TUM

TECHNISCHE UNIVERSITÄT MÜNCHEN
INSTITUT FÜR INFORMATIK

Crafting a Method Engineering Metamodel
Approach, Methods, Results

Marco Kuhrmann, Michaela Tiessler

TUM-I1410

Crafting a Method Engineering Metamodel

Approach, Methods, Results

Marco Kuhrmann, Michaela Tiessler
Technische Universität München
Institut für Informatik, Software & Systems Engineering
Boltzmannstr. 3
85748 Garching, Germany
kuhrmann@in.tum.de, *Michaela.Tiessler@gmx.net*

Summary

Method engineering is a research area that addresses the need for the construction and the flexible and situation-specific composition of methods. Our research objective is to develop a *Method Engineering Metamodel* that serves software process improvement & management in all it facets.

The report at hands documents two years of method engineering research. It summarizes the research methods and the respective outcomes. The report serves as data sink in which we summarize all (tentative) findings. Furthermore, the report comprises the first outcomes, which are a step towards the creation of a method engineering metamodel that lays the foundation to be implemented in various tools.

Keywords

Method Engineering, Situational Method Engineering, Software Engineering, Software Process, Software Process Improvement, Software Process Metamodels, Literature Review, Metamodeling, Crafting

CR-Classification: D.2, D.2.9

Contents

List of Figures

List of Tables

1 Introduction

Method engineering[1] is a research area that addresses the need for the construction and the flexible and situation-specific composition of methods. In 1987, Basili and Rombach [7] fostered the discussion on more flexibility of software processes. The tailoring of a software process should address project goals and environments.

In response to the demand of providing more flexibility in designing and adapting software processes, *method engineering* was proposed as a new paradigm. Especially Brinkkemper [13, 11, 12] and Harmsen [20] provided fundamental research on method engineering. Based on their research, several contributors, e.g. Henderson-Sellers and Gonzales-Perez, worked on the adaptation of method engineering to software process metamodels [22, 19]. Furthermore, approaches that used basic method engineering ideas were proposed to support authoring and designing methods [60, 15, 59]. Today, several software process frameworks, including SPEM [57] and ISO 24744 [32], sate to implement basic method engineering concepts.

Problem Statement & Research Objective Although method engineering was proposed in the mid 1990's only few studies dealing with analyzing the state-of-the-art and, in particular, the feasibility when applying method engineering concepts in practical settings. In particular, a comprehensive, accepted, and proven method engineering metamodel is yet not available. Such a metamodel would support the definition of approaches to organize and steer software process improvement (SPI), and such a metamodel would also allow for development of supporting tools, e.g. to support process design, process life cycle management, or process enactment.

The research objective of our research on method engineering is thus to develop a *Method Engineering Metamodel* that serves software process improvement & management in all it facets. We aim to analyze the state-of-the-art in method engineering research by combining different methods from empirical software engineering to, ultimately, develop a (conceptual) metamodel for method engineering.

Contribution We contribute first steps toward an integrated *Method Engineering Metamodel*, which is based on the documented knowledge from literature as well as our intensive analyses of software process frameworks [45]. In order to craft this model, we used different techniques from empirical software engineering, e.g. *Systematic Literature Review* (SLR; [40]) to detect relevant literature, and we investigated the used terminology and approaches in which we developed software process metamodels and artifact models and continuously evaluated these models.

Beyond the inferred method engineering metamodel, we also provide a critical discussion of the outcome and relate the metamodel to a metamodel, which was explicitly designed to support life cycle management of comprehensive software processes and software process lines (SPL; [70]).

Remark: The technical report at hand serves as data sink that comprises all data required to reproduce our research. Especially, we name and describe all methods used during the investigation. Results of our work that is already published is only referred, but explained as needed to understand the rest of the report.

1 To ease understandability, we use the term *method engineering* to subsume all schools of method engineering, i.e. method engineering and situational method engineering.

1.1 Related Work

In this section, we give a brief overview of related work (a detailed discussion can be depicted from [48, 49]). We structure the related work section into (standard) literature on (Situational) Method Engineering, and work, which we published on this topic.

Tolvanen et al. [78] provided a first review on method engineering to show future research directions in 1996. In 1997, Hofstede and Verhof [75] provided the first study on the state-of-the-art. They discussed the definition of methods and method fragments, the selection of method fragments, storage, formalisms, the retrieval and the assembly of method fragments. Taking into account that method engineering was a rather "young" concept at this time, Hofstede and Verhof provided a comprehensive collection of relevant concepts and terms. However, their contribution is more of philosophical nature as they discussed available concepts rather than providing any research type classification for those concepts. Still, they concluded that *"much more empirical research is needed to substantiate the claims associated with the potential benefits of situational method engineering."* In 2009, Rolland [68] reviewed the state-of-the-art and compared method engineering-related concepts and the terminology used. This work should provide *"a survey of the main results obtained for the two issues of defining and assembling [reusable method] components."* The survey also stays, however, at a more philosophical level and, thus, is comparable to the one provided by Hofstede and Verhof [75], which did not follow the (today) established procedures of a systematic literature review and/or classification of available contributions according to their research type facets (cf. Petersen et al. [58]). This also holds for the contributions of Henderson-Sellers and Ralyté [26] that continued the discussion of the state-of-the-art in 2010.

Previously Published Material The work that we summarize in this technical report was done in the context of a broader research program in which we investigate the role of *software process improvement* (SPI) and *software process management* (SPM). To this end, a number of contributions was published in this context: In [48], we provided first results of a systematic literature review (SLR) conducted to determine the maturity of the method engineering domain in general. An extended version of this paper that adds further research questions and an initial in-depth analysis is presented in [49]. The SLR also serves as the basis for the research presented in this report. To this end, a discussion on the SLR and, especially, on the methods applied to investigate the domain can be found in Sect. 2 (SLR) and Sect. 3 (analysis techniques).

Since SME is considered as general life cycle approach, several complementing topics that are of special interest for process engineers were contributed: In [33], we investigated the flexibility of software processes by investigating tailoring criteria, which is important in order to provide process consumers with the required flexibility to adopt a software process to the respective context. Such information is also important, when it comes to the design of flexible methods and, thus, is relevant in the context of crafting a flexible method engineering metamodel.

In [45], we investigated the current state of the art regarding today's software process frameworks to work out which SME-related capabilities are already implemented and which features would be necessary to add in order to allow for an artifact-based method engineering. Since we identified a number of gaps in SME, many components especially in the context of an integrated life cycle management need to be added. The outcome of this work is documented in the ArSPI model [42], which is an approach to organize and conduct SPI in a company-wide SPM strategy. A complementing experimental validation can be depicted from [44].

1.2 Outline

The remainder of this report is structured as follows: In Sect. 2 we introduce the overall research design and put emphasis on the literature review that builds the backbone of our investigation.

In Sect. 3, we present details from the analysis procedure and introduce the instruments applied during the analysis, namely tag clouds, social network analysis, and the identification of research type facets. In Sect. 4, describe the construction of the method engineering terminology that emerges from the selected key contributions and, finally, in Sect. 5, we infer and critically discuss a method engineering metamodel. Further detailed information regarding the collected data (data tables and numbers) can be found in the appendix.

2 Literature Review

In order to craft a unified metamodel for (situational) method engineering, the determination of the state of the art is a key task. Method engineering as research area was defined almost 20 years ago and, therefore, the analysis of the maturity of this particular domain, the analysis of the proposed concepts, and the analysis of the general feasibility of method engineering are the first steps to be done. For this, we conducted a comprehensive literature review according the the *systematic literature review* (SLR) method.

In this section, we present our research design. We discuss the research questions, the case selection, and the procedures for the data collection, the analyses, and for supporting the validity. As this work also reflects previously published contributions, we present the overall research designs on which also our contributions [48, 49] are based, and we extend the research design in order to address the remaining research questions. This section basically follows the structure proposal by Runeson et al. [71].

Chapter Overview

2.1 Research Questions

According to our contribution [48, 49], we define the research questions and follows:

	Research Question
RQ1	How many papers on method engineering were published over the years? *The first research question aims at investigating which publications were contributed in which year. This shall give us the opportunity to analyze particular trends in a quantitative manner.*
RQ2	Which research type facets do the contributions address? *The second research question aims at structuring the publication flora according to the research type facets proposed by Wieringa et al. [80] to investigate whether the contributions where of more conceptual nature or of more empirical nature. The classification of the research type in combination with the year of publication shall round out the trend analysis and needs an in-depth analysis whereby we consider our study to be not exclusively a mapping study where we classify the publications according to the abstracts and the keywords, but need deeper insights to analyze the state of evidence. One reason is that many contributions classified by the authors as, for example, a "study" need more clarification regarding the type of study, e.g., validation research or evaluation research.*
RQ3	Are there prominent contributors recognizable and how they are related to each other? *This research question aims at analyzing to which extent single authors where present in the publication flora and how the different authors relate to each other to identify networks in which certain concepts and a certain terminology were shaped.*
RQ4	What Terminology is used and how does the concepts fit into a metamodel for method engineering? *This research question aims at analyzing the method engineering domain regarding the fine-grained concepts that can be used to define a unified metamodel for (situational) method engineering.*

The technical report at hand is a "cumulative" work in which we build on previously published contributions. The research questions *RQ1*, *RQ2*, and *RQ3* were already covered in two previously published contributions. In [48], we mainly covered *RQ1* and *RQ2*. In [49], we added a detailed discussion regarding *RQ3*. These contributions lay the foundation for the answering *RQ4*, which we address with the report at hand.

In subsequent sections, we first provide a brief summary of the literature studies related to our research.

2.2 Case Selection

Our contributions [48, 49] aim to systematize the (situational) method engineering domain. As instrument, we selected a combination of a *systematic literature review* (SLR) and a *mapping study*. Peterson et al. [58] propose to initiate a mapping study by (1) constructing the repository via a search of primary papers, (2) screen those papers for inclusion and exclusion according to their relevance to the research questions, and (3) construct the classification scheme of the maps according to the keywords and the abstracts.

However, we need a deviation from the standard procedure for two reasons (discussion from [49]):

1. Inherent in the research area is that many contributions cannot be allocated to a common area "method engineering"; for instance, many publications arise from other research communities (e.g. information systems development) that investigate concepts of software processes

and tailoring of any facet, e.g., “organizational tailoring”, “static tailoring”, “dynamic tailoring”, or “software process customization” in general. Those exemplary terms and concepts already show how the various interpretations of method engineering hamper the definition of the search strings and the inclusion and exclusion criteria in advance.

2. We deviate from the standard way of constructing maps according to the keywords proposed in the publications as we are especially interested in aspects, which we cannot extract from the keywords as they are pre-defined by an external classification scheme (independently of given keywords). For instance, we are interested in the research type facets, which are defined according to a fixed set of criteria in [80], not necessarily matching the keywords used by the authors.

For those reasons, we refer to the case selection by following a more pragmatic, yet more time intensive procedure. We first structure an initial set of publications to lay the foundation for the search string definition following the principles of snow-balling [40]. We use a primary set of publications and manually search for secondary references that are based on the contributions' references sections to find further contributions. This first research step was implemented in an examination project[1] and resulted in a set of standard contributions used for testing our research questions, search strings, and for structuring the publications.

Author	Publications
Brinkkemper	[10, 13, 11]
Harmsen	[20]
Henderson-Sellers	[28, 26]
Hofstede	[75]

Table 2.1: Overview: Primary sources for the snow-balling procedure.

For this primary search, we refer to the authors and publications summarized in Table 2.1, which we use later also as control values. The second step is the automated search in several literature databases, which we introduce in the following.

2.3 Data Collection Procedures

The data collection is an automated search in several literature databases[2]. The queries are built on the basis of the keyword lists given by the primary sources (Table 2.1) and terms most commonly used in the area of software processes. The authors of the set of the primary sources also serve as control values (the automated search result set has to contain the contributions of those authors, see also the previous section).

As main data sources, we rely on established literature databases, which we consider most appropriate for a search, and a meta search engine (DBLP) to fill potential gaps of the other selected literature databases. The internal discussion about which databases to select was based on our experiences in the software process engineering domain (e.g. which conferences are in the field and which journals are of interest). In consequence, we selected the following databases:

- ACM Digital Library
- SpringerLink
- IEEE Computer Society Digital Library
- Wiley
- Elsevier

1 Stute, O. Method Engineering – Prinzipien und Konzepte. TU München, 2012 (in German).
2 The search was extended and redone over time, e.g. in the context of revising [49].

- DBLP

If there is a paper listed in one of those databases, but is only referred from another database, we allocate the result to the database that generates the item, regardless of the actual publication location.

In addition to those databases, we also take papers into account that are not referred by the databases, but have to be considered as key contributions, e.g. PhD thesis as the one of Harmsen [20]. For such contributions, we add a category "misc". To structure the data, we created a spreadsheet that was later used to screen and select the contributions.

2.3.1 Query Definition

We defined our queries as follows: For the beginning, we took a sample of relevant papers, analyzed them in order to identify and iteratively refine the search strings, and validated them against a pre-defined list reference authors to be part of the search results (see Sect. 2.2).

The initial set of key words was: *software, development, process, tailoring, method, methodology, customization, customisation, adaption, adaptation, ISO, CMMI, SPICE, standard, compliance, study, experience, weaving, situational, engineering, practice*.

Based on the primary searches and the analysis of the primary sources via snow-balling, we conclude the following search strings.

	Query String
S_1	(process **OR** method **OR** methodology) **AND** (tailoring **OR** adaption **OR** customization **OR** customisation) *Search string S_1 addresses such publications that deal with software processes, methods, and tailoring in general.*
S_2	process tailoring **AND** (practice **OR** experience **OR** study) *In S_2, we search for contributions on tailoring software processes.*
S_3	software process **AND** (standard **OR** CMMI **OR** ISO **OR** SPICE) **AND** compliance *Search string S_3 is introduced to also get contributions in the area of software process improvement (SPI), which is a field of interest when constructing, adapting and optimizing methods.*
S_4	method **AND** (engineering **OR** weaving) **OR** situational method engineering *S_4 explicitly considers contributions on method engineering and situational method engineering.*

Table 2.2: Queries for the literature search procedure.

Due to the complexity of the publication sets provided by the different research communities, we do not further distinguish between primary strings and secondary strings (see also the discussion in Sect. 2.2).We used the search strings and aforementioned literature databases for the data collection. Each result set was transferred to a spreadsheet.

2.3.2 Selection Criteria

Having the single result sets available, all results were combined and used as basis for the data analysis. Since the considered literature databases (for instance the Wiley database) eventually limit the complexity of the queries, we took into account at most the first 160 search results for each data source. Due to the nature of the investigated domain, we expected a considerable number of contributions, even such that are out of scope for the study at hands. Therefore, we defined inclusion criteria (IC_x) and exclusion criteria (EC_x) to filter the result set (see Table 2.3).

	Criteria
IC_1	The contribution's title refers to method engineering.
IC_2	The contribution's keyword list contains terms related to method engineering.
IC_3	The contribution's abstract refers to method engineering, at least it relates to method engineering(-like) concepts, such as software process adaptation or tailoring.
IC_4	The contribution's full text introduces, discusses, or compares method engineering or concepts related to method engineering (incl. terminology, concepts, tools, etc.).
IC_5	The contribution reports on experiences w.r.t. method engineering.
EC_1	The contribution is not on Software Engineering or computer science in general.
EC_2	The contribution refers to method engineering only in its related work section without further contributing to this topic.
EC_3	The contribution occurred multiple times in the result set.
EC_4	The contribution's language is neither English nor German.
EC_5	The full text of the contribution not available.

Table 2.3: Overview: study selection criteria (inclusion: IC, exclusion: EC).

2.4 Analysis Procedures

In the following, we describe our analysis procedure, which consists of a preliminary analysis preparation and a subsequently conducted in-depth analysis.

2.4.1 Analysis Preperation

To get the initial set of data to be analyzed, we performed an automated search that required us to filter and prepare the result set. The data analysis was prepared by harmonizing the data and performing a 3-staged voting process to prepare the in-depth analyses.

Harmonization. After a first analysis of the search results, we saw many contributions occur multiple times in one result set or that many contributions were out of scope. To make the selection of the contributions more efficient, we thus first cleaned the result set by eliminating multiple occurrences. Furthermore, we removed papers that not dealing with computer sciences (e.g., from the medicine or chemistry domain) or papers of which the full text was not available. In this harmonization stage, we first applied the exclusion criteria EC_1, EC_3, EC_4, and EC_5 (Table 2.3). For EC_3 we furthermore checked, whether the contribution occurs multiple times because of matching several search strings. In such a case, the first occurrence remained in the result set and all copies are removed. If a contribution occured multiple times because of a situation in which a conference paper was followed by a journal article, the journal article—or the most comprehensive paper—remained in the result set and all other entries were removed from the result set.

Voting. We performed a 3-staged voting process to classify the papers as relevant or irrelevant and to build a set of contributions for further investigation by applying the exclusion criterion EC_2 and the inclusion criteria IC_1, IC_2, and IC_3.

Result Table

The integrated result table, which was created during the analysis, contains three columns. The first two columns are used in the first voting stage (one column per researcher). A cell in the column is filled either with 1 (the contribution is relevant) or 0. If a contribution is finally rated with 2, it is automatically in the set of contributions for further investigation. However, if a contribution is rated with 0, it is excluded from further investigation. Only if a contribution is rated with 1, it is marked to be judged in the secondary voting. The criteria for the secondary voting were (1) the title of the contribution, (2) the keyword list, and (3) the abstract.

In the second voting stage, we only considerd contributions that were not finally decided in the first stage and called in a third reviewer. This third reviewer also worked with the integrated table and voted by following the same criteria as in the first voting stage.

In the third and last voting stage, which is done by two researchers, we analyzed the results of the second stage, but extend the evaluation to the complete contribution by further conducting a in-depth analysis of the paper going beyond the title, the keyword list, and the abstract (inclusion criteria IC_4 and IC_5). The goal of this final stage was to figure out the key contributions on method engineering that are relevant for the in-depth analyses.

Result Set. Table 2.4 summarizes the set of the papers resulting from the collection and preparation phases. We summarize for each database the total number of results, the cleaned number of results after the first harmonization (removing duplicates), and after the multi-staged voting of the papers for their relevance. The overall list of the publications taken for the analysis can be taken from the reference section (respectively, from [49] including the classification and the assignments done during the mapping study).

Database	Total	Clean	Voting	Relevant
ACM Digital Library	210	210	44	14
Springer Link	60	60	22	18
IEEE Digital Library	210	210	22	11
Wiley	1120	381	34	5
Elsevier	50	50	23	12
DBLP	244	86	22	19
Misc	4	-	-	4
SUM	1898	997	172	83

Table 2.4: Overview: Secondary Sources.

2.4.2 In-depth Analysis

For the in-depth analysis, we applied a *social network analysis* (Sect. 3) in order to identify the key contributors (RQ3) that then serve as sources to answer *RQ4*. The identified key contributors and the respective contributions are listed in Table 2.5 (cf. [49] for a detailed analysis of the social network graph).

2.4.3 Investigating RQ4 – Crafting the Metamodel

Based on these sources, we intensively investigated the remaining contributions in order to answer *RQ4*. We searched for (situational) method engineering terminology and concepts, and structured

Author	Publications
Brinkkemper	[1, 13, 11, 14, 21, 79]
Harmsen	[4, 5, 14, 21, 20]
Henderson-Sellers	[1, 22, 23, 24, 25, 26, 27, 28, 29, 51, 61, 62]
Karlsson	[1, 35, 36, 37, 38, 81]
Gonzales-Perez	[1, 19, 23, 24, 25, 27]
Ralyte	[1, 23, 24, 26, 54, 63, 64, 65, 66]
Rolland	[64, 65, 66, 67, 68, 69]

Table 2.5: Overview: Sources of key contributors.

the out comes using a spreadsheet which then served as basis to create an initial glossary (App. A). Finally, by using the glossary, we created a metamodel for method engineering (Sect. 5). The steps to create the glossary and the metamodel are described in detail in the respective chapters.

2.5 Quality Assessment

To ensure the quality of the result set, we rely on researcher triangulation (cf. Sect. 2.4.1). We applied a rigorous procedure to analyze and select the contributions relevant for further investigation. In addition, we also applied different techniques to continuously check the result set, e.g. we created a number of *tag clouds* (Sect. 3.2) in order to check the abstracts and the keywords for entries not relevant to our research.

To increase the validity of the inferred metamodel, we rely on design workshops in which a group of 2-4 researchers worked on the term-wise analysis and modeling. Furthermore, a complementing Master's project was conducted in which the resulting model (or parts of it) were analyzed from the tool development perspective.

3 Analysis

In this section, we present the instruments used in the analysis of the relevant literature. Due to the challenges coming with analyzing large literature pools, we relied on a selection of different (tool-based) instruments to ensure the suitability of the selected literature and its analysis. We first describe the process of creating the tag-clouds, which were used to analyze whether all selected contributions are relevant for the study. For the identification of significant key contributors, we used the tool Gephi, which we briefly introduce, before discussing the instrument of research type facets to analyze the maturity of the investigated literature.

Chapter Overview

3.1 Introduction

One of the major challenges in analyzing the outcomes of a literature search is ensuring the suitability of the result set, its structuring, and the selection of meaningful instruments to conduct an in-depth analysis of the selected contributions. For handling the result set, we first had to clean the result set. In Sect. 3.2 we thus discuss, how we instrumented tag clouds to (1) check the validity of the result set and (2) to create a reference in which the authors' self-perception meets the objective classification using research type facets. In Sect. 3.3, we introduce the Gephi tool, which we used to investigate the key contributors of the considered domain and to analyze relationships among the authors. Finally, we briefly discuss the research type facets [58] that we used to objectively categorize the selected contributions in Sect. 3.4.

3.2 Tag Cloud

Tag clouds are a visual representation of a weighted list of words. Usually, words are arranged in alphabetical order and the size of the words is scaled according to the frequency of the occurrence of the respective word/term. Tag clouds have a broad appeal in the internet. Especially search engines or e-commerce application often use tag clouds to visualize the most widely used requests or topics of interest.

In the context of our investigation, we used tag clouds to filter the most important concepts and contents. Furthermore, we used tag clouds to check the validity of the result set emerged from the literature search, e.g. if the word "chemistry" occurred in a tag cloud, we knew that we still have to filter/clean the result set before starting the analyses.

3.2.1 Tool-based Creation of Tag Clouds

Since tag clouds are popular nowadays in the age of Web 2.0, a variety of tools supporting the creation of tag clouds exist. A selection of tools supporting the generation of tag clouds can be depicted from Table 3.1.

Tool	References
WORDLE	`http://www.wordle.com`
TAGCROWD	`http://tagcrowd.com`

Table 3.1: Selection of tools supporting the generation of tag clouds.

There are tools which mainly focus on the graphical representation, others focus more on the content. Most content-focused tools are able to remove common words of the chosen language, e.g. "of", "the". However, many tools cannot handle large amounts of text, or cannot save the generated tag cloud. For instance, *Wordle* provides rich functionality to change the settings regarding the layout of the generated cloud, however the only possibility to save the result is a screenshot. Finally, we used the tool *TAGCROWD* for several several reasons:

- The tool supports multiple languages and thus allows for skipping common words of the respective language as well as selected individual word (reduction of the terms considered in the analysis).
- The tool allows for investigating the frequency of word use.
- The tool allows for configuring the set of words to be displayed, e.g. words occurring just once or twice can be excluded from further analyses.
- The tools allows for grouping "similar" word (upper/lower case variants etc.).

3.2.2 Creating the Tag Cloud

We now briefly describe the use of the selected tool to generate the tag clouds. The handling quite simple: The text that shall be used to generate the tag cloud can be uploaded in three different ways: The easiest way is to paste the text of maximal 500 kB into a textfield. It is also possible, to add a web page URL or to upload an text file of maximal 500 kB. The resulting tag clouds can, finally, be saved as pdf-file.

3.2.3 Approach and Results

In order to finally analyze the investigated literature, three different types of tag clouds were created. The first tag cloud was generated using the abstracts of all the papers, which were relevant for the study (Figure 3.1). The second tag cloud was created using the the abstracts of those papers that were considered relevant for further investigation (Figure 3.2). Finally, the third tag cloud should be generated using only the keywords of all papers considered relevant. However, some papers did not contain a sufficient set of keywords and, thus, only one tag cloud was created using only those papers of the special selection (Figure 3.3).

Figure 3.1: Tag cloud generated from the abstracts of all papers in the cleaned result set.

In addition to the use of tag clouds to visualize the outcomes of the filtering procedure, tag loads can also be used in order to determine the validity of the (cleaned) result set. For instance, if the search is on papers from the software engineering domain, papers, e.g., from chemistry, are out of scope. Since, the elimination of papers according to the title and can be incomplete, a tag cloud can be used to screen abstracts and/or keyword lists in order to find “intruders” that point to papers considered off-topic.

Figure 3.2: Tag cloud generated from the abstracts of papers categorized as R*

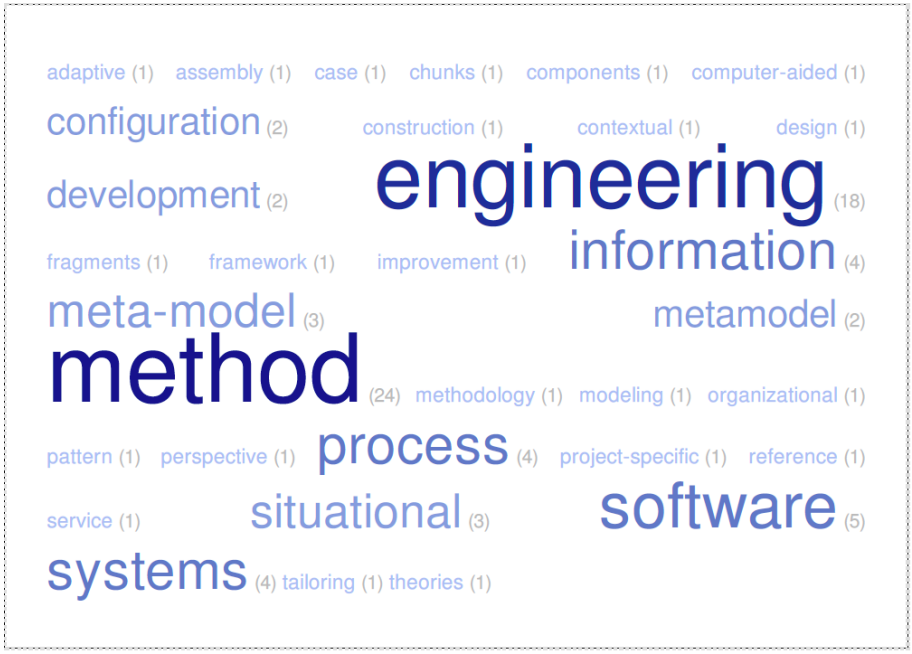

Figure 3.3: Tag cloud generated from the Keywordsof papers categorized as R*

Anyway, we recommend to continuously generate tag clouds to also check whether the cleaning process leads to a result set that meets the expectations, e.g. if the search is—as in our case—on method engineering, continuously generating tag clouds shows an increasing size of the term "method engineering" in the tag cloud (precision of the result set).

3.3 Social Network Analysis

In order to get an overview about the network in which authors collaborated, we decided to use Gephi (`http://gexf.net`) to create social networks from data tables. Gephi was used to analyze the authors, their collaboration, and the year of the collaborations to distill networks in which, e.g., concepts were defined, or terms were created. Furthermore, the resulting networks were used to figure out which contributors are recognized as the major drivers in the field and, eventually, which contributions have to be considered of special importance (we categorized these as *R** [49]) and should become subject to in-depth analyses.

In this section, we briefly introduce the Gephi tool and describe how we used Gephi to analyze the data set.

3.3.1 Gephi – An Overview

Gephi is an open source tool for creating, analyzing, and manipulating networks, also with larger amounts of edges and nodes (up to 50,000 nodes and 1,000,000 edges). The first release was published in 2008. The tool provides several ways of data import with different features, for example as a spreadsheet or a GEXF-File (Graph Exchange XML Format).

The use of Gephi is not always straightforward. The user interface provides three views: *Graph* (Overview), *Data Laboratory*, and *Preview*:

Overview	In this view, users have three windows of which one is the "graph", and the others are the data tables and the preview In the graph perspective, the social network is only foreshadowed by the nodes and their connections without labels and weighting.
Data Laboratory	The data laboratory allows to add or delete edges and nodes and to manually manipulate the data set.
Preview	As the name says, this view presents a preview so that the user can see what the network (could) look like.

3.3.1.1 Input/Import

Gephi provides different ways to import data[1] with different features (cf. Figure 3.4). For the actual studies, we decided to upload our data as a `csv` file and the `gexf`-Format, but we had to notice that Gephi only provides the possibility to upload either the nodes or the edges as a `csv`-file, and fill in the edges/nodes by hand. If both, edges and nodes, are uploaded together in a `csv`-file, Gephi is not able to combine the source- and target-IDs to the predefined node-IDs. Even if one exports data from Gephi as `csv`-files and import them in a new workspace it is not able to assign the edges to the nodes.

Thats why we decided to use the `gexf` format. We imported the dataset of nodes and edges as a `gexf`-file. The following listing shows an example of the used `gexf`-file:

1 `https://gephi.org/users/supported-graph-formats/`, cf. Figure 3.4

Figure 3.4: Possible formats for data upload and features.

```
<gexf version="1.1">
<meta lastmodifieddate="2010-03-03+23:44">
  <creator>Gephi 0.7</creator>
  </meta>
  <graph defaultedgetype="undirected" idtype="string" type="static">
        <nodes count="160">
          <node id="0.0" label="Agerfalk,␣P.␣J."/>
          <node id="1.0" label="Aharoni,␣A."/>
          <node id="2.0" label="Asadi,␣M."/>
          ...
        </nodes>
        <edges count="␣␣">
          <edge id="0" source="61.0" target="64.0" paperid="1" label="2007"/>
          <edge id="1" source="61.0" target="47.0" paperid="1" label="2007"/>
          <edge id="2" source="61.0" target="11.0" paperid="1" label="2007"/>
          ...
        </edges>
```

Problems and Experiences As mentioned before, the use of Gephi was not always straightforward. For instance, two problems occurred during data upload: During data import, an error occurred, which pointed stated that Gephi does not support parallel edges, and that those edges were ignored for the import. As the number of edges was far below the announced limits of Gephi, this problem could quickly solved by choosing "directed edges." As the raw number of collaboration was only of minor interest for our investigation, collaborations of authors in the same year were then combined to one edge (our main focus was to detect the networks of collaborations and the key contributors).

However, this lead to the next problem: In the classes for edges and nodes that are predefined in Gephi (using the GEFX format), it is only possible to add weightings to edges, but not to nodes. Both, Gephi and the GEXF format, provide solutions (or at least work-arounds) to solve that problem. For instance, one can directly edit the imported data record in Gephi and add a new column that should be used for weighting, e.g., the number of papers an author was involved in. Another solution would be to overwrite the predefined classes for the nodes in Gephi and to add required attributes, e.g., for the weightings. For the actual investigation, we decided to manually extend the structure of the data table, and, thus, enable Gephi to calculate weighting on the extended data structure.

3.3.1.2 Layout and Settings

Having imported the data for consideration, Gephi, by default, creates a simple network according to the information available in the input data set (Figure 3.5). This initial graphs only reflects the raw data and is not configured at all.

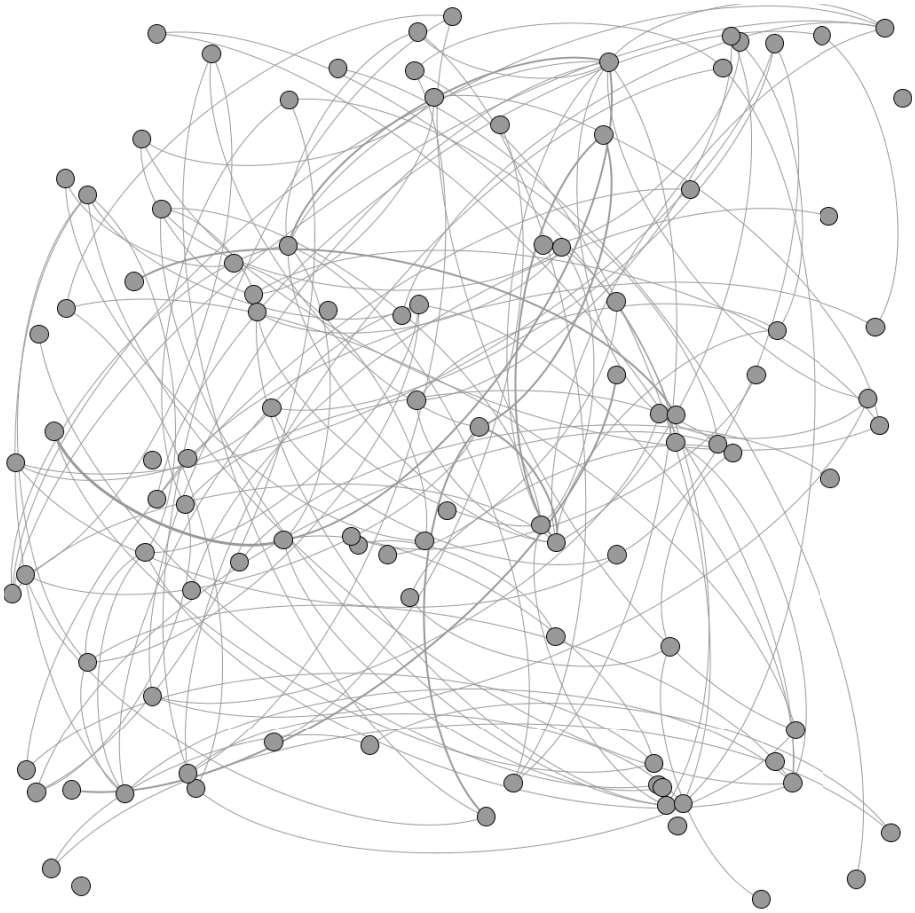

Figure 3.5: Edges and nodes after data import (non-configured, raw view).

Graph Layout Algorithms To overcome the shortcomings of the randomly generated graph, Gephi provides the user with manifold options the configure the layout. In Table 3.2, we briefly summarize the algorithms implemented by Gephi that allow for manipulating the layout of the generated graph. Moreover, it is always possible to manually modify an auto-generated layout, e.g., by changing node positions.

Weighting To support the analysis, we looked for weighted graphs in which the relative size of the nodes reflects the importance of a particular contributor in the field. However, as mentioned before, some kind of collaboration would require to have multi-graphs (in this specific context: multiple and parallel edges), a features that was not yet implemented in the version of Gephi that we used for our studies. To overcome the shortcoming, we decided to (manually) group parallel edges and to add the weights as attributes (as discussed before).

Furthermore, we wanted to show the year of collaboration as label of an edge. We implemented this by using directed edges that were labeled with the year of collaboration. That is, we could also realize “parallel” edges, which were distinguished by the year of the contribution. The outcome is

Algorithm	Description
(Counter-)Clockwise Rotate	With the same settings for the parameter angle, both algorithms do exactly the same, namely they both turn the graph the predefined angle. For example for the value 90 the graph turns 90 degrees to the left, for -90 to the right.
Contraction/ Expansion	As the names of the algorithms tell these algorithms reduce or increase the size of the graph by decreasing/increasing the distance between the nodes. The size of the nodes stays the same. With the same parameter for the scale factor, both algorithms do the same. Again the only difference between the both algorithms are the predefined settings of the scale factor.
ForceAtlas	"Quality Layout: a linear attraction linear-repulsion model with few aproximations(BarnesHut). Speed cutomatically computed."
Fruchterman Rheingold	The Fruchterman Rheingold Algorithm is a force directed layout algorithm. For illustration of force directed networks physical phenomenons are used. The nodes are represented by steel rings, which push each others away like electrical force, whereas the edges represent springs, which pull the node which are connected close together.
Label Adjust	This algorithm should make all labels readable by repositioning them but it didn't move anything by applying this algorithm
Random Layout	As the name says, the nodes get distributed randomly over an area of predefined size.
Yifan Hu	"Original Yifan Hu's attraction-repulsion model. Reduce the computional cost by restricting force calculation to the neighborhood. The algorithm stops itself, as it has an adaptiong cooling scheme."
Yifan Hu Proportional	Modified version of Yifan Hu, that uses proportional displacement scheme.
Yifan Hu Multilevel	The Yifan Hu Multilevel is also a modified version of Yifan Hu, which is also able handle big amounts of data by reducing computation.

Table 3.2: Gephi layouting algorithms (summarized).

shown in Figure 3.6. In the resulting graph shows that there are many non-coherent small networks (isolated graphs) and one big network of authors with the key contributors, which are listed in table 2.5. This graph was then used to determine the key contributors and to select their contributions for further in-depth analyses.

3.4 Research Type Facet

In order classify the contributions and the investigate their maturity and "soundness", we applied techniques well-known from systematic mapping studies as recommended by Peterson et al. [58]. Furthermore, we use a classification according to research type facets as proposed by Wiering et al. [80] (cf. Sect. 2.2). As research type facets, we use the following categories for classification:

Validation Research Techniques investigated are novel and have not yet been implemented in practice. Techniques used are for example experiments, i.e., work done in the lab.

Evaluation Research Techniques are implemented in practice and an evaluation of the technique is conducted. That means, it is shown how the technique is implemented in practice (solution implementation) and what are the consequences of the implementation in terms of benefits and drawbacks (implementation evaluation). This also includes to identify problems in industry.

Solution Proposal A solution for a problem is proposed, the solution can be either novel or a significant extension of an existing technique. The potential benefits and the applicability of the solution is shown by a small example or a good line of argumentation.

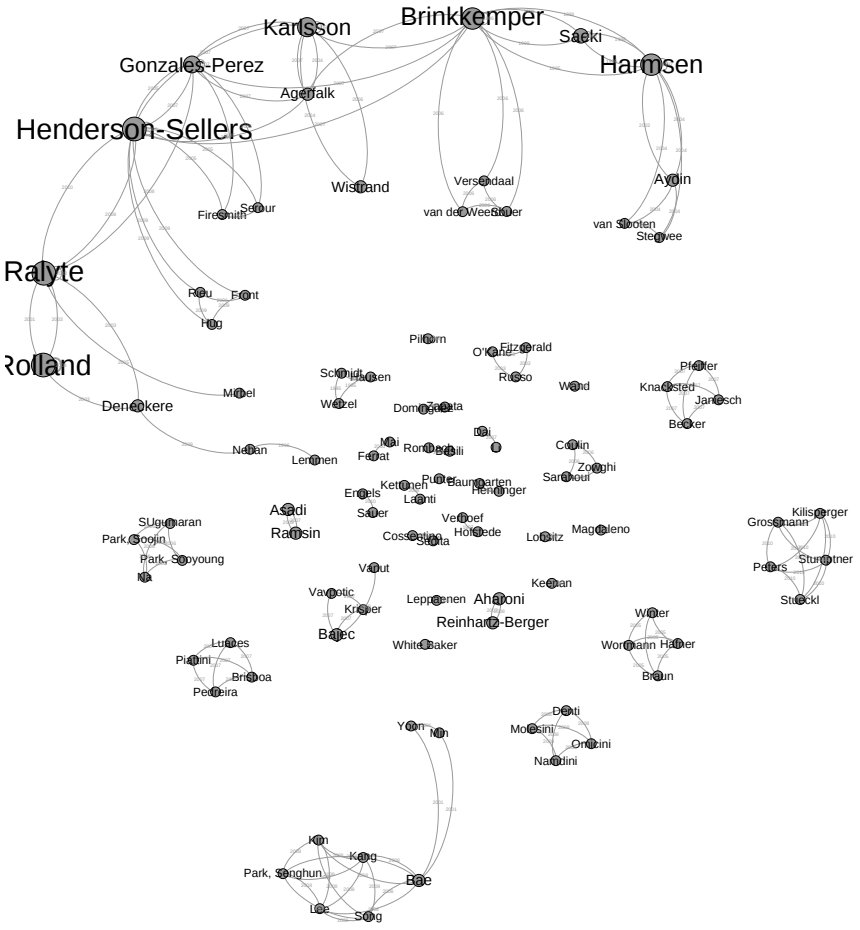

Figure 3.6: Network of authors (all contributors, configured network).

Philosophical Papers These papers sketch a new way of looking at existing things by structuring the field in form of a taxonomy or conceptual framework.

Opinion Papers These papers express the personal opinion of somebody whether a certain technique is good or bad, or how things should been done. They do not rely on related work and research methodologies.

Experience Papers Experience papers explain on what and how something has been done in practice. It has to be the personal experience of the author.

The evaluation according to the research type facets is presented in detail in [49]. Figure 3.7 shows the visualization of the evaluation results. The figure shows most of the analyzed contributions to be classified into *solution proposals* and *philosophical papers*. In [49], we thus conclude that the Method Engineering research community keeps looking for feasible concepts. Furthermore, we cannot yet judge whether Method Engineering "made it" into practice. So far, our findings show a number of proposals, however, similar to Hofstede and Verhoef [75] we have to conclude

that empirical evidence on the feasibility is still missing. A detailed discussion can be depicted from [49]. In the remainder of the report at hand, we further discuss our understanding of Method Engineering, and we also discuss our approach regarding its systematization.

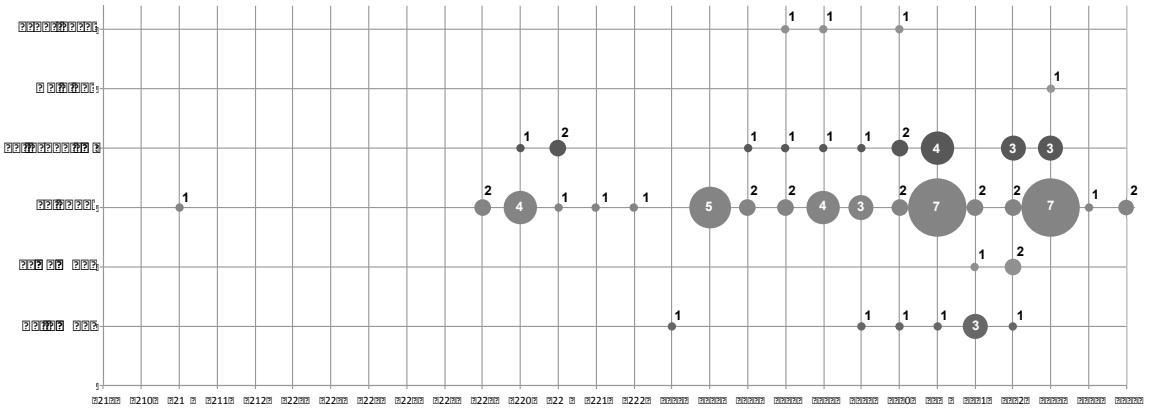

Figure 3.7: Evaluation of the research type facets of the result set.

4 Method Engineering Terminology

As our first analyses on (situational) method engineering showed a blurry and partially inconsistent terminology, the first step to craft a metamodel is the in-depth analysis of the terminology and the concepts associated with the used terminology. In this section we first create an method engineering glossary, which we use to develop a consolidated terminology that serves as taxonomy for the method engineering domain.

Chapter Overview

4.1 Research Method

In order to analyze the SME terminology, we analyzed the the contributions from the SLR (Sect. 2) for concepts and definitions. To this end, we created a spreadsheet having the data structure as shown in Table 4.1.

Data	Description
Term	This cell names the term crafted from literature.
Alternatives	In this cell, alternatives to the previously identified term are named.
Definitions	If available, a (precise) definition is given. For some terms, different authors provide individual definitions. All definitions are collected and listed in the cell for later analysis.
Further Information	Sometimes, the identified authors provide examples that can be used to sharpen a definition. Such information is stored in this cell.
Further References	If a term is mentioned by an author, but the author refers to another source instead of giving an own definition, the respective external sources are collected here.
Contributions	"Contributions" is a set of columns. There is one column for every identified author from the SLR who was considered of special relevance. In the cells, the letter "D" indicates this contribution defining a term and the letter "E" shows the reuse and extension of a previously defined term.

Table 4.1: Data structure for the SME terminology analysis.

As a first step, the spreadsheet was filled with the information from the contributions to build an initial glossary. After the spreadsheet was complete, the glossary was created. The glossary served as the basis for the third step in which a taxonomy was created using an UML modeling tool. The raw data as well as the initial glossary can be depicted from Appendix A.

4.2 Creating the Method Engineering Glossary

The glossary that contains all relevant SME terms was created using the data structure introduced in Table 4.1. Appendix A comprises the details regarding the terms and the respective definitions and classifications. A summary of the results from the analysis can be depicted from Table 4.2. In the following, we use Table 4.2 for discussing and selecting the (final) glossary used for the creation of the taxonomy.

In order to create the taxonomy, the resulting terms from the literature study were analyzed and categorized. For the purpose of creating a metamodel, we were primarily interested in those terms that were categorized as *structural* terms. However, the analysis of the found terms also showed many terms that are of methodical nature, or that describe model elements to be considered as attributes. To this end, in Table 4.2, we list all terms, summarize the categorization and indicate whether a term is skipped or used for the creation of a metamodel. If a term is skipped, we perform no further in-depth analysis.

Table 4.2 shows the summary of the terms considered in the metamodel crafting procedure. The table shows 37 terms that were considered relevant after screening the key contributions, which were identified in the SLR. After several detailed analysis workshops, finally, 22 terms remained in the candidate set for further investigation (cf. Sect. 5). The terms—that were considered relevant—capture structural concepts required to construct an artifact model. For this, we especially investigated concepts, such as `Actvitiy`, `Actor`, and `Product`. These terms were investigated in

detail in order to collect structure, properties, attribute candidate, and a notion of the relationships between the artifact candidates.

No.	Term	struct.	meth.	tech.	relevant
01	Abstraction Level		✓		
02	Activity	✓			✓
03	Actor	✓			✓
04	Actor Role		✓		
05	Base Method	✓			✓
06	Conceptual Method Fragment	✓			✓
07	Guideline	✓			✓
08	Metamodel	✓			
09	Meta-Modeling		✓		
10	Method	✓			✓
11	Method Base			✓	
12	Method Chunk	✓			✓
13	Method Chunk Repository			✓	
14	Method Component	✓			✓
15	Method Configuration	✓			✓
16	Method Engineering		✓		
17	Method Fragement	✓			✓
18	Model		✓		
19	Modular Method	✓			✓
20	Process	✓			✓
21	Process Configuration	✓			✓
22	Process Domain Metamodel	✓			✓
23	Process Fragment	✓			✓
24	Process Manager Fragment				
25	Process Model		✓		
26	Process Role	✓			✓
27	Process Type	✓			✓
28	Product	✓			✓
29	Product Fragement	✓			✓
30	Product Model	✓			✓
31	Repository Fragement			✓	
32	Situational Method		✓		
33	Situational Method Engineering		✓		
34	Software Development Process		✓		✓
35	Task Model	✓			✓
36	Technical Method Fragment			✓	
37	Tool Fragment			✓	

Table 4.2: SME terms and evaluation (overview).

On the other hand, we decided to exclude *technical terms*, e.g., `Method Chunk Repository` or `Tool Fragement` from the construction procedure, as we consider such aspects of little importance for a general artifact model (we get back on this topic in Sect. 5).

One finding that becomes obvious from Table 4.2 is the heterogenous (partially competing) terminology that we already complained about in [49]. For example, we found different types of `Fragments`, and we also found the concept `Method Chunk`—both frequently discussed in literature, e.g., [24, 23, 28].

5 A Method Engineering Metamodel

A metamodel for method engineering should serve the analysis of software processes, the definition of a (conceptual) process language, and the implementation of software processes using tools. In this section, we present how the method engineering metamodel was crafted. Furthermore, the implication on software process analysis and design as well as for software process implementation will be initially discussed.

Chapter Overview

5.1 Introduction

In order to construct a metamodel for method engineering, we have to answer the question first, which kinds of models shall be created on the basis of a method engineering metamodel. To this end, we fist need to define our notion of method engineering and where method engineering contributes to the construction and customization of a software process.

Screening the literature on (situational) method engineering, we find an inconsistent terminology that is partially underspecified, or contradictory defined. Therefore, from the analysis of the state of the art in SME literature, we craft an initial metamodel and test the result for its feasibility regarding an appropriate modeling support for software processes.

5.2 Initial Metamodel – A Method Engineering Taxonomy

In Figure 5.1, we show the initial metamodel that emerged from the literature analysis. In this initial metamodel, all terms harvested from the in-depth analysis were considered, clustered, and integrated if they could substantially contribute the the metamodel.

In the metamodel, we find different areas focussing on different topics. First, as basic design paradigm the artifact-based design approach was applied. This design approach is based on an artifact metamodel [53], which is a generalized metamodel crafted from the V-Modell XT metamodel [77] (based on the work [41, 76]). The key element is a *composite pattern* in which two elements are the key players: a `Fragment` is a composite element that has a `FragmentType`, and comprises further elements of `FragmentType`.

The `FragmentType` is, furthermore, the core element used for creating methods (class `Method`), and composites and configurations. Furthermore, a `FragmentType` is the abstract base class for further specialized fragment types. In the model—based on the SLR results—we find the children `RoleFragment`, `ProcessFragment`, and `ProductFragment`. Between these fragment types, we find *fragment dependencies*, which are initially defined as association class `FragmentDependency` linking instances of `Fragment`. `FragmentDependency` is the base class for further refined children:

- `RoleProcessFragmentDependency` links role and process fragments.
- `RoleProductFragmentDependency` links role and product fragments.
- `ProcessProductFragmentDependency` links process and product fragment.

The different fragment and fragment dependency types together form the concept of a *Method Chunkt* as proposed in literature on SME. In the presented model, we do not have an explicitly defined method chunk element, as the characteristics of a method chunk are an inherent property of the metamodel.

The second part of the metamodel shows the step-wise composition of (atomic) elements into comprehensive structures. For instance, the metamodel defines a `Product` to be comprised of different product fragments. Furthermore, a collection of products forms the `ProductModel`. The same pattern is applied for processes and roles.

Hint: The most recent definition of a software process is done using the combination of different (sub-)models that together form a software process, namely a role model, an artifact model, a process model, and so forth. This is different to the notion of a software process used in SME. In today's literature, the set of models (role, artifact, etc.) defines the pools of different process assets that is normally defined as *prescriptive software process* [55].

An `Activity` is designed as a reusable asset that manipulates a `Product` and which is performed by a `Role` (that comprises different role fragments and also links the `Actor` class that, itself, connects role-, process-, and product fragments) and, finally, comprises different processes. That is, an `Activitiy` is a composite that can be part of a `Method`.

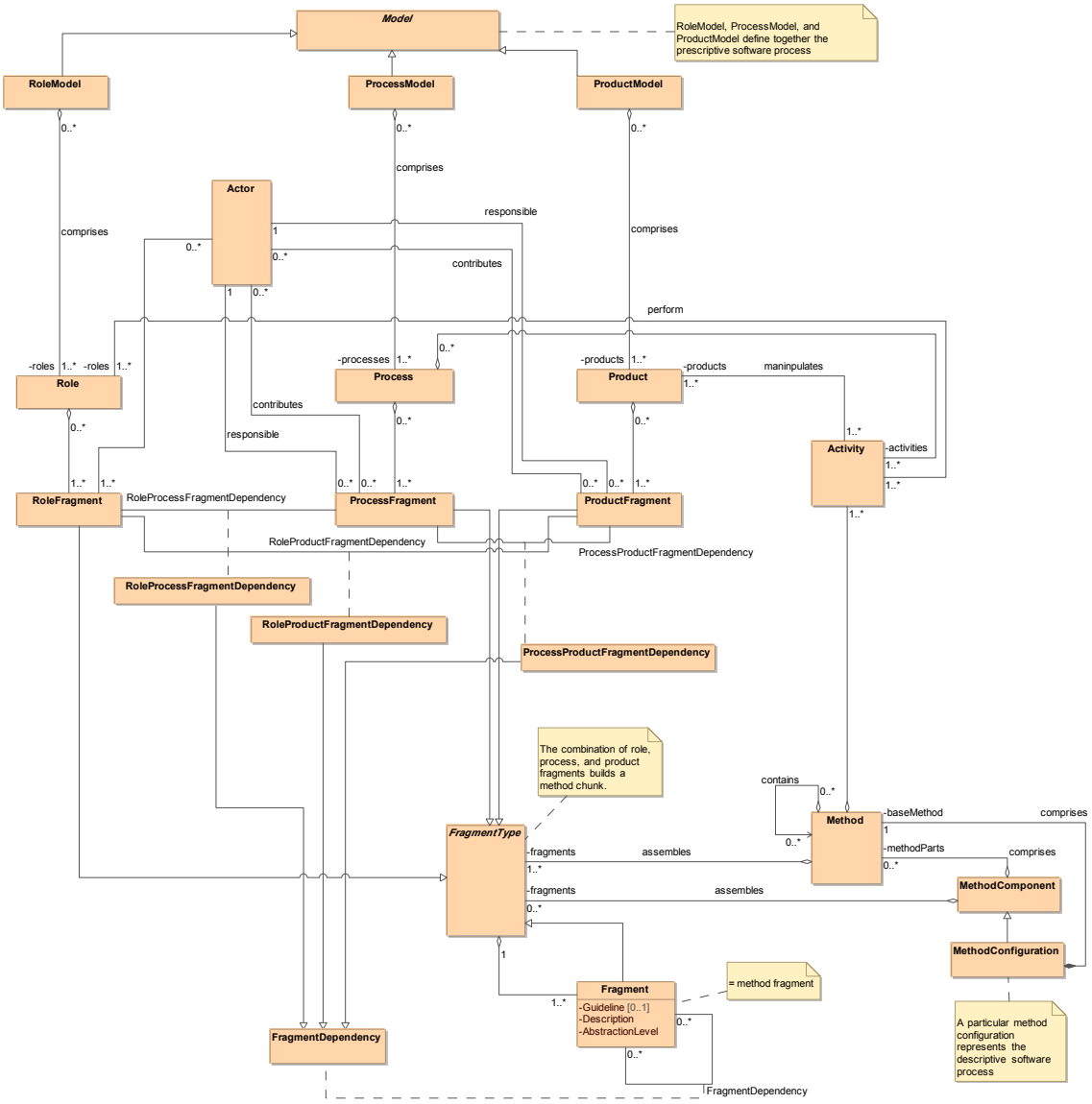

Figure 5.1: Initial metamodel crafted from the literature review.

Finally, a `Method` is package that comprises at least one `FragmentType`, (optional) further methods, and at least one `Activity`. To this end, the `Method` ensures that (at least) one `Activity` is comprised, which itself comprises roles, processes, and products that are each composed of fragments. That is, it is ensured that methods are not empty (in terms of being just placeholders), but always contain assets that allow for describing certain development and/or management activities. A `Method` can also be considered to be an *atomic* container that contains a cohesive and consistent set of fragments. A comprehensive software process is thus an integrated set of methods. The model contains two elements that allow for assembling comprehensive methods. The first element is the `MethodComponent`, which represents a reusable combination of methods that can, e.g. build a (partial) predefined software process. This construct is comparable to the *Method Plug-In* as defined in SPEM [57] or the *Process Module* as defined in the V-Modell XT [77]. A `MethodComponent` comprises either methods or fragments, whereas the set of comprised elements must not be empty (at least one `Method` or one `FragmentType` must be present). A special representation of a `MethodComponent` is the `MethodConfiguration`. The `MethodConfiguration` explicitly contains one `Method`, which is the *base method* for the entire configuration.

All other capabilities are inherited from `MethodComponent`, which means that a `MethodConfiguration` comprises a base method and further methods and/or fragments that together form a constant process. As a configuration is comparable to a tailored software process (whereas we yet do not distinguish between static and dynamic tailoring), according to Münch et al. [55] a particular `MethodConfiguration` can be mentioned to be a *descriptive software process*.

5.2.1 Discussing the Initial Metamodel

The metamodel shown in Figure 5.1 reflects the outcomes of the in-depth analysis of the systematic literature review conducted to determine the current state of the art in method engineering. This metamodel can be used to model software processes or parts of it. However, compared to today's software process metamodels [45, 74] the inferred metamodel has several flaws:

- The inferred metamodel has a still unsatisfactory terminology. For instance, in SME, literature intensively discusses fragments or, competing, method chunks and, furthermore, a number of terms and concepts does not contribute to process construction given state of the art software process metamodels.
- Although, SME claims to support the construction of flexible methods, the inferred metamodel does not allow for integrating life cycle models. However, the inferred metamodel is based on the terminology gathered from the outcomes of the SLR only and, thus, is limited to the results from the SLR that may contain gaps. A number of SME-related metamodels were already proposed (a collection and summary can be depicted from [26]). Every metamodel that is presented by [26] aims to address a (slightly) different focus; often, these metamodels are overlapping or discussing selected aspects from different angles. The inferred metamodel shown on Figure 5.1 presents an integrated view that is based on the commonalities reported across the analyzed contributions from the SLR and, thus, adds another perspective. It remains unclear, how this particular metamodel adequately catches the SME idea, although it is crafted from the (few) agreed concepts.
- The inferred metamodel does not contain any lifecycle information, as such information was not provided in a reusable manner in the investigated contributions. Again, [26] collected some life cycle models that mostly cover the construction of methods or their improvement, respectively, e.g. [65, 66]. A generic approach is for instance described in [64].

5.2.2 Beyond the Result Set

As already mentioned, the distilled metamodel is based on the set of identified key contributions from our literature study. We consider this metamodel incomplete. For instance, newer work, e.g., as contributed by Engels and Sauer [16], provides a more detailed perspective on method engineering. Their method engineering proposal puts more emphasis on tools and, thus, is more focused on the precision of the concepts. Figure 5.2 (extracted from [16]) presents a top-level perspective on their approach.

Comparing the MetaME model with the one distilled from our investigation, we find some similarities as well as significant differences. First, explicit 'models', e.g., `RoleModel` and `ProductModel` as proposed by the study's result set, cannot be found in [16]. Instead, [16] shows some inspiration from the SPEM metamodel, e.g., the central position of the class `Work`, which is the root for activities and processes. Furthermore, the MetaME model adds more context information to the model, e.g., by relating roles to an organization, and regarding a project as part of the method engineering model.

Beyond a structure model (Figure 5.2), MetaME also defines a process model to instrument the metamodel. The construction procedure for MetaME is presented in Figure 5.3 (extracted from [16]). The construction procedure is, however, limited to the construction of a (new) method,

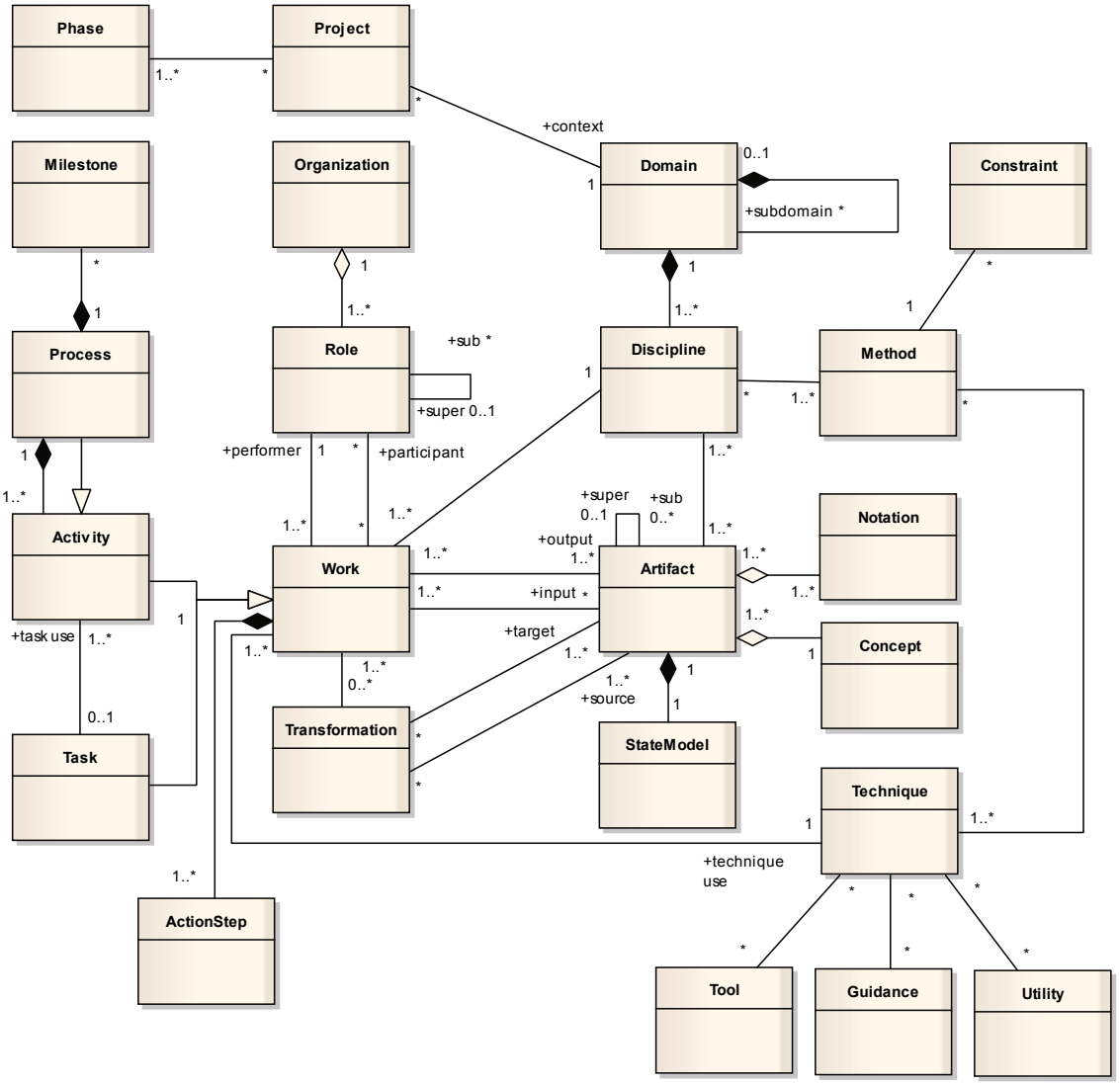

Figure 5.2: Metamodel of MetaME (according to Engels and Sauer [16]).

and shows similarities to the *Essence* approach [30]. Management of processes (or of particular methods) does not become obvious from this procedure.

We consider this being a major flaw that we also observe in the other method engineering literature. Method engineering on the organizational level is considered in, e.g., Becker et al. [8] (reference modeling techniques), Karlsson and Ågerfalk [37] (creation of reusable process assets), and Kellner et al. [39] (presenting the IDEAL model).

5.3 Proposing a Metamodel supporting Life Cycle Management

We want to conclude this report by proposing a slightly different perspective on method engineering. For this, in this section, we first discuss what our notion of method engineering is, what the purpose of method engineering should be and, consequently, which problems should be addressed by a method engineering approach, and, finally, what requirements emerge from that. Afterwards, based on our experiences in software process modeling and metamodelling, we construct metamodel that reflects the aforementioned points, and discuss trade-offs and potential flaws.

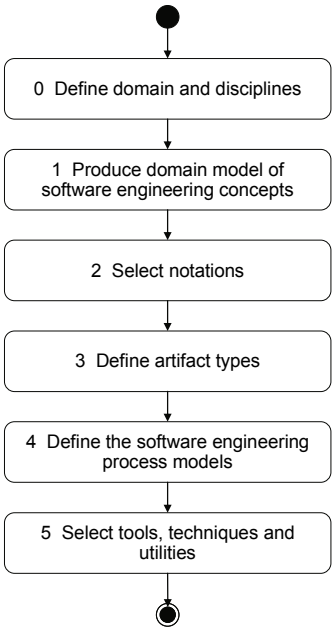

Figure 5.3: Method construction procedure of MetaME (according to Engels and Sauer [16]).

5.3.1 Method Engineering – What for?

According to the well-known definitions, we consider method engineering an approach to create flexible development methods. What does this mean in detail?

- Software development is operated by development teams, which are hosted by organizations. For this, a development method must clarify the notion of a particular method: Is it a method addressing the organization layer, or is it a method describing a project, or a method that is executed within a project.

  In the past, we used the terms *macro-* and *micro processes* to provide a differentiation of those methods. Münch et al. [55] speak of *engineering-* and *development processes*. However, flexibility is tightly coupled to the respective context, e.g., at the organization layer, flexibility usually addresses a family of processes that follow a common blueprint in order to fulfill compliance requirements (cf. software process lines, Rombach [70]), and on the project layer, flexibility can be understood as a pool of equivalent methods to be selected to solve a particular problem.

- Software processes—same as software systems—age and evolve. Therefore, a method engineering approach must address the process life cycle. However, the life cycle of process, again, depends on the kind of the process. For instance, small processes/methods that are used within a project, and that do not impact an organization-wide process, have—if at all—a different life cycle than an organization-wide deployed standard process. Context and granularity of the process dictate requirements regarding the process life cycle. Furthermore, evolving processes are hard to manage. For instance, in [56, 73, 72], the group around Münch investigates analyses techniques to figure out particular modification caused by process evolution. However, they investigate the evolution in an *ex-post* manner. Another approach is presented by Kuhrmann et al. [46] in which we analyzed the feasibility of a metamodel-based approach to direct process variant management and evolution.

- Software processes can become quite comprehensive. They can contain up to several thousands of process elements (e.g., roles, activities, work products, and so forth). Thus, assembling a (project-specific) method is a challenging task that requires powerful tailoring

mechanisms (Martínez-Ruiz et al. [52] speak of *tailoring constructors*). In order to provide meaningful tailoring instruments, processes/methods need a solid basis on which they can be constructed. In [45, 74], wis used to term *Software Process Engineering Framework* (short: process framework) to describe the infrastructure to allow for such design and comprehensive management tasks. The heart of a process framework is a well-defined software process metamodel that provides process engineers with a process language in which processes can be described properly. In [45], we found only the two metamodels SPEM [57] and V-Modell XT [77] providing sufficient support. Moreover, the SEMDM metamodel [32], which claims to implement method engineering comprehensively was found practically irrelevant, as no empirical evidence on its application nor its feasibility was found.

In a nutshell, in order to establish a meaningful method engineering, several requirements should be addressed:

Req 1: *A method engineering approach must link the* conceptual *approach to analyze, construct, and manage a software process to the respective process frameworks used to realize the process.*

As established process frameworks exist, method engineering—in our understanding—serves as methodical framework to use these process frameworks. Therefore, a method engineering approach should provide the guideline on how use these frameworks, which causes further sub-requirements:

- A method engineering approach must provide a common terminology to describe (concrete) process elements as well as the methodical aspects of process construction.
- A method engineering approach must provide interfaces or "hot spots" that can be used to bind particular process assets to construction and management tasks.
- A method engineering approach must be platform-agnostic in order to support different frameworks, but also to be applicable with different process construction, management, and improvement approaches.

Req 2: *A method engineering approach must integrate different levels of abstraction. Especially, a method engineering approach must provide—at least—organization- and project perspectives.*

As mentioned before, developing, deploying, and managing a software process happens on the organization layer as well as on the project layer. To this end, a method engineering approach must support (1) the company-wide definition of software processes (incl. their initial definition, their deployment, their management, and their continuous improvement), and (2) the project-specific adoption of software processes. For this, method engineering must include clearly defined tailoring mechanisms (e.g., based on so-called *customization levels* [47, 41]) to support a variety of development and customization scenarios, e.g., process-line-based software processes, and project-specific (micro) methods.

Req 3: *A method engineering approach must address the process life cycle.*

Most of the available literature on (situational) method engineering addresses the method construction (selection, assembly, etc.) from a "per-project" perspective. However, many companies establish software processes that require maintenance and improvement on a long-term basis. Therefore, method engineering must pay attention to the evolution of the software process, e.g., by also providing interfaces to administration processes, such as configuration-, change-, release-, and quality management. For example, a company-wide process is improved: A major problem occurs when discussing, e.g., the right deployment strategy, and the right training strategy [3, 2, 47]. Every decision that is made accordingly impacts the company, projects, and, eventually, the people. Therefore, a method engineering approach should support process engineers preparing such decisions and anticipating the effects on the different stakeholder groups in order to define appropriate strategies for, e.g., improvement and deployment.

5.3.2 An Artifact-based Metamodel for Method Engineering and the Software Process Life Cycle

In order to address the aforementioned requirements, we present a metamodel to lay the foundation towards an artifact-based method engineering approach. The presented metamodel was defined during the construction of the *Artifact-based Software Improvement & Management* model (ArSPI, [42, 43]). The illustrated metamodel links software process improvement projects and software process frameworks by providing a common terminology, which is based on the artifact-based design approach [53]. In this sense, the presented model is *not* a 'classic' method engineering approach. Moreover, this metamodel is an instrument to establish an artifact-based software process improvement and management that comprises software process analysis, construction, deployment, and long-term management and improvement. For this, a comprehensive method or *process engineering* approach, in this context, requires the following components:

1. A terminology model that captures the basic concepts and terms, and allows for coupling further methodical and technical aspects.
2. A software process improvement (SPI) approach to provide the methodical aspects.
3. A software process engineering framework to provide the technical aspects, e.g., metamodels and tools.
4. An organization to host and perform SPI endeavors.
5. Software development projects to use processes, and to provide feedback for further improvements.

In Fig. 5.4, we illustrate the relationship of the presented metamodel to the other mentioned components. In the following, we introduce the metamodel, and we provide a brief discussion on the remaining components. Furthermore, we provide references to related work conducted in this context to shape out the big picture.

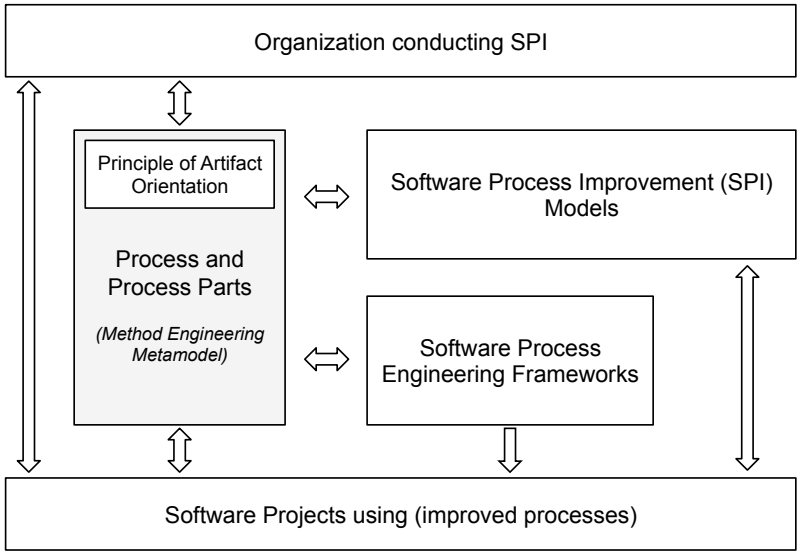

Figure 5.4: Relationship of the presented metamodel to the key elements of software process improvement and management.

5.3.2.1 The Basic Metamodel

In the following, we present the *basic metamodel*, which is referred in Fig. 5.4 as "Process and Process Parts." Figure 5.5 illustrates the basic model, which consists of two parts. The first

part is represented by the package `Artifact Orientation Base Classes`. This package provides the basic structures of artifact orientation according to [53]; namely the basic notion of an `Artifact`, its structure and content. In Fig. 5.5, we only mention the most relevant base classes, and omit further elements used for coupling artifact models and processes. Since the basic artifact model relies on the *composite pattern*, all artifacts (artifact types) used for the design of a method engineering approach are (hierarchically) structured by design, and, thus, we need not explicitly design composition abilities into the model as done in the crafted model (Fig. 5.1).

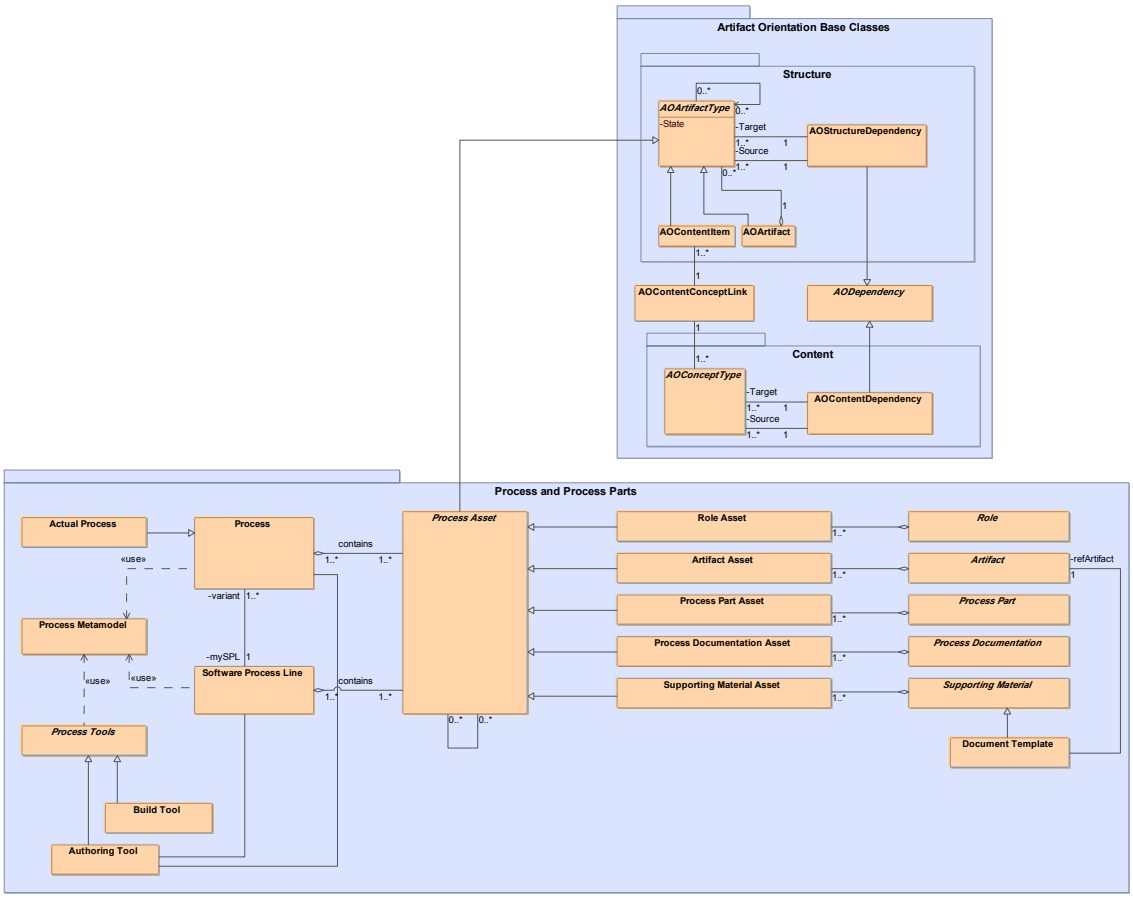

Figure 5.5: Basic method engineering metamodel (process parts and terminology).

The second part of Fig. 5.5 shows the basic metamodel addressing the processes and process parts. This base model addresses the aforementioned requirements 1 and 2. The heart of the model is the class `ProcessAsset`, which serves several purposes:

1. It provides an abstraction of process-related objects, which are subject to context- and process analysis and design. Thus, it allows for designing processes in a general manner without statically linking the elements of interest to a particular design language—one can talk about the concepts rather than technical issues.
2. It provides an abstraction of concepts, which are implemented in software process metamodels. Thus, designed concepts can be mapped to different realization platforms, e.g., SPEM-based or V-Modell-XT-based process models.
3. It provides a general element to express the composition of processes, and, also, a generic element that is subject to several administration and management processes, e.g., the subjects of a change management process are always process assets.

A `ProcessAsset` is also the key element to describe software processes. In the left part of Fig. 5.5, the relation of a `Process` and a set of `ProcessAsset`s is shown. We consider a particu-

lar software process to be a collection of process assets, whereby the configuration of the process assets is managed by the process assets themselves (according to [41]). Furthermore, beyond single (stand-alone) software process, we also provide a notion of how large-scale software process lines and single process variants relate to each other. We assume both, processes and process lines, rely on a software process metamodel, which serves as process language definition. Based on this metamodel, different supporting process tools, e.g., editors, build tools, and enactment tools, can be developed in order to support the different process stakeholders.

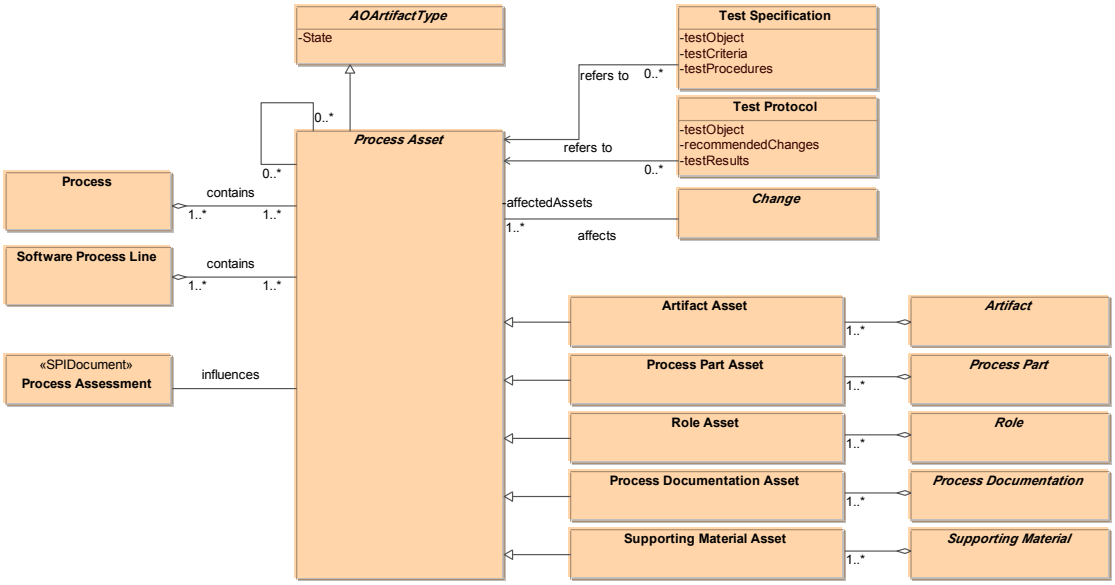

Figure 5.6: Basic links of the process asset.

These few structures build the basis to set up an artifact-based method engineering approach. They clarify the terminology by providing a generic design concept—the `ProcessAsset` (Fig. 5.6), which establishes links to concrete process metamodels, allows for platform-independent analysis and design approaches, and links higher-integrated concepts such as processes and process lines. Furthermore, the base model provides links to the complementing administration and management procedures, which we explain in the following sections.

5.3.2.2 Connecting Development and Life Cycle Models

In Fig. 5.7, we show the classes that link the engineering model and the life cycle models. In particular, we focus on the project organization and management parts as well as on the process life cycle management. As mentioned before, the `ProcessAsset` is the key element of the model to support the process development and (long-term) management activities. In terms of the (overall) management, process assets are considered in the context of processes (class: `Process`). A `Process`—especially the derived class `ActualProcess`—is usually subject to SPI projects, which includes process development, management, and improvement. For this, several management activities, which aim to improve a process, are linked to a `ProcessRelease`. A `ProcessRelease` is the version of a process that is shipped to a company and deployed for use. All management and improvement activities thus address a specific `ProcessRelease`[1], which is the basis for improvements.

1 This approach is comparable to a software life cycle management: If a software is maintained and/or improved, feature requests, bugs or issues always refer to a particular version of a deployed software. Improvements are thus based on an actual software release.

The particular way how an actual process is treated during the maintenance and improvement cycles is documented in a `ProcessLifeCycleSupport` (documentation) in which all required management and administration activities are documented. In order to establish a meaningful process management ecosystem, at least the following processes must be established:

- Change Management
- Measurement and Evaluation
- Training
- Deployment and Further Development

In the change management, feature requests and issues regarding an deployed process release are recorded and managed. The collected changes are input for the quality management, which is—on the company level—responsible to shape the new releases based on the collected issues. The change management must define all procedures that are applied in order to gather problems with a deployed process. Hence, the change management addresses a process as such, it collects problems across several projects of a company, and, thus, is an administration process that needs to be established at the company level[2].

Measurement and evaluation needs to be installed in order to gather information on the effectiveness and efficiency of deployed software processes. That is, as these activities address all projects using a particular process, measurement and evaluation is a family of administrative activities at the company level. The measurement activities, basically, comprise two perspectives: 1) the engineering perspective, and 2) the use perspective. In the engineering perspective, companies and process engineers are interested in the way the process fulfills, e.g., technical requirements, and certification goals. For example, if a company aims to reach a certain certification, e.g., ISO 9000 or a certain CMMI level, a process needs to be assessed in regular intervals. These goals are of an organizational—and often of a strategical—nature. On the other hand, user-based evaluation aims to investigate the perception of process consumers, i.e. if a process is valuable, e.g., by providing sufficient guidance. Depending on the specific goals—be it goal- or problem-driven ones—metrics need to be defined to gather information regarding the process implementation. Example metrics are for instance, CMMI levels, cost estimation precision, bug fixing time/cost, and so forth (cf. Kan [34] for examples). *Note*: A continuous evaluation of a software process is necessary in order to determine whether a process is still efficiently implement, and, moreover, whether a process really supports a company to achieve the set goals. This means, a process needs to be valuable, e.g., by helping a company to get a certification, and by providing projects with meaningful support.

In the training, all activities need to be planned that are necessary to train the company's personnel. The development of an adequate training strategy is, however, an challenging task [3]. Stakeholder groups need to be identified, training material needs to be created accordingly, and training needs to be scheduled. Notably, if a company has a process that evolves over time, it must be carefully determined, what the differences are, how training programs need to be updated, and who needs to be trained (again).

Finally, in order to establish a software process, deployment plans need to be worked out. In such a strategy, the deployment strategy must be defined (this means, it needs to be defined which version of the process is considered the actual process, what is the point from which the new process is mandatory to implement, and so forth). Based on the deployment strategy, all other management and administration processes need to be defined, e.g., a training plan highly depends on the selected deployment strategy (when to train whom).

As the deployment strategy is the heart of the management activities, it is also connected to the `Roadmap`, and, thus, is a key element of the whole software process improvement program. The `Roadmap` as such is further linked to a `Vision`, which is the source for the `Goals` that shall be addressed by improving a software process. The `Vision` and the `Goals` cause and influence the `ProcessRequirements` and thus link the engineering and management activities.

2 Comparable to ITIL-based change management processes.

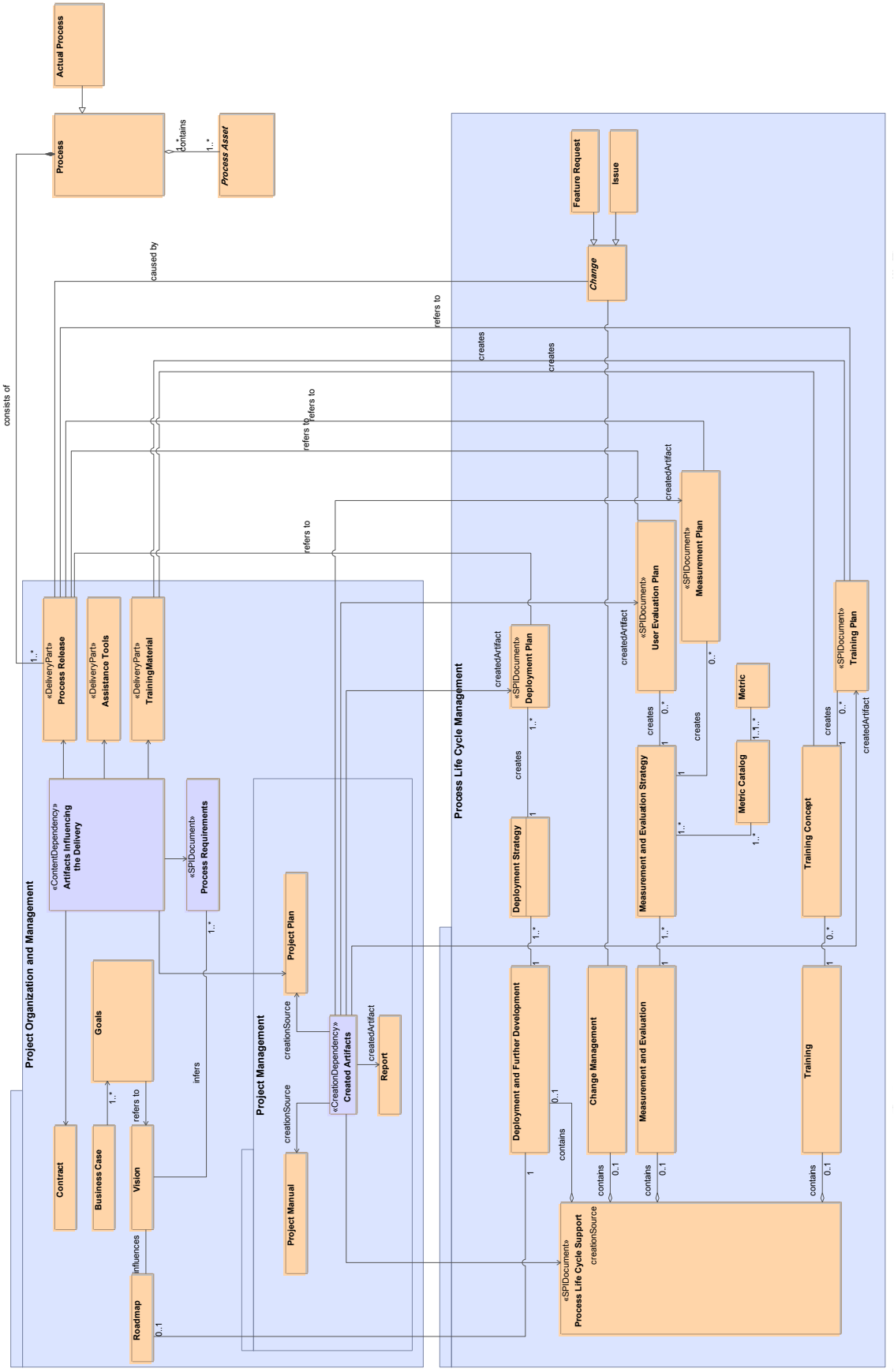

Figure 5.7: Basic classes to link engineering and life cycle models.

In a Nutshell Summarized, the engineering and management tasks need to be connected to each other to implement an integrated and comprehensive process improvement program. As part of such an improvement program, several management and administration activities need to be planned and established at the company level. Basically, only few key elements establish the basic links to bring the engineering and the management activities together: a `Roadmap` together with a `Vision` defines `Goals`, which again lead to `ProcessRequirements`. `ProcessRequirements` are realized in a process development project, which creates a `ProcessRelease` that is shipped to the company, and that is published as `ActualProcess`. Referring the (new) `ActualProcess`, a `ProcessLifeCycleSupport` defines all the management and administration activities necessary to monitor and manage the process. The definitions regarding `Deployment and Further Development` include and update the `Roadmap`; the definitions regarding the `Change Management` establish processes to investigate the feasibility of a deployed process, and thus, close the cycle.

All other (remaining) activities are based on these few elements, e.g., process development and quality assurance are based on the `ProcessRelease`, which is a `Process` that comprises several `ProcessAsset`s, and that is subject to quality assurance.

5.3.2.3 Quality Assurance and Improvement

As mentioned before, a method engineering metamodel must also address certain management activities. One of the most important is the quality management, which, in the context of software process improvement, at least comprises the quality assurance procedures and the software process improvement activities.

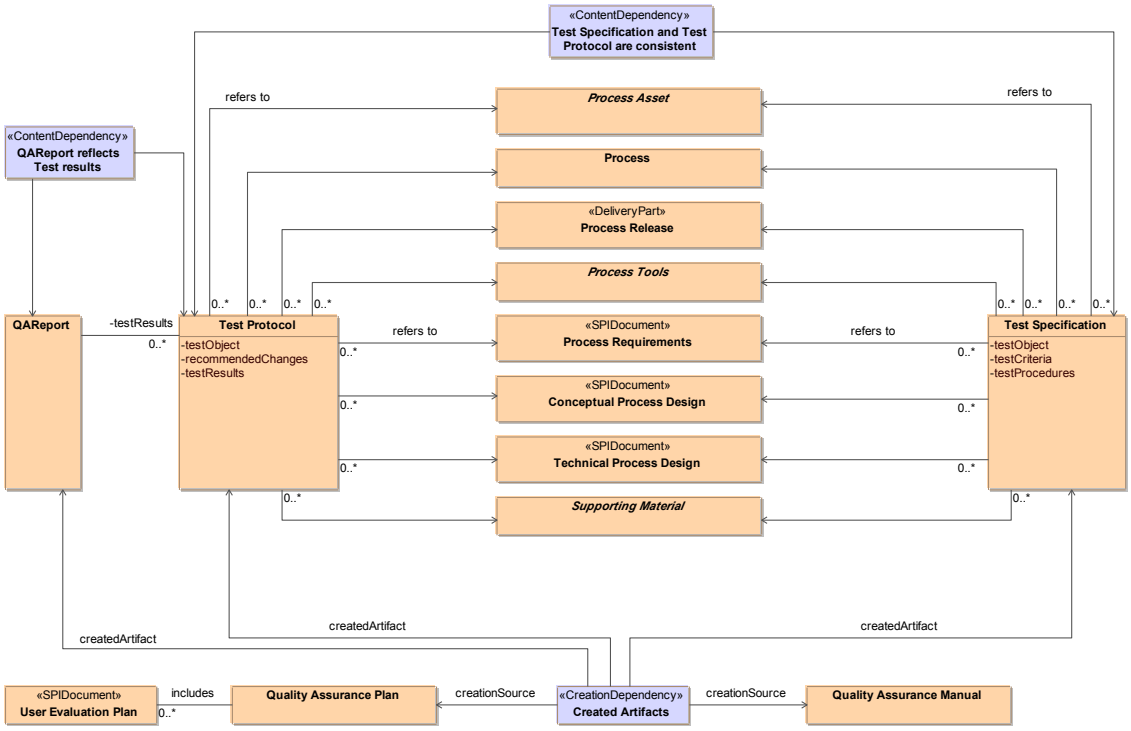

Figure 5.8: Linking process assets to quality assurance procedures.

In Fig. 5.8, we refine the perspective of Fig. 5.6. In this figure, we take the general SPI perspective including all relevant process-related parts as well as central SPI-related artifacts. The figure shows how the quality assurance artifacts related to the core process-related ones. In an “ideal world”,

all created artifacts are quality assured. A `QualityAssurancePlan` and a `QualityAssuranceManual` define the procedures to be used and schedule the quality assurance measures. Test specification define the concrete procedures and objects under test, while respective test protocols document the outcomes of the test runs. The results are eventually reported in quality assurance report[3].

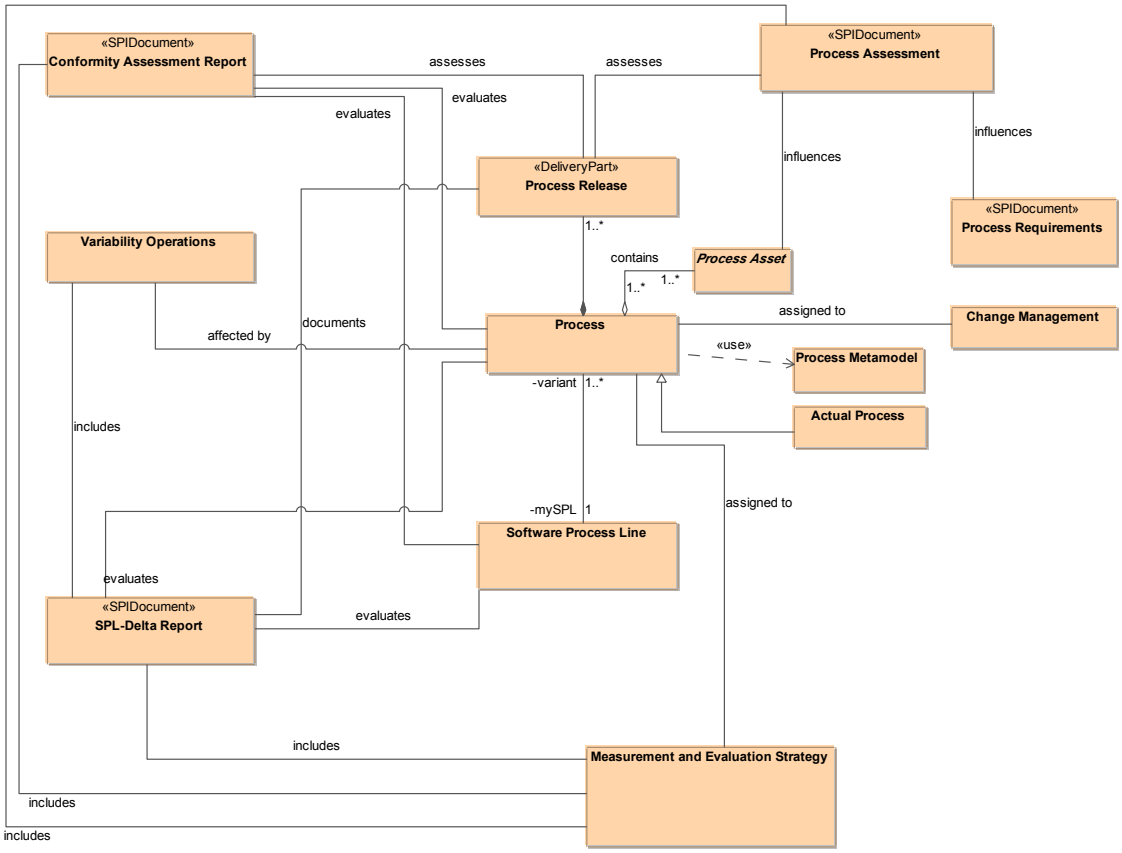

Figure 5.9: Linking process assets to SPI procedures.

In Fig. 5.9 we focus on the links to SPI procedures. The model provides a comprehensive perspective comprising asset-based and process-line-based SPI. That is, improvements can be done based on single `ProcessAssets` as well as on entire software process lines. Key elements are the classes `ProcessAssessment`, `ConformityAssessmentReport`, and `SPLDeltaReport`. The first artifact represents the "classic" process assessment, e.g., using CMMI or ISO 15505 (SPICE) procedures. The latter ones address specific topics relevant to the development, maintenance, and improvement of large-scale processes, which are based on the process lines concept. For such processes, deviation from a given reference process need to be determined in order to investigate the degree to which the conformance is ensured. Moreover, if certification was an improvement goal, the `ConformityAssessmentReport` also serves the analysis of complying to external standards. The figure shows these elements being part of the definition of the management procedures (as already described in the previous section). Furthermore, the figure shows the basic relationships among the different model elements.

3 This general approach is adopted from the quality assurance and management approach as defined by the German V-Modell XT [18]

6 Summary and Conclusion

In the report at hand, we summarized our research on method engineering. We presented summaries of the contribution published so far, presented the instruments used to conduct our research (to allow for replication), and we extended our investigation by an analysis of the concepts harvested from the comprehensive literature studies. We conclude that 1) method engineering as a discipline is still in the negotiation phase, as no agreed and empirically validated concepts are available; 2) if taking the selected key contributions and trying to generate a common metamodel, we find several flaws that make the feasibility in terms of sufficient support during process design and improvement questionable, and 3) key elements that are required to establish a sustainable process improvement and management are yet not addressed by the available method engineering literature.

For this, and taking recent software process metamodels and process frameworks into account, we argue that method engineering should be considered a methodical framework to handle these process frameworks and implementing process improvement and management based on these frameworks. That is, method engineering as discipline needs to be enriched by further, especially management- and administration-related processes complementing the core SPI endeavors. In the report at hand, we thus presented our notion of method engineering, derived the requirements necessary to implement method engineering from an engineer's perspective, and, finally presented a metamodel addressing the defined requirements.

The metamodel, which was presented in this report, reduces the key elements required in engineering methods to the minimum. We introduced a terminology, which was derived and refined from the established terminology from the *software product line domain*. We introduced the concept of the `ProcessAsset`, which is based on the metamodel for artifact orientation. Based on this concept, we construct comprehensive process models (engineering perspective) and link the resulting elements to the life cycle processes that comprise management- and administration activities, e.g., project management, change management, and quality management. Furthermore, we explicitly introduce the concept of a `ProcessLifeCycleSupport` (documentation) to define and establish the required processes.

Are we done with it?—The presented model, however, is a first step toward a more systematic method engineering approach. Missing is the final definition of a concise framework that couples the presented engineering model with sufficient methodical support. Future work thus comprises:

- Refinement of the proposed metamodel
- Refinement of the complementing methodical framework
- Further evaluation of the approach

For some of the aforementioned points, we already have initial data and validation. That is, the major task is the integration of all concepts and bringing the pieced together. With the report at hand, we lay the foundation to direct further research on artifact-based method engineering.

A Method Engineering Glossary Data

A.1 Data Collection

In the following, we show the raw spreadsheet-based data. This information is used to create the glossary, which we show in its initial state in Sect. A.2.

Since the data in the following tables is based on our internal literature search and analysis process, the references used in the "Contributions" columns need to be mapped to the entries of the reference section of this report. The mapping is done according to Table A.1.

Data Entry		Reference
2009Rol#23	↦	[68]
1997Har#136	↦	[20]
1997Hof#37	↦	[75]
2002Ral#38	↦	[63]
2003Fit#154	↦	[17]
2004Kar#44	↦	[37]
2005Mir#35	↦	[54]
2005Bra#129	↦	[9]
2006Kar#26	↦	[38]
2006Lep#81	↦	[50]
2007Baj#127	↦	[6]
2007Bec#128	↦	[8]
2009Rol#83	↦	[69]
2009Hug#141	↦	[29]
2010Jan#78	↦	[31]
2010Eng#135	↦	[16]
2010Hend#139	↦	[26]

Table A.1: Mapping of the references of the data analysis to the reference section.

Term	Alternative	Definition	Further information	Further Refs.	2009Rol#23	1997Har#136	1997Hol#37	2002Ral#38	2003Fit#154	2004Kar#44	2005Mir#35	2005Bra#129	2006Kar#26	2006Lep#81	2007Baj#127	2007Bec#128	2009Rol#83	2009Hug#141	2010Jar#78	2010Eng#135	2010Hen#139
Abstraction Level		The Abstraction level of a product fragment is the state of its instance in terms of degree of abstraction.				D															
		The abstraction level defines whether a fragment is a conceptual fragment (e.g. a model or modelling primitive), or a technical fragment (e.g. A CASE-tool)								D											
Activity		Activity is a step to perform a function in an information system.				D															
		Activities are construction tasks which create certain results, i.e. which create certain specification documents										D									
			Examples: translating user needs into software Requirements, transforming the software requirements into design, implementing the design in code, testing the code, and sometimes, installing and checking out the software for operational use																		
Actor		An Actor is a function involved in an IS engineering project.	Actor is a person or machine performing an activity.			D															
		Actors perform individual actions based on the chosen systems development method.											D								
Actor role		An Actor Role is the type of function an actor has with respect to manipulating and receiving product fragment instances.				D															
Base Method		The main characteristic of method configuration is that it uses one specific method as a basis for creating specific configurations—a base method								D											
		It is a formal representation of how a particular organisation is performing its projects.													D						
Conceptual Method Fragment		A Conceptual Method Fragment is a non-executable method fragment which is described as complete as possible, without taking into account the actor or actor type that will possibly use it.				D															
	Conceptual Fragment		e.g. a model or modelling primitive							D											
Guideline		A guideline is defined as 'a statement or other indication of policy or procedure by which to determine a course of action'		2000 American Heritage Dictionary							D										
Meta Model		The meta model specifies the conceptual data model of the results, thereby guaranteeing the consistency of the entire method.										D									
		A meta-model is a model of a model																		D	
Meta Modelling		Meta-modeling is the principle governing the description of methods. By meta-modeling we mean the modeling of models that make up a method.															D				
		Meta modelling is the principle governing the description of methods. By meta modelling we mean the modelling of models that make up a method.			D																
		Meta-modeling is, according to [GH08], "the act and science of creating meta-models, which are a qualified variant of models."		2008 Gonzales Perez																D	
Method		a set of loosely coupled method chunks expressed at different levels of granularity									D										
		a process which is planned and systematic in terms of its means and purpose, and which leads to technical skill in resolving theoretical and practical tasks		1995 Lorenz								D									
		Methods describe systematic procedures to overcome problems.														D					
		The term method comes from the Greek word 'methodos' which means 'investigation mean'. As an example, Harmsen (1997) defines this investigation means as 'a collection of procedures, techniques, product descriptions and tools for effective, efficient and consistent support of the IS engineering process'. Different authors (Seligmann et al . 1989; Smolander et al . 1991; Kronlof 1993; Brinkkemper 1996; Prakash 1997) describe them in different ways, but all converge on the idea that any method should provide 'a rigorous process to generate a set of product descriptions based on some well defined notations' (Booch 1991).		…														D				
		a method is ''a rigorous process to generate a set of models describing various aspects of software being built using some well-defined notation".		1991 Booch														D			
		A (software/systems development) method can be defined as an approach to perform a software/systems development project, based on a specific way of thinking, consisting, inter alia , of guidelines, rules and heuristics, structured systematically in terms of development activities, with corresponding development work products and developer roles (played by humans or automated tools) (adapted from [Brinkkemper 06] – see also [Henderson-Sellers 95]).		1995 Henderson-Sellers, 2006 Brinkkemper																		D
	Software Engineering Method	the full set ofelements needed to describe a software development endeavor, such as a software development project, in all relevant aspects																		D	
Method Base		for storing method components				D															
		Critical to the support of engineering situational methods is the provision of standardised method building blocks, referred to as method fragments, that can be stored in and retrieved from a so-called method base		1996 Brinkkemper			D														
Method Chunk		A method chunk is an autonomous and coherent part of a method supporting the realization of some specific ISD activities.	Both of these notions, method fragment and method chunk , represent the basic blocks for constructing 'on the fly' methods								D										
		a method chunk is a combination of one process focussed fragment plus one product-focussed fragment																			D
Method Chunk Repository		One of the core elements in the SME framework is the repository of method chunks. The role of this repository is to store reusable parts of methods as prospective method building blocks.									D										
Method Component		A method component is a self-contained part of a systems development method expressing the transformation of one or several artefacts into a defined target artefact and the rationale for such a transformation.											D								
		They defined the method component construct as a self-contained part of a system engineering method expressing the process of transforming one or several artifacts into a defined target artifact and the rational for such a transformation.		2004 Wistrand											D						
		Method components are reusable descriptions of parts of existing methods, i.e. of product and process meta-models.															D				
		Selfcontained part of a system engineering method expressing the process of transforming one or several artefacts into a defined target artefact and the rationale for such a transformation		2004 Wistrand																	D

Term	Alternative	Definition	Further information	Further Refs.	2009Roi#23	1997Har#136	1997Hof#37	2002Rol#38	2003Fit#154	2004Kar#44	2005Mir#35	2005Bra#129	2006Kar#26	2006Lep#81	2007Bal#127	2007Bec#128	2009Rol#83	2009Hug#141	2010Jan#78	2010Eng#135	2010Hen#139
Method Configuration		particular form of situational method engineering. Method configuration means to adapt a particular method to various situated factors. The focus is thus on one method as a base for configuration rather than on a set of methods as a base for assembly. Nonetheless, during method configuration that particular method might need to be enhanced with additional fragments from other methods as well. The important thing is that method configuration always takes one particular method as a starting point.								D											
Method Configuration	Process Tailoring	Another, a somewhat different kind of approach, called method configuration or process tailoring, claims that many organisations choose one commercially available system development methodology and focuses on how to adapt this methodology to situational factors of the project at hand.		2004 Karlsson, 2004 Rossi						D	E										
		Method Configuration is defined to mean the adaptation of a particular method, called the base method, according to various situational factors.		2004 Karlsson										D							
		many organizations choose one commercially available method and focus on how to adapt this method to situational factors of the project at hand.															D				
Method Engineering		Method Engineering is the systematic analysis, comparison, and construction of Information Systems Engineering Methods.				D															
		Method engineering is the engineering discipline to design, construct and adapt methods, techniques and tools for the development of information systems.		1996 Brinkkemper			D														
		The structured approach for creating and tailoring systems engineering methods has been termed method engineering		1999 Brinkkemper						D											
		We call method engineering (ME) the process of developing, customizing and/or configuring a situational SE/ISDmethod, or parts thereof, in a certain organizational and technological context.												D							
		The method engineering deals with the creation of methods that are specifically attuned to the needs of a particular organisation or project.													D						
		method engineering represents the effort to improve the usefulnes of systems development methods by creating an adaption framework whereby methods are created to match specific organisational situations			D																
		Method engineering is an engineering discipline that deals with the development of methods, techniques, and tools for the development of software systems		1996 Brinkkemper																D	
		Method engineering, is defined as the engineering discipline to design, construct and adapt methods, techniques and tools for systems development, a definition analogous to the IEEE definition of software engineering		2006 Brinkkemper																	D
Method Fragment		A method fragment is a description of an IS engineering method, or any coherent part thereof.				D															
		Critical to the support of engineering situational methods is the provision of standardised method building blocks, referred to as method fragments, that can be stored in and retrieved from a so-called method base. A method fragment may be defined as a coherent part of a metamodel, which may cover any of the modelling dimensions at any level of granularity. In this section, we will explain what we consider to be a modelling dimension and what we consider to be a level of granularity.		1996 Brinkkemper			D															
		A method fragment is a coherent piece of a systems engineering method or part thereof, facilitating representation of a method's three basic constituents discussed above		1999 Brinkkemper						D											
		a reusable part of a method		1994 Harmsen											D						
		the term method fragment was coined by [Harmsen 94] (and also by [Brinkkemper 96]) by analogy with the notion of a software component – see also [Saeki 93, Rolland 96]. It can be regarded as an atomic element of a method.																			D
Model		A model is, according to scientific theory, a representation of a natural or artificial original that focuses on those characteristics and properties of the original that are relevant for the given purpose of modeling, and abstracts from irrelevant properties.																		D	
Modular Method		A modular method means a collection of interconnected method fragments/chunks,									D										
Process		way of working		1991 Wijers			D														
		A process is 'the route followed' to reach the target, i.e. the product		1992 Olle				D													
		a process is „the route followed" to reach the target, i.e. The product		1992 Olle	D																
Process Configuration		The idea that lies behind the Process Configuration Approach is relatively simple and can be explained as follows: 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 or project situations it is compulsory, advisable or discouraged to use a particular component.													D						
Process Domain Metamodel		contains the main concepts of the different existing process metamodels such as work unit, role, strategy, etc.; The process domain metamodel, comprises the main concepts from the existing process metamodels of different viewpoints.																D			
Process Fragment		A Process Fragment is a description of an activity to be carried out within a method.				D															
Process Manager Fragment		A Process Manager Fragment is an executable part of a support tool process manager, or a link to a process manager part.				D															
Process Model		The process model describes the steps and activities prescribed by the method.				D															
		The process model describes how to construct the corresponding product model.									D										
		A process model prescribes a way of working to reach the desired target. It describes at an abstract and ideal level, the way of organizing the production of the product: the stages, the activities which they include, their scheduling, and flow constraints to move from one stage to another. It plays the role of a 'mould' to generate processes.															D				
		A process model prescribes a way of working to reach the desired target. It describes at an abstract and ideal level, the way of organizing the production of the product: the stages, the activities which they include, their scheduling, and flow constraints to move from one stage to another. It plays the role of a 'mould' to generate processes.			D																
Process Role		A Process Role represents the respect in which a process fragment manipulates a product fragment.				D															
Process Type		The Process type of a process fragment is the step category to which it belongs.				D															
Product		way of modelling		1991 Wijers			D														
		Products (models, diagrams, et cetera) will both be results of the method's prescribed actions and prerequisites for subsequent prescribed actions in the method.								D											
		A product is the result of the application of a method, the target of ASD		1992 Olle													D				
		a product is the result of the application of a method, the target of IS development		1992 Olle	D																

Term	Alternative	Definition	Further information	Further Refs.	2009Rol#23	1997Har#136	1997Hof#37	2002Ral#38	2003Fli#154	2004Kar#44	2005Mir#35	2005Bra#129	2006Kar#26	2006Lep#81	2007Baj#127	2007Bec#128	2009Rol#83	2009Hug#141	2010Jan#78	2010Eng#135	2010Hend#139
Product Fragment		A Product Fragment is a specification of a product delivered and/or required within a method.				D															
Product Model		The product model of a method defines a set of concepts, relationships between these concepts and constraints for a corresponding schema construction.									D										
		A product model prescribes the expected characteristics of products. It is the 'mould' for the production of products.															D				
Repository Fragment		A Repository Fragment is a support tool repository structure or part thereof.				D															
Situational Method		A situational method is an IS engineering method tailored and tuned to a particular situation.				D															
		The situational method is often thought of as a combination of fragments from different methods, and the incentive for situational method engineering is thus a need to integrate the strengths of different methods		1999 Brinkkemper, 1998 Goldkuhl						D											
Situational Method Engineering		Situational Method Engineering is the sub-area of Method Engineering directed towards the controlled, formal and computer-assisted construction of situational methods out of method fragments.				D															
		The aim of frameworks for situational method engineering is usually to guide the process of selecting and integrating different method fragments into congruent and consistent methods relevant for the situation at hand		1997 Hofstede						D											
	Modular Method Construction	Situational method engineering may be defined as adapting a method to the needs of a particular Project		1995 Odell			D														
		'the discipline to build project-specific methods, called situational methods, from parts of the existing methods, called methods fragments'		1996 Brinkkemper				D													
		the discipline of Situational Method Engineering (SME) focuses on the creation of new techniques and tools allowing to construct projectspecific methods 'on the fly' instead of looking for universally applicable ones.		1992 Kumar							D										
		SME deals with developing project-specific methods or adapting existing ones to specific project situations		1994 Harmsen, 1992 Kumar										D							
		focus in particular on the recombination of method fragments and, thus, are using the mechanism of aggregation.		1997 Harmsen, 1996 Brinkkemper, 1998 Brinkkemper											D						
		Situational method engineering (SME) is the construction of methods which are tuned to specific situations of development projects. Each system development starts then, with a method definition phase where the development method is constructed on the spot.		1992 Welke													D				
		Situational method engineering is the construction of methods which are tuned to specific situations of development projects. Each system development starts then, with a method definition phase where the development method is constructed on the spot.			D																
		encompasses all aspects of creating a development method for a specific situation																			D
Software Development Process		The process by which user needs are translated into a software product.		IEEE Standard 610.12																D	
Task Model		The task model defines the tasks that need to be accomplished, again in the form of transformations.																		D	
Technical Method Fragment		A Technical Method Fragment is a method fragment implemented as a IS engineering support tool or part thereof.				D															
	Technical Fragment		e.g. a CASE-tool							D											
Tool Fragment		A Tool Fragment is a technical method fragment for entering, transferring or checking information regarding the IS.				D															

A.2 Initial Glossary

In the following, we present the initial glossary extracted from the found data (Sect. A.1). Every glossary item is structured as follows:

⇒ Term: *[name of the term]*

Category	*The term is categorized either as "methodical", as "structural" or as "technical" item. Structural terms are the basis to infer the metamodel while the methodical ones provide context information.*
Source	*The source of the term is given by an author reference.*

Definition: *A (consolidated) definition is provided per term.*

Input: *The sources that were used to craft the definition.*

In the definition of the terms, we highlighted other referred terms. Furthermore, we included remarks in which we clarified the notion of the term, which was found in a team discussion on the respective terms.

⇒ Term: Abstraction Level

Category	methodical
Source	[20, 37]

Definition: The abstraction level defines whether a fragment is a conceptual fragment, or a technical one.

Input: The Abstraction level of a *product fragment* is the state of its instance in terms of degree of abstraction.

The abstraction level defines whether a fragment is a conceptual fragment (e.g. a model or modeling primitive), or a technical fragment (e.g. a CASE-tool).

⇒ Term: Actvitiy

Category	structural
Source	[20, 9]

Definition: Activities are construction tasks, which create certain results.

Input: Activity is a step to perform a function in an information system. Activities are construction tasks, which create certain results, i.e. which create certain specification documents.

Remark: In our notion, an activity represents an atomic method that produces or manipulates 1 result by using 1 technique. An activity comprises n tasks (elementary steps), and an activity is also used for planning. That is, an activity represents a *work package* (as subject to planning activities).

⇒ Term: Actor

Category	structural
Source	[20, 38]

Definition: Actors perform individual actions based on the chosen systems development method.

Input: An Actor is a function involved in an IS engineering project.

Actors perform individual actions based on the chosen systems development method.

⇒ Term: Actor Role

Category	methodical
Source	[20]

Definition: The Actor role describes which function an actor has in Information system Development Process.

Input: An Actor Role is the type of function an actor has with respect to manipulating and receiving product fragment instances.

Remark: From the perspective of meta-modeling, the actor role usually represents the type of a process asset representing a role.

⇒ Term: Base Method

Category	structural
Source	[37, 6]

Definition: A Base Method is the foundation to adapt a particular method to various situated factors.

Input: The main characteristic of method configuration is that it uses one specific method as a basis for creating specific configuration a base method.

It is a formal representation of how a particular organization is performing its projects.

Remark: In terms of a reference process, the "common notion" of the base method *is* the software process; if the process is based on a software process line, the base method defines the reference process of the process line.

⇒ Term: Conceptual Method Fragment (Conceptual Fragment)

Category	structural
Source	[20, 37]

Definition: —

Input: A Conceptual Method Fragment is a non-executable method fragment, which is described as complete as possible, without taking into account the actor or actor type that will possibly use it.

⇒ Term: Guideline

Category	structural
Source	[54]

Definition: A guideline is any piece of actionable description of a fragment, e.g. procedures, tool guides, and so forth.

Input: A guideline is defined as "a statement or other indication of policy or procedure by which to determine a course of action".

Remark: A guideline is an additional and optional piece of information that complements the description of an fragment. While the description describes the fragment (e.g., purpose), the guideline states how to use a fragment in the development procedure.

⇒ Term: Metamodel

Category	structural
Source	[9, 16]

Definition: —

Input: The metamodel specifies the conceptual data model of the results, thereby guaranteeing the consistency of the entire method. A meta-model is a model of a model.

Remark: In the context of the work at hand, we use the term metamodel as defined by software process metamodels, e.g. SPEM [57] or ISO 24744 [32].

⇒ Term: Meta-Modeling

Category	methodical
Source	[69, 68, 16]

Definition: —

Input: Meta-modeling is the principle governing the description of methods. By meta-modeling we mean the modeling of models that make up a method. Meta modeling is the principle governing the description of methods. By meta-modeling we mean the modeling of models that make up a method. Meta-modeling is, according to [GH08], "the act and science of creating meta-models, which are a qualified variant of models."

⇒ Term: Method

Category	structural
Source	[54, 9, 8, 69, 29, 26, 16]

Definition: A method comprises activities that created and/or modify artifacts, performed by responsible and contribution roles that use a defined technique (consisting of concrete methods, e.g. unit testing, notation, and tools) to create and or modify the artifacts of interest.

Input: A set of loosely coupled method chunks expressed at different levels of granularity.

A process, which is planned and systematic in terms of its means and purpose, and which leads to technical skill in resolving theoretical and practical tasks.

Methods describe systematic procedures to overcome problems.

The term method comes from the Greek word "methodos" which means "investigation mean". As an example, Harmsen (1997) defines this investigation means as "a collection of procedures, techniques, product descriptions and tools for effective, efficient and consistent support of the IS engineering process". Different authors (Seligmann et al. 1989; Smolander et al. 1991; Kronlof 1993; Brinkkemper 1996; Prakash 1997) describe them in different ways, but all converge on the idea that any method should provide "a rigorous process to generate a set of product descriptions based on some well defined notations" (Booch 1991).

A method is a "rigorous process to generate a set of models describing various aspects of software being built using some well-defined notation."

A (software/systems development) method can be defined as an approach to perform a software/systems development project, based on a specific way of thinking, consisting, inter alia, of guidelines, rules and heuristics, structured systematically in terms of development activities, with corresponding development work products and developer roles (played by humans or automated tools) (adapted from [Brinkkemper 06] – see also [Henderson-Sellers 95]).

The full set of elements needed to describe a software development endeavor, such as a software development project, in all relevant aspects.

⇒ Term: Method Base

Category	technical
Source	[20, 75]

Definition: —

Input: For storing method components.

Critical to the support of engineering situational methods is the provision of standardized method building blocks, referred to as method fragments, that can be stored in and retrieved from a so-called method base.

Remark: This is a technical aspect that is no more covered, as we demand a process-engineering framework and the corresponding infrastructure to provide the required technical support to store process assets/fragments.

⇒ Term: Method Chunk

Category	structural
Source	[54, 26]

Definition: A method chunk is a concept that describes a "prototypical" method that comprises configurations of product, role, and/or product fragments. A method chunk is not purposed to be directly applied—it is always part of an integrated method.

Input: A method chunk is an autonomous and coherent part of a method supporting the realization of some specific ISD activities.

A method chunk is a combination of one process focussed fragment plus one product-focused fragment.

⇒ Term: Method Chunk Repository

Category	technical
Source	[54]

Definition: —

Input: One of the core elements in the SME framework is the repository of method chunks. The role of this repository is to store reusable parts of methods as prospective method building blocks.

Remark: see "Method Base"

⇒ Term: Method Component

Category	structural
Source	[38, 6, 69, 26]

Definition: A method component is a self-contained and reusable part of a (set of) method(s) and/or fragments. A method component can be considered as a reusable container comprising methods or parts of it.

Input: A method component is a self-contained part of a systems development method expressing the transformation of one or several artifacts into a defined target artifact and the rationale for such a transformation.;

They defined the method component construct as a self-contained part of a system engineering method expressing the process of transforming one or several artifacts into a defined target artifact and the rational for such a transformation.;

Method components are reusable descriptions of parts of existing methods, i.e. of product and process meta-models.;

Self-contained part of a system engineering method expressing the process of transforming one or several artifacts into a defined target artifact and the rationale for such a transformation;

Remark: For a concrete materialization of the concept "method component", cf. SPEM [57] – "method plug-in".

⇒ Term: Method Configuration

Category	structural
Source	[37, 9, 50, 69]

Definition: A method configuration is a special materialization of a method component that comprises methods and/or fragments and adds configuration information in order to provide a meaningful and consistent package. A method configuration is based on a base method.

Input: Particular form of situational method engineering. Method configuration means to adapt a particular method to various situated factors. The focus is thus on one method as a base for configuration rather than on a set of methods as a base for assembly. Nonetheless, during method configuration that particular method might need to be enhanced with additional fragments from other methods as well. The important thing is that method configuration always takes one particular method as a starting point.

[Context: process tailoring] Another, a somewhat different kind of approach, called method configuration or process tailoring, claims that many organizations choose one commercially available system development methodology and focuses on how to adapt this methodology to situational factors of the project at hand.

Method Configuration is defined to mean the adaptation of a particular method, called the base method, according to various situational factors.

Many organizations choose one commercially available method and focus on how to adapt this method to situational factors of the project at hand.

⇒ Term: Method Engineering

Category	methodical
Source	[26, 75, 37, 50, 6, 68, 16, 26]

Definition: Method engineering is the engineering discipline to design, construct and adapt methods, techniques and tools for the development of information systems.

Input: Method Engineering is the systematic analysis, comparison, and construction of Information Systems Engineering Methods.;

The structured approach for creating and tailoring systems engineering methods has been termed method engineering; We call method engineering (ME) the process of developing, customizing and/or configuring a situational SE/ISD method, or parts thereof, in a certain organizational and technological context.;

The method engineering deals with the creation of methods that are specifically attuned to the needs of a particular organization or project.;

method engineering represents the effort to improve the usefulness of systems development methods by creating an adaption framework whereby methods are created to match specific organizational situations;

Method engineering is an engineering discipline that deals with the development of methods, techniques, and tools for the development of software systems;

Method engineering, is defined as the engineering discipline to design, construct and adapt methods, techniques and tools for systems development, a definition analogous to the IEEE definition of software engineering

⇒ Term: Method Fragment

Category	structural
Source	[26, 75, 37, 6, 26]

Definition: A method fragment is a description of an IS engineering method, or any coherent part thereof.

Input: Critical to the support of engineering situational methods is the provision of standardized method building blocks, referred to as method fragments, that can be stored in and retrieved from a so-called method base. A method fragment may be defined as a coherent part of a metamodel, which may cover any of the modeling dimensions at any level of granularity. In this section, we will explain what we consider to be a modeling dimension and what we consider to be a level of granularity.;

A method fragment is a coherent piece of a systems engineering method or part thereof, facilitating representation of a method's three basic constituents discussed above a reusable part of a method;

"the term method fragment was coined by [Harmsen 94] (and also by [Brinkkemper 96]) by analogy with the notion of a software component – see also [Saeki 93, Rolland 96]. It can be regarded as an atomic element of a method."

⇒ Term: Model

Category	methodical
Source	[16]

Definition: —

Input: A model is, according to scientific theory, a representation of a natural or artificial original that focuses on those characteristics and properties of the original that are relevant for the given purpose of modeling, and abstracts from irrelevant properties.

⇒ Term: Modular Method

Category	structural
Source	[54]

Definition: —

Input: A modular method means a collection of interconnected method fragments/chunks,

⇒ Term: Process

Category	structural
Source	[75, 69, 68]

Definition: A process is a coherent collection of process fragments.

Input: way of working;

A process "is the route followed" to reach the target, i.e. the product;

Remark: The term process is defined on different levels of abstraction and thus has different meanings depending on the respective abstraction level. For instance, from the top-level perspective, the term process means the overall process that is also meant by the term method/configuration. On a fine-grained level, the term process may also describe a part of an activity.

⇒ Term: Process Configuration

Category	structural
Source	[6]

Definition: A process configuration is a (project-specific) consistent selection of processes.

Input: The idea that lies behind the Process Configuration Approach is relatively simple and can be explained as follows: 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 organization and thus reflects its actual performance on the projects (base method). The configuration is done by processing the rules that tell in what circumstances or project situations it is compulsory, advisable or discouraged to use a particular component.

⇒ Term: Process Domain Metamodel

Category	structural
Source	[29]

Definition: —

Input: contains the main concepts of the different existing process metamodels such as work unit, role, strategy, etc.; The process domain metamodel, comprises the main concepts from the existing process metamodels of different viewpoints.

Remark: This concept addresses the need for creating a generic instrument that can be applied using different software process metamodels, e.g. SPEM [57], ISO 24744 [32], or the V-Modell XT metamodel [77].

⇒ Term: Process Fragment

Category	structural
Source	[20]

Definition: A process fragment is a specific fragment comprising different process parts. A process fragment is part of a process and is linked to fragment of same *and* other types.

Input: A Process Fragment is a description of an activity to be carried out within a method.;

⇒ Term: Process Manager Fragement

Category	
Source	[20]

Definition: —

Input: A Process Manager Fragment is an executable part of a support tool process manager, or a link to a process manager part.

⇒ Term: Process Model

Category	
Source	[20, 54, 69, 68]

Definition: A process model is a (consistent) collection of processes (that again comprise process fragments).

Input: The process model describes the steps and activities prescribed by the method.;

The process model describes how to construct the corresponding product model.;

A process model prescribes a way of working to reach the desired target. It describes at an abstract and ideal level, the way of organizing the production of the product: the stages, the activities which they include, their scheduling, and flow constraints to move from one stage to another. It plays the role of a ?mould? to generate processes.;

⇒ Term: Process Role

Category	structural
Source	[20]

Definition: —

Input: A Process Role represents the respect in which a process fragment manipulates a product fragment.

⇒ Term: Process Type

Category	structural
Source	[20]

Definition: —

Input: The Process type of a process fragment is the step category to which it belongs.

⇒ Term: Product

Category	structural
Source	[75, 37, 69, 68]

Definition: A product (artifact) is any tentative result of an activity that is created, consumed, or modified.

Input: way of modeling;

Products (models, diagrams, etc.) will both be results of the method's prescribed actions and prerequisites for subsequent prescribed actions in the method.;

A product is the result of the application of a method, the target of ASD;

a product is the result of the application of a method, the target of IS development

⇒ Term: Product Fragment

Category	structural
Source	[20]

Definition: A product fragment is a specific fragment comprising different artifacts. A product fragment is part of a product and is linked to fragment of same *and* other types.

Input: A Product Fragment is a specification of a product delivered and/or required within a method.

Remark: Instead of the term product, we also use the term *artifact* as defined in [53].

⇒ Term: Product Model

Category	structural
Source	[54, 69]

Definition: A product model is a (consistent) collection of products (that again comprise product fragments).

Input: The product model of a method defines a set of concepts, relationships between these concepts and constraints for a corresponding schema construction.;

A product model prescribes the expected characteristics of products. It is the "mould" for the production of products.

⇒ Term: Repository Fragement

Category	technical
Source	[20]

Definition: —

Input: A Repository Fragment is a support tool repository structure or part thereof.

⇒ Term: Situational Method

Category	methodical
Source	[20, 37]

Definition: A situational method is an IS engineering method tailored and tuned to a particular situation.

Input: The situational method is often thought of as a combination of fragments from different methods, and the incentive for situational method engineering is thus a need to integrate the strengths of different methods.

Remark: In terms of customizing/tailoring a software process, a situational method is a *tailored* method. However, one needs to differentiate between static and dynamic tailoring (cf. [47]).

⇒ Term: Situational Method Engineering

Category	methodical
Source	[20, 37, 75, 63, 54, 6, 8, 69, 68, 26]

Definition: Situational method engineering (SME) is the construction of methods which are tuned to specific situations of development projects.

Input: Situational Method Engineering is the sub-area of Method Engineering directed towards the controlled, formal and computer-assisted construction of situational methods out of method fragments.;

The aim of frameworks for situational method engineering is usually to guide the process of selecting and integrating different method fragments into congruent and consistent methods relevant for the situation at hand;

Situational method engineering may be defined as adapting a method to the needs of a particular Project; [[aka Tailoring]]

"the discipline to build project-specific methods, called situational methods, from parts of the existing methods, called methods fragments";

"the discipline of Situational Method Engineering (SME) focuses on the creation of new techniques and tools allowing to construct project-specific methods ?on the fly? instead of looking for universally applicable ones.";

SME deals with developing project-specific methods or adapting existing ones to specific project situations;

focus in particular on the recombination of method fragments and, thus, are using the mechanism of aggregation.;

Situational method engineering is the construction of methods which are tuned to specific situations of development projects. Each system development starts then, with a method definition phase where the development method is constructed on the spot.;

⇒ Term: Software Development Process

Category	methodical
Source	[16]

Definition: —

Input: The process by which user needs are translated into a software product.

Remark: A software development process is differently defined in different context. In the context of method engineering, software development processes and method are distinguished, whereas such a differentiation is not made in other context.

⇒ Term: Task Model

Category	structural
Source	[16]

Definition: —

Input: The task model defines the tasks that need to be accomplished, again in the form of transformations.

⇒ Term: Technical Method Fragment (Technical Fragment)

Category	technical
Source	[20, 37]

Definition: —

Input: A Technical Method Fragment is a method fragment implemented as a IS engineering support tool or part thereof.

⇒ Term: Tool Fragment (cf. Tech. Fragment)

Category	technical
Source	[20]

Definition: —

Input: A Tool Fragment is a technical method fragment for entering, transferring or checking information regarding the IS.

Bibliography

[1] ÅGERFALK, P. J., BRINKKEMPER, S., GONZALEZ-PEREZ, C., HENDERSON-SELLERS, B., KARLSSON, F., KELLY, S., AND RALYTÉ, J. *Situational Method Engineering: Fundamentals and Experiences*. Springer, 2007, ch. Modularization Constructs in Method Engineering: Towards Common Ground?

[2] ARMBRUST, O. Leitfaden zur modelleinführung im rahmen der organisationsspezifischen anpassung des v-modell xt. Forschungsbericht 013.08/D, Fraunhofer Institut Experimentelles Software Engineering, 2008.

[3] ARMBRUST, O., EBELL, J., HAMMERSCHALL, U., MÜNCH, J., AND THOMA, D. Prozesseinführung und -reifung in der praxis: Erfoolgsfaktoren und erfahrungen. In *Proceedings des 14. Workshop der Fachgruppe WI-VM der Gesellschaft für Informatik e.V. (GI)* (apr 2007), no. ISBN: 978-3-8322-6111-5, Shaker Verlag, pp. 3–15.

[4] AYDIN, M. N., AND HARMSEN, F. Making a Method Work for a Project Situation in the Context of CMM. In *4th International Conference on Product Focused Software Process Improvement* (2002), Springer.

[5] AYDIN, M. N., HARMSEN, F., AND SLOOTEN, K. An agile information systems development method in use. *Turk J Elec Engin* (2004).

[6] BAJEC, M., AND VAVPOTIC, D. Practice-driven approach for creating project-specific software development methods. *Information and Software Technology* (2007).

[7] BASILI, V. R., AND ROMBACH, H. D. Tailoring the software process to project goals and environments. In *9th International Conference on Software Engineering (ICSE)* (1987), IEEE Computer Society Press.

[8] BECKER, J., KNACKSTEDT, R., AND PFEIFFER, D. Configurative method engineering–on the applicability of reference modeling mechanisms in method engineering. In *AMCIS 2007 Proceedings* (2007).

[9] BRAUN, C., WORTMANN, F., HAFNER, M., AND WINTER, R. Method Construction- A Core Approach to Organizational Engineering. In *ACM Symposium on Applied Computing* (2005).

[10] BRINKKEMPER, S. *Formalisation of Information Systems Modelling*. PhD thesis, Radboud University, 1990.

[11] BRINKKEMPER, S. Method engineering: engineering of information systems development methods and tools. *Information and Software Technology 38*, 4 (1996).

[12] BRINKKEMPER, S., LYYTINEN, K., AND WELKE, R. J. *Method engineering: principles of method construction and tool support...* Springer, 1996.

[13] BRINKKEMPER, S., AND SAEKI, M. Meta-modelling based assembly techniques for situational method engineering. *Information Systems* (1999).

[14] BRINKKEMPER, S., SAEKI, M., AND HARMSEN, F. Assembly Techniques for Method Engineering. In *10th International Conference Advanced Information Systems Engineering* (1998).

[15] DOMINGUEZ, E., AND ZAPATA, M. A. Noesis: Towards a situational method engineering technique. *Information Systems 32*, 2 (2007).

[16] ENGELS, G., AND SAUER, S. A meta-method for defining software engineering methods. *Graph transformations and model-driven engineering* (2010).

[17] FITZGERALD, B., RUSSO, N. L., AND O'KANE, T. Software development method tailoring at Motorola. *Communications of the ACM 46*, 4 (2003).

[18] FRIEDRICH, J., HAMMERSCHALL, U., KUHRMANN, M., AND SIHLING, M. *Das V-Modell XT - Für Projektleiter und QS-Verantwortliche kompakt und übersichtlich*, 2. ed. No. ISBN: 978-3-540-76403-8 in Informatik im Fokus. Springer, 2009.

[19] GONZALEZ-PEREZ, C. *Situational Method Engineering: Fundamentals and Experiences*, vol. 244 of *IFIP — The International Federation for Information Processing*. Springer, 2007, ch. Supporting Situational Method Engineering with ISO/IEC 24744 and the Work Product Pool Approach.

[20] HARMSEN, A. F. *Situational Method Engineering*. PhD thesis, Universiteit Twente, 1997.

[21] HARMSEN, F., AND BRINKKEMPER, S. Design and implementation of a method base management system for a situational CASE environment. In *Asia Pacific Software Engineering Conference* (1995), IEEE Comput. Soc. Press.

[22] HENDERSON-SELLERS, B. Method engineering for OO systems development. *Communications of the ACM 46*, 10 (2003), 73.

[23] HENDERSON-SELLERS, B., GONZALEZ-PEREZ, C., AND RALYTÉ, J. Situational method engineering: Fragments or chunks? In *CAiSE Forum* (2007), J. Eder, S. L. Tomassen, A. L. Opdahl, and G. Sindre, Eds., vol. 247 of *CEUR Workshop Proceedings*, CEUR-WS.org.

[24] HENDERSON-SELLERS, B., GONZALEZ-PEREZ, C., AND RALYTE, J. Comparison of Method Chunks and Method Fragments for Situational Method Engineering. In *19th Australian Conference on Software Engineering* (2008).

[25] HENDERSON-SELLERS, B., GONZALEZ-PEREZ, C., SEROUR, M. K., AND FIRESMITH, D. G. Method engineering and COTS evaluation. In *ACM SIGSOFT Software Engineering Notes* (2005), ACM.

[26] HENDERSON-SELLERS, B., AND RALYTE, J. Situational method Engineering: State-of-the-Art Review. *Journal of Universal Computer Science* (2010).

[27] HENDERSON-SELLERS, B., SEROUR, M., MCBRIDE, T., GONZALEZ-PEREZ, C., AND DAGHER, L. Process construction and customization. *Journal of Universal Computer Science 10* (2004), 326–358.

[28] HENDERSON-SELLERS, B. Method engineering: Theory and practice. In *Information Systems Technology and its Applications* (2006).

[29] HUG, C., FRONT, A., RIEU, D., AND HENDERSON-SELLERS, B. A method to build information systems engineering process metamodels. *Journal of Systems and Software 82*, 10 (2009).

[30] JACOBSON, I., NG, P.-W., MCMAHON, P. E., SPENCE, I., AND LIDMAN, S. *The Essence of Software Engineering: Applying the SEMAT Kernel*. Addison Wesley, 2013.

[31] JANIESCH, C. Situation Vs. Context: Considerations on the Level of Detail in Modelling Method Adaptation. In *43rd Hawaii International Conference on System Sciences* (2010).

[32] JOINT TECHNICAL COMMITTEE ISO/IEC JTC 1, SUBCOMMITTEE SC 7. Software engineering – metamodel for development methodologies. Tech. Rep. ISO/IEC 24744:2007, International Organization for Standardization, 2007.

[33] KALUS, G., AND KUHRMANN, M. Criteria for Software Process Tailoring: A Systematic Review. In *Proceedings of International Conference on Software and Systems Process (ICSSP 2013)* (may 2013), ACM Press, pp. 171–180. available at http://dl.acm.org/.

[34] KAN, S. H. *Metrics and Models in Software Quality Engineering*, 2 ed. Addison-Wesley Longman, 2002.

[35] KARLSSON, F. A WIKI-BASED APPROACH TO METHOD TAILORING. In *3rd International Conference on the Pragmatic Web* (2008), ACM Press.

[36] KARLSSON, F. Method tailoring as negotiation. In *CAiSE Forum* (2008), Z. Bellahsène, C. Woo, E. Hunt, X. Franch, and R. Coletta, Eds., vol. 344 of *CEUR Workshop Proceedings*, CEUR-WS.org, pp. 1–4.

[37] KARLSSON, F., AND ÅGERFALK, P. Method configuration: adapting to situational characteristics while creating reusable assets. *Information and Software Technology 46*, 9 (2004).

[38] KARLSSON, F., AND WISTRAND, K. Combining method engineering with activity theory: theoretical grounding of the method component concept. *European Journal of Information Systems 15*, 1 (2006).

[39] KELLNER, M., BRIAND, L., AND OVER, J. A method for designing, defining, and evolving software processes. In *4th International Conference on the Software Process* (1996), pp. 37–48.

[40] KITCHENHAM, B. Procedures for Performing Systematic Reviews. Tech. Rep. TR/SE0401, Keele University, 2004.

[41] KUHRMANN, M. *Konstruktion modularer Vorgehensmodelle*. PhD thesis, Technische Universität München, 2008.

[42] KUHRMANN, M. ArSPI: An Artifact Model for Software Process Improvement and Management . Forschungsbericht TUM-I1337, jul 2013. available at http://mediatum.ub.tum.de/?id=1170019.

[43] KUHRMANN, M., AND BEECHAM, S. Artifact-based software process improvement and management: A method proposal. In *International Conference on Software and Systems Process (ICSSP)* (may 2014), ACM Press, pp. 165–169. available at http://dx.doi.org/10.1145/2600821.2600839.

[44] KUHRMANN, M., FERNANDEZ, D. M., AND KNAPP, A. A First Investigation About the Perceived Value of Process Engineering and Process Consumption. In *Proceedings of the 14th International Conference on Product Focused Software Development and Process Improvement (PROFES)* (jun 2013), no. 7983 in LNCS, Springer-Verlag Berlin Heidelberg, pp. 138–152. Full title: Who Cares About Software Process Modelling? A First Investigation About the Perceived Value of Process Engineering and Process Consumption, available at http://link.springer.com/.

[45] KUHRMANN, M., FERNANDEZ, D. M., AND STEENWEG, R. Systematic Software Process Development: Where Do We Stand Today? In *Proceedings of International Conference on Software and Systems Process (ICSSP 2013)* (may 2013), ACM Press, pp. 166–170. available at http://dl.acm.org/.

[46] KUHRMANN, M., FERNÁNDEZ, D. M., AND TERNITÉ, T. Realizing software process lines: Insights and experiences. In *International Conference on Software and Systems Process (ICSSP)* (may 2014), ACM Press, pp. 110–119. available at http://dx.doi.org/10.1145/2600821.2600833.

[47] KUHRMANN, M., AND HAMMERSCHALL, U. Anpassung des V-Modell XT - Leitfaden zur organisationsspezifischen Anpassung des V-Modell XT. Projektbericht TUM-I0831, Technische Universität München, 2008.

[48] KUHRMANN, M., MENDEZ FERNANDEZ, D., AND TIESSLER, M. A Mapping Study on Method Engineering - First Results. In *Proceedings of the 17th Evaluation and Assessment in Software Engineering (EASE 2013)* (2013), ACM Press, pp. 165–170. available at http://dl.acm.org/.

[49] KUHRMANN, M., MENDEZ FERNANDEZ, D., AND TIESSLER, M. A mapping study on the feasibility of method engineering. *Journal of Software: Evolution and Process* (2014).

[50] LEPPAENEN, M. Conceptual evaluation of methods for engineering situational ISD methods. *Software Process: Improvement and Practice 11*, 5 (2006).

[51] LOW, G., MOURATIDIS, H., AND HENDERSON-SELLERS, B. Using a situational method engineering approach to identify reusable method fragments from the secure tropos methodology. *Journal of Object Technology 9*, 4 (2010), 93–125.

[52] MARTÍNEZ-RUIZ, T., MÜNCH, J., GARCÍA, F., AND PIATTINI, M. Requirements and constructors for tailoring software processes: a systematic literature review. *Software Quality Journal 20*, 1 (2012), 229–260.

[53] MENDEZ FERNANDEZ, D., PENZENSTADLER, B., KUHRMANN, M., AND BROY, M. A Meta Model for Artefact-Orientation: Fundamentals and Lessons Learned in Requirements Engineering. In *Proceedings of the 13th International Conference on Model Driven Engineering Languages and Systems (Models)* (2010), D. Petriu, N. Rouquette, and O. Haugen, Eds., vol. 6395, Springer-Verlag Berlin Heidelberg, pp. 183–197.

[54] MIRBEL, I., AND RALYTE, J. Situational method engineering: combining assembly-based and roadmap-driven approaches. *Requirements Engineering 11*, 1 (2005).

[55] MÜNCH, J., ARMBRUST, O., SOTO, M., AND KOWALCZYK, M. *Software Process Definition and Management*. Springer, 2012.

[56] OCAMPO, A., AND SOTO, M. Connecting the Rationale for Changes to the Evolution of a Process. In *Intl. Conf. on Product-Focused Software Process Improvement* (2007).

[57] OMG. Software & Systems Process Engineering Metamodel Specification (SPEM) Version 2.0. Tech. rep., Object Management Group, 2008.

[58] PETERSEN, K., FELDT, R., MUJTABA, S., AND MATTSSON, M. Systematic mapping studies in software engineering. In *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering* (Swinton, UK, UK, 2008), EASE'08, British Computer Society, pp. 68–77.

[59] PLIHON, V. MENTOR: An Environment Supporting the Construction of Methods. In *Asia Pacific Software Engineering Conference* (1996).

[60] PUNTER, T., AND LEMMEN, K. The MEMA-model: towards a new approach for Method Engineering. *Information and Software Technology* (1996).

[61] QUMER, A., AND HENDERSON-SELLERS, B. Construction of an agile software product-enhancement process by using an agile software solution framework (assf) and situational method engineering. In *Proceedings of the 31st Annual International Computer Software and Applications Conference - Volume 01* (Washington, DC, USA, 2007), COMPSAC '07, IEEE Computer Society, pp. 539–542.

[62] QUMER, A., AND HENDERSON-SELLERS, B. Framework as software service (fass) - an agile e-toolkit to support agile method tailoring. In *ICSOFT (2)* (2010), pp. 167–172.

[63] RALYTE, J. Requirements Definition for the Situational Method Engineering. In *Working Conference on Engineering Information Systems in the Internet Context* (2002).

[64] RALYTE, J., DENECKERE, R., AND ROLLAND, C. Towards a Generic Model for Situational Method Engineering. In *Advanced Information Systems Engineering*. Springer, 2003.

[65] RALYTE, J., AND ROLLAND, C. An Approach for Method Reengineering. In *20th International Conference on Conceptual Modeling Yokohama* (2001).

[66] RALYTE, J., AND ROLLAND, C. An Assembly Process Model for Method Engineering. In *Advanced Information Systems Engineering* (2001).

[67] ROLLAND, C. A primer for method engineering. In *Proceedings of the conferance INFORSID* (1997).

[68] ROLLAND, C. Method Engineering: State-of-the-Art Survey and Research Proposal. In *Conference on New Trends in Software Methodologies, Tools and Techniques* (2009), IOS Press.

[69] ROLLAND, C. Method engineering: towards methods as services. *Software Process: Improvement and Practice 14*, 3 (2009).

[70] ROMBACH, D. Integrated Software Process and Product Lines. In *International Software Process Workshop (SPW 2005)* (2005).

[71] RUNESON, P., AND HÖST, M. Guidelines for conducting and reporting Case Study Research in Software Engineering. *Empirical Software Engineering 14*, 2 (2009), 131–164.

[72] SOTO, M., AND MÜNCH, J. The deltaprocess approach for analyzing process differences and evolution. Tech. Rep. IESE-Report No. 164.06/E, Fraunhofer Institut Experimentelles Software Engineering, October 2006. Submitted for Publication to Software Process: Improvement and Practice.

[73] SOTO, M., AND MÜNCH, J. Process model difference analysis for supporting process evolution. In *EuroSPI* (2006), pp. 123–134.

[74] STEENWEG, R., KUHRMANN, M., AND MÉNDEZ FERNÁNDEZ, D. Software Engineering Process Metamodels – A Literature Review. Tech. rep., TUM, 2012.

[75] TER HOFSTEDE, A., AND VERHOEF, T. On the Feasibility of Situational Method Engineering. *Information Systems* (1997).

[76] TERNITÉ, T. *Variability of Development Models*. PhD thesis, Technische Universität Clausthal, 2010.

[77] TERNITÉ, T., AND KUHRMANN, M. Das v-modell xt 1.3 metamodell. Tech. Rep. TUM-I0905, Technische Universität München, 2009.

[78] TOLVANEN, J.-P., ROSSI, M., AND LIU, H. Method Engineering: Current Research Directions and Implications for Future Research. In *Proceedings of IFIP TC8, WG8.1/8.2 Working Conference on Method Engineering* (1996).

[79] VAN DE WEERD, I., BRINKKEMPER, S., SOUER, J., AND VERSENDAAL, J. A situational implementation method for web-based content management system-applications: method engineering and validation in practice. *Software Process: Improvement and Practice 11*, 5 (2006).

[80] WIERINGA, R., MAIDEN, N., MEAD, N., AND COLETTE, R. Requirements engineering paper classification and evaluation criteria: a proposal and a discussion. *Requirements Engineering 11*, 1 (2005), 102–107.

[81] WISTRAND, K., AND KARLSSON, F. Method Components – Rationale Revealed. In *Advanced Information Systems Engineering*. Springer, 2004.