developers, the practitioner model developers’ projects were still ranked higher than the ad-hoc method tailoring developers’ projects, however the difference in the scores was not of a magnitude that would constitute a statistical significance.

The results of this initial research, while promising, are significant for several reasons. First, mean scores were higher for the practitioner model developers than they were for ad-hoc method tailoring developers for almost all of the dependent variables. This can tentatively be used as an indicator that the practitioner model has a role as a legitimate model. Second, there were areas that showed a statistically significant improvement of the practitioner model developers over ad-hoc method tailoring developers. Given the limitations of the experiment elaborated earlier, this could be an indicator that the practitioner model has the potential to improve the development process. Finally, the results of this experiment give support to the argument that the practitioner model can be used to integrate several system development methodologies, not by breaking those methodologies down into method fragments, but by identifying the isomorphic characteristics of those methodologies.

The practitioner model presented in this research directly addresses the problems inherent with other development methodology adaptation approaches. This general
systems approach facilitates an IS community effort to normalize system development methodologies. The adherence to design science guidelines lends itself to the legitimacy of the model. Practitioners who use this method will not have to learn methodologies that are not normalized. Thus, they will have a shorter learning curve to implement this technique versus the other method tailoring techniques. Our research community can work collaboratively to reduce ambiguity in methodologies by using the theoretical foundation presented here.

Future research is needed in several areas. Field experiments are needed that will test the practitioner model in a more realistic setting and against other methodology tailoring approaches. Also, specific methodologies and instantiations of the practitioner model need to be developed and evaluated accordingly.

References

Utilizing Theories to Reduce the Subjectivity of Method Engineering Processes

Andreas Gehlert and Daniel Pfeiffer

1 Technische Universität Dresden, Helmholtzstr. 13, 01069 Dresden, Germany, gehlert@wise.wiwi.tu-dresden.de, WWW homepage: http://wiseweb.wiwi.tu-dresden.de
2 European Research Center for Information System, University of Münster, Leonardo-Campus 3, 48149 Münster, Germany, pfeiffer@ercis.de, WWW homepage: http://www.ercis.de

Abstract. The objective of this paper is to utilize theories to reduce the subjectivity in method engineering processes. Since the support for the development of novel method fragments is especially weak, we concentrate on this method engineering aspect only. We show how theories can be incorporated into an existing method engineering process to guide method fragment development. We argue that theoretical support allows predicting the success or failure of a method and reduces the subjectivity in the method engineering process. One result of the paper is a proposal for a process model, which utilizes theories. Furthermore, we show that our approach is feasible by analyzing the usability of existing theories for the method engineering discipline.

1 Introduction

Although conceptual modeling is considered as the core of the Information Systems (IS) discipline by several researchers [1, 2], Patel et al. [3, p. 374] found in an empirical study that designers have failed to use this tool in practice. Patel et al. [3] concluded that difficulties in reconstructing reality with the modeling grammar led to reduced confidence in the value of conceptual modeling as such. Theories of conceptual modeling might help to overcome the problem of representing reality and to guide practitioners in the modeling process [4-6].

Modeling methods used in conceptual modeling projects are artifacts created in a method engineering process. In a broader sense, the method engineering discipline is concerned with the selection, adaptation and design of situation specific modeling methods. The underlying assumption of this research is that there is no universal
method, which can be used in all situations [7-12]. Therefore, it is necessary to create and/or to tailor a method to the specific characteristics of the situation [13-15]. A method is not considered as a single monolithic bloc but rather as consisting of a set of fragments [15-18], which differ in terms of their scope and their granularity [12, pp. 403].

Method engineering approaches can be distinguished based on their starting point. Ralyté et al. [19] describe four different approaches for method engineering projects (cf. Fig. 1):

1) The **assembly-based approach** reuses method fragments to construct a new method [13, 15-17, 20]. This approach assumes that these fragments have been detached from existing methods, provided with a description and stored in a method base. Based on the specific characteristics of a project these fragments can be selected from this repository and assembled by following predefined rules.

2) The **paradigm-based approach** takes a meta-model that belongs to a certain theoretical framework as starting point [21]. This meta-model is specialized, abstracted or adapted according to the objective of the project [20, 22].

3) The **extension-based approach** focuses on an existing method and provides novel additions to it. The objective of this approach is to enhance a method with new features that are helpful to meet the requirements of the project. Examples for the extension-based approach are the enrichment of activity diagrams with temporal aspects [23] or an extension of class diagrams to model web applications [24].

4) The **ad-hoc approach** is concerned with the construction of a novel method ‘from scratch’ [19]. This strategy is required when necessary method fragments or meta models are not available. This can be the case when the project deals with a new application domain that is not yet covered by a specific method or when the project characteristics differ significantly from former situations.

Fig. 1. Process model for method engineering (adaptation from [21, p. 97])
Approaches one to three take parts of a method as given and deal with their recombination or adaptation. However, it remains unclear how to initially construct method fragments, how to find the relevant constructs and how to combine them. Consequently, we focus on the ad-hoc approach as it is the only one that considers how a method fragment is actually built.

The construction of a method fragment is a design process [25, p. 45]. As design artifacts are inherently subjective [26], we argue that the usage of theories as source of shared knowledge reduces the subjectivity of method engineering processes.

Typical design decisions in a method engineering project include the question whether a distinction between types and properties should be made, whether optional properties should be allowed, or whether inheritance hierarchies should be modeled upside-down. Without the existence and application of a generally accepted theory, these design decisions depend on the (subjective) knowledge of the method engineer. Consequently, the construction of the method fragment lacks scientific rigor.

Traditional IS theories such as the technology acceptance model [27], general systems theory [28] or contingency theory [29] are on a granularity level, which is too abstract to provide answers to these fine-grained questions. However, if traditional IS theories are too general to be useful for a theory-guided method engineering project the question of the feasibility of the approach arises. Hence, in the course of this paper we address the following research questions:

1. How can a method engineering process be guided by theory? We aim at enriching the generic method engineering process in the case of an ad-hoc strategy with theoretical knowledge (cf. Fig. 1). Thus, we propose to derive appropriate design decisions which are traceable and justified by empirical studies.

2. What are possible theoretical foundations of method engineering? This question addresses the feasibility of our approach. We show that empirical research from the ontological discipline (subsection 3.1) and research from cognitive science (subsection 3.2) has the granularity, which is suitable for method engineering processes.

The paper proceeds as follows: In the next section we show how theoretical findings can be incorporated into method engineering processes and discuss briefly their impacts. In section three we describe possible theoretical foundations of method engineering from a conceptual and representational perspective. We derive hypotheses and provide the corresponding empirical findings. The paper closes with a summary of the main results and an outlook to future research.

2 Theory-Guided Method Construction

Concrete process models that describe how method fragments are actually built are very rare. One of the most developed processes was introduced by Greiffenberg [30]. He argues that the construction of a meta-model comprises five different phases: Firstly, the concepts of the modeling grammar are defined and described in a concept dictionary. After a validation of these concepts, existing meta-model fragments are selected for the new meta-model. The last two steps comprise the
construction of the meta-model (e.g., the mapping of the concepts to concrete constructs) and the validation of this meta-model [30, pp. 166].

The meta-model construction process proposed by Greiffenberg is a classical design science process [e.g., 25], which emphasizes the evaluation of the latter artifact. However, it has been argued elsewhere that this evaluation is difficult because of the inherent subjectivity in design science processes [e.g., 26, p. 9]. The opponents of classical design science processes propose to incorporate theories.

We consider a theory as a set of universally quantified statements [31, p. 202]. It is composed of hypotheses that strive to explain certain behavior or structure of the world. A theory requires an empirical evaluation to prove its adequacy.

The reasons for theory usage in design processes are threefold. Firstly, using established theories in designs enables to predict its failure or success. Secondly, implementing theory-based designs can be used to evaluate the implemented hypotheses [26, p. 9]. Thirdly, the resulting increased rigor in the method engineering discipline helps to reduce the currently existing method jungle [32, p. 365].

The first argument is especially interesting for the method engineering discipline, since its methods are usually used in the early stages of the systems development cycle. Boehm argues that errors made in these early phases lead to exponentially higher costs [e.g., 33]. Thus, the prediction of failure or success of a modeling grammar is expected to result in high payoffs (similar arguments in [6]).

The first question which arises when designing modeling methods is how and what theories should be used. The decomposition of a method into its fragment can help to answer this question. It is generally agreed that a modeling method comprises at least two different fragments, the modeling grammar fragment and the process model fragment [e.g., 12, pp. 403]. A modeling grammar in turn can be decomposed into a set of constructs described with a meta-model of the abstract syntax and a notation described with a meta-model of the concrete syntax. To provide theoretical guidance for the construction of the modeling process is beyond the scope of this paper.

Since Wand and Weber conclude that conceptual modeling lacks a coherent theory [34, p. 2], the question arises whether our approach is feasible if existing theories do not cover all aspects of the modeling grammar under design. If this is the case, we argue that new hypotheses should be defined. These new hypotheses reflect the decisions made in the design science process. The conditional form of these statements allows their empirical testing. Thus, using hypotheses to describe design decisions is an extension to what is known as method construction rationale in the literature [35, p. 373].

The second research question addresses the feasibility of the approach. The approach can only be defended if there are existing theory fragments, which are ready to use in the method engineering discipline. This discussion is addressed in the next section. Our arguments presented so far are summarized in Fig. 2.
3  Theoretical Foundations of Method Engineering

The aim of this section is to show that there are empirically tested theory fragments, which have a suitable granularity to be used in a method engineering process. Since a modeling grammar consists of a conceptual aspect (abstract syntax) and a representational aspect (concrete syntax / notation), we investigate existing fragments in both areas.

3.1  Hypotheses about Conceptual Aspects of Modeling Grammars

A possible source for hypotheses about conceptual aspects of modeling grammars is ontological research. Ontological research strives to guide the creation of modeling grammars with the help of the basic categories of reality. Ontological analyses of modeling grammars have been mainly performed based on the Bunge-Wand-Weber (BWW) Ontology [e.g. 36, 37], Chisholm Ontology [e.g. 38], and the General Ontological Language (GOL) [e.g. 39, 40]. In the following we will focus on the BWW Ontology as an example for ontological research as it offers the richest set of empirical data.

Most recently, the general debate continued whether ontologies should be used as theories in the analysis, design and implementation of modeling grammars. While the opponents of this approach argue “… that the project of developing theoretical foundations of conceptual modeling on the basis of philosophical ontology is neither feasible nor defensible” [41, p. 74], the proponents mainly put forward the
interesting results achieved so far with this approach. In this paper we see an ontology as an artifact from which empirically testable hypotheses can be derived.

Empirical studies in the BWW domain mainly address the so-called representation model [42–47]. To the best of our knowledge, there is only one study testing the good decomposition model [48], and no empirical studies on the state tracking model. In the interest of brevity, we focus on the representation model only.

The main aspect of the representation model of the BWW is to propose a set of hypotheses for modeling grammars. It assumes that models created with modeling grammars, which contradict the hypotheses derived from the BWW representation model, are more difficult to interpret. In this case the modeling grammar is said to be ontologically deficient [36, pp. 226]. The hypothesis of the representation model can therefore be formulated as follows:

H 1: Models of an ontologically deficient grammar are more difficult to understand than models of ontologically clear grammars.

To formulate more precise hypotheses, which can be empirically tested, the scope of H 1 must be limited in a deductive way. There are two possibilities to achieve this limitation: Firstly, the term “understanding” can be clarified and tightened. Secondly, the coverage of the BWW representation model can be restricted by concentrating on a specific part of the ontology.

Authors used three different approaches to specialize the term understanding:

1. Weber [42] interpreted understanding as comprehension, e. g. the proportion of a model, which participants in his study remembered in a free recall experiment. Gemino and Wand [46] operationalized the term similarly and used a cloze test to measure comprehension.

2. Burton-Jones and Weber [43], Bodart et al. [44], Parsons and Cole [47], and Gemino and Wand [46] used problem solving measures to indicate how well a model is understood. In contrast to comprehension questions where surface-level understanding is sufficient, answers to problem solving questions require deep-level understanding that needs to be applied to the respective question.

3. Burton-Jones and Weber [43] and Bodart et al. [44] also used the measure “perceived ease of understanding” with an instrument, which Davis et al. initially developed in the technology acceptance model research [49]. Similar measures were used by Burton Jones and Weber ([43], "perceived ease of use"). While comprehension and problem solving questions are directly related to the model-user’s knowledge of the model, the perceived ease of understanding and the perceived ease of use instruments investigate the user’s subjective evaluation of the utility of the respective models. However, the “perceived” measures were not significant in any experiment [43, 46, 48].

Early empirical work in the BWW area traces back to a 1996 paper from Weber (cf. table 1). He provides empirical evidence that users of conceptual modeling grammars distinguish between properties and things [42]. He concluded that modeling grammars, which do not distinguish between these concepts, should not be used for conceptual modeling. In a later paper Shanks et al. [6] were able to show...
that even if a modeling grammar offers the possibility to distinguish between these concepts, they might not be used appropriately. In this case, the authors provide empirical evidence that the resulting models are more difficult to understand. As a consequence, the distinction between properties and things should be carefully implemented into conceptual models (cf. Table 1 and Fig. 3).

Table 1: Overview of empirical studies about the BWW Ontology

<table>
<thead>
<tr>
<th>Hypotheses</th>
<th>Implications for Method Engineering</th>
</tr>
</thead>
<tbody>
<tr>
<td>H 1.1 Models are easier to comprehend if things and properties are clearly distinguished [6, 42].</td>
<td>Modeling grammars should distinguish between properties and things by providing separate constructs for both concepts.</td>
</tr>
<tr>
<td>H 1.2 If properties of relations are avoided, the models are easier to comprehend [48].</td>
<td>A modeling grammar should not include the concept of properties of relations.¹</td>
</tr>
<tr>
<td>H 1.3 If properties of relations are avoided, problem solving performance increases [43].</td>
<td></td>
</tr>
<tr>
<td>H 1.5 If optional properties are avoided in conceptual modeling, deep-level understanding as well as problem-solving performance increases [44, 46].</td>
<td>Modeling grammars should provide the possibility to model optional properties. Optional properties should only be used if a surface-level understanding is required.</td>
</tr>
<tr>
<td>H 1.6 If optional properties are used in conceptual models, surface-level understanding increases [44].</td>
<td></td>
</tr>
<tr>
<td>H 1.7 Problem solving capabilities are higher, if composites are represented explicitly in conceptual models [50].</td>
<td>A modeling grammar should provide a separate construct to express composites. When applying this modeling grammar, users should refrain from using associations to express composites.</td>
</tr>
</tbody>
</table>

Another stream of research deals with the question whether properties should have properties. This discussion started in 1999 when Burton-Jones and Weber \[43\] showed in an experiment that models, which use the concept of properties for relations (property of a mutual property), are not as useful for problem solving than models, which avoid properties of relations. However, the perceived ease of use measure of a model was not affected by this concept. In a later study Burton-Jones and Meso \[48\] showed that models are more difficult to understand if properties of relations are used. Both studies suggest that properties of relations should be generally avoided in conceptual models and its modeling grammars.

Following up this debate, Bodart et al. \[44\] investigated the usage of optional properties. They found in an experiment that surface-level comprehension increases if optional properties are allowed in conceptual models. However, problem solving capabilities increase if optional properties are avoided. The latter findings were replicated by Gemino and Wand \[46\]. Both studies suggest that the concept of optional properties should be implemented into modeling grammars. However, if

¹ Cardinalities in the BWW domain are not properties of relations but describe state laws and are, thus, allowed for any relation.
deep-level understanding or problem solving is required, the concept should not be used in models.

Shanks et al. [50] investigated the role of part-whole relations in conceptual models. In an experiment they found empirical evidence that composites should be expressed as separate things (e.g., separate entity-types). The alternative representation as association leads to reduced problem solving capabilities of the model users. Consequently the modeling grammar should preferably offer a separate symbol to represent part-whole relations and this symbol should be carefully used in models.

Fig. 3 summarizes the dependencies between the hypotheses discussed so far.

![Dependencies between the BWW hypotheses](image)

### 3.2 Representational Theories

Representational theories build upon the findings in cognitive psychology. To the best of our knowledge, there is no general accepted framework, such as ontologies at the conceptual level, in which these research projects can be embedded. Consequently, we show only that there is theoretical support for the representation of modeling grammars as well. We do not aim to set up a framework for this research stream nor do we claim that the studies cited here are complete.

The main hypotheses why diagrammatic representations of modeling grammars should be used at all were given by Larkin and Simon [51]. The authors argue that the two-dimensional representation of graphs lead to a visual grouping of elements that enhance human information processing and problem solving. However, the authors did not investigate which representational properties lead to a “good” diagram.

This gap was filled by a series of papers. As in subsection 3.1, the question what a “good” representation is, needs to be clarified before the empirical study. Different operationalizations were used in the literature including:

- **Model correctness**: In a study of Batra et al. the authors investigated whether a diagram leads to higher model correctness compared to a textual notation [52, p. 130].

---

Optional properties avoided

Distinction between things and properties

Hypothesis

Conceptual hypothesis

Properties of relations avoided

Ontological completeness
Problem solving: In another study Kim et al. [53] investigated how a diagrammatic representation influences the problem solving capabilities of the diagram’s users.

Comprehension: The last study covered here was conducted by Purchase et al. [54]. The authors investigated how different forms of representation of inheritance hierarchies, associations and cardinalities affect the comprehension of the diagram.

The findings of the selected studies are summarized in Table 2 and Fig. 4 and are described in more detail below.

Table 2: Overview of empirical studies of model representations

<table>
<thead>
<tr>
<th>Hypotheses</th>
<th>Implications for Method Engineering</th>
</tr>
</thead>
<tbody>
<tr>
<td>H 2.1 If a diagrammatic representation of a modeling grammar is used, the resulting models have a higher correctness [52].</td>
<td>Modeling grammars should offer a diagrammatic representation.</td>
</tr>
<tr>
<td>H 2.2 If visual cues and an overview diagram over the complete model are used in models, which consist of more than one diagram, the problem solving capabilities are higher than in models without these instruments [53].</td>
<td>Modeling grammars should provide means to represent visual cues between different diagrams as well as means to construct an abstract overview over the entire model. If models consist of more than one diagram, the visual cues provided by the grammar should be used. Additionally, an overview-model should be added.</td>
</tr>
<tr>
<td>H 2.3 If inheritance hierarchies are modeled upside-down, the model comprehension capabilities increase [54]. If inheritance hierarchies are represented as hyperedges the model comprehension capabilities increase. If the name label of associations does not interfere with the association line, the model comprehension capabilities increase. If cardinalities are annotated using a min..max notation, the model comprehension capabilities increase</td>
<td>The empirical findings have no effect on the modeling grammar as long as the grammar enables the representations proposed by Purchase et al. Modelers should model inheritance hierarchies upside-down and as hyperedges. Association labels should generally not interfere with the association line and the min..max notation of cardinalities should be preferred.</td>
</tr>
</tbody>
</table>

Early work on the influence of the representational form on models traces back to a paper of Batra et al. The authors investigated whether diagrammatic notations such as ER-diagrams should be preferred over textual grammars such as the relational model. Participants in a laboratory experiment had to create a model according to a natural language description using a diagrammatic representation (ER-model) and a textual representation (relational model). The authors compared the models with their textual description and recorded the correctness of the respective diagrams [52, pp. 130]. The authors found that ER-models scored significantly higher than the
relational model. Consequently, representing a *modeling grammar* diagrammatically leads to models, which are more correct than textually represented models.

In the study of Kim et al. the authors investigated the common situation of representing models with a set of diagrams rather than with a single diagram only. The authors investigated different integration forms of these diagrams. They found that the problem solving performance was significantly higher in those models, which utilized visual cues between the diagrams and provided a separate diagram representing the overview over the complete model [53]. Consequently, *modeling grammars* should provide diagrammatic means to model visual cues between different diagrams. Additionally, the *grammar* should provide constructs to represent and overview of the entire model.

In another study Purchase et al. [54] investigated the effects of different representations of inheritance hierarchies, associations and cardinalities on the comprehension of a diagram and the ability to detect errors in the diagram. Correct and incorrect diagrams were evaluated in an experiment. The participants were advised to match the diagram against the textual description and to identify the errors in these diagrams. The authors found significant effects only for the comprehension task. They conclude that inheritance hierarchies should be generally modeled upside-down and as a hyperedge (opposed to representing an inheritance hierarchy by a set of separate edges). Furthermore, association name labels should not interfere with the association line. Lastly, comprehension increases if cardinalities are modeled as min..max notation. Consequently a *modeling grammar* should describe rules on how to use the before-mentioned elements.

Fig. 4. Dependencies between the representational hypotheses
Conclusions and Further Research

We illustrated in the last section that empirically testable hypotheses can be derived from ontological research and cognition science. Hence, we could show that a theoretical support for method engineering is feasible. The suggested process model incorporates theoretical knowledge and, thus, allows decreasing the subjectivity of the method construction and to increase its traceability. Furthermore, this enables an empirical evaluation of the applied hypotheses and, therefore, fosters the development of a holistic method engineering theory.

Currently, we focus on hypotheses that claim to improve the quality of a modeling grammar in general. However, there are also hypotheses that are specific to a certain domain and which are not applicable under arbitrary conditions. Such hypotheses can be explicated for example from the configuration rules of adaptive reference models [55] or from the configuration packages of method configuration approaches [56]. It is due to further research to extent the process model by including domain knowledge and corresponding hypotheses.

The ratio between the number of established hypotheses and new hypotheses that are used in a method engineering project provides a measure for the likelihood of a successful application of the resulting artifact. The probability to achieve the desired project result increases ceteris paribus with the number of established and well evaluated hypotheses that are used in the construction process. Hence, this ratio can be interpreted as a measure for product quality. The validity of this assumption must be shown in future research.

The literature on conceptual and representational aspects of modeling grammars has mainly focused on single, isolated hypotheses so far. However, the available results do not necessarily mean that a combination of these isolated hypotheses will lead to the same outcome also. Some of the hypotheses could contradict each other or lose their intended effects when they are used together. Future empirical research must evaluate whether a combination of hypotheses is feasible and advisable.

References


The adoption of method engineering principles for the creation of organization specific IT processes

Boštjan Žvanut¹ and Marko Bajec²
1 University of Primorska, College of Health Care Izola
   Polje 42, 6310 Izola-Izola, Slovenia
   bostjan.zvanut@vszi.upr.si
2 University of Ljubljana, Faculty of Computer
   and Information Science
   Tržaška 25, 1000 Ljubljana, Slovenija
   marko.bajec@fri.uni-lj.si
   http://infolab.fri.uni-lj.si/marko/

Abstract. This article presents the idea of introducing method engineering concepts into the field of IT process construction. This field lacks of scientifically based systematic approach for IT process construction. Many important decisions about the structure of IT process are made according to the IT process consultant’s personal experience and subjective opinion. In this paper we propose a predefined approach for IT process construction which is based on the (scientifically) established method engineering principles and adapted to IT process construction specifics.

1 Introduction

Increasingly, top management is realising the significant impact that information can have on the success of the enterprise. Management expects heightened understanding of the way information technology (IT) is operated and the likelihood of its being leveraged successfully for competitive advantage. To provide the information that the enterprise requires to achieve its objectives, the enterprise needs to manage and control IT resources using a structured set of IT processes to deliver the required information services [1].

IT processes and other business processes in the organizations are defined by the following interrelating factors: organizational structure, culture, technology and
The adoption of method engineering principles for the creation of organization specific IT processes

people [2]. They form a complex system from both sociological and technological point of view. Practice shows that it is impossible to define universal IT processes suitable for every organization. For that reason we have to construct IT processes that are acceptable for the organization and at the same time suitable for optimal use of IT resources.

In the IT literature, as well as in the related research fields, no relevant researches or approaches for IT process construction were found. We found only documents that are based on best practices [1, 3, 4]. The research in the software development field has on the other hand contributed many ideas that tell how to tackle these issues if software development is under consideration. The question is, are these principles and techniques that prove to be useful for the creation and configuration of software development methods also useful for the construction of an arbitrary IT process. After all, a software development process is nothing else than an example of IT process.

The paper is organized as follows. In section 2 we describe the approach we adopted for our research. Section 3 and 4 give the reader an introduction to the COBIT framework and a brief overview of the main principles of the method engineering. The core of the paper is in section 5, where our idea of introducing suitable method engineering approach into the IT process construction is presented. Finally, in section 6, concluding remarks are given.

2 Research approach

In the research work on which we report in this paper the principles of the Collaborative Practice Research (CPR) [5] have been applied. CPR combines action research, experiments and conventional practice studies to strike a useful balance between relevance and rigour [5]. The decision to use CPR was derived from the good experiences that we had gained before when this approach was employed in similar research projects (see e.g. [6]). In this section we briefly describe the main idea of CPR and explain how we adapted it for the needs of our research.

CPR takes as the main consideration the difficulty of establishing effective and well functioning relations between research and practice. Ideally, the research process should be tightly connected with practice to get firsthand information. According to CPR we split our research project in three major phases:

1. Phase 1: analysis of the state of the art: to make improvements of particular IT processes in particular organisations, a good and solid understanding of how these processes are currently performed is mandatory. In addition, we should be familiar with best practices that tell how these processes should be ideally performed. In our research we took COBIT IT processes as a baseline and studied their theoretical background. To start with, we selected few IT processes as described in COBIT and studied how they are performed in particular organisations.

2. Phase 2: development of new methods and approaches: by deeper understanding of the evaluated IT processes, their negative aspects and weaknesses become
evident. According to CPR the second phase is dedicated to the design of new approaches, techniques and tools that could help to improve the IT processes under consideration. In our research, possible improvements were identified based on the understanding and knowledge gained in the previous phase.

3. Phase 3: application of new methods and approaches in practice: important phase of CPR is to test developed methods and approaches through attempts to improve practice. In our research, this phase was not performed yet in its entirety. For now, the detected deficiencies of the assessed processes were only discussed with representatives of the participating organisations while their implementation has not take place yet.

It has to be emphasised here that the main objective of our research work is actually out of the scope we just described. Our goal is not just to identify deficiencies and take actions to improve particular IT processes but to design an approach and supporting tool that will help us to make these steps automatically.

3 IT processes defined by COBIT

The IT Governance Institute (ITGI) (www.itgi.org) defines IT processes with a framework COBIT (Control Objectives for Information and Related Technologies) [1]. COBIT is a framework and supporting toolset that allow managers to bridge the gap between control requirements, technical issues and business risks. It enables the development of clear policy and good practice for IT control throughout enterprises. COBIT is continuously kept up to date and harmonised with other standards. Hence, it has become the integrator for IT best practices and the umbrella framework for IT governance that helps in understanding and managing the risks and benefits associated with IT. The process structure of COBIT and its high-level business-oriented approach provide an end-to-end view of IT and the decisions to be made about IT.

The framework was developed to be employed by IT service providers, users of IT process, management, business process owners and especially IT auditors. COBIT has strong focus on control and therefore it does not provide sufficient details about the structure of IT processes and their elements.

For our research, the following documents of the COBIT framework were studied:

- Framework: explains how COBIT organises IT governance objectives and best practices by IT domains and processes, and links them to business requirements;
- Control objectives: provides generic best practice management objectives for all IT activities;
- Control Practices: provides guidance on why controls are worth implementing and how to implement them;

In COBIT, 34 IT processes are defined, divided in 4 domains:

- Plan and organize: e.g. Define a Strategic IT Plan, Define the Information Architecture, Determine Technological Direction etc.;
The adoption of method engineering principles for the creation of organization specific IT processes

- **Acquire and implement**: e.g. Identify Automated Solutions, Acquire and Maintain Application Software etc.;
- **Deliver and Support**: e.g. Manage Service Desk and Incidents, Manage the Configuration etc.;
- **Monitor and Evaluate**: e.g. Monitor and Evaluate IT Performance, Monitor and Evaluate Internal Control etc.

COBIT provides for each IT processes a high-level control objective statement with key goals, metrics and detailed control objectives (as generic action statements of the minimum management best practices to ensure the process is kept under control). The following figure (Fig. 1) presents the COBIT’s a high-level control objective statement for IT process **DS8 - Manage Service Desk and Incidents** (domain Deliver and Support) with key goals and metrics.

![DS8 Manage Service Desk and Incidents](image)

**Fig. 1.** COBIT’s a high-level control objective statement of IT process DS8 - Manage Service Desk and Incidents [1]
4 Insight to method engineering principles and approaches

4.1 Method engineering

Method engineering is an engineering discipline that deals with the design, construction and adaptation of methods, techniques and tools for the development of information systems. Similarly as software engineering is concerned with all aspects of software production, so is method engineering dealing with all engineering activities related to methods, techniques and tools [7]. A specific direction in method engineering is situational method engineering (SME), which deals with developing project-specific methods or adapting existing ones to specific project situations [6, 8, 9, 10].

In the method engineering literature, a number of approaches can be found that propose how to create project-specific software development methods (for a review see [11]). Although these approaches have never been widely acknowledged or practised by software engineers [9], they offer strong theoretical bases that demonstrate how software development methods can be assembled and/or adapted to particular circumstances on-the-fly.

As pointed out in the introduction, the goal of our research is to generalise the principles of SME into an approach for the engineering of an arbitrary IT process. In other words, we would like to see whether the principles suggested by SME can be used also for other kinds of processes, not just for software development process. To this end, we carefully studied various SME approaches to see which one suits best to our goals. At the end it turned out (predominantly due to its simplicity) the most suitable is the approach that was actually developed in our research group – the Process Configuration Approach or shortly PCA [6].

Fig. 2. High-level architecture of the PCA
4.2 Process configuration Approach

The idea that lies behind the PCA is relatively simple and can be explained as follows (see Fig. 2): for each individual project a specific process configuration (project-specific method) is created. This is done by selecting components from a method that has been specifically designed for the organisation and thus reflects its actual performance on the projects (base method). The configuration is done by processing the rules that tell in what circumstances for project situations it is compulsory, advisable or discouraged to use a particular component. The rules are part of the base method and are defined together with other base method elements.

Base method is the most important prerequisite for the PCA. It is a formal representation of how a particular organization is performing its projects.

For the purpose of the PCA we designed a generic data structure that can be used to capture an arbitrary metamodel. The idea of a generic data structure is to allow method engineers to design metamodels according to their perceptions of how a method should be formally represented.

![Fig. 3. PCA generic data structure](image-url)

Fig. 3 illustrates the main components of the aforementioned generic data structure, base method and project-specific method. The classes representing metamodel are: a metaelement (it can be of two types: content element, such as activity, tool, discipline, role, etc. or process flow element, such as decision node, join and synchronisation) and metalink (links between metaelements). By using such a generic data structure, a base method is represented as a structure of instances of the metaelements and metalinks, and a project-specific method as a selection of the elements and links of the base method.

Typical base method encompasses various situations that may occur when projects are performed. In other words, it comprises a number of elements and their
alternatives which describe several possible ways to perform a particular project. The paths and method structure, however, are not fixed in the PCA. They are defined by the rules that tell which elements to consider in specific circumstances and consequently which path to take. As depicted in Fig. 3, rules apply directly to the links that bind elements of the method (see the element Condition).

5 Adopting PCA for the creation of an arbitrary IT process

5.1 PCA-IT

As illustrated in Fig. 2, in PCA a base method is configured so that it suits best to a particular project. This is done by taking into account the rules that tell which fragments of the base method to use in a particular project situation described by a set of project characteristics. In this section we demonstrate that the same approach is useful also for the construction of an arbitrary IT process. We will label this approach as PCA-IT.

Fig. 4. The duality of organization-specific IT process construction and project specific method construction

The main idea lies in the definition of the best possible IT processes which are perfect for perfect organisations. This means that they include all possible fragments and what is more important, they conform to the highest possible maturity level (e.g.
The adoption of method engineering principles for the creation of organization specific IT processes

they are defined, measured and optimised) [12] thus assuring the highest possible quality. Of course, in typical organisations it is rarely the case that we can afford to set up perfect IT processes as this brings in a lot of overhead in terms of people and money. The question is what is than an optimal level a particular IT process should reach in a particular organisation. We believe this level could be assessed by careful consideration of various socio-technical characteristics of the organisation in question. These are than the input that helps us to configure a perfect IT process into an IT process that suits best for the needs of a particular organisation. In Fig. 4 the duality between PCA and PCA-IT is represented.

5.2 IT process representation

Even though the data structures that are used in PCA to store base methods are generic (see Fig. 5) and thus suffice also for the representation of any kind of IT process, we introduced some extensions that are specifically intended for the representation of IT processes. Both, the core and additional elements of the model are represented in figure below (Fig. 5).

The additional classes in the model include:

- **COBIT’s IT process**: it defines the 34 COBIT processes;
• **Partial IT Process**: it defines partial IT processes or sub-processes a particular IT process consists of;
• **Selected IT Process**: it defines which IT processes are selected for a particular organization;
• **Selected Partial Process**: it defines which partial IT processes are selected for a particular organization.

Both Partial Process and Selected Partial Process have their Elements/Selected Elements which can be part of many Partial Processes (e.g. a particular application can be used in many partial processes).

The representation of both, perfect and organisation-specific IT processes is similar to the one in PCA. Perfect IT processes are represented as structures of instances of the metaelements and metalinks, while organization-specific IT processes correspond to a selection of the elements and links of the perfect processes.

As already mentioned, an important part of our research is to identify socio-technical characteristics and related rules that affect the IT process construction. These rules are applied in different conditions that affect the IT processes (see Fig. 5). We divide these conditions in four distinct groups: **conditions of process existence**, **conditions of partial process existence**, **structure conditions** and **process flow conditions**.

### 5.3 The main activities of PCA-IT

PCA-IT can be divided in three activities:

1. **Identification of As is process**;
2. **Construction of Should be process**;
3. **Identification of Possible changes**.

In the activity “Identification of As is process” we observe how selected processes are currently performed in a particular organisation. The goal is to identify which elements of the corresponding perfect IT processes are already performed and which are ignored.

Next is the construction of »Should be« processes which is supported by PCA-IT. The first step is to identify socio-technical parameters of the organisation which are than used, together with the corresponding perfect processes, as input into PCA-IT. Based on the values of general socio-technical parameters and conditions of process existence, conditions of partial process existence, structure conditions and process flow conditions, the perfect processes are configured into organisation-specific IT processes.

Finally, the “should be” process is compared against the “as-is” process to identify the changes that would be required to improve the existing to the desired situation. Of course, not always all changes are feasible due to the constraints of the existing situation or due to the availability and quality of concrete IT resources (applications, information, infrastructure and people).
Fig. 6 shows how general socio-technical parameters are considered to estimate COBIT’s maturity attributes and to determine how strategically important is the assessed IT process for the observed organisation. These attributes are then taken into account when selecting the IT processes and their components that are most relevant for the organisation in question. Furthermore, these same attributes are considered when checking the process flow and structure rules during the IT process configuration. For the detailed description on how structure and process flow rules are checked and fired please refer to [6].

Fig. 6. The parameters affecting the process construction

5.3.1 COBIT’s maturity attributes
COBIT’s maturity model can be used as a measurement system that helps us to estimate how well IT is being managed in a particular organisation. To this end it provides a set of discrete values according to which we can estimate the maturity of a particular IT process. These values are [1]:

- **0: Non-existent.** Complete lack of any recognisable processes. The enterprise has not even recognised that there is an issue to be addressed.
- **1: Initial.** There is evidence that the enterprise has recognised that the issues exist and need to be addressed. There are, however, no standardised processes; instead there are ad hoc approaches that tend to be applied on an individual or case-by-case basis. The overall approach to management is disorganised.
- **2: Repeatable.** Processes have developed to the stage where similar procedures are followed by different people undertaking the same task. There is no formal training or communication of standard procedures, and responsibility is left to the individual. There is a high degree of reliance on the knowledge of individuals and, therefore, errors are likely.
- **3: Defined.** Procedures have been standardised and documented, and communicated through training. It is, however, left to the individual to follow these processes, and it is unlikely that deviations will be detected. The procedures themselves are not sophisticated but are the formalisation of existing practices.
- **4: Managed.** It is possible to monitor and measure compliance with procedures and to take action where processes appear not to be working effectively.
Processes are under constant improvement and provide good practice. Automation and tools are used in a limited or fragmented way.

- **5: Optimised.** Processes have been refined to a level of best practice, based on the results of continuous improvement and maturity modelling with other enterprises. IT is used in an integrated way to automate the workflow, providing tools to improve quality and effectiveness, making the enterprise quick to adapt.

The maturity models are built up starting from the *generic qualitative model* to which principles from the following attributes are added: *Awareness and communication, Policies, standards and procedures, Tools and automation, Skills and expertise, Responsibility and accountability, Goal setting and measurement.* Higher quality of these attributes means higher maturity level. For example, at maturity level 0, the required skills and expertise is not even identified while at level 5 “the organisation formally encourages continuous improvement of skills…” Since for the maturity level all the attributes are important, its absolute value can only be as high as the value of the “poorest” attribute. For more details please see [1].

Fig. 7. The impact of COBIT’s attributes on IT process construction conditions

As we will see later, an estimation of the absolute maturity level a particular IT process reaches in a particular organisation does not suffice for the process construction using PCA-IT. By defining an absolute maturity level we lose information about the exact value of each IT process maturity attribute. These values are important for IT process construction, as they define conditions of process existence, conditions of partial process existence, structure conditions and process flow conditions. The following table shows how different attributes affect different conditions (Fig. 7.).

<table>
<thead>
<tr>
<th>Maturity Attributes</th>
<th>Condition of Partial Process Existence</th>
<th>Structure Condition</th>
<th>Process Flow Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Awareness and Communication</td>
<td><img src="" alt="Diagram" /></td>
<td><img src="" alt="Diagram" /></td>
<td><img src="" alt="Diagram" /></td>
</tr>
<tr>
<td>Policies, Standards and Procedures</td>
<td><img src="" alt="Diagram" /></td>
<td><img src="" alt="Diagram" /></td>
<td><img src="" alt="Diagram" /></td>
</tr>
<tr>
<td>Tools and Automation</td>
<td><img src="" alt="Diagram" /></td>
<td><img src="" alt="Diagram" /></td>
<td><img src="" alt="Diagram" /></td>
</tr>
<tr>
<td>Skills and Expertise</td>
<td><img src="" alt="Diagram" /></td>
<td><img src="" alt="Diagram" /></td>
<td><img src="" alt="Diagram" /></td>
</tr>
<tr>
<td>Responsibility and Accountability</td>
<td><img src="" alt="Diagram" /></td>
<td><img src="" alt="Diagram" /></td>
<td><img src="" alt="Diagram" /></td>
</tr>
<tr>
<td>Goal Setting and Measurement</td>
<td><img src="" alt="Diagram" /></td>
<td><img src="" alt="Diagram" /></td>
<td><img src="" alt="Diagram" /></td>
</tr>
</tbody>
</table>
5.3.2 Strategic importance of the IT process for the organization

Strategic importance of a particular IT process of the organization depends on various factors, e.g.: involvement of customers in the IT process, added value of the IT process, etc. In practice, if a particular IT process is strategically important for the organization (and the top management is aware of this fact), the management is prepared to invest more into it. The investments in IT resources (that are relevant for the IT process) are crucial for achieving higher levels of COBIT’s maturity attributes. Therefore the strategic importance of the IT process for the organization moderates the effect of COBIT’s attributes and has an important role in the IT process construction.

5.3.3 General socio-technical parameters

Each level of COBIT’s maturity attributes is explicitly defined in COBIT’s framework [1]. Although simple, these attributes are in practice difficult to measure. The reason lies in the fact that many organizations are not aware of their IT processes (in many cases their IT processes are not formalized at all) and therefore is difficult to define COBIT’s maturity attributes for concrete IT processes.

For this reason we introduce in our model general socio-technical parameters (e.g. organization’s degree of centralization, organization’s degree of formalization, number of employees, etc.) which we believe can help us to estimate COBIT’s maturity attributes for observed IT processes (see Fig. 6). General socio-technical parameters are independent of the organization’s awareness of IT processes and therefore easier to measure. Another reason for introducing general socio-technical parameters in our model is that they can help us to determine the strategic importance of the IT process for the organization (Fig. 6.).

5.4 The current state of the research

Currently we are working on the identification of the socio-technical parameters that are significant for the construction of IT processes. Specifically, we are interested in the characteristics that will help us to perform reliable estimations of the COBIT’s maturity attributes as well as to determine how important specific IT processes are for a particular organisation. A draft of the model that comprises characteristics important for the IT process construction has been already developed but it is not presented here since it has not been tested yet and may contain characteristics that are not really significant for the problem domain.

Another activity we are working on currently is the definition of the COBIT IT processes that correspond to the highest maturity level and thus assure the highest possible quality (perfect IT processes). For now we managed to define a perfect version of the COBIT process DS8 – Manage Service Desk and Incidents which we tested with a prototype tool that supports PCA-IT. The results are shown in the next section.
5.5 Example

In the following subsection we wish to demonstrate how PCA-IT can be used to construct an organisation-specific IT process DS8 based on its perfect version and socio-technical characteristics of the organisation. As emphasised before the socio-technical parameters we use are not fully defined yet. Therefore we will use PCA-IT with the assumption that exact values of the COBIT maturity attributes for the selected process are already known. Due to space restrictions we will present only a part of the perfect DS8 IT process and its configured version.

Organization’s parameters

COBIT’s IT process: DS8 - Manage Service Desk and Incidents
Strategic importance of process for selected organization: Very critical
COBIT’s attributes for DS8 in concrete organization (Fig. 8.):

![Figure 8: COBIT’s maturity attributes for a process DS8 of a concrete organization](image)

**Perfect IT-process**

In the study of the DS8 process we came to the conclusion that perfect process DS8 consists of three partial processes (Fig. 9): (a) basic partial process DS8, (b) supervision of DS8 and (c) trend analysis of DS8. Each partial process has its own partial process existence condition (C1, C2 and C3) derived form COBIT’s maturity levels [1].
The adoption of method engineering principles for the creation of organization specific IT processes

C1: $\text{AC} \geq 3 \text{ AND } \text{PSP} \geq 3$
C2: $\text{AC} \geq 4 \text{ AND } \text{GSM} \geq 4$
C3: $\text{AC} = 5$
*abbreviations are explained in Fig. 8.

Fig. 9. Partial processes of COBIT’s IT Process DS8: Manage Service Desk and Incidents

Fig. 10. (due to space restrictions) shows only some elements of one partial process DS8-basic partial process with its elements and related conditions.

»Should be« process construction
In this example the process DS8 was marked as “very critical” (for the organization) and therefore it will not (negatively) moderate the value of COBIT’s maturity attributes. According to the concrete values of organization COBIT’s maturity attributes (see Fig. 8), we realize that only condition 1 (C1) from figure 9 and conditions 1 and 3 (C1 and C3) from figure 10 are satisfied. The result of construction is a »Should be« process, tailored for the selected organization (Fig. 10.).
58 Boštjan Žvanut and Marko Bajec

C1: TA < 3
C2: TA >= 3
C3: AC < 5 AND PSP < 5
C4: AC = 5 AND PSP=5
*abbreviations are explained in Fig. 8.

Fig. 10. Example of some elements of perfect Basic Partial Process DS8

Fig. 11. Example of a result of the IT process DS8 construction for the concrete organization

6 Conclusion and further work

In IT process design, consultants use different frameworks and standards that are based on good/best practices. This ad-hoc approach lacks of scientific rigour and many important decisions are made according to the IT process consultant’s personal
experience and subjective opinion. In this paper we proposed a predefined approach for IT process construction that builds on the established method engineering principles. For the needs of constructing an arbitrary IT process we have adopted the Process Configuration Approach which is a variation of the situational method engineering. In the paper we demonstrated how this approach can be extended to support the creation of an arbitrary IT process based on the socio-technical characteristics of the observed organisation.

As already acknowledged this paper is about the research in progress and should be treated this way. Further work includes the detailed definition of perfect versions for some other important COBIT’s IT processes (e.g. Manage changes, Manage the configuration etc.) with respective conditions of process existence, partial process existence, structure conditions and process flow conditions. Next is to confirm the validity of the evaluation model that defines which socio-technical characteristics are important for PCA-IT. Finally, our goal is to test the proposed approach in real practice, e.g. by its implementation and evaluation in participating organizations.

Another possible direction for further research is to test the approach also in other non IT disciplines where the processes are generally well defined, but their implementation in practice (according to the organization’s socio-technical parameters) differs from organization to organization (e.g. Health Care).

References

1. IT Governance Institute, COBIT 4.0 (IT Governance Institute, Rolling Meadows, IL, USA, 2005).
Practical Use of Method Engineering
Analysing Benefits and Challenges

Josef Blasini and Susanne Leist
University of Regensburg
Universitätsstraße 31, 93040 Regensburg, Germany
{josef.blasini, susanne.leist}@wiwi.uni-regensburg.de

Abstract. Over the last decade Method Engineering has emerged as the research and application area for using methods, especially but not only for systems development. Although numerous methods have been developed to that purpose, there are only few publications which concentrate on highlighting the application of Method Engineering in a business environment. This discrepancy presents the starting point of this analysis. It aims at deriving hypotheses which offer a first explanatory approach and can be verified in a subsequent empirical survey.

1 Introduction

Method engineering is the discipline to design, construct, and adapt methods, techniques and tools for the development of information systems [1] (see also [2], [3]). The starting point of many publications in this discipline is the multiplicity of the existing software development methods. Despite the availability of a vast number of methods it has, however, so far been preferred to develop a new method for each occurring problem. The potential of already existing methods was not exploited [4] and by putting up with unnecessary double applications the (already) high number of methods was constantly growing. To counteract this tendency approaches were developed in Method Engineering which support the construction of new methods on the basis of already existing ones. In the meantime a lot of such approaches have been developed and fields of application have been worked on. The method construction is, for instance, particularly suited for projects, e.g. in Change Management with only a small number of recurring problems (see [5]). Thus Method Engineering has established itself as a scientific discipline and has become the object of many papers. On closer examination it does, however, show that – despite the enormous practical benefits derived from the re-use of existing methods – only few
publications pinpoint explicitly the application in business environments, i.e. in concrete business projects (see [3, 310]). The existing publications show as well that the subject is predominantly pushed by science and is only applied upon instigation of the latter in business environments ([6], [5], [7]). Practitioners have not yet discovered Method Engineering. It is thus the aim of this paper to propose hypotheses about the low prevalence rate in business environments which are verified in a subsequent empirical survey and to ascertain approaches for the further development of method construction.

Therefore the following sections contain a short overview of the different methodological approaches as well as of the tools developed in Method Engineering. Afterwards the benefit potentials of Method Engineering are worked out which serve as the basis for phrasing the hypotheses. An outlook on the subsequent examination concludes this paper.

2 Research in Method Engineering

2.1 Goals and Approaches in Method Engineering

Method Engineering comprises the construction of methods which is domain-specific, goal-oriented and user-specific. Consequentially early research work concentrates on the analysis, the comparison or the support of the selection of methods, techniques or technologies (cf. e.g. [8], [1], [9], [10], [11], [12], [13]). Basically methods for the development of information systems are focussed. As a result the authors established the term and the constituent elements of a method (e.g. [14], [2], [15]).

In almost all of the relevant literature, methods are described as procedures which are both purposeful and methodical. They consist of components which are constructed by means of defined rules and principles (see e.g. [16] [17, 125], [18], [4, chap 2.1, 1], [19, 90-93], [20, 1], [21, 202], [22, 197]). The constitutive components of a method are activities, roles, results, meta model, techniques, and tools (see [15, 11-17], [23, 148]). In Method Engineering, however, it is not the components that are utilised, but method fragments (or method chunk) to construct methods. A method fragment is the description of a method or any of its coherent parts [17, 125].

Recent publications extend the focus of contemplation and examine the application of Method Engineering in different domains (e.g. organisational change [5], agent-oriented Method Engineering [24, 38] or healthcare [7]). Therefore, method engineering aims at the development not only of information systems but also of so-called work systems, especially IT-reliant work systems (as regards the term work system see [25]. At the same time different construction procedures are assumed (e.g. "configuration" [26], "assembling" [27], "extending with pattern" [28]). In addition influencing factors on the construction procedure, i.e. characteristics of the project and the development environment, the problem domain and the given goal are examined to be able to include them in the construction
procedure [29], [1], [5]. Depending on the emphasis of the examination (e.g. the specific situation of the project or the environment of the development (context)), different methodical approaches are developed (Situational Method Engineering, e.g. according to [30], Contextual Method Engineering, e.g. according to [31]).

All in all the following core areas of Method Engineering can be outlined (see [2, 1-5] [19, 4-5]):

- **Specification and design of methods**: Methods must be developed on the basis of the particular problem and specified harmonically in view of a later (partial) re-use. Thus the prerequisite is a consistent and well-structured description of the method fragments.

- **Comparison and evaluation of methods**: Method Engineering offers a basis for the comparison of methods and their consistent evaluation. In this context the concept of the meta modelling provides an important instrument for the comparative analysis not only of individual method fragments (e.g. modelling techniques, languages) and constructed methods, but also of Method Engineering approaches as such.

- **Method integration and adaptation**: In order to construct new methods it is necessary to revert to existing methods and to adapt them to the respective (project) situation in a goal-oriented and contextual way. Thus methods underlie continuous development to suit different situations, e.g. by means of versioning.

- **Management of method knowledge**: The know-how derived from the process of development must be administrated and stored. This refers, above all, to the documentation of the influence of specific situations, contexts and goals on the suitability of individual method fragments. In addition, experience (positive as well as negative) made with the methods and the method construction used ought to be documented in a company-wide data base (compare [2, 4]).

### 2.2 Tools for Method Engineering

In parallel with the theoretical analysis of Method Engineering tools were developed which support the construction procedure. This was evident, since the first constructed methods were devised for the development of information systems. The basic idea was the computer-aided construction of methods and their subsequent utilisation in software development projects.

The important and essential tasks in this context consisted primarily in the development of meta models (data- and process-oriented) which define the modelling resp. the description languages of the methods and served as the basis for the repositories. Examples for such languages are ASDM ([32], among others used in [2]), CoCoA (among others used in [33]), GOPRR, OPRR (among others used in [21]) or NIAM ([34], among others used in [35]). Apart from the latter object-oriented languages such as Object-Z (compare [36]) were developed as well as a language which was particularly geared at the specification and the manipulation of method fragments (MEL [1, 278-279]).

Development tools support the construction procedure (Computer Aided Method Engineering (CAME)) (e.g. Decamerone, GraphicalDesdigner, JKogge, Maestro II, MetaEdit+, MetaView); their individual focus, however, is very different. In
principle, all CAME tools focus the design of the method. Some of them provide facilities for specifying, storing, and selecting method fragments and for assembling method fragments into a situational method (e.g. Decamerone). As regards others the support is, however, limited to the adaptation of the modelling languages, e.g. in the form of an extension of the objects of a modelling language (e.g. MetaEdit+). Other tools, in turn, do not only support the manipulation of the modelling languages, but also the activities of the method, as has for instance been considered in [37]. Since these tools do not modify the method, but only individual components, namely the development processes of the modelling languages used, they are – more precisely – referred to as Meta-Case tools. As regards some of them (e.g. Maestro II) special transformation functions have been defined which allow for transferring models and representations into the corresponding CASE tool.

On the whole many of the tools were only developed as prototypes. That is why only few mature products are currently available (as for instance Cubetto Toolset 1.2 or MetaEdit+ 4.5).

2.3 Benefit of Method Engineering

In view of the aim to compose or adequately change methods (domain-specific, goal-oriented and user-specific) a method base consisting of method fragments is constructed in a first step. With each new project the existing method base is reverted to and changes resp. extensions in the context of the specific project will be entered in the method base in the form of new method fragments. The basis is the central structure (repository) of the method base on which the description of the method fragments depends. This application results in the following benefit potentials:

1. The most obvious benefit appears to be the advantage to be able to execute specific extensions (domain-specific, goal-oriented and user-specific) by combining aptly existing method fragments. In the simplest case this can be the adaptation of a modelling language by means of changing its notation or by extending its modelling constructs. In so doing new views onto the problem can be taken and modelled; this increases the number of applications of this method.

2. Specific extensions can, however, also be achieved by including new method fragments (e.g. adding a new technique or a new modelling language). Here Method Engineering has a supportive role, since by means of the repository it sets the rules as to how new method fragments are included into the method base: thus such an extension is made possible in the first place.

3. By way of the repository of the method base not only extensions of the method fragments, but also transformations between methods become possible. For instance, results of a project already concluded can be allocated to another project irrespective of whether the same modelling language has been used or not. In the ideal case result documents can be described without information losses by means of a different modelling or programming language.

4. By way of storing transfer rule reports, documentations, programme code, etc. can be automatically generated from the models, likewise syntax checks can be carried out. Advantages result, above all, from ensuring that developments take place in good time and from possibilities to make transfers from a model into
different application systems e.g. mobile phone, PC, a different model, a
different notation, etc.

5. Experiences in the application of methods are systematically gathered and are
available for use in subsequent projects. The quality of the project results can
thus increase.

The above details essentially show that the application of Method Engineering
can support three basic success factors flexibility, time and quality:
- A higher flexibility is achieved by means of combining existing method
  fragments or integrating new method fragments (benefit 1 and 2).
- Likewise the project execution time is reduced, if it is possible to revert to
  existing results, because the latter do not have to be manually collected or
  entered again or manually transferred into programme code (benefit 3 and 4).
- Ultimately the quality of the projects will increase, too, if relevant results and
  experience from projects already concluded find their way into new ones.

Reducing information losses (because of the discontinuity of different formats)
through meta model-based transformations increases as well the project quality
(benefit 3, 4 and 5).

The said benefit potentials can basically be achieved irrespective of whether a
tool is available which has a supportive role when selecting, extending or
transforming the method fragments. With an increase in method fragments in the
method base, however, the complexity grows as well, since individual method
fragments can show interdependencies. Therefore a greater benefit can be achieved
with the application of a tool.

3 Challenges in Method Engineering

Even if the above explanations have pointed at the benefit potentials of Method
Engineering, it shows, as described in the beginning, that Method Engineering is
mainly instigated by research institutes and is rarely incorporated into practical
projects. If one assumes that companies act rationally, it can only be that the
implementation of Method Engineering presents challenges which obviously
compensate the benefit. A closer examination of these challenges plights to find
starting points for the further development of Method Engineering and to reach a
higher prevalence rate. Challenges that have to be overcome when implementing
Method Engineering will therefore be outlined in the following, they serve as a basis
for the phrasing of hypotheses which are supposed to be verified resp. falsified. In
doing so the examination will be limited to those challenges which counteract the
benefit of Method Engineering.

Reasons for a compensation of the flexibility which can be achieved by
combining existing method fragments or integrating new method fragments mostly
arise from the users’ attitudes. It could, for instance, be possible that on the strength
of past – very positive – experience regarding the application of the established
methods the responsible project staff assume that continuing to apply the well-known
methods will lead to better results in shorter time. Further on there could be the worry that the changes or the adaptations of the methods could breed quality losses, since consistency checks do not take place or interdependencies between the method fragments are ignored. These assumptions suggest the following first hypothesis which ought to be verified.

Hypothesis 1: The responsible project staff prefers to apply established methods rather than developing new ones.

Savings of time which arise from the re-use of existing results, for instance, because multiple data entries or manual transfers of specialised concepts into programme code can be avoided requires a certain level of quality of the existing data. The actual level of quality is based on the requirements of the subsequent project or the subsequent phase (implementation) which revert resp. reverts to the existing data. It has not been attained, if the existing results are not relevant, incompletely registered or insufficiently documented. It has not been attained, either, if the results are presented in such a way that the transfer in the modelling language of the new project is only possible when putting up with high losses of information; or if the specialised models are described so imprecisely that they have to be intensively reworked before their implementation. These assumptions lead to the second hypothesis which has to be validated.

Hypothesis 2: The potential for re-use of the existing results is too small due to the insufficient quality of the available data.

The reason for the improvement of the project quality is the re-use, since it is not only possible to revert to existing result documents, but also to the experience in previous projects. The prerequisite is the generation of a method base which in addition to the method fragments also documents the experience in applying the method fragments. At the same time quality management processes are to be established when including or changing the method fragments. If there are only few method fragments for a project, it is quite probable that existing method fragments are to be adapted or new method fragments be introduced. The larger the method base is, the better new projects can be carried out by combining existing method fragments. At the same time it becomes less necessary to apply extensions to the method fragments. Thus the benefit of the project increases with the number of integrated method fragments. If the method base is generated from project to project, visible benefits will only occur much later which could easily call into question the acceptance of Method Engineering. This motivates the third hypothesis:

Hypothesis 3: The efforts (and expense) to generate a method base and to administrate it compensate its benefit.

4 Outlook

This paper presents a short survey of Method Engineering and, explicitly, the benefit which can be derived from the (situational) applications of constructed methods and techniques. Following to the aim of the paper three hypotheses were
phrased to describe the approaches for the – hitherto – rare practical application of Method Engineering. These hypotheses are the basis for a continuative examination in which they are first of all firmed up by means of detailed questions and subsequently are verified in guideline-based interviews with experts. Hypothesis 1, for instance, has to be specified by means of questions regarding the methods applied, concerning the significance of methods in projects, regarding the experiences as to the adaptation of methods and concerning the quality management. The results of the empirical survey have to be differentiated in view of the different domains in which Method Engineering is applied, the type of the construction process and whether influencing factors on the construction process are taken into account. After systematising the results approaches have to be devised which support the further development of Method Engineering and increase its practical benefit.

References


25. S. Alter, 18 reasons why IT-reliant work systems should replace "the IT artifact" as the core subject matter of the IS field, *Communications of the Association for Information Systems*, **12**, 366-395 (2003).


