Criteria for Software Process Tailoring: A Systematic Review

Georg Kalus, Marco Kuhrmann
Technische Universität München – Software & Systems Engineering
Munich, Germany
{kalus,kuhrmann}@in.tum.de

ABSTRACT

Independently from which software process was selected for a company or a project, the selected software process usually cannot be applied without any customization. Although the need to tailor a software process to specific project requirements seems to be widely accepted and unquestioned, the way of doing the tailoring remains unclear and is, therefore, often left to the expertise of process engineers or project managers. What are the criteria to be applied in the tailoring? What are dependencies between different criteria and how should certain criteria influence the software process? In this paper we investigate concrete tailoring criteria for the tailoring of software processes. To this end, we present a collection of 49 tailoring criteria as the outcomes of a systematic literature review. We further analyze the impact of the discovered tailoring criteria by relating them to a set of 20 exemplary tailoring actions, which affect the project-specific software process. Our outcomes show that the factors influencing the tailoring are well understood, however, the consequences of the criteria remain abstract and need to be interpreted on a project-per-project basis.

Categories and Subject Descriptors

D.2.9 [**Software Engineering Management**]: Software process models

General Terms

Management, Experimentation

Keywords

Software Process, Tailoring, Systematic Literature Review

1. INTRODUCTION

Tailoring is defined as "the act of adjusting the definition and/or particularizing the terms of a general description to derive a description applicable to an alternate (less general) environment [20]." It is commonly accepted that any software process needs to be tailored to the particular project's requirements, as it becomes, otherwise, a project risk [5, 6, 37]. Popular software processes, e.g., PRINCE2 [1], Rational Unified Process [28], recommend—at least—to tailor the standard process before applying it in a project. Even agile methods recommend the selection of appropriate practices for a software project. Software process metamodels as a means to describe software processes, e.g. SPEM [35], usually define support to customize software processes. For instance, SPEM contains language constructs that support (delivery) process configurations (coarse-grained operations to configure a software process in general) and variability operations as a means to fine-tune certain software process elements, e.g. workflows. On the other hand, SPEM does not name criteria that support the selection of a particular process configuration nor does SPEM support a process engineer in determining adequate criteria during the process design. The same holds, e.g. for the German V-Modell XT [49] in which several coarse- and fine-grained tailoring criteria are defined, or the Swiss Hermes method [17] that also defines some selection criteria for a project-specific software process. Furthermore, in the field of agile methods no common guideline is available to support the selection of concrete agile practices, e.g. does pair programming work in particular settings, or is the "Follow-the-Sun" approach appropriate to operate a project. Therefore, the following questions often remain unanswered:

- What are the characteristics of a project that shall be considered for tailoring?
- What do the characteristics mean for the resulting software process?

Much research has been conducted to answer those questions, often independent of a concrete software process, e.g. [52]. The majority of current research is given by experience reports in which the feasibility of certain process-related management decisions was evaluated, but missing a systematization and a generalization.

1.1 Problem Statement

Although the need for tailoring software processes is unquestioned, current research in this area mostly addresses the feasibility of process-related project management decisions that deal with certain project situations. Yet, missing are an understanding of what are basic criteria to be considered during the process tailoring and, consequently, a catalog of documented and proven tailoring criteria that

provide guidance to process engineers and project managers when selecting and tailoring a software process for a concrete software project.

1.2 Research Objective

To overcome the shortcomings stated above, we conduct a systematic literature survey (SLR) to get an overview over the state-of-the-art in software process tailoring. We create a catalog of tailoring criteria reported in literature and analyze those criteria for their implications on project-specific software processes and project environments.

1.3 Contribution

We contribute a study on the state-of-the-art in software process tailoring, which results in a catalog of tailoring criteria that are documented in literature and outline the implication of applying those criteria.

1.4 Outline

The remainder of the paper is organized as follows. In Sect. 2, we discuss work related to our contribution, what gaps are left open, and how we intend to close those gaps. We introduce the study design in Sect. 3, before discussing the results of the study in Sect. 4. In Sect. 5, finally, we conclude the paper with a discussion of our results, what implications the results have, the limitations of our study, and future work.

2. RELATED WORK

Pedreira et al. [38] conducted a study to get a deeper understanding about the current practice in software tailoring. They conclude, that one of the problems is that existing approaches for software process tailoring are defined in specific environments and vote for developing a general framework. Xu and Ramesh [52] conduct an interview-based study on software process tailoring in order to support project managers to tailor software processes and, therefore, investigate challenges software projects face. Based on those challenges, Xu and Ramesh derive requirements w.r.t. the tailoring, group those requirements and, finally, develop strategies to cope with the challenges. Xu and Ramesh conclude that tailoring affects both, the software process as well as the project environment. They also note that excessive tailoring can "also involve significant overhead, undermining process repeatability and consistency across development projects", and, therefore, vote for creating company- or project-wide tailoring guidelines.

Cockburn [13] presents three high-level criteria *communication*, *criticality*, and *priority*, which he considers to be adequate to select a project-specific methodology. A number of criteria to tailor software processes can be found in the literature on (Situational) Method Engineering (SME; [11, 21]). Since empirical work is basically absent in the SME domain [43, 31], those criteria are "only" proposed in that particular literature yet missing an evaluation. However, although staying on a conceptual level, contributions such as the S^3 method (Harmsen et al. [22]) also vote for a joint analysis of influence factors and success criteria to select an appropriate software process.

Existing studies either provide first information on software process tailoring or propose tailoring criteria [13]. Pedreira et al. [38] initiated the discussion by analyzing the state-of-the-art in software process tailoring in general. Xu and Ramesh [52] conducted a study, which is based on interviews and provided a first catalog of challenges and measures to deal with changing project situations. However, those contributions stay on a generic level, yet missing guidelines that support the application of such criteria in projects. It remains unclear, e.g. how to implement Xu and Ramesh's strategies in a concrete project environment, and what the consequences of applying those strategies are.

In this paper, we contribute a systematic literature review to develop a catalog of tailoring criteria. We do not rely on interviews, but on reported experiences to create an initial catalog, which is backed up by literature. Furthermore, we screen standard literature on software project management for measures that are referred to be "standard measures" when dealing with changing project environments that require process customization to develop a notion of software process tailoring and its consequences to software processes and project environments.

3. STUDY DESIGN

We design the study as a combination of methods used for a systematic literature review (SLR, Kitchenham et al. [27]). The following study design itself is structured according to the proposal of Runeson et al. [40].

After defining the research questions of the study, we describe how we select the case. Finally, we describe how we collect and analyze the data, before concluding with a discussion on the validity procedures.

3.1 Research Questions

Our overall goal is to investigate the current state-of-the-art in software tailoring. To this end, we aim at answering the following research questions:

RQ 1 Which tailoring criteria are defined in literature?

RQ 2 What implications do the tailoring criteria have?

To answer RQ 1 we conduct a systematic literature review in which we use "snow-balling" to collect key words that are used to build the search strings to be used for an automated literature database search. To answer RQ 2 we use the resulting set of relevant contributions as input for an in-depth analysis w.r.t. concrete tailoring criteria and implications of applying those criteria to a project-specific software process.

3.2 Case Selection

We refer to the case selection by following a pragmatic, but time-intensive procedure. We first structure the publications and, thus, lay the foundation for the search strings by following the principles of snow-balling [27]. 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 re-

Table 1: Key contributions for snow-balling.

Author	Contribution
Pedreira et al.	[38]
Brinkkemper, Sjaak	[11]
Harmsen et al.	[22, 21]
Xu Peng et al.	[52]
Méndez Fernández et al.	[33]

search step was implemented in the context of a cooperation project with the T-Systems International GmbH, the Telekom Laboratories (T-Labs), and the Technische Universität München[1], and resulted in a set of standard contributions used for testing research questions, search strings, and structuring the publication flora. For this primary search, we refer to the authors and publications summarized in Table 1, which later on also serve as control values. The second step was the automated search in several literature databases, which we introduce in the following.

3.3 Data Collection Procedures

The data collection is an automated search in several literature databases. The queries are built based on the keyword lists given by the most common terminology in the area of software processes. As main data sources, we rely on established literature databases, which we consider most appropriate for a search. The internal discussion about which databases to select is based on our experiences in the process engineering domain (e.g. which conferences are in the field and which journals are of interest). In consequence we select the following databases:

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

If there is a paper listed in one of those databases, but is only referred, we count it for the database that generates the item, regardless of the actual publication location. In addition to those databases, we also take contributions into account that are not referred by the databases, but have to be considered as key contributions, e.g. books such as [25, 19, 42, 17]. For such contributions, we add a category "misc". To structure the data, we create a spreadsheet that contains the attributes shown in Table 2.

Table 2: Data collection table (simplified).

Info. Set	Attributes
Meta data	ID, Citation-key,
Content	Authors, Title, Abstract, Year
Voting	Relevance (defined during further analysis and voting by the different authors), Comments

3.3.1 Query Definition

To harvest the publication flora, we define our queries. For the beginning, we take a sample of relevant papers, analyze them in order to identify and iteratively refine the search strings, and validate them against a pre-defined list reference authors to be part of the search results (see Sect. 3.2). The initial set of key words is: {*software, development, process, tailoring, method, methodology, customization, customisation, adaption, adaptation, ISO, CMMI, SPICE, standard, compliance, study, experience, weaving, situational, engineering, practice*}.

[1]The cooperation aimed at improving the tailoring process of T-System's standard software process [42]. The research, which is presented in the paper at hands, is the basis for defining a tailoring concept, which considers project parameters as well as dependencies among certain parameters. A first technical prototype of the *feature-based tailoring* is described in [18]. The overall concept is currently under evaluation at T-Systems.

Table 3: Final search strings used for query.

	Search string
S1	(software process tailoring)
S2	(method engineering)
S3	(process **or** method **or** methodology) **and** (tailoring **or** adaption **or** customization **or** customisation)
S4	process tailoring **and** (practice **or** experience **or** study)
S5	software process **and** (standard **or** CMMI **or** ISO **or** SPICE) **and** compliance
S6	method **and** (engineering **or** weaving) **or** situational method engineering

Based on the initial searches and the analyses, we conclude the search strings shown in Table 3. Due to the complexity of the publication flora in the different research communities, we do not further distinguish between primary strings and secondary strings (Sect. 3.2). We use the search strings and aforementioned literature databases for the data collection. Each result set is transferred to a spreadsheet with the structure shown in Table 2. Having the single result sets available, all results are combined and used as basis for the data analysis. For each data source, at most the top 160 search results are taken into account.

3.4 Analysis Procedures

In the following, we describe our analysis procedure. We describe the analysis preparation as well as the in-depth analysis.

3.4.1 Analysis Preparation

To get the initial set of data to be analyzed, we perform an automated search that requires us to filter and prepare the result set. The data analysis is prepared by harmonizing the data and performing a 2-staged voting process to prepare the in-depth analyses.

Harmonization. After a first sight of the search results, we see that many contributions occurred multiple times in one result set. To make the selection of the contributions more efficient, we first clean the result set by eliminating multiple occurrences.

Voting. We perform a 2-staged voting process to classify the papers as relevant or irrelevant and to build a set of contributions for further investigation. The integrated result table therefore contains three columns (attribute "relevance" in Table 2). 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 voting were (1) the title of the contribution and (2) the abstract.

In the second voting stage, we only consider contributions that are not finally decided in the first stage and call in a third reviewer. This third reviewer also works with the

integrated table and votes by following the same criteria as in the first voting stage. The goal of this stage is to figure out the contributions that are relevant for the in-depth analyses.

3.4.2 In-depth Analysis

After the preparatory voting, the final set to be further investigated is defined. On this set, the in-depth analysis is performed using the complete contribution. In the following, we summarize the analysis procedures used to answer our research questions.

RQ 1 – Tailoring Criteria in Literature.
To analyze which tailoring criteria were described in literature, we screen the result set and create a spreadsheet that names the criteria, contains brief descriptions, and cites the contributions in which the criteria were named.

RQ 2 – Implications of Applying Tailoring Criteria.
To analyze the implication of applying certain tailoring criteria, we analyze each tailoring criterion for evidence from literature or—at least—a rationale. Based on the evidence and rationale we identify actions that *can* be chosen to affect a project-specific software process and assign those actions to the tailoring criteria.

3.5 Validity Procedures

To increase the validity of our study, we refer to two particular procedures. First, we analyze the area of investigation in advance and conduct a snow-balling procedure to infer and iteratively readjust the search strings by following the snow-balling procedure. This increases the construct and the external validity, since we perform our analysis on a small, but representative set of publications.

Second, we refer to researcher triangulation within a rigorous multi-staged voting procedure (Sect. 3.4.1). The voting procedure allows us to select the relevant papers from the irrelevant ones and to classify them appropriately. This procedure is accompanied by an in-depth analysis of the contents of the papers going beyond the abstracts, which we see as necessary as software process tailoring still remains a multi-facetted area with various interpretations in the provided concepts and the terminology used.

4. RESULTS

In the following, we describe the results of our study and conclude with a short interpretation. Table 4 summarizes the set of papers resulting form the collection and preparation phases. We summarize the databases, 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. Summarized 127 contributions were selected for the in-depth analyses.

Table 4: Survey Results

Source	Number	Number (cleaned)	Selected
ACM	210	210	44
Springer	60	60	13
IEEE	210	210	22
Wiley	1120	329	44
Misc			4
Sum	1600	809	127

4.1 Tailoring Criteria in Literature (RQ 1)

To answer research question 1, we analyze the contributions for tailoring criteria. We collect the found criteria and structure them in Table 5. Each item of the table is named, briefly described, and refers to the contributions from the result set. Furthermore, Table 5 contains a column that refers to appropriate actions that affect the project-specific software process when applying the tailoring criteria (cf. Sect. 4.2).

Table 5 lists 49 tailoring criteria from literature. Since the criteria are not clearly positioned, the table contains the groups *Team*, *Internal Environment*, *External Environemnt*, and *Objectives*, which we choose to categorize the criteria according to their emphasizes. For each criterion, a rationale (*What does a concrete criterion exactly mean?*) and possible implications (*What might happen when not considering this particular criterion?*) are taken from literature, e.g., [48, 47, 1, 39], to argue why a specific criterion needs to be regarded in certain project settings.

Analysis. More than half (28) of the criteria address environmental aspects, e.g., availability of stakeholders, users and the management, or project parameters, e.g. budget or contracting. Nine criteria address team-related parameters, e.g., size, distribution pattern, and, finally, 12 criteria address aspects w.r.t. the project's objectives, e.g. degree of innovation or complexity. The most frequently mentioned criterion is *team size* (eight mentions: [52, 15, 16, 22, 33, 7, 2, 14]).

Interpretation. A number of criteria overlap to a certain extend, e.g., *management support* and *management availability* (taken from [52, 22]). Those criteria consider a specific project situation from different perspectives, e.g., if the management does not support a project, it does not matter whether the management is available, whereas given management availability does not necessarily imply a (useful) support. Other criteria, e.g., *client availability* [24, 23] and *user availability* [7, 22, 14] can represent—depending on the actual project setting—the same stakeholder focus group, and, furthermore, the criterion *stakeholder availability* [52, 33] can be a means to generalize concrete stakeholder focus groups, e.g., the management, the client, the users.

4.2 Implications of Applying Tailoring Criteria (RQ 2)

To answer research question 2, we investigated given literature, e.g., [52], as well as literature that is common in project management, especially such that is dealing with risk management and influence factors on software projects, e.g., [48, 47, 1, 39]. Table 6 lists 20 exemplary actions, which are often named as appropriate measures in software process tailoring.

We relate the actions from Table 6 to the tailoring criteria (Table 5, column *Actions*) to get insights into the effects the selection of a certain tailoring criterion has. The exemplary assignment shows that a criterion can be related to several actions and, vice versa, an action can also serve several criteria.

Table 5: Catalog of tailoring criteria and actions

Name	Rationale & Implication	Actions	Ref.
Team			
Size	The team size is an indicator for the effort of team coordination [50]. While smaller teams located in a single room can directly communicate the need for formalization increases if the team grows (also if distributed/virtual teams become relevant). Team size is one of the key criteria when selecting a software process, e.g., [17, 14].	ADF, ADI, ASI, ACP, ACM	[52, 15, 16, 22, 33, 7, 2, 14]
Distribution	Team distribution influences the interaction pattern in a project [24, 23]. Teams located in a single room can directly communicate while distributed teams need a more formalized communication.	ADF, ADI, ASI, ACP, ACM	[3, 14, 29, 33]
Turnover	If a team member leaves a team, knowledge will also be lost. Furthermore, if new team members enter a team, effects of group dynamics according to Tuckman [45] will occur [41].	ADF, ADI, ASI, ACP, ACM, AKM	[52]
Previous cooperation	If the team worked together in previous projects the need for getting familiar with the other team members may decease which, in turn, may cause a less formal communication.	ADI, ACM	[52]
Good cooperation	If the team works in a good and collaborative way the need for formalized communication/documentation may decrease.	ADI, ACM	[52]
Domain knowledge	Little or missing knowledge w.r.t. the actual domain is a risk [47, 48].	AIW, ATT, AKM	[52, 22, 33]
Tool knowledge	Little or missing knowledge w.r.t. the actual tools is a risk [47, 48].	ATT, ATI	[52]
Technology knowledge	Little or missing knowledge w.r.t. the technology is a risk [47, 48].	ATT, APD, ASI	[52]
Process knowledge	Little or missing knowledge w.r.t. the process to be used is a risk [47, 48].	ADF, ATT, AKM	[52, 22]
Internal Environment			
Protoyping	The creation of prototypes is a strategy for risk mitigation [9, 10], and performance improvements [8], which is applied to projects that explore a new domain or a new technology, if the requirements are volatile, or if several solutions shall be evaluated.	ARE, APD, AFL, AIW	[25, 19]
Clear project proposal	A clear project proposal is an essential artifact. It contains basic goals and requirements [47, 48, 53] essential for the project's success. A blurry proposal is a risk.	ARE, AMI, APD, AFL, AIW	[52, 22]
Management availability	Top management is required to solve problems and to make project progress decisions. In critical project settings, a missing top management availability is a risk [47, 48].	AMI, ADF, ACP	[52]
Management support	The top management should actively support a project. This is important to have the management's support in critical situations [47, 48].	AMI, ADF, ACP	[52, 22]
Project budget	Project budget influences the degree of formalism in a project. A little project budget usually implies a non-formalized process (less documentation), but also requires a strict controlling w.r.t. costs.	AFI, ADI, ACM	[52]
Project duration	The project duration is a factor directly influencing the software process. While a "long" duration might cause risks (i.e. team turnover), a "short" duration is similar to a little project budget. Consequently, a software process needs to pay attention to the duration [8].	APD, AFL, APP	[52, 33]
Project type	Depending on the project- or service type, different facets of a software process need to be emphasized, e.g., kind of requirements elicitation, addressed life cycle phases, system migration. This criteria influences the software process's structures in general.	ARE, AAR, AIT	[42, 18, 25, 19]
Project role	Each project has a specific role that characterizes a project in relation to other projects. The typical setting is a client and a contractor where the contract defines the actual roles. Depending on the project role, the corresponding software process is entirely influenced [36].	ARE, AAR, AIT	[25, 19]

Table 5 – continued from previous page

Name	Rationale & Implication	Actions	Ref.
Sub contractors	Certain tasks in a project can be done by sub contractors, and, thus, a software process should support synchronization by defining artifacts to be exchanged, and process interfaces for collaboration [36, 23]. Furthermore, when sub contractors are in the projects, the contractor's project role changes; he becomes the client for the sub contractors.	ARE, AAR, AIT	[42, 18, 25, 19]
Financial controlling	The emphasis on a financial controlling is important if a project is critical w.r.t. the project budget. An intensive financial controlling results in a self-contained documentation as well as in an increased participation in the planning and decision making processes.	AMI, AFI, ADF, ACP	[25, 19]
Measurement	Measurement using KPIs is important, e.g., to provide the top management with status information, to measure the project's performance and to coduct data that are important for a company-wide controlling [6, 34]. Measurement causes additional effort in a project and also requires a more detailed reporting.	AMI, ADF, ACP	[25, 19]
Technical support	An unknown technical environment with little support may cause project risks [47, 48].	ATI, ATT, AKM	[52]
Programming language	A new programming language can cause project risks. Furhermore, a concrete programming language can influence the software architecture in a project.	ATT, AAR, AIT	[42, 18]
COTS products	The integration of COTS products or components can, on the one hand, shorten the development time, but on the other hand, require the team to pay attention, e.g., to legal implications, to test and integration procedures [46, 44].	AIT, APD, AFL	[42, 18, 25, 19]
Operating system	An operating system limits, e.g., the available programming languages, tools and potentially available COTS components. On the other hand, system requirements can also limit the supported operating systems for the intended solution.	AAR, AIT	[42]
Database system	Support for databases is, especially in business information systems, essential.	ARE, AAR, AIT, APD, AFL	[42]
Tool infrastructure	Tools that shall be used in a software project—or in a specific phase such as requirements engineering or coding—have to be defined. The definition of a tool infrastrucutre has certain implications, such as: tools that are not available have to be bought.	ATI	[42, 29, 30]
External Environment			
Legal aspects	Projects are critical in terms of legal aspects (i.e. contracting [32]) for several reasons. When not delivering the ordered software in time nor with the defined functionality recourse claims can occur. Furthermore, especially software with high requirements w.r.t. safety & security requires detailed documentation according to laws and regulations. This causes the software process to be very formal.	ARE, ADF	[15]
Number of Stakeholders	The higher the number of involved stakeholders the more time is required to negotiate all needs and requirements. Furthermore, the coordination effort increases [26]. A software process should pay attention to the number of stakeholders by defining adequate communication and reporting pattern.	ACI, AUI, ADF, ACP, AIW	[52]
Stakeholder availability	Similar to the availability of the top management, the availability of stakeholders is a success factor. Missing stakeholder availability is a project risk as it may cause delays [26, 47, 48].	ACI, AUI, ADF, ACP, AFL	[52, 33]
Stakeholder background	If the stakeholders' background is not adequate (e.g., due to a new domain, a new technology, or a new innovative product), the software process should provide different strategies, e.g., for requirements elicitation to support for learning curves [26].	ACI, AUI, ATT, AKM, AIW	[52, 33]
Requirements stability	The stability of requirements directly influences the entire approach in a project [26, 47, 48, 53], e.g., the strategy for requirements elicitation, architecting the solution, implementation and test.	APD, AFL, ARE	[52]
Client process	In case a client has its own software process and wants to align contractors with this process, the used software processes have to support process interfaces or certain procedures to comply with the client's process requirements, i.e. ensuring certain CMMI levels.	ARE, AIT, AFL	[42]

Table 5 – continued from previous page

Name	Rationale & Implication	Actions	Ref.
Client availability	The client's availability influcences the customer satisfaction since the client can continuously monitor progress. For instance, agile methods propose the *on-site-customer* [24, 23]. Regular deliveries in short cycles can compensate a missing availability.	AFL, ACI, AUI	[52, 33]
Type of contract	The type of the contract [32] directly influences the software process, e.g., a fixed-price model vs. time & material leads to different strategies in handling change requests.	ACI, AUI, ADF, ACP	[42]
User availability	The end user availability is important during the requirements engineering, and the integration and testing phases [24, 23].	AUI, ARE, AAR, AIT, AFL, APD	[7, 22, 14]
User background	The end user background/domain knowledge is important to decide about the required training. Also, appropriate (acceptance) testing strategies should be selected according to the end users' knowledge.	AUI, AIW, ATT, AFL, APD	[52, 33, 22]
Trainings	The requirements regarding training, e.g. during the acceptance testing or the deployment cause additional effort (planning of tranings, creation of training material) [48, 8].	ATT	[42]
Objectives			
Complexity	The complexity of a project directly influcences the process [4, 51, 12], e.g., by creating prototypes or following a *divide and conquer* strategy (creating sub projects). A higher complexity usually causes a more comprehensive software process, more (formalized) communication, a more formalized configuration and change management and so on.	ADF, AIW, ACP, ARE, AAR	[22]
Degree innovation	The degree of innovation may cause a more explorative approach to mitigate project risks [47, 48].	APD, AFL	[33, 22]
Legacy system	A legacy system needs to be considered during the requirements engineering and, usually, limits the solution space, e.g., by compatibility requirements, data migration.	APD, AIT, ARE, AAR	[25, 19]
Legacy system documentation	If there is no documentation for a legacy system available, higher effort in analyzing the legacy system has to be expected. In such a case, a software process should provide appropriate strategies, e.g., reverse engineering.	APD, AIT, ARE, AAR	[52, 33]
Domain	A domain implies certain standards, norms, regulations, and laws that need to be considered in a project. According to such requirements an adequat software process has to fulfill such requirements. Depending on the actual domain, further potential criteria exist that should be regarded in the tailoring, e.g., special requirements for Cloud software.	ARE, AAR	[42]
Conceptual solution	A software process can contain domain-specific knowledge and best practices, e.g., blue prints, architecture pattern, to support the development of the conceptual solution.	AIW, ARE, AAR	[42]
Technical solution	The technical solution can be supported by specific contents regarding, e.g., web development, desktop applications, application of design pattern, coding guidelines for programming languages and so on.	ARE, AAR	[42]
Safety & Security	Safety & security usually cause a comprehensive documentation of a project. The software process should, therefore, provide templates and hints about the documentation.	ADF, ACP, ARE	[42, 25, 19]
Hardware development	If hardware development is also part of a project, the software process has also to provide corresponding artifacts (e.g., hardware specifications and designs, test hardware, logistics) that are consistently integrated in the software process.	ARE, AAR, AIT	[25, 19]
Neighboring systems	A software system is, usually, part of an integrated ecosystem. Interfaces to this ecosystem need to be defined as they directly influence the integration and test strategies.	ARE, AAR, AIT	[42]
User Interface	If a software has special requirements w.r.t. the user interface, the design, implementation, and test should be part of the software process.	ARE, AIT	[42, 25, 19]
System integration test	Requirements w.r.t. system integration should be reflected by the software process, e.g., by defining integration strategies, providing a fundamental integration of project management, development, and quality assurance.	ARE, AAR, AIT, ATI	[42, 25, 19]

Table 6: Derived actions clustered into groups.

	Stakeholder-related actions
ACI	intensify customer involvement
AUI	intensify end user involvement, e.g. ui testing
AMI	ensure management involvement
AUT	intensify end user trainings
	Project life cycle actions
ARE	put emphasis on requirements engineering
AAR	put emphasis on system architecture
AIT	put emphasis on integration and test
AFI	put emphasis on financial project management
APD	put emphasis on prototype development
AFL	put emphasis on fast feedback loops, e.g., continuous delivery and deployment, on-site-customer
APP	put emphasis on planning pattern for time critical development, e.g., Follow-the-Sun, time boxing
	Project organization actions
ADF	expand project documentation, e.g., formalized documentation using templates
ADI	reduce documentation, e.g., replace paper-based documentation by stand-ups
ASI	increase number of (micro-)iterations
ACP	formalize project communication pattern
ACM	foster open project communication
ATI	select appropriate tools w.r.t. the process's weight
	Knowledge building/preservation actions
AIW	intensify meetings/workshops, e.g., specific for certain stakeholder focus groups, shorten intervals between workshops
ATT	provide trainings, e.g., role-specific trainings
AKM	provide knowledge management infrastructure

Interpretation. Considering a tailoring criterion to be relevant in a particular project situation can significantly influence the resulting software process. For instance, the criterion *tool infrastructure* is related to the action *ATI*. However, it remains unclear what is meant by selecting appropriate tools. Cockburn [15] votes for selecting the tools w.r.t. the actual process' weight that, in turn, can result into completely different tool ecosystems, which make it difficult for a company to implement a common tool infrastructure (according to Xu and Ramesh, this can undermine repeatability of processes across development projects [52]). It also remains unclear who decides which tools are appropriate, e.g., if such a decision is made by the team itself, a whiteboard could also be mentioned to be an appropriate tool (which can also be a problem: In a company, a team decided to use Scrum, but was not allowed to install a whiteboard as regulations for fire safety forbid the installation $\mapsto$ a decision w.r.t. tools turned into an organizational issue).

Although, the selected actions from Table 6 are referred in literature to be standard measures, implications on the software process and, further, on the project operation can so far not be derived from our data.

5. CONCLUSION

The paper at hands closes a gap in the software process tailoring literature by conduction a systematic literature review to investigate tailoring criteria. We contribute a catalog of 49 tailoring criteria, which we extracted from literature and that we backed up by an in-depth analysis in which we investigated the rationale and the implications of the investigated tailoring criteria. The catalog is a means to support the management to understand influence factors for software projects and, furthermore, to develop a notion of the implications such influence factors have on the requirements w.r.t. the project-specific software process as well as on, e.g., the project organization, project planning, budget, and training demand.

Furthermore, we analyzed the consequences of the investigated tailoring criteria on software projects. To this end, we formulate 20 actions, which are referred to be standard measures to deal with certain project situations and relate them with the tailoring criteria. Our results do, so far, not allow for explaining how concrete actions affect a software project as software projects underly manifold influences.

Althogh our results do not allow for deriving concrete implications of tailoring criteria on software projects, it is necessary to prepare software processes to provide support for the definition of tailoring criteria as the necessary variability of software processes in inherently given by the complexity of todays software projects. The contributed catalog of tailoring criteria that we provide with the paper at hands, therefore, lays the foundation for further investigation.

5.1 Relation to Existing Evidence

In 2007 Pedreira et al. [38] conducted a first systematic literature survey to analyze software process tailoring. They investigated the state-of-the-art in software process tailoring in general to foster further research. A study on influence factors and strategies to deal with tailoring is contributed by Xu and Ramesh [52]. However, although providing challenges and strategies for mitigation, their contribution does not state how to apply the proposed tailoring strategy in specific project environments. Concrete experiences and empirical evidence on the feasibility of proposed strategies and criteria is, however, still yet missing.

The study at hands does neither provide a strategy nor a new approach to tailor software processes. Our contribution aims primarily at investigating and structuring the publication flora to develop a catalog of reported and proven tailoring criteria. Our study is a further step towards developing a notion of tailoring criteria and their implication to software projects in order to lay the foundation for future research. To this end, we built our research on the work of Xu and Ramesh, extended the list of tailoring criteria, and provided reasons w.r.t. the importance of specific criteria that are based on standard literature.

5.2 Impact/Implications

To create a complete and generalizable list of tailoring criteria is—if possible at all—a challenging task, wherefore, our research has several implications. Practitioners can apply our contributed catalog in their own projects and, furthermore, companies can use the contributed catalog as a starting point to define appropriate tailoring criteria for their software processes and projects.

Although we close a gap in the literature on software pro-

cess tailoring, it remains unclear whether the contributed list of tailoring criteria is complete. Furthermore, a number of the investigated criteria stays on a fairly generic level, which makes it difficult to apply them to concrete software processes. Further research is necessary to extend and refine the contributed catalog. Our research also shows that there still is a gap in literature w.r.t. the implications of tailoring criteria. Further studies are necessary to close this gap and to foster the discussion on software process tailoring, especially to give empirical evidence on the feasibility of concrete actions that were performed because of applying a specific tailoring criterion.

Finally, to support software process tailoring, also software process metamodels have to provide process engineers with appropriate means to design tailoring criteria and to integrate them with a software process. Software process metamodels, such as SPEM, do not provide mechanisms that are mature enough to define a tailoring that goes beyond coarse-grained export configurations.

5.3 Limitations

This paper aims at creating a catalog of tailoring criteria and, thus, has some limitations. Deeper insights and analyses w.r.t. conceptual, methodical, and technical aspects of software process tailoring are not part of this study. Furthermore, this study does not aim at creating taxonomies, generalized concepts, or strategies to tailor software processes. This study neither does provide any solution or improvement proposal, as the scope was to harvest the publication flora and to develop an integrated and structured catalog of tailoring criteria to foster the discussion on software process tailoring.

5.4 Future Work

In response to the implications our research has, future work addresses the in-depth investigation of tailoring criteria and their implications to lay the foundation to conduct studies on the feasibility of software process tailoring.

On the "technical" layer, the analysis of software process metamodels for their capabilities w.r.t. support a fine-grained tailoring is another facet to be considered in future research.

6. ACKNOWLEDGMENTS

We want to thank Daniel Méndez Fernández for fruitful discussions on previous versions of this paper and for the support during the analyses of the publications.

7. REFERENCES

[1] *Managing Successful Projects with PRINCE 2*. The Stationery Office Ltd., 2009.

[2] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta. Agile software development methods - review and analysis. Technical Report 478, VTT PUBLICATIONS, 2002.

[3] T. J. Allen. *Managing the Flow of Technology: Technology Transfer and the Dissemination of Technological Information Within the R&D Organization*, volume 1 of *MIT Press Books*. The MIT Press, December 1984.

[4] D. Baccarini. The concept of project complexity—a review. *International Journal of Project Management*, 14(4):201–204, Aug. 1996.

[5] V. R. Basili and H. D. Rombach. Tailoring the Software Process to Project Goals and Environments. In *ICSE '87: Proceedings of the 9th international conference on Software Engineering*, pages 345–357, Los Alamitos, CA, USA, 1987. IEEE Computer Society Press.

[6] V. R. Basili and H. D. Rombach. The TAME project: towards improvement-oriented software environments. *IEEE Transactions on Software Engineering*, 14(6):758–773, June 1988.

[7] K. Beck and C. Andres. *Extreme Programming Explained: Embrace Change (2nd Edition)*. Addison-Wesley Professional, 2004.

[8] J. D. Blackburn, G. D. Scudder, and L. N. Van Wassenhove. Improving speed and productivity of software development: a global survey of software developers. *IEEE Transactions on Software Engineering*, 22(12):875–885, 1996.

[9] B. W. Boehm. A spiral model of software development and enhancement. *Computer*, 21:61–72, May 1988.

[10] B. W. Boehm. Software risk management: principles and practices. *IEEE Software*, 8(1):32–41, 1991.

[11] S. Brinkkemper. Method engineering: engineering of information systems development methods and tools. *Information and Software Technology*, 38(4):275 – 280, 1996. Method Engineering and Meta-Modelling.

[12] A. Camci and T. Kotnour. Technology Complexity in Projects: Does Classical Project Management Work? *Technology Management for the Global Future*, 2006.

[13] A. Cockburn. Selecting a project's methodology. *IEEE Softw.*, 17:64–71, July 2000.

[14] A. Cockburn. *Crystal Clear: A human-powered methodology for small teams*. Addison-Wesley Professional, first edition, 2004.

[15] A. Cockburn. *Agile Software Development: The Cooperative Game (agile software development series)*. Addison-Wesley Professional, 2006.

[16] G. Coleman and R. O'Connor. Investigating software process in practice: A grounded theory perspective. *J. Syst. Softw.*, 81(5):772–784, May 2008.

[17] Confédération Suisse. The HERMES Method. Online: `http://www.hermes.admin.ch`, 2011.

[18] D. Costache, G. Kalus, and M. Kuhrmann. Design and Validation of Feature-based Process Model Tailoring - A Sample Implementation of PDE. In *Proceedings of the 8th European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEM/FSE 2011)*, pages 464–467. ACM Press, Sept. 2011.

[19] J. Friedrich, U. Hammerschall, M. Kuhrmann, and M. Sihling. *Das V-Modell XT*. Informatik im Fokus. Springer, 2. edition, 2009.

[20] M. Ginsberg and L. Quinn. Process tailoring and the software capability maturity model. Technical Report CMU/SEI-94-TR-024, Software Engineering Institute, Pittsburgh, PA, Nov. 1995.

[21] A. F. Harmsen. *Situational Method Engineering*. PhD thesis, University of Twente, Utrecht, January 1997.

[22] F. Harmsen, I. Lubbers, and G. Wijers. Success-driven selection of fragments for situational methods: The s3 model. In *Proceedings of the Second International*

Workshop on Requirements Engineering: Foundations of Software Quality, volume 13 of *Aachener Beiträge zur Informatik*, pages 104–115, 1995.

[23] R. Heeks, S. Krishna, B. Nicholsen, and S. Sahay. Synching or Sinking: Global Software Outsourcing Relationships. *Software*, 2001.

[24] J. D. Herbsleb and A. Mockus. An empirical study of speed and communication in globally distributed software development. *IEEE Transactions on Software Engineering*, 29(6):481–494, June 2003.

[25] R. Höhn and S. Höppner. *Das V-Modell XT*. eXamen.press. Springer, 2006.

[26] J. J. Jiang, G. Klein, S. P. J. Wu, and T. P. Liang. The relation of requirements uncertainty and stakeholder perception gaps to project management performance. *Journal of Systems and Software*, 82(5):801–808, May 2009.

[27] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman. Systematic literature reviews in software engineering – A systematic literature review. *Information and Software Technology*, 51(1), 2009.

[28] P. Kroll and P. Kruchten. *The Rational Unified Process Made Easy – A Practinioner's Guide to RUP*. Addison-Wesley, 2003.

[29] M. Kuhrmann and G. Kalus. Providing Integrated Development Processes for Distributed Development Environments. In *Workshop on Supporting Distributed Team Work at Computer Supported Cooperative Work (CSCW 2008)*, Nov. 2008.

[30] M. Kuhrmann, G. Kalus, M. Then, and E. Wachtel. From Design to Tools: Process Modeling and Enactment with PDE and PET. In *Proceedings of Third International Workshop on Academic Software Development Tools and Techniques (WASDeTT-3), co-located with the 25th IEEE/ACM International Conference on Automated Software Engineer*, Sept. 2010.

[31] Kuhrmann, M., Méndez Fernández, D., and Tiessler, M. A Mapping Study on Method Engineering – First Results. In *International Conference on Evaluation & Assessment in Software Engineering*, 2013.

[32] Y. Lichtenstein. Puzzles in software development contracting. *Communications of the ACM*, 2004.

[33] D. Méndez Fernández, S. Wagner, K. Lochmann, A. Baumann, and H. de Carne. Field study on requirements engineering: Investigation of artefacts, project parameters, and execution strategies. *Inf. Softw. Technol.*, 54(2):162–178, Feb. 2012.

[34] R. J. Offen and R. Jeffery. Establishing software measurement programs. *IEEE Software*, 14(2):45–53, 1997.

[35] OMG. Software & systems process engineering metamodel specification version 2.0. Technical report, Object Management Group, April 2008.

[36] M. Paasivaara and C. Lassenius. Collaboration practices in global inter-organizational software development projects. *Software Process: Improvement and Practice*, 8(4):183–199, 2004.

[37] D. L. Parnas and P. C. Clements. A Rational Design Process: How And Why To Fake It. *IEEE Transactions on Software Engineering*, 12(2):1–10, Jan. 1986.

[38] O. Pedreira, M. Piattini, M. Luaces, and N. Brisaboa. A Systematic Review of Software Process Tailoring. *SIGSOFT Software Engineering Notes*, 32(3):1–6, 2007.

[39] Project Management Institute. *A Guide to the Project Management Body of Knowledge*. Project Management Institute, fourth edition, 2009.

[40] P. Runeson and M. Höst. Guidelines for Conducting and Reporting Case Study Research in Software Engineering. *Empirical Software Engineering*, 14(2):131–164, 2009.

[41] E. Salas, D. Rozell, B. Mullen, and J. E. Driskell. The Effect of Team Building on Performance: An Integration. *Small Group Research*, 30(3):309–329, June 1999.

[42] T-Systems. SE Book. internal handbook.

[43] A. H. M. ter Hofstede and T. F. Verhoef. On the feasibility of situational method engineering. *Information Systems*, 22(6-7):401 – 422, 1997.

[44] M. Torchiano and M. Morisio. Overlooked aspects of COTS-based development. *IEEE Software*, 21(2):88–93, 2004.

[45] Tuckman, B. W. Development sequence in small groups. *Psychological Bulletin*, 1965.

[46] J. Voas. COTS software: the economical choice? *IEEE Software*, 15(2):16–19, 1998.

[47] L. Wallace and M. Keil. Software project risks and their effect on outcomes. *Communications of the ACM*, 47(4):68–73, Apr. 2004.

[48] L. Wallace, M. Keil, and A. Rai. How Software Project Risk Affects Project Performance: An Investigation of the Dimensions of Risk and an Exploratory Model*. *Decision Sciences*, 35(2):289–321, May 2004.

[49] Weit e.V. The V-Modell XT Online Portal. Online `http://www.v-modell-xt.de/`.

[50] J. Wolfe and T. I. Chacko. Education TEAM-SIZE EFFECTS ON BUSINESS GAME PERFORMANCE AND DECISION-MAKING BEHAVIORS. *Decision Sciences*, 14(1):121–133, Jan. 1983.

[51] W. Xia and G. Lee. Grasping the complexity of IS development projects. *Communications of the ACM*, 47(5):68–74, May 2004.

[52] P. Xu and B. Ramesh. Using Process Tailoring to Manage Software Development Challenges. *IT Professional*, 10(4):39–45, July 2008.

[53] D. Zowghi and N. Nurmuliani. A study of the impact of requirements volatility on software project performance. In *Asia Pacific Software Engineering Conferenc*, pages 3–11. IEEE Comput. Soc, 2002.