On the complex nature of MDE evolution and its impact on changeability

Regina Hebig, Holger Giese

Noname manuscript No.
(will be inserted by the editor)

On the Complex Nature of MDE Evolution and its Impact on Changeability

Regina Hebig · Holger Giese

Received: date / Accepted: date

Abstract In Model-Driven Engineering (MDE) a particular MDE setting of employed languages, automated and manual activities has major impact on productivity. Furthermore, it has been observed that such MDE settings evolve over time. However, currently not much is known about this evolution and its impact on the MDE setting's maturity, i.e. on changeability and other *productivity dimensions*. Research so far focuses on evolution of separate building blocks, such as (modeling-) languages, tools, or transformation, only.

In this article we address the lack of knowledge about evolution of MDE settings by investigating case studies from different companies. The first results reveal (1) that there is evolution that affects the composition of an MDE setting (*structural evolution*) and has the potential to strongly impact aspects, such as changeability, and (2) that this structural evolution actually occurs in practice. Based on these first results we investigated, (3) whether there are cases in practice, where structural evolution already altered the risks for changeability given by the respective MDE setting. Therefore, we search and identify examples for such *evolution steps* on MDE settings from practice and collected 6 case studies on *evolution histories* in detail. As a result, we show in this paper that structural evolution (a) is not seldom in practice and (b) sometimes leads to the introduction of changeability risks.

R. Hebig
Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, LIP6, F-75005, Paris, France.
E-mail: regina.hebig@lip6.fr

H. Giese
Hasso Plattner Institute
at the University of Potsdam, Germany
E-mail: holger.giese@hpi.uni-potsdam.de

Keywords Model-driven engineering · Evolution · Empirical research

1 Introduction

Model-driven engineering (MDE) techniques, such as the use of models and (semi-)automated analyzes, transformations, or code generation are applied to improve productivity of software development as well as the quality of software. A chosen combination of these MDE techniques can affect diverse productivity related aspects, such as complexity of work, degree of automation, or changeability [11]. We refer to such a specific combination of (modeling) languages, automated activities (e.g. transformations), manual activities, tools, and artifacts as *MDE setting*[1] within this paper. A freely accessible example for such an MDE setting can be seen in Figure 1[2]. Manual activities, e.g. "Manipulate Implementation", and automated activities, e.g. "Generate Java Code", consume, manipulate, or create the different model and code artifacts, e.g. "Interface Implementation". These artifacts can be specified using different (modeling) languages, e.g. Ecore or Java. Finally, manual and automated activities are supported by tools, such as editors and frameworks like the Eclipse Modeling Framework.

It is known that single elements of an MDE setting, e.g. languages and automation steps, such as transformations or code generators, are subject to evolution in practice [9,27,40]. Consequently, the question arises

[1] In context of Software Configuration Management an MDE settings might be considered as model-based configuration.

[2] The EMF generation is described in detail in `http://www.vogella.de/articles/EclipseEMF/article.html`.

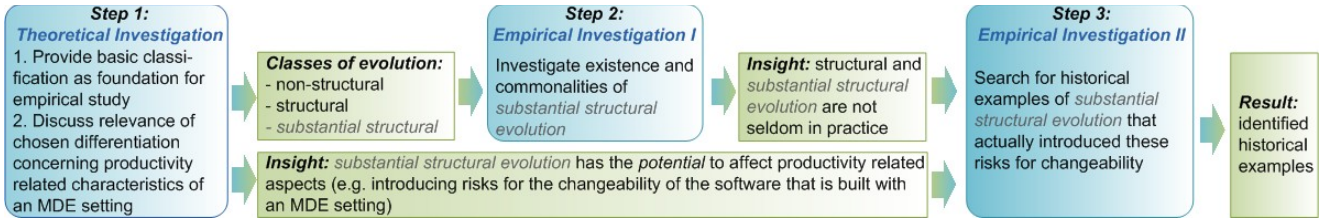

Fig. 2 Overview on the three applied investigation steps

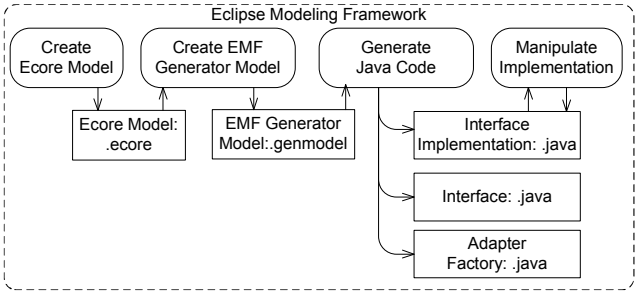

Fig. 1 Example for an MDE setting: generation of model code with EMF (illustrated in an Activity Diagram [31] based notation). The Ecore model is used to generate an EMF generator model, which is basis for the later generation of Java code.

what impact changes on an MDE setting as a whole do have. Do some types of changes have the potential to mature and change the way how this MDE setting affects different productivity related aspects? If so, there arise the two additional questions, whether these changes actually occur for MDE settings in practice and whether in consequence the effects of an MDE setting on productivity related aspects actually change in practice. However, nearly all existing studies focus on evolution of single languages or automation steps, only (e.g. [22],[26],[28],[27], or [6]) and do not study these raised questions on the overall maturation of an MDE setting. The only more complex case of evolution that can be found in literature is called "abstraction evolution" and affects the number of used languages [8].

In this paper, we investigate the raised questions in three steps, as summarized in Figure 2.

(1) We chose to introduce a basic classification of possible *atomic change* types on MDE settings, which allows us to systematically capture and differentiate observable changes. We discuss the relevance of the chosen differentiation concerning productivity related characteristics of an MDE setting. As a result of this discussion, we identify two groups of changes that affect the composition/structure of an MDE setting (later referred to as "*structural changes*" and "*substantial structural changes*"). We discuss that these groups have the risk and potential to affect the changeability support of an MDE setting (i.e. how an MDE setting can imply risks for the changeability of software that is built with this MDE setting).

(2) Second, we study whether changes that affect the composition/structure of an MDE setting actually occur in practice. Therefore, two different data sets are investigated, namely a set of 6 MDE settings that have been captured in context of a former investigation at SAP, as well as a set of 7 reports from literature that focus on MDE in practice. The results show that changes that affect the composition/structure of an MDE setting occur in practice and can be found for more than 25% of the MDE settings in practice.

(3) The first two steps show that changes that affect the composition/structure of an MDE setting have the *potential* to introduce risks for changeability into an MDE setting and that these changes are not seldom in practice. However, it is still not known whether the changes that affect the composition/structure of an MDE setting and do occur in practice actually introduce the risks for changeability. Therefore, we apply a third investigation step, which aims at identifying historical examples of occurred *structural evolution steps* that did affect the changeability support of the respective MDE setting. Therefore, we capture 6 *evolution histories* of MDE settings from 4 companies, all in all spanning 32 *evolution steps*. We use a pattern-based technique (presented in [11]) to analyze how the MDE settings that we captured affect changeability. In a next step we analyzed the *evolution histories* in order to identify *evolution steps* that introduced the identified changeability issues. As a result we could identify examples for *structural evolution steps* that worsened or improved the changeability support of the respective MDE settings.

In the following we introduce the contribution of this paper, which is an extended version of [14]. There the risks and potentials of different *structural evolution steps* were theoretically discussed and the existence and commonness of such *evolution steps*, which affect the composition/structure of an MDE setting, have been studied (*covering steps (1) and (2)*). In this paper, we extend this research in several directions. We enlarge

the set of captured data on MDE settings and their *evolution histories* from three to six case studies.

As new contribution, we extend the whole examination by identifying examples for *evolution steps* in industry that actually led to a worsened or improved changeability (*adding step (3)*). In addition, we substitute the early list of eight observations presented in [14] in favor of a more detailed discussion of four selected observations that is backed up by a more extensive data set.

While in [14] we opened the research question about the role of evolution that affects the composition/structure of an MDE setting in practice by showing that an impact of this evolution on productivity is plausible (i.e. that this evolution can have an impact and that this evolution exists in practice), in this paper we are able to provide the so far missing step to show that evolution that affects the composition/structure of an MDE setting really impacts changeability in practice. Therefore, this paper is able to establish the evolution that affects the composition/structure of MDE settings as a research question of high practical relevance and provides a basis for future research.

This paper is structured as follows. First we discuss different change types in Section 2. In Section 3, we investigate the existence of evolution that affects the composition/structure of an MDE settings in practice. Afterward, we investigate whether *evolution steps* that affect the composition/structure of an MDE setting actually led to the introduction of changeability issues in Section 4. In Section 5 we discuss the results and add some observations on motivations of evolution that affects the composition/structure of an MDE setting. We discuss related work in Section 6. Finally, we conclude and discuss implications that these results have for researchers and practitioners.

2 Theoretical Investigation: A Classification for Atomic Change Types

Available classifications of evolution in context of MDE concern the substitution of languages, tools, and automations, only. To study empirically, whether other types of changes can occur, too, it is first necessary to clearly identify what kind of changes might be of interest when studying evolution of complete MDE settings. Therefore, this section focuses on providing a classification of *atomic changes* types that are possible for an MDE setting. Note that in this paper, the evolution of software that can be built with an MDE setting (e.g. evolution of code and models) is out of focus. Although interaction of evolution of the MDE setting and evolution of the built software might be interesting subject to research, we focus in this paper on the general potential impact of an MDE setting, independent of the concrete software built.

Further, it will be discussed whether a differentiation between the possible change types is really relevant from the perspective of studying the impact on the productivity related characteristics of an MDE setting.

2.1 Atomic Change Types

The main idea of the classification is to identify a set of ***atomic changes*** that can occur to MDE settings and are the building blocks for the actual changes that happen during *evolution steps*.

Definition 1 (Atomic change) The set of *atomic changes* can be constructed in a straightforward manner, based on the elements listed in the definition of MDE settings (artifacts, manual and automated activities, languages, and tools). Possible changes are that existing elements are replaced or that the number of elements changes (i.e. that elements are added or removed from the MDE setting).

Exchanging existing elements does not affect the structure of an MDE setting. Therefore we call these changes types ***non-structural***.

Definition 2 (Non-structural change) *Non-structural changes*, where only existing elements are exchanged.

Elements in an MDE setting that might be exchanged during evolution are implementations of automated activities, the used (modeling) languages, and tools. In contrast, concrete actions within manual activities and data of artifacts are not assets of an MDE setting and are manifested during development with that MDE setting, only. Therefore, these elements cannot be exchanged in context of the evolution of an MDE setting.

The resulting *non-structural* change types are "exchange or evolution of an automated activity (e.g. any model operation or code generation)" (*A1*), "exchange or evolution of a used language" (*A2*), and "exchange or evolution of a used tool" (*A3*).

Definition 3 (Structural change) Changes in the set of elements affect the structure of an MDE setting and therefore are *structural changes*.

The resulting atomic *structural* change types concern adding or removing artifacts (referred to as *A4*), adding or removing languages (referred to as *A5*), adding or removing tools (referred to as *A6*), adding or removing manual activities (referred to as *A7*), and adding or removing automated activities (referred to as *A8*).

Table 1 Summary of *atomic change* types for MDE settings.

Change	Elements of an MDE Setting				
	Artifact	Language	Tool	Manual Activity	Automated Activity
exchange element		x	x		x
add or remove element	x	x	x	x	x

Based on the *atomic changes* captured in the classification (as summarized in Table 1), there are *complex changes*.

Definition 4 (Complex change) Complex changes consist of multiple *atomic changes* and can - beyond just changing the set of elements - affect the interrelations between elements of an MDE setting.

An example for a *complex change* that shall be considered here in addition to the *atomic changes* is the change of the order of manual and automated activities: Each change in the set of automated (*A8*) or manual activities (*A7*) leads to a change in the order of activities. We call the relative positioning of automated activities within (and behind) the manual activities *order of manual and automated activities*. Only some changes in the set of manual or automated activities also change this order of manual and automated activities. For example, an automated activity might be introduced between two manual activities. Note that it is possible that an added or removed manual or automated activity does not lead to the change in the order of manual and automated activities. For example, a new manual activity might simply be embedded between two already existing manual activities. Thus, each automated activity that was before the change not followed or preceded by a manual activity is afterward the change also not followed or preceded by a manual activity. Subsequently, we refer to this *complex change* as change type *C1*. Figure 3 summarizes the considered change types.

2.2 Discussion of Impact

Before studying whether the different identified change types play a role in practice, the relevance of the chosen differentiation between these change types shall be discussed here. Therefore, we discuss whether the different change types have similar or different potential to impact productivity related characteristics of an MDE setting.

An MDE setting can affect effort and costs for developers and a company in different ways. For example, the *degree of automation* of development depends on the set of automated activities, and the *complexity* of work depends on the set of languages and artifacts that need to be handled in parallel. Similarly, the *changeability and maintainability* of the software under construction, i.e. the software that is built with an MDE setting, might be affected via different aspects of an MDE setting [12]. For example, a concrete risk for changeability is the loss of manually created content, when a generation step needs to be re-executed. Whether this risk exists depends, amongst other influences, on the question whether generated artifacts in the MDE setting are manipulated by manual activities. E.g., when the artifact "EMF Generator Model" is not manually adapted as illustrated in Figure 1, activity "Create EMF Generator Model" might be newly executed in correspondence to a change of artifact "Ecore Model" without causing problems. However, if artifact "EMF Generator Model" was manually enriched after the first generation, as illustrated in Figure 4, a new execution of activity "Create EMF Generator Model" might lead to the loss of these manually created contents. This in turn leads to extra effort during each change and reduces the changeability.

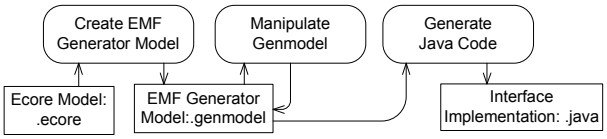

Fig. 4 Alternative version of a part of the MDE setting from EMF example, where additional changes in generator model are added by a manual activity.

Further, the effort required for *integration* and *maintenance of consistency* between artifacts is impacted by the set of tools and artifacts in an MDE setting. Finally, tools and implementations of automated activities are business assets of a company, which need to be maintained [35] and therefore affect the *cost of ownership* for a company. In the following, we refer to the listed aspects as *productivity dimensions*:

Definition 5 (Productivity dimensions) The term is used in this paper to refer to characteristics of an MDE setting that directly or indirectly would have to be in included in a hypothetical calculation of productivity, i.e. the overall effort and costs spend during software development. *Productivity dimensions* that are discussed in this section are: *costs of ownership*, *effort required for integration and maintenance of consistency*, *degree of automation*, *complexity of work*, and *changeability and maintainability of the software under construction*.

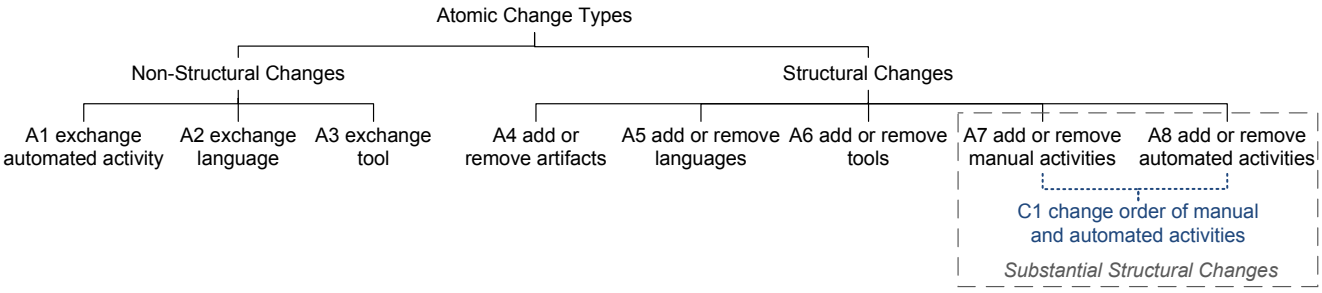

Fig. 3 Overview on *atomic change* types together with *complex change* type *C1*

Which of these *productivity dimensions* can be affected by the different change types is shortly discussed in the following. Changes in an automated activity (*A1*) and changes in the number of automated activities (*A8*) can affect the *degree of automation*. Changing a used (modeling) language (*A2*) or the number of used languages (*A5*) can have benefits concerning the degree of abstraction, but yields the risk that the developers lack the know-how to use that language (affecting the *complexity* of the MDE setting) [3]. Similarly, a growing number of models (*A4*), necessary manual activities (*A7*), or tools (*A6*) increases *complexity* for developers. In addition, a change in the number of models affects the need to *maintain the consistency* of different models [18]. Further, new tools can lead to additional activities to move artifacts between them increasing the *integration effort*. As tools and implementations of automated activities have to be maintained, changes in either number of tools *A6* or automated activities *A8* can affect the *cost of ownership*.

Finally, a main risk results from the addition of automated or manual activities if this leads to a change of the order of manual and automated activities (*A7*, *A8*, and *C1*). This can lead to constellations, where automatically created artifacts are touched manually. With it the above described risk for changeability is introduced. For example, the arising need to add a default values for attributes might cause the addition of the manual activity "Manipulate Genmodel" to the EMF example, changing a part of the MDE setting from Figure 1 to the situation shown in Figure 4. Due to the tangible character of this possible impact, we partly will consider the three change types separately in the remainder of this paper:

Definition 6 (Substantial structural changes) We refer to *structural changes* that can introduce constellations where automatically created artifacts are touched manually as *substantial structural changes*. Examples are *A7*, *A8*, and *C1*, as shown in Figure 3.

This discussion leads to two insights: first, every type of change has possible and plausible consequences and second, the change types differ concerning affected *productivity dimensions* and the way how these dimensions are affected.

2.3 Terminology: Structural Evolution

As mentioned above the changes in this categorization are atomic. Evolution bases on such changes. Following terms are used in this paper to refer to different forms of evolution:

Definition 7 (Evolution step) An *evolution step* leads from one version of an MDE setting to a next version. Within an *evolution step* multiple atomic or *complex changes* might occur simultaneously.

Definition 8 (Evolution history) The *evolution history* of an MDE setting, can span multiple *evolution steps*

Definition 9 (Structural evolution step) An *evolution step*, where the set of changes contains at least one *structural change*.

Definition 10 (Substantial structural evolution step) *Evolution step*, where the set of changes contains at least one *substantial structural change*.

2.4 Limits and Decisions on Next Study Steps

The discussion in this Section showed that, judging the potential of impact, the kinds of changes differ with respect to the potentially affected *productivity dimensions*. However, this discussion has the limitation that it is very difficult to predict the extent of the impact of a change. For that reason we could discuss possible kinds of impacts, only. A more detailed discussion of the impacts on a theoretical level would be very difficult, since many different parameters would need to be taken into account, just to name some of them:

- Size, domain, and flexibility requirements of the software that is built with the MDE setting,

- Extent of the change, such as a degree of difference between two languages that are exchanged,
- Effort of changed, removed or added activities, or
- Share of changed artifacts on the system under construction.

To learn more about *structural changes* it is possible to go for empirical investigations. Would it be possible to design a study without restriction on resources and limitations of available data, a possible approach could be to search for correlations between occurrences of *structural changes* and improvements or deterioration of different *productivity dimension*.

However, existing data on evolution of MDE settings is very rare. Collecting new data is not only resource intensive, but also highly dependent on industrial cooperation. Thus, data sets with a high number of cases cannot be expected. A further obstacle is that such an approach requires a for each studied *productivity dimension* a method that allows to identify or measure impacts of an MDE setting on that *productivity dimension*.

Facing these challenges, we decided to approach the further investigation in two steps, as illustrated in Figure 2. In a first step, we perform a study to learn more about the frequency of structural evolution, i.e. about how many MDE projects have to face structural evolution. This study is independent from the need for methods that analyze the impact of an MDE setting on productivity. Nonetheless, it can provide us with the information what forms of structural evolution are not seldom. While this study step cannot prove the existence of impacts on the different *productivity dimensions*, it can, taken together with the theoretical discussion, show up existing potentials for impacts. This first empirical study is presented in Section 3.

In a second step, we perform a study to show the existence of impacts. Therefore, we decided to focus in the empirical part on one specific aspect of changeability. The reason is, that an analysis technique is available that allows to assess the impact of an MDE setting on this aspect. Although, this study is restricted to that changeability aspect, we are convinced that this step is important. It can show for the first time that impacts of structural evolution on at least one aspect of productivity play a role in practice. This second empirical study step in presented in Section 4.

3 Empirical Investigation I: Existence and Relevance of Structural Evolution

In context of the first investigation step (see Figure 2), we have discussed and categorized the possible changes on MDE settings above. However, there is currently little knowledge whether the *structural changes* actually occur in practice and if so, whether they are common. We approach this question in the second investigation step, which will be presented in this section.

Therefore we formulate the following hypotheses:

$H_{existence}$: *Structural evolution and substantial structural evolution occur in practice.*

H_{common}: *Structural evolution and substantial structural evolution occur for more than 25% of the MDE settings in practice (i.e. they are common).*

Note that we number this "common" with 25%, here. This is, because for it is important to have a percentage to test against for statistical consideration of frequency. Therefore, we choose the percentage, such that it reflects a situation, where companies in general have to take into account that the phenomenon affects parts of their MDE settings.

3.1 Data Collection and Analysis

To evaluate the hypotheses, data about evolution in practice is required. However, such data is rare. For being able to nonetheless retrieve justifiable and generalizable statements, we use the concept of triangulation as described in [34] and combine the data from two independent sources, each with its own advantages and disadvantages.

As the first data source, we use data records from a former exploratory and descriptive field study that we performed with the focus on capturing the structure of MDE settings in practice ([13]). The observed cases were not chosen with the topic of evolution in mind, which reduces the selection bias. However, the disadvantage of this first data set is that all case studies stem from a single company and that all data was collected by our team only. As the second data source, we use reports about MDE in practice that can be found in literature ([29], [10], [36], [20], [3], [17]). Although a selection bias cannot be excluded for literature studies, the advantage of this data source is that it provides us with a broader spectrum of companies and domains and that the reports are captured by different research teams. Thus, the second data source does not suffer from the problems of the first data source and vice versa.

In the remainder of this Section, we describe the methods of data collection for the SAP case studies (Section 3.1.1) and the method of identification and selection of literature reports (Section 3.1.2).

3.1.1 SAP Case Studies

We performed an exploratory and descriptive field study [4] in cooperation with SAP. The focus of the study was to learn about the characteristics of MDE in practice. The choice of the six captured case studies (as summarized in Table 2) was made by our contact persons within the company. We used semi-structured interviews. In contrast to questionnaires, interviews have the advantage that misunderstandings can be better identified and compensated [39]. This allowed us to combine the collection of complex MDE settings with more open questions about the motivations and reasons for the use of MDE techniques. The question covered following key topics (considering artifacts as models or source code):

- Used tools, modeling and programming languages
- Artifacts used and created during development
- Existing and created relations between artifacts
- Activities used to change, enrich, translate, generate, merge, compile, or interpret artifacts
- Degree of automation of individual activities
- Order of activities
- (Semi-) automated quality assurance activities
- Responsible roles for different activities

For each case study, we performed two telephone interviews, which lasted between 30 and 60 minutes each. The interviewees were developers that participated in the creation of tools for the MDE setting or used it. Between the initial and the final interview, we performed several rounds of feedback to ensure correctness of the captured data. More details about this field study can be found in [13]. As result we gained a descriptive model of each MDE setting as well as records from the more exploratory parts of the interviews. We systematically went through these records, searching for hints or more concrete information on evolution. Where possible, we assigned concrete change types to these hints or rated them as structural or non-structural.

3.1.2 Literature Reports

As a second data source we performed a small meta study. We systematically searched through

- The proceedings of the MODELS conference from 2007 to 2011 and ECMFA, respectively ECMDA-FA conferences, from 2007 to 2012,
- The proceedings of the Workshop on Models and Evolution ME, as well as its predecessors MCCM (Workshop on Model Co-Evolution and Consistency Management) and MoDSE (Workshop on Model-Driven Software Evolution) from 2007 to 2011,
- The proceedings of the OOPSLA Workshops on Domain-Specific Modeling from 2007 to 2011,
- As well as the Software and Systems Modeling journal (SoSyM) from 2007 to 2012, including papers published online first until end of July 2012.

In addition, we performed online key word search and followed references in reviewed papers. In particular we used the ACM digital library for keyword search in the proceedings of the ICSE conference. We searched for reports on the application of model-driven techniques or domain-specific modeling languages in practice. Note that we focused on non-purchase tool chains. We identified thirteen reports that describe MDE introduction or usage ([29,10,38,24,7,33,2,36,20,3] and three case studies in [17]). We filtered the reports to ensure that the captured period of time that is long enough to be able to observe evolution. Thus, reports that focus only on the initial introduction of MDE or on settings that were used for a single project only, were not suitable. Therefore, we excluded five reports ([38,24,7,33,2] as well as the telecom case study in [17], where the described example was only used during one project). Finally, we chose seven reports ([29] (CsTe), [10] (CsBA), [36] (CsFO), [20] (CsFBL), [3] (CsMo), as well as the case studies of the printer company (CsPC) and the car company (CsCC) [17]), which stem from different domains, such as the telecommunication industry, financial organizations, and development of control systems. Again, we systematically went through the reports and annotated hints or concrete information about evolution with change types where possible.

3.2 Data

Following, an overview about the changes identified for MDE settings from the SAP case studies and the literature reports is given.

3.2.1 SAP Case Studies

For the case study VC, no hints about evolution could be found in the records. In contrast, for the case studies SIW, Oberon, and BRF, hints on evolution were identified in the records. Such hints are often part of descriptions of the improvements reached by the introduction of the current setting. For example, one record from case study Oberon included the statement that the development functionality that is now provided by one tool was split between several tools before. From this record it can be concluded that the number of tools changed (*A6*).

For the case studies BO and BW, more precise information is available. On the one hand, the *evolution history* of the case study BO was covered in detail in

Table 2 Summary on captured SAP case studies

Case Study	Full Name	Number of modeled activities	Years in use
BO	Development of Business Objects for the feature package 2.0	19	> 2
BRF	Personalization of business processes using the tool business rule framework	23	> 3
BW	Definition and automated execution of reporting with the tool BW	19	> 14
Oberon	Development of a user interface and application based on business objects using the tool Oberon	25	> 3
SIW	Development of web services using the tool service implementation workbench	16	> 8
VC	Development of a user interface for SAP Net Weaver applications using the tool visual composer	30	> 5

context of the later investigation of *evolution histories*. On the other hand, the records for case study BW included a more detailed description of a former version of this MDE setting. Therefore, the difference to this former MDE setting is used to derive information about the evolution that happened. In these two cases, additional information about non-structural evolution could be captured.

The change types for which hints could be identified are summarized in Table 3. It is illustrated whether there is a more precise documentation of the changes or whether there are hints on a change type, only.

3.2.2 Literature Reports

In the following, the change types that can be identified for the different case studies from literature reports are presented and summarized in Table 3.

The report of the case study CsFO from Shirtz et al. [36] ends with a note that better integration of different tools (*A6*) and more automation of the construction phase (*A8*) are planned in future.

Karaila et al. describe in CsFBL ([20]) several changes that happened to the tool support for the language FBL. The tool vendor started with providing the visual programming language FBL together with an editor and a code generator. The user interface of the editor was later on extended. Over time they introduced the tool "function test" to enable developers to debug FBL (*A6*). The introduction of automated verification or debugging operations changes the order of manual and automated tasks. As a result manual programming is followed by automated debugging and further manual correction before the automated generation is applied (*A8*,*C1*). Further they report on the introduction of templates to allow programming on a higher level of abstraction. Developers have to choose and configure templates by specifying parameters. A chosen template with parameters is then automatically translated to FBL and from there code is generated. Thus, language (templates instead of FBL) and generation implementation (transformation plus generation) changed (*A1*,*A2*).

Fleurey et al. [10] present an approach to adapt a migration process towards the specific needs of the current application. A case study of the actual application of this migration process to migrate a banking application is also presented (CsBA). Interestingly, the changes that are actually applied in CsBA differ strongly from the proposed changes. For example, it was necessary that the resulting system conforms to the development standards of the customer. Thus, it was not sufficient to produce code, but to provide corresponding models that were synchronized with the code, such that round-trip engineering on the migrated system was possible. Therefore, they replaced the code generation with an automated UML extraction. They integrated the Rational Rose code generator used by the customer to generate code skeletons out of the models (*A6*). Further, they added a generation to migrate the remaining code from the platform-independent model (extracted from the original code) into the code skeletons (*A1*, *A2*, *A4*,*A5*). Conforming to the round trip engineering, some manual migration tasks have to be applied to the models (*A7*). The corresponding reapplication of the Rational Rose code generation adds an additional automated step to the MDE settings (*A8*). Thus, instead of being only followed by manual migration, the automated migration is followed by manual migration activities on the Rational Rose model, a generation of code and further manual migration activities on the code. The order of manual and automated tasks change, as manual migration is intermixed with automated code generation (*C1*).

In [29] a DSML for the generation of configuration files in telecommunication industry is presented (case study CsTe). It is also reported that the DSML was

Table 3 Identified change types in literature reports (○ = hints on changes; ● = documented changes)

	SAP Case Studies						Meta-Study						
	SIW [13]	Oberon [13]	BO [13]	BW [13]	BRF [13]	VC [13]	CsFO [36]	CsFBL [20]	CsBA [10]	CsTe [29]	CsMo [3]	CsPC [17]	CsCC [17]
Changes in General	○	○	●	●	○		○	●	●	●	●	○	●
Atomic Non-structural changes			●	●				●	●		●		●
A1 exchange automated activity			●	●				●	●	●			
A2 exchange language			●					●	●		●		●
A3 exchange tool		○											
Atomic structural changes	○	○	●	●	○		○	●	●	●	●		
A4 add or remove artifacts			●		○				●	●	●		
A5 add or remove languages			●		○				●	●	●		
A6 add or remove tools	○	○	●		○		○	●	●		●		
A7 add or remove manual activities			●	●	○				●	●			
A8 add or remove automated activities	○	○	●	●	○		○	●	●	●	●		
Complex Changes	○		●					●	●	●			
C1 change order of manual / automated activities	○		●					●	●	●			

changed later on. The verification language EVL to incrementally check the correctness of the models during development was integrated. This intermixes manual modeling activities with an added automated analysis for correctness (*A8*,*C1*). Further, the generation of the configuration files was exchanged, by an implementation that is based on a composition system conforming to the approach reported in [19]. Motivation for this change was the wish to reach higher levels of abstraction. The number of input models and languages changed from one to a flexible number (*A4*,*A5*). Also the number of manual modeling activities changes for the developer, who has to create a number of different DSL models (*A7*).

In [3], Baker et al. described how the use of MDE within Motorola changed over time (case study CsMo). This includes reports about changing tools (*A6*) and a changing number of languages and used models (*A4*,*A5*) due to the introduction of Message Sequence Charts (MSC) and SDL. Further, it is reported about changes in MSC (*A2*) that enabled the introduction of automated generation of test cases (*A8*).

Although the report about the printer company in [17] (CsPC) includes hints that the MDE setting under study changed, the information is not sufficient to make assumption about the actual type of change. Similarly, the report about the car company in [17] (CsCC) includes not much detailed information about the actual change. At least this report explicitly includes the information that the used modeling language changed (*A2*).

3.3 Hypotheses Testing

As summarized in Table 3, all types of structural evolution that were identified in Section 2 actually occur in practice. This validates the first hypothesis $H_{existence}$.

The evaluation of the second hypothesis H_{common} requires a statistical test. In each of the two data sets 5 cases with structural evolution have been identified and in each of these cases also hints for substantial structural evolution have been found. As defined in hypothesis H_{common}, the phenomenon of structural evolution is considered to be common if it exists in more than 25% of all MDE settings.

The null hypothesis h_{0_all} is that the percentage of MDE settings that are subject to structural evolution in practice is below or equal to 40%. Correspondingly, the alternative hypothesis h_{1_all} is that the percentage of MDE settings that are subject to structural evolution in practice exceeds 40%. The tested percentage of 40% substitutes the required 25% here, in the interest to approach the actual percentage. When h_{0_all} can be rejected this is an even stronger confirmation of H_{common}.

To enable a comparison, the probable percentage is also tested for the two single data sets. Since both data sets are very small, only the minimal percentage of 25% is tested. The corresponding null hypotheses are as follows: $h_{0\_smallSet}$ is that the percentage of MDE settings that are subject to structural evolution in practice is below or equal to 25%. The corresponding alternative hypothesis $h_{1\_smallSet}$ is that the percentage of MDE settings that are subject to structural evolution in practice exceeds 25%. The hypothesis $h_{0\_smallSet}$ will be tested separately for both data sets.

Binomial Test: For the selection of an appropriate test, we had to consider that our sample size is quite small and that we have no assumptions on the general distribution of the data. For such situations non-parametric statistical tests can be used. The data at hand is categorical data (i.e. structural evolution and non-structural evolution) of a single sample. The non-parametric test that can be used to test whether the probability of one of the categories is smaller than a given percentage (e.g. 25%) is a one sided binomial test, which will be used in the following. The two studied categories are MDE settings that underlie structural evolution vs. MDE settings that do not. For the test a significance level of 5% is chosen.

Further, we retrieve the power for the three binomial tests, i.e. the probability that the null hypothesis is rejected, while the alternative hypothesis is correct.

Therefore, we first we calculated the minimum number of occurrences required for a rejection of the null hypothesis. This is the smallest number for which the cumulative probability that this or a smaller number of occurrences can be observed under the tested probability is greater or equal to $0,95$ ($1-$ the significance level). The minimum number m of occurrences required for acceptance is $m = 4$ for sample size $n = 6$, $m = 5$ for sample size $n = 7$, and $m = 9$ for sample size $n = 13$.

Based on that m we can retrieve the power of the binomial test for all possible actual probabilities for the occurrence of *structural changes*, as summarized in Figure 5. It can be seen that, due to the small sample sizes, the power of the tests is not very good. For example, a power of 0.8 can only be reached, when the actual probability for the occurrence of *structural changes* is around 80% to 90%. Due to that the tests are running with the risk that the null hypothesis is accepted while the alternative hypothesis is true.

Results: In Table 4 the results of the test are summarized. The hypothesis h_{0_all} can be rejected. The probability that 10 MDE settings that underlie structural evolution are identified for a sample size of 13, while the percentage of such MDE settings is below or equal to 40%, is less than 5% (p-value = 0.007793). Thus, h_{0_all} is not plausible.

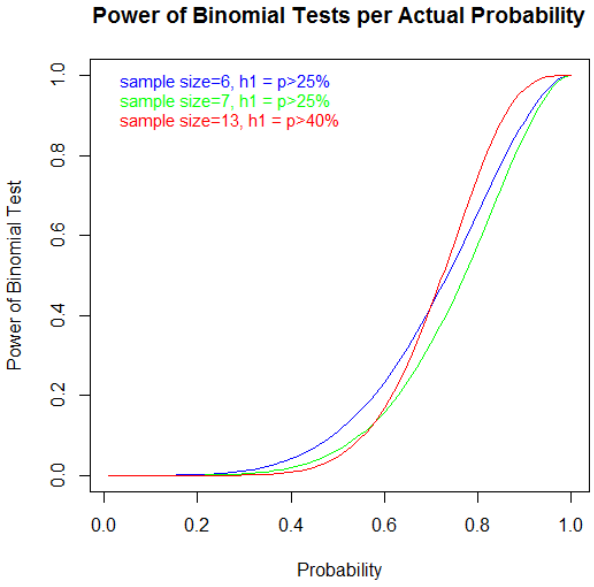

Fig. 5 Power of the three binomial tests per actual probability

For each of the two single case studies the hypothesis $h_{0\_smallSet}$ can be rejected, too. Here, the probability that 5 MDE setting that underlie structural evolution are identified for a sample size of 6 or 7, respectively, while the percentage of such MDE settings is below or equal to 25%, is in both cases less than 5% (p-values = 0.004639 and 0.01288). Thus, $h_{0\_smallSet}$ is not plausible. This adds to the validity, since it shows that the commonness of structural evolution is not specific for one of the data sets.

Furthermore, despite the weak power of the three tests, the null hypothesis was in all three cases rejected. Although, there was a high risk that the null hypothesis is accepted while the alternative hypothesis is true, it was possible to reject the null hypothesis. In each case the found number of occurrences was higher or equal to the respective minimum number for acceptance of the alternative hypothesis m.

Since in all cases, where we found structural evolution, we also found *substantial structural changes*, the results of examination of both data sources supports the hypothesis that structural and substantial structural evolution occur for more than 25% of the MDE settings in practice (H_{common}).

Table 4 Results of binomial test to check whether probabilities for *structural changes* exceed 40%

Date	# occurrences	sample size	h_0	h_1	p-value	95% confidence interval	Result
SAP	5	6	$p \leq 25\%$	$p > 25\%$	0.004639	0.4182 - 1	$h_{0\_smallSet}$ is rejected
Meta study	5	7	$p \leq 25\%$	$p > 25\%$	0.01288	0.3413 - 1	$h_{0\_smallSet}$ is rejected
All	10	13	$p \leq 40\%$	$p > 40\%$	0.007793	0.5053 - 1	h_{0_all} is rejected

3.4 Threats to Validity

In this section we discuss validity of the results following the categorization of threats to validity that was introduced by Wohlin et al. [39]. To address the observational/non-experimental character of the study, threats to validity are discussed under the following viewpoints: Construction validity: *"Are data and effects captured appropriately?"*, External validity: *"To what extent can the results be generalized?"*, and Conclusion validity: *"Are the conclusions that are drawn correct?"*. The fourth category that is introduced by Wohlin et al. (internal validity) is specific to experiments and therefore not relevant for the studies presented in this paper.

Construction Validity Due to the character of both data sources, the information about change types is most probably incomplete, which might have led to an under-representation of structural evolution within the results. Fortunately, such an under-representation cannot lead to a wrong acceptance of the hypotheses $H_{existence}$ and H_{common}.

External Validity Two biases that might influence the observed frequency of structural evolution are the selection bias and the bias that arises by the corporate culture or domain. While a selection bias is not probably for the the six SAP case studies, the fact all of them stem from the same company might lead to a bias, e.g. due to the corporate culture or domain. In contrast, the 7 case studies from the meta study stem from different countries, companies, and domains, which allows to draw conclusions that are not specific to a single domain. However, although a systematic selection method was applied, a small selection bias can never be excluded for meta studies. Thus, the biases that might affect both data sets are different. In consequence, since the observed frequencies of structural evolution are comparable for both data sets, it can be concluded that the observed frequencies are not caused by selection bias or biases due to corporate culture.

Conclusion Validity Although larger scaled empirical studies may help to get more accurate information in future, the data used here is sufficient to determine whether *structural changes* are sufficiently common in practice to be a relevant object for further research.

3.5 Summary

In context of our overall investigation, which was illustrated in Figure 2, this second investigation step reveals that structural evolution has not only theoretical relevance concerning possible impacts, but also practical relevance concerning its commonness.

4 Empirical Investigation II: Impact on Changeability

So far, it was theoretically discussed that structural and substantial structural evolution can improve or worsen an MDE setting's support for changeability and other *productivity dimensions* (see Section 2). Further, it was shown in Section 3 that structural and substantial structural evolution actually occur in practice.

However, while both results form a strong hint that the effect on *productivity dimensions* might really change for MDE settings in practice due to evolution, actual examples for such changes are missing. To address this question, we present in this section the third investigation step as illustrated in Figure 2. Therefore, we formulate the following hypothesis:

H_{impact}: *In practice there are cases, where structural evolution already led to changes to an MDE setting's impact of changeability.*

However, to prove this hypothesis it is necessary to identify a set of actual historical examples on *evolution steps* that caused measurable changes in an MDE setting's influence on changeability aspects. The identification of such examples is the goal of this third investigation step.

In order to gain data about occurred *evolution steps*, we again perform a descriptive and exploratory field study[3] to also capture information about the *evolution*

[3] Project's home pages: `http://www.hpi.uni-potsdam.de/giese/projects/mde_in_practice.html?L=1`

history of MDE settings. The examination that will be presented here is to answer the question whether the collected data includes hints on or actual evidence for such *evolution steps*. As mentioned in Section 2.4, techniques to measure the impact of an MDE setting on *productivity dimensions* are rare. However, there is a technique to identify risks for changeability that are implied by an MDE setting [11] (for details see below in Section 4.3.1). This technique enables a targeted search for changeability risks that had been introduced by structural evolution. A success in this search would lead to the identification of examples of *structural evolution steps* that prove the hypothesis H_{impact}.

In the following, the applied methods for data collection and investigation are presented. Further, the collected case studies and *evolution steps* are introduced. Then it is discussed to what extent occurrences of changeability risks can be ascribed to specific *evolution steps*. Finally, the results are summarized and the validity of the conclusions is discussed.

4.1 Data Collection Method

The first challenge was to get access to case studies. Most companies are very cautious about providing detailed data to external researchers (as is necessary for case studies). This problem was approached in this study by using personal contacts to different companies. In addition we contacted alumni students of the Hasso Plattner Institute. The request was accompanied by a short description of the project and the data to be captured. Eventually the contacted persons passed the request on to colleagues. When the persons in charge of such a project were interested in the study they answered the request.

All in all, this led to responses from five projects (from four companies). Unfortunately, in two companies the management did not agree to the participation due to confidentiality reasons. However, three companies agreed to participate in the study: Capgemini (3 MDE settings), VCat, and Carmeq. In addition, one of the contact persons of the former study at SAP agreed to resume the participation in the study and helped to document the *evolution history* for one of the captured MDE settings.

We took a conscious decision to go on with research of a few detailed case studies. This form of research has some important advantages when it comes to understanding complex phenomena and their drivers. Focusing on case studies makes it possible to gain coherent pictures of complex MDE settings. This includes the different ways in which artifacts are used or reused in automated and manual activities, the order of activities, and the tools used for supporting the activities. This is even more important since MDE settings had not been captured explicitly within the cooperating companies before.

We used semi-structured telephone interviews, since interviews have the advantage that misunderstandings can be resolved directly [39], which is beneficial when capturing complex phenomena. The structure of the interviews provided the basis to capture the MDE settings systematically. The semi-structured character of the interviews also supports an exploratory mode within the study.

To address the new issues we changed the method of eliciting the case studies, by substituting the rounds of feedback that where performed per email (as described in Section 3.1) with a third interview. In addition, we included questions on how the MDE settings evolved over time and asked for motivations and responsibilities for the captured changes.

The collected information about the *evolution history* was analyzed. The *evolution steps* were further categorized according to the change types presented in Section 2.

4.2 Case Studies

As summarized in Table 5[4], all captured MDE settings were in use for between two and eight years. Each captured model includes between 10 and 30 activities. For each of the different cases studies between one and seven *evolution steps* were captured (33 *evolution steps* overall, as summarized in Tables 6 and 7). This section further summarizes details on the stories behind the *evolution histories* of the 6 captured case studies.

Development of Business Objects for the Feature Package 2.0 (BO) The MDE setting of the case study BO was already captured during the SAP study. For this object of study an old historic version was captured in detail. The case study BO was already subject to seven *substantial structural evolution steps* in a period of around six years. The MDE setting started as almost classical code centric approach with some activities to ensure tracing between code and data model. Later on, a modeling tool was introduced followed by further tools that supported partial generation of the code and eventually, one of these generation tools was adopted as standard (*evolution step S1*). To improve

[4] Due to a request, names of artifacts and activities from the Capgemini case studies are substituted within this paper, in order to ensure confidentiality of the actually investigated projects.

Table 5 Summary on captured case studies for third investigation step

Case Study	Company	Full Name	Number of modeled activities	Years in use	Number of captured evolution steps
BO	SAP	Development of Business Objects for the feature package 2.0	19	> 2	7
VCat	VCat Consulting GmbH	Development of TYPO3 based websites	10	7	2
Carmeq	Carmeq GmbH	Development of AUTOSAR standard documents at Carmeq	25	8	5
Cap1	Capgemini	Capgemini case study 1	16	4	7
Cap2a	Capgemini	Capgemini case study 2	18	3	5
Cap2b	Capgemini	Capgemini case study 3	27	5	6

quality of behavioral implementation an interpretable modeling language was introduced in addition to the code (*evolution step S2*). The introduction of an additional modeling tool was motivated by the aim to introduce further quality assurance (*evolution step S3*). A next improvement was reached by introducing a semi-automated support for data migration between the modeling tools (*evolution step S4*). Here, a manual data migration activity that had to be executed between two automated activities, was substituted by a semi-automated support, i.e. an automated activity followed by a manual activity. The two automation activities before and after the initial manual activity are still preceded or followed by a manual activity, respectively. Consequently, the order of manual and automated activities did not change. In order to reduce the number of tools the three modeling tools were integrated into a single new modeling tool (*evolution step S5*). A variant of the MDE setting was created to enable simple use at the cost of a reduced set of supported features (*evolution step S6*). Finally, the generation functionality was moved to this new modeling tool (*evolution step S7*). Details on this evolution and the underlying decisions can be found in [14].

Development of TYPO3 based Websites (VCat) The fourth case study was collected in cooperation with VCat Consulting GmbH[5]. The documented MDE setting supports development of websites that rely on the underlying content management systems TYPO3. For the case study VCat two *substantial structural evolution steps* from a period of around seven years are captured. The first version of the MDE setting included a manual task to copy and clean TYPO3 instances from old projects, such that configurations could be reused. To improve this process and decrease the probability that content from old projects is preserved in the copied TYPO3 instance without notice, an automated instantiation and configuration of new TYPO3 instances was introduced (*evolution step S1*). Further the use of open source TYPO3 extensions should be improved in future. For several reasons, extensions need to be adapted before they are used. To support reuse of these adaption among the different projects at VCat, there are concrete plans to introduce an internal extension repository within VCat (*evolution step S2*).

Development of Standard Documents for AUTOSAR (Carmeq) The fifth case study was captured in cooperation with Carmeq GmbH[6]. The captured MDE setting is used to create documents of the AUTOSAR standard[7], including models and tables. For the case study Carmeq five *structural evolution steps* from a period of around eight years are captured. Three of the five *evolution steps* are *substantial structural evolution steps*. Initially, the AUTOSAR documentation was created without explicit modeling. To deal with inconsistencies between documents, a central model of the standardized software as well as an UML profile for AUTOSAR were introduced (*evolution step S1*). An automated generation of figures and tables on the basis of the central model was introduced. Later on macros were implemented to support the integration of figures and tables into the standard documents (*evolution step S2*). To support quality assurance for the generated figures and tables the modelers started to use diff-tools for comparison between old and new versions of the artifacts (*evolution step S3*). Further, a CI server was introduced, such that the generator is executed centrally (*evolu-*

[5] `http://www.vcat.de/` (last access at April 1st, 2014)

[6] `http://www.carmeq.de/` (last access at April 1st, 2014)

[7] `http://www.autosar.org/` (last access at April 1st, 2014)

Table 6 Identified *structural changes* in *evolution steps* of the case studies BO, VCat, and Carmeq (• = documented change; * = the MDE setting resulting from this *evolution step* was captured completely)

	SAP (BO)							VCat		Carmeq					Cap1						
	S1	S2	S3	S4*	S5	S6	S7	S1*	S2	S1	S2	S3	S4	S5*	S1	S2	S3	S4	S5	S6	S7*
Atomic Structural Changes																					
A4 add or remove artifacts	•	•			•	•	•	•	•	•		•				•	•		•	•	
A5 add or remove languages	•	•	•		•	•	•			•		•				•	•		•		
A6 add or remove tools	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•		•	•	•
A7 add or remove manual activities	•	•	•	•	•	•	•	•			•							•	•		
A8 add or remove automated activities	•	•	•	•	•	•	•	•	•	•	•	•			•	•	•		•	•	•
Complex Changes																					
C1 change order of manual / automated activities	•		•		•	•		•		•	•	•			•	•	•	•	•	•	•

tion step S4). Finally, an alternative implementation for some parts of the generator (e.g. the automated import between two of the modeling tools) was introduced (*evolution step S5*).

Capgemini First Case Study (Cap1) The case study (Cap1) was captured in cooperation with Capgemini[8] and is used in a project that has run for four years. In this project Capgemini builds software for a customer. There are two interacting MDE settings involved. The first MDE setting is used by the customer to collect requirements and create or prepare parts of the specification. The second MDE setting is applied within Capgemini to create prototypes, generate the specification, and to implement the software. This second MDE setting and its history were captured within this study. For the case study Cap1 seven *substantial structural evolution steps* from a period of around four years are captured. The project started with a standard code generator, which was soon substituted by a project specific generator. In addition, a semi-automated support for the export of the data between two modeling tools was added (*evolution step S1*). To improve merge of the old version of the model and the version that is result of the semi-automated export, a first version of a diff-tool was introduced (*evolution step S2*). Due to changes the MDE setting of the customer the support for the export as well as the diff-tool were taken out of operation (*evolution step S3*). Further the automation of the export between the main modeling tool and the code generator was improved (*evolution step S4*). Later on automated support for the implementation of a user interface based on mock-ups was introduced (*evolution step S5*). A change in the development process led to a situation that modified versions of the model are created by different teams and need to be merged. To support this merge a second version of the diff-tool was reintroduced (*evolution step S6*). Finally, to address a new need on additional documentation an additional generator was introduced (*evolution step S7*).

Capgemini Second and Third Case Studies (Cap2a and Cap2b) Also the second and third case studies (Cap2a and Cap2b) were captured in cooperation with Capgemini. The two MDE settings are parts of the same project. This project aims at providing the MDE settings that are used by a customer of Capgemini. Both settings can be applied in the same customer projects. They aim at reaching similar goals for different use cases.

For the case study Cap2a five *evolution steps* from a period of around three years are captured. Three of the five *evolution steps* are *structural evolution steps* (and two of them are *substantial structural evolution steps*). The project started with a first generator that was substituted later on by a more flexible version (*evolution step S1*). This substitution was planed from the beginning and was motivated by the need to rapidly provide a working MDE setting to the customer. The underlying meta model was permanently changed over the time. To create new output artifacts the generator implementa-

[8] `http://www.capgemini.com/` (last access at April 1st, 2014)

Table 7 Identified *structural changes* in *evolution steps* of the case studies Cap2a and Cap2b (• = documented change; * = the MDE setting resulting from this *evolution step* was captured completely)

	Cap2a					Cap2b					
	S1	**S2**	**S3**	**S4**	**S5***	**S1**	**S2**	**S3**	**S4**	**S5**	**S6***
Atomic Non-Structural Changes											
A1 exchange automated activity	•	•	•	•						•	
A2 exchange language										•	
A3 exchange tool											
Atomic Structural Changes											
A4 add or remove artifacts		•			•	•	•	•	•		•
A5 add or remove languages	•	•			•	•	•	•	•		•
A6 add or remove tools	•					•					
A7 add or remove manual activities	•					•					•
A8 add or remove automated activities					•		•	•	•		•
Complex Changes											
C1 change order of manual / automated activities					•			•			

tion was extended (*evolution step S2*). Later on a part of the generator was reimplemented, such that independence of the formerly used implementation technology is reached (*evolution step S3*). Finally, checks have been optimized over time, such that they can be applied automatically and regular (*evolution steps S4* and *S5*).

For the case study Cap2b six *evolution steps* from a period of around five years are captured. Five of the six *evolution steps* are *substantial structural evolution steps*. The first version of the MDE setting included a generator for the creation of the documentation. To improve the usability, the generator was integrated to a modeling tool and adapted the meta model that was already used in case study Cap2b (*evolution step S1*). In order to support creation of additional resulting documents three further generation activities were embedded into the existing generator over the time (*evolution steps S2*, *S4*, and *S6*). Addressing quality assurance, basic consistency checks for the models were introduced (*evolution step S3*) and the introduction of further consistency checks (following the example of case study Cap2b) is planned. Finally, the output format had to be changed at least one time (*evolution step S5*).

4.3 Investigation Method

In the following the method applied for this investigation is described.

4.3.1 Analysis Technique for Changeability Risks

As mentioned above, the subject of this investigation is the identification of risks for changeability for the software that is built with an MDE setting. Based on the Software Manufacture Model language for the representation of MDE settings [15], we already introduced in [11] a technique to identify changeability risks in MDE settings. There, two anti-patterns[9][10] are presented that allow to capture and describe structures of activities in Software Manufacture Models that can be associated to changeability risk. These two anti-patterns are *subsequent adjustment* and *creation dependence*.

Subsequent Adjustment: Here the first activity in this anti-pattern is an automated activity that generates an artifact (e.g. model or code) without taking former versions of this artifact into account. The second activity is a manual activity that changes parts of the generated artifact. When two activities that match to the activities in *subsequent adjustment* are re-executed to perform a change to the software, this can cause a loss of the manually created content in the generated artifact, since the generation, does not consider the former artifact version and simply over-generates the old artifact. For example, as it was illustrated in

[9] Here the term anti-pattern is used to refer to the facts that the patterns are "anti-pattern", i.e. pattern that describe negative effects [1,5]

[10] In [15] the anti-patterns are introduced as proto-pattern, i.e. patterns for which the number of documented occurrences in practice is not yet sufficient to refer to them as pattern[1]. For simplicity we refer to them as anti-patterns in this paper.

[15], the activities "Generate Java Code" and "Manipulate Implementation" from the example shown in Figure 1 match to this anti-pattern, when developers do not respect protected regions. Thus, a new execution of "Generate Java Code" due to changes in the generator model can lead to loss of manually implemented parts in the code.

Creation Dependence: The only activity in this pattern creates an artifact and in the same moment couples the artifacts via a references to parts of another "referenced" artifact. Former versions of the created artifact are not considered during the execution of this activity. In the MDE setting does not include alternative activities to manipulate the created artifact, an activity that matches to anti-pattern *creation dependence* can work as an amplifier that increases existing negative effects on changeability. The reason is that the activity couples an additional trigger to the complete recreation of the artifact: a change in the "referenced" artifact. In consequence, existing problems with changeability of the created artifact (or artifacts that are build based on the created artifact) are amplified. For example, activity "Create EMF Generator Model" from the example shown in Figure 1 matches to *creation dependence*. Here the activity needs to be newly executed when the input artifact "Ecore Model" changes in order to reestablish the reference to this input artifact from the created "EMF Generator Model". However, this later artifact might have been subject to manual changes (note that this is not illustrated in Figure 1), which get lost with the recreation of the artifact.

As mentioned the patterns describe risks, since external factors can prevent the manifestation of the risk as an actual problem. These factors are called *mitigating factors*. For example, within a software development project not necessarily all artifacts will become subject to change. If there is no need to change artifacts that are part of an occurrence of the *subsequent adjustment* pattern, the associated changeability problem will not manifest. Similarly, an occurrence of *creation dependence* only amplifies existing risks for changeability. Thus, it is possible that there is no risk that can be amplified.

Note that for the investigation of measurable changes in the characteristics of an MDE setting no technique for complete assessment of the changeability support of an MDE setting is required. In contrast it is sufficient to investigate whether local risks for changeability change. With the help of the anti-pattern, it is possible to identify pairs of activities (in case of the *subsequent adjustment*) or single activities (in case of *creation dependence*) that cause the corresponding risks within an MDE setting. In the following, we use this localization of changeability risks for the investigation of evolution effects.

4.3.2 Approach

Due to the character of the captured data, we have data on the *evolution steps* for the captured MDE settings, but only respectively one (historical) version of each MDE setting is completely captured, i.e. only for one historical version per MDE setting a Software Manufacture Model, which is the basis for the identification of the anti-patterns, is available. The *evolution steps* are captured in form of change descriptions, and allow no detailed reconstruction of the actual activity details. As a consequence it is not possible to directly identify the *evolution steps* that affected the occurrence of the anti-pattern by analyzing all historical versions of an MDE setting and comparing the located patterns. For example, it might be interesting, whether located patterns form older versions of the MDE setting disappear in younger versions or whether a pattern occurrence can only be located in the youngest version of an MDE setting.

Instead another approach is followed for this analysis. First, we analyzed the available Software Manufacture Models to identify occurrences of the two anti-pattern. In a second step, we reconsidered the *evolution steps* in reverse order, starting at the version of the MDE setting that was analyzed. Thereby, we rated for each *evolution step* whether this *evolution step* introduced an activity that is part of an anti-pattern occurrence. This way it is possible to identify *evolution steps* that introduced such a anti-pattern occurrence and with it the corresponding changeability risk. In addition, we searched through the records from the interviews, to identify for each of the occurrences whether the risk manifested in practice or whether we can identify mitigating factors.

In the most cases the currently used version of the MDE setting is the one which was captured completely. An exception is the case study BO where an older version of the MDE setting was captured in detail, which is the version that resulted from the fourth *evolution step* (S4) and was replaced with the fifth *evolution step* (S5). The Software Manufacture Model captured for VCat represents the currently used version. However, this version is the result of the first *evolution step* (S1), while the second captured *evolution step* will be applied in future to replace this current setting.

4.3.3 Limitation

Due to the above described constraint on the applicable approach, a limitation is that it is only possible to

consider changeability issues in the version of an MDE setting that is captured in detail. Thus, this approach cannot be used for the identification of changeability problems in former or later historical versions of the MDE setting. Due to this, there cannot be a systematic search for examples for *evolution steps* that did remove changeability problems. As a result, the described analysis approach allows to identify *evolution steps* that led to the introduction of changeability risks that have not been remove by later *evolution steps* so far, only.

4.4 Effects of Evolution on Changeability

In the following it is examined whether a direct influence of evolution on occurrences of the two anti-pattern *subsequent adjustment* and *creation dependence* can be observed within the six available case studies. Within the Software Manufacture Models that are captured for the six case studies, 6 occurrences of anti-pattern *creation dependence* could be identified. As summarized in Table 8, for 5 of these 6 occurrences, it is possible to identify *evolution steps* that introduced the matched activities into the MDE setting. For example, an activity that is matches to anti-pattern *creation dependence* was most probably introduced in the third *evolution step* of the SAP business object case study (BO S3), where a couple of validation activities were added in context of a technological change. Only one of the occurrences of *creation dependence* was most probably part of the corresponding MDE setting from the start (see Table 8, case study Cap1).

8 occurrences of anti-pattern *subsequent adjustment* have been detected for the six case studies. For 5 of these 8 occurrences *evolution steps* that introduced these occurrences can be identified. For example, the the automated code generation activity that is part of an occurrence of anti-pattern *subsequent adjustment* was already introduced to the SAP business object case study in the first *evolution step* (BO S1). However, the manual activity that is part of the same occurrence of *subsequent adjustment* was only introduced in the second *evolution step* (BO S2). With that second *evolution step* the occurrence of the anti-pattern was introduced to the MDE setting. Three of the occurrences of *subsequent adjustment* were most probably part of the corresponding MDE setting from the start.

Further, the fact that the detailed version of the MDE setting of the SAP business object case stud yBO is *not* the currently used version, allows the observation that occurrences of a anti-pattern can also be removed by *evolution steps*. One match of anti-pattern *subsequent adjustment*, which was introduced with *evolution step* S2, was removed later on in the *evolution step* S5, where the MDE setting was refactored by better integrating multiple tools.

Theoretically, the introduction of anti-pattern *subsequent adjustment* can be caused by different changes. In all five examples where an introducing *evolution step* could be identified, this *evolution step* is a *substantial structural evolution step*. The match of *subsequent adjustment* was two times introduced by the addition of an automated activity that matched to *initial creation*. Once an automated activity was exchanged by another automated activity and a following manual activity (in order to produce the result in another format). In a fourth case both matched activities were introduced together. Finally, one match was introduced by the introduction of the manual activity, only.

For three of the five occurrences of *subsequent adjustment* there were introduced by *evolution steps* the associated loss of content during changes actually occurred. For the other two introduced occurrences of *subsequent adjustment* the risks did not manifest to problems at the time of the interviews, since there is usually no need to change the involved artifacts.

Considering all 26 captured *substantial structural evolution steps* in these 6 case studies, 7 (more than one quarter) could be ascribed with the actual introduction of anti-patterns.

4.5 Observed Evolution Effects

Based on the six case studies, it is not possible to identify a correlation between the actual number of structural or *substantial structural evolution steps* that occurred to an MDE setting and the number of matches of the anti-patterns. However, for 10 of the 14 identified anti-pattern occurrences, we could identify the *evolution steps* that introduced these occurrences. For three of the 10 introduced anti-pattern occurrences the associated risk already manifested to a problem. With these results from practice we can confirm the hypothesis H_{impact} that *in practice there are cases where structural evolution already led to changes to an MDE setting's impact of changeability.*

4.6 Threats to Validity

Again the three categories, construction validity, external validity, and internal validity (Wohlin et al. [39]), are discussed.

Construction Validity Due to the complexity of the captured information it cannot be excluded that there are small faults in the created models of the MDE settings,

Table 8 Summary of identified *evolution steps* that introduced occurrences of anti-patterns within MDE settings of the case studies. Mitigating factors are describes as follows: **no change happens** = during development usually no changes occur that trigger the new execution of the involved activities; **no problem to amplify** = there is no changeability problem in the MDE setting that is subject to the amplification.

Case Study	Pattern Occurrence	Identified Indications that the Risks Manifest in Projects	Introducing Evolution Step	Removing Evolution Step
BO	*subsequent adjustment* occurrence 1	✓	S1	
	subsequent adjustment occurrence 2	✓	S2	S5
	subsequent adjustment occurrence 3	*mitigating factor:* no change happens	S3	
	creation dependence occurrence 1	*mitigating factor:* no problem to amplify	S3	
VCat	*subsequent adjustment* occurrence 4	*mitigating factor:* no change happens	–	
	subsequent adjustment occurrence 5	*mitigating factor:* no change happens	–	
	subsequent adjustment occurrence 6	*mitigating factor:* no change happens	S1	
Cap1	*creation dependence* occurrence 2	✓ (amplification of *subsequent adjustment* occurrence 7)	–	
	subsequent adjustment occurrence 7	✓	–	
Cap2b	*creation dependence* occurrence 3	*mitigating factor:* no problem to amplify	S4	
	creation dependence occurrence 4	*mitigating factor:* no problem to amplify	S4	
	creation dependence occurrence 5	*mitigating factor:* no problem to amplify	S1	
	creation dependence occurrence 6	*mitigating factor:* no problem to amplify	S1	
Cap2a	*subsequent adjustment* occurrence 8	✓	S1	

which might impact the results of the applied changeability analysis. However, we included rounds of feedback in the study designed. This way the probability of faults can be reduced.

External Validity While it remains possible that the results cannot be generalized to all domains of software engineering, the use of case studies from different companies and domain allows to draw conclusions that are not specific to one domain only.

Conclusion Validity The number of pattern matches under study is with 6 and 8 relatively low. However, it was not the goal to make statements about frequencies, but about the existence of *evolution steps* with measurable effects, only. The captured data is sufficient to conclude that changes in changeability risks of MDE settings are caused by *evolution steps* in practice.

4.7 Summary

In this third investigation step (see Figure 2), we were able to show the existence of cases, where risks for changeability were actually introduced by substantial structural evolution in practice. Further, we could identify one example for a *substantial structural evolution step* that removed a changeability risk.

5 Discussion

When we put the three presented investigations together, as shown in Figure 2, we gain a comprehensive picture of the nature of structural evolution and its impacts on changeability. The theoretical consideration showed that substantial structural evolution has the potential to introduce (or remove) changeability risks. The second investigation showed that structural evolution and substantial structural evolution is common in practice. Finally, we could show in a third step that there is more than just a potential that the occurring *structural evolution steps* impact changeability, but that changeability risks and problems are actually introduced by substantial structural evolution in practice. Furthermore we could identify one example of an *evolution step* that removed a changeability problem.

In addition to the investigation of the impact on changeability, we could also make some additional observations on how changes are combined, on trade-offs that are made, when an MDE setting evolves, and on motivations and triggers of structural evolution.

5.1 Method

Therefore, we systematically inspected and coded the records from the interviews following the constant comparison method described in [34]. At the start, a set of preformed codes was used. These referred to the moti-

vation for an *evolution step*, the institution or role that triggered the *evolution step*, and the institution or role that implemented the *evolution step*. During the inspection of the records, codes were added when necessary (e.g. for external influences on the evolution). Based on these codes it was possible to derive observations.

In [14] eight of these observations had been shortly introduced. Due to the enhanced number of case studies, we were able to refine four of these observations, which will be presented in detail in the following.

5.2 Observations

Trade-Offs: Structural changes *are often trade-offs, e.g. with respect to costs and manageability.*

For example, implementing a smaller new generation step is easier to manage than applying a change to an existing automated activity. A further factor in such a trade-off is the weight that is given to the different *productivity dimensions*. An example is the *evolution step* S3 in case study BO. In favor of better consistency between conceptual models and business models, it was accepted that links between conceptual models and implementation are no longer maintained during development.

Part of such trade-offs are also decisions to a delay of MDE evolution. For example, the third *evolution step* in case study Cap2a included a complete new development of a generator on the basis of a new technology. The corresponding idea existed for a while and was only implemented when a number of other big changes to the generator became necessary. In many of the observed cases it was decided to increase the *degree of automation* or tool support by adding new automated activities instead of adapting existing automated activities like transformation steps. For example, in case study Carmeq additional *importers* were added, instead of adapting the existing *importer*.

Thus, a *substantial structural change* that has the potential to cause drawbacks for the changeability is accepted in favor of costs and manageability of the *structural evolution step*.

The factors involved in such trade-offs change over time. Costs that can be invested for an *evolution step* can change strongly. For example, the *evolution step S4* in case study Cap1 was implemented by a developer in his leisure time. The weight that is given to different *productivity dimensions* can also change. For example, *evolution step S1* in case study BO was mainly driven by the desire to increase the *degree of automation*. For a long time explicit conceptual modeling had a priority. Later on the priorities change, such that efficiency and total cost of ownership became more important. As a consequence, *evolution step S5* led to reduction of the number of tools and the number of inconsistencies at the cost of a loss of graphical modeling capabilities, the loss of functionality to simulate status models, as well as loss of the ability to model design alternatives.

Environment: *Changes in an MDE setting can be driven by the need to take other MDE settings into account.*

Sometimes structural evolution is caused by changes in the environment of the MDE setting. Changes in an MDE setting that is used in a cooperating project or company can lead to new opportunities for the integration of both MDE settings. For example, in Cap1 the customer's MDE settings was changed, such that it based on the same framework (Enterprise Architect) as the MDE setting of Capgemini, afterwards. As a consequence, the automation and support for merging and exporting could be removed in *evolution step* S3. A similar mechanism worked, when the MDE setting of the customer changed, such that mock-ups were modeled in the Enterprise Architect, too. As a consequence *evolution step* S5 was enabled, where a partial generation of user interfaces on the basis of the mock-up model was introduced. Another example can be found in case study Cap2b. Here, functionality to generate an HTML catalog was introduced in *evolution step* S2 of Cap2a. After that, an HTML generator was introduced in *evolution step* S4 of case study Cap2b. The resulting HTML catalog is an extension of the HTML catalog that can be generated with Cap2a for a same project. Thus, both MDE settings can be combined to create a common result.

Pragmatic Developer's Decision: *Some* evolution steps *are not planned centrally, but are caused by developers who add automation steps to ease their daily work.*

Examples, where developers evolved the MDE setting that they used themselves, can be found in the four of the case studies. In the first *evolution step* in BO similar automations of the same implementation aspects were introduced independently by developer teams that worked on different projects. Eventually, one of the automations was chosen as standard. In case study Cap1 four *evolution steps* (S2, S4, S6, and S7) are triggered this way. As aforementioned, the introduction of the tool for detecting model differences to automatically correct typical errors in an output format (S4) was even performed by a developer in his leisure time. Finally, in case study Carmeq the macros and diff-tools (S2 and S3) were introduced by developers and in case study

VCat the planned introduction of the new repository for reuse was triggered by the developers (and was supported by the management). Apart from evolution that is triggered and implemented directly by developers, they might also propose changes to the MDE setting that are then implemented by the tool vendor. For example, in Cap2b the use of HTML generation to gain a new possibility for navigating through the model was an idea of developers that use the MDE setting.

There are also introductions of automation steps that are not triggered by developers. For example, the first *evolution step* in case study VCat and the first and fifth *evolution step* in the case study Cap1 were requested by the management or customers, respectively. However, it seems that developers have relevant insights into potentials for further automation of development. More importantly, it seems that developers even have an own interest in automating parts of the development.

Manage the MDE Setting: *The motivation of some* evolution steps *is to reduce the complexity of MDE settings, which can be considered as "refactoring".*

An example of this is the introduction of the new repository (MDRS) in business object development (*evolution step S5* in case study BO). This refactoring was completed in *evolution step* S6, were also the status and action management was integrated into the MDRS.

In case study Cap1 a less obvious clean-up occurs. Here the *evolution steps* S3 and S5 succeed changes in the MDE setting of the customer, which can be rated as clean-up. In both cases development activities that are performed by the customer were moved to the development tool that is used at Capgemini. Consequently a better integration of both MDE settings was achieved.

5.3 Threats to Validity

As in Section 3.4 validity of this study part is discussed for the three categories construction validity, external validity, and conclusion validity introduced by Wohlin et al. [39].

Construction Validity A social threat is that interviewees might omit information due to the fear of being evaluated (*evaluation apprehension* [39]). This might concern change motivations that lay in disadvantages of or problems with the preceding MDE setting. While that cannot be ruled out in general, incompleteness can be accepted. Nonetheless, the collected data provide valuable insights into existing motivations for evolution.

External Validity As mentioned above a selection bias towards case studies with long *evolution histories* cannot be excluded. It is possible that motivations and triggers for evolution are different in projects where MDE settings often change compared to projects where MDE settings rarely change. Consequently, motivations might be different for MDE settings that change rarely.

Finally, all case studies stem from companies within the same country (Germany). Thus, it is possible that the observations, such as the observation on *pragmatic developer's decision*, cannot be generalized to countries with very different culture.

Conclusion Validity The data on the 32 *evolution steps* stems from four different companies (SAP, Capgemini, Carmeq, and VCat). To ensure that the made observations are not company specific, we only included them, if they based on examples of at least two companies.

6 Related work

This section focuses on related work on evolution in context of MDE.

Case studies on MDE settings: First, this includes on the one hand the papers that have been identified and discussed in the literature study ([29] (CsTe), [10] (CsBA), [36] (CsFO), [20] (CsFBL), [3] (CsMo), the case studies of the printer company (CsPC) and the car company (CsCC) [17], see Sections 3.1.2 and 3.2.2 for more details). These papers include reports on MDE usage in practice and, thereby, partly also documented hints on structural evolution. However, none of these papers targeted at the systematic documentation of evolution or even structural evolution of MDE settings.

Studies on evolution occurrence in practice: For specific forms of non-structural evolution, studies exist that describe their occurrence and relevance in practice. For example, Herrmannsdoerfer et. al studies forms of evolution that happen to modeling languages (metamodel evolution) [16]. Unfortunately, we are not aware of similar studies discussing occurrences of structural evolution in practice.

Categorizations and surveys on types of MDE evolution: There are several categorizations and surveys on evolution (summarized in Table 9):

There are different surveys and research agendas that classify types of changes in model-driven software evolution or in software evolution in general. For example, Stammel et al. provide an extensive survey on techniques and approaches in software engineering that enable flexibility and evolution of software [37]. The

spectrum of approaches reaching from agile methods via model-driven software development to architectural patterns is discussed on a high level of abstraction.

A collection of possibilities to support software evolution with models is presented by Karanam and Akepogu in [21]. However, within MDE approaches models are also part of the implementation of the software and need to be evolved when the software evolves. Khalil and Dingel present in [22] a survey on techniques that support the evolution of UML models. They consider model evolution as a subset of software evolution. Thus, the discussed approaches deal with evolution of the different models that are used for the specification of a software system. Discussed evolution tasks concern for example the change propagation between different models.

That evolution in MDE can be more than evolution of the software is summarized by van Deursen et al. in [8]. Here, challenges in supporting the different forms of evolution are discussed. Considered forms of evolution are "regular evolution", which is software evolution, "meta-model evolution", which is a form of language evolution (discussed as change type *A2*), "platform evolution", which means the evolution of code generators and tools (discussed as change types *A1* and *A3*), and "abstraction evolution", which means the introduction of an additional modeling language to the set of languages (which is one part of change type *A5*). "Abstraction evolution" as is introduced by van Deursen et al. is a special case of structural evolution and has potential side effects on other change types, which are not further described in [8]. For example, *A4* and *A7* can be affected, when the introduced additional language is used to express additional artifacts (and does not only substitutes the language that is used for an already existing artifact). Mens et al. supplement the challenges listed by van Deursen et al. with a stronger focus on the evolution of models [26].

Meyers et al. subdivide metamodel- and model evolution into four primitive scenarios, describing how evolution of models, metamodels, or transformations enforces co-evolution among each other [28,27]. The resulting scenarios are combinations of the change types *A1* and *A2*.

Finally, Corrêa et al. classified change types in context of to software product lines [6]. The categories include the change of the meta-model (summarized as language evolution *A2*), changes of features and changes of models, which are both forms of software evolution, and changes within a transformation (discussed in our classification as change type *A1*).

Subsuming, most categorizations have a central view on software evolution, which is as such not part of our categorization. Further, *non-structural changes* are considered in most categories, while only in [8] a very specific case of *structural changes* is considered.

Discussion of evolution impacts: Finally, we could not find related work that discussed the impacts of MDE evolution, especially structural MDE evolution, on productivity aspects.

Evolution in other contexts: Besides the context of MDE, evolution has a long history in data base research. Here the co-evolution of data during schema evolution was studied. As summarized by Roddick in 1992 in an annotated bibliography, much research was done on schema evolution [32]. Schema evolution can be compared to co-evolution of meta-models and models (or language evolution). For example, Lerner presents a framework for the automated creation of transformations for migrating data from one schema to a new one [23]. Noy and Klein discuss in [30] the difference between ontologies and database schemata in order to identify additional challenges for the evolution of ontologies and corresponding co-evolution of data. From the viewpoint of the types of MDE evolution that are discussed in this paper, research on schema evolution focuses on the *non-structural change* type *A2*. The schema can be seen as the language, while the data can be compared to instances of this language.

Summary: To sum up, related work on evolution mainly focuses on non-structural forms of evolution. In contrast, this paper introduced a classification of structural evolution and goes further by studying the the impacts of these changes on an MDE setting's influence on *productivity dimensions*, such as changeability.

7 Conclusion and Implications

In this paper, we investigated the nature of structural evolution of MDE settings and its impact on changeability. The presented investigation consisted of three steps (see Figure 2). First, we classified possible types of *atomic changes* in MDE settings and discussed the potential of these different changes to affect productivity-related aspects of an MDE setting. As a result we identified that structural and especially substantial structural evolution can have strong impacts and might even change how an MDE setting supports changeability.

As a second step, we used two different data sources, namely a set of 6 MDE settings capture at SAP and a set of literature reports on MDE, to study whether structural evolution is relevant in practice. The results support the hypothesis that structural evolution and substantial structural evolution occur in practice and occur for more than 25% of the MDE settings in practice (i.e. are common).

Table 9 Change types covered by existing categories in other categorizations of changes in context of MDE (● = covered by categorization; ● = only specific situations covered by a category; ○ = side effects of changes in other categories possible but not specified;)

*Kind of Changes**Categorization*	Stammel et al. [37]	Karanam et al. [21]	Khalil et al. [22]	Mens et al. [26]	Meyers et al. [28] [27]	Corrêa et al. [6]	van Deursen et al. [8]
Software Evolution	●	●	●	●	●	●	●
Atomic Non-Structural Changes							
A1 exchange automated activity					●	●	●
A2 exchange language					●	●	●
A3 exchange tool							●
Atomic Structural Changes							
A4 change number of artifacts							○
A5 change number of languages							●
A6 change number of tools							○
A7 change number of manual activities							○
A8 change number of automated activities							○
Complex Changes							
C1 change order of manual / automated activities							○

In a third step, we investigated whether there are actual examples of *evolution steps* that did affect changeability. Therefore, we collected 6 MDE settings and their *evolution histories* from 4 different companies. For the collect 32 *evolution steps*, we rated the applied change types. Subsequently, we used a pattern-based analysis technique to localize causes for changeability problems within an MDE setting. We applied this analysis technique to the 6 captured MDE settings and rated whether one of the *evolution steps* introduced the identified parts of the MDE setting. As result we identified 7 *evolution steps* that actually led to the introduction of anti-pattern matches and with it risks and partly manifested problems on changeability.

In addition, we present 4 observations on motivations and triggers behind *structural evolution steps* in detail. The results give a first insight on the mechanisms behind structural evolution.

To sum up, this paper shows for the first time, that structural evolution occurs in practice and that it has impact on the changeability of the evolving MDE settings.

7.1 Future Studies on Structural Evolution

To better understand the degree of generality of our results and to improve the accuracy of knowledge, some future studies might be highly of interest. In general holds that the main challenge for performing these future studies will be to gain the required data.

For example, the estimation of frequency so far bases on a very small data set and thus allows us only to investigate a lower limit for the probably frequency of structural evolution. Here future studies that base on bigger data sets can help to gain more precise information about this frequency.

Further, gaining additional data on structural evolution from other countries than Germany and from further domains can help in future work to learn more about the possibility to generalize the results of the first and second study step, as well as the made observation.

So far we focused on the evolution of "real" MDE settings, i.e. MDE settings that actually include modeling and/or generation activities. However, as the first *evolution step* of case study BO shows structural evolution might also affect non-MDE development settings. However, it seems that there are two tendencies that differentiate most MDE settings from most non-MDE settings: First, programming languages are in most cases backwards compatible, while this is does often not succeed for modeling languages. Second, MDE settings, as many of the examples in this paper, base often on tools that are build or adapted within the company where they are used, which is probably rather seldom for non-

MDE settings. These two tendencies might cause that structural evolution plays a different role in non-MDE settings than in MDE settings. Thus, while the theoretical definition of MDE setting is defined widely enough to cover these cases, it would be interesting to study whether structural evolution actually occurs in non-MDE programming environments and whether it has similar impacts to changeability.

An intuitive understanding of structural changes would be that they occur during the specific phase short after MDE introduction until the MDE setting reaches a stable state, only. However, it is difficult, and in context of this paper not possible, to gain enough data to really prove the opposite. Nonetheless, our data about the evolution histories of the second empirical study is not supporting that idea. In contrast, in some of our case studies structural evolution steps occurred during a phase of 5, 7, or even 8 years. It might be interesting to investigate in future studies whether most MDE settings do reach a stable state after which structural evolution can be excluded, or not.

To gain in future more knowledge about the impact of evolution on the two changeability anti-pattern *subsequent adjustment* and *creation dependence*, future studies can target on two improvements, compared to the study at hand. First, the a data set with a bigger size and no selection bias might be collected and used to learn about the percentage with which *structural evolution steps* lead to changes in the occurrence of changeability anti-pattern. Also a correlation analysis might this way become possible. Second, it might be interesting to collect data sets with complete (analyzable) documentation of each historical version of an MDE setting. This way it will become possible to learn more about the degree to which *structural evolution steps* lead to the improvement of changeability (i.e., to the removal of anti-pattern).

Finally, the empirical study of actual impacts, so far could only cover the impact of structural evolution on a specific aspect of changeability (the two so far discussed changeability risks). Future studies on the impact of structural evolution, similar to our second study step might be performed, when techniques become available that allow the analysis of the impact of an MDE setting for additional aspects of changeability and also other productivity aspects.

7.2 Consequences on Future Research and Practice

In the following we discussed consequences of the presented results on future research and practice:

7.2.1 Predicting Consequences of Evolution Decisions

Research often focuses on designing and introducing innovative and specialized MDE settings or single MDE techniques. These approaches are sometimes evaluated empirically by reporting on success or by experiments (e.g. as in [25]). However, when MDE settings are the result of structural evolution, results from empirical evaluations that have been performed on the initial MDE setting are no longer helpful to predict the quality or risks of the new MDE setting. There will be no time to perform a new empirical study on the new productive MDE setting. Consequently, more well-founded knowledge is required about the causality of an MDE setting's structure and the observable benefits. Based on such knowledge, techniques to statically analyze and predict benefits and different characteristics of an MDE setting could be developed. Such techniques can then be used by practitioners to balance trade-offs when planning the next *evolution step*.

Apart from such overall analysis techniques, support for evolution decisions might also be provided with respect to concrete *evolution steps*. Best practices of *evolution steps* might be collected and shared.

Finally, we need to learn more about the impact of MDE evolution, during running software projects. For example, is there a mutual impact between evolution of the MDE setting and evolution that happens to the software that is build with the MDE setting?

7.2.2 Shifting Trade-Offs

Trade-off plays an important role during evolution decisions. In combination with the fact that different forms of evolution lead to diverse changes in the characteristics of an MDE setting, this leads to the implication that trade-offs need to be better understood.

It is possible to reach the same goals with different MDE settings and thus also by different *evolution steps* on MDE settings. Therefore, it is necessary to learn when and why practitioners choose specific forms of evolution to reach a given goal. Researchers might provide techniques and tools that shift the involved trade-offs, such that less risky *evolution steps* become advantageous. For example, frameworks for combining and extending DSLs, like the one presented in [19] might be a first step in this direction.

In this context, the question arises about how much effort is required to apply an *evolution step* to an MDE setting. To estimate the costs associated with an *evolution step* itself, techniques to assess and evaluate the "evolvability" of MDE settings are required.

7.2.3 Refactoring

Beyond single *evolution steps*, there is the question of the extent to which a strategic planning of evolution can be used to reach long term goals. This includes the option and task to plan refactoring of MDE settings. During periods when requirements on the created software are stable and resources are available, such a refactoring can be used to correct negative side effects of former *evolution steps*. For example, a goal might be to improve the integration of different tools or to decrease complexity for the developers.

7.2.4 Change Management

Finally, the observations on structural evolution not only have implications for research, but also for practice. For example, the observations indicate that sometimes developers trigger and implement *evolution steps* on MDE settings on their own initiative. Considering the risks and potential that are associated with structural evolution, it might be a meaningful step to establish a management of change requests for MDE settings within a company. Such a change management would allow developers to contribute to the improvement of the MDE settings, while the risk of uncoordinated evolution of MDE settings can be reduced.

Acknowledgements First of all the authors appreciate the valuable feedback of Florian Stallmann and Andreas Seibel, who co-authored the conference paper [14]. We are also grateful to the participants of our studies with SAP, Carmeq, VCat and Capgemini and especially Axel Uhl, Cafer Tosun, Gregor Engels, and Marion Kremer for their support in choosing the case studies and for making this research possible. We are thankful to Hannelore Liero and Hilmar Buchholz for hints and feedback on the statistical analysis. Further we thank the HPI Research School on Service-Oriented Systems Engineering for funding parts of this research. Last but not least, we want to thank the anonymous reviewers of this article for their constructive feedback, which helped us to improve the paper.

References

1. Appleton, B.: Patterns and Software: Essential Concepts and Terminology. Object Magazine Online **3**(9) (1997). URL `citeseer.ist.psu.edu/247320.html`
2. Aschauer, T., Dauenhauer, G., Pree, W.: A modeling language's evolution driven by tight interaction between academia and industry. In: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 2, ICSE '10, pp. 49–58. ACM, New York, NY, USA (2010)
3. Baker, P., Loh, S., Weil, F.: Model-Driven Engineering in a Large Industrial Context – Motorola Case Study. In: L. Briand, C. Williams (eds.) Model Driven Engineering Languages and Systems, *LNCS*, vol. 3713, pp. 476–491. Springer (2005)
4. Basili, V.R.: The role of experimentation in software engineering: past, current, and future. In: Proceedings of the 18th international conference on Software engineering, ICSE '96. IEEE Computer Society, Washington, DC, USA (1996)
5. Brown, W.H., Malveau, R.C., Mowbray, T.J.: AntiPatterns: refactoring software, architectures, and projects in crisis. Wiley (1998)
6. Corrêa, C.K.F., Oliveira, T.C., Werner, C.M.L.: An analysis of change operations to achieve consistency in model-driven software product lines. In: Proceedings of the 15th International Software Product Line Conference, Volume 2, SPLC '11, pp. 24:1–24:4. ACM, New York, NY, USA (2011)
7. Deng, G., Lu, T., Turkay, E., Gokhale, A., Schmidt, D.C., Nechypurenko, A.: Model Driven Development of Inventory Tracking System. In: Proceedings of the ACM OOPSLA Workshop on Domain-Specific Modeling Languages (2003)
8. van Deursen, A., Visser, E., Warmer, J.: Model-Driven Software Evolution: A Research Agenda. In: D. Tamzalit (ed.) Proceedings 1st International Workshop on Model-Driven Software Evolution (MoDSE), pp. 41–49. University of Nantes (2007). URL `http://www.sciences.univ-nantes.fr/MoDSE2007/`
9. Favre, J.M.: Languages evolve too! Changing the Software Time Scale. In: Proceedings of the Eighth International Workshop on Principles of Software Evolution, IWPSE '05, pp. 33–44. IEEE Computer Society, Washington, DC, USA (2005)
10. Fleurey, F., Breton, E., Baudry, B., Nicolas, A., Jézéquel, J.M.: Model-Driven Engineering for Software Migration in a Large Industrial Context. In: G. Engels, B. Opdyke, D. Schmidt, F. Weil (eds.) Model Driven Engineering Languages and Systems, *LNCS*, vol. 4735, pp. 482–497. Springer (2007)
11. Hebig, R., Gabrysiak, G., Giese, H.: Towards Patterns for MDE-Related Processes to Detect and Handle Changeability Risks. In: 2012 International Conference on Software and Systems Process (2012)
12. Hebig, R., Gabrysiak, G., Giese, H.: Towards Patterns for MDE-Related Processes to Detect and Handle Changeability Risks. In: Proceedings of the 2012 International Conference on on Software and Systems Process (2012)
13. Hebig, R., Giese, H.: MDE Settings in SAP. A Descriptive Field Study . Tech. Rep. 58, Hasso-Plattner Institut at the University of Potsdam (2012)
14. Hebig, R., Giese, H., Stallmann, F., Seibel, A.: On the Complex Nature of MDE Evolution. In: A. Moreira, B. Schaetz (eds.) Model Driven Engineering Languages and Systems, 16th International Conference, MODELS 2013, LNCS. Springer, Miami, USA (2013)
15. Hebig, R., Seibel, A., Giese, H.: Toward a Comparable Characterization for Software Development Activities in Context of MDE. In: Proceedings of the 2011 International Conference on Software and Systems Process, ICSSP '11, pp. 33–42. ACM, New York, NY, USA (2011). URL `http://doi.acm.org/10.1145/1987875.1987884`
16. Herrmannsdoerfer, M., Vermolen, S.D., Wachsmuth, G.: An extensive catalog of operators for the coupled evolution of metamodels and models. In: Software Language Engineering, pp. 163–182. Springer (2011)
17. Hutchinson, J., Rouncefield, M., Whittle, J.: Model-driven engineering practices in industry. In: Proceeding

of the 33rd international conference on Software engineering, ICSE '11, pp. 633–642. ACM, Waikiki, Honolulu, HI, USA (2011)
18. Hutchinson, J., Whittle, J., Rouncefield, M., Kristoffersen, S.: Empirical assessment of MDE in industry. In: Proceeding of the 33rd international conference on Software engineering, ICSE '11, pp. 471–480. ACM, New York, NY, USA (2011). URL http://doi.acm.org/10.1145/1985793.1985858
19. Johannes, J., Fernandez, M.: Adding Abstraction and Reuse to a Network Modelling Tool Using the Reuseware Composition Framework. In: T. Kühne, B. Selic, M.P. Gervais, F. Terrier (eds.) Modelling Foundations and Applications, *LNCS*, vol. 6138, pp. 132–143. Springer (2010)
20. Karaila, M.: Evolution of a Domain Specific Language and its engineering environment – Lehman's laws revisited. In: Proceedings of the 9th OOPSLA Workshop on Domain-Specific Modeling (2009)
21. Karanam, M., Akepogu, A.: Model-Driven Software Evolution: The Multiple Views. In: Proceedings of the International MultiConference of Engineers and Computer Scientists, vol. 1 (2009)
22. Khalil, A., Dingel, J.: Supporting the Evolution of UML Models in Model Driven Software Development: A Survey. Tech. rep., School of Computing, Queen's University Kingston, Ontario, Canada (2013)
23. Lerner, B.S.: A model for compound type changes encountered in schema evolution. ACM Transactions on Database Systems (TODS) **25**(1), 83–127 (2000)
24. Mansurov, N., Campara, D.: Managed Architecture of Existing Code as a Practical Transition Towards MDA. In: N. Jardim Nunes, B. Selic, A. Rodrigues da Silva, A. Toval Alvarez (eds.) UML Modeling Languages and Applications, *LNCS*, vol. 3297, pp. 219–233. Springer (2005)
25. Martínez, Y., Cachero, C., Matera, M., Abrahao, S., Luján, S.: Impact of MDE approaches on the maintainability of web applications: an experimental evaluation. In: M. Jeusfeld, L. Delcambre, T.W. Ling (eds.) Proceedings of the 30th international conference on Conceptual modeling, ER'11, pp. 233–246. Springer-Verlag, Berlin, Heidelberg (2011)
26. Mens, T., Blanc, X., Mens, K.: Model-Driven Software Evolution: An alternative Research Agenda. In: The 6th BElgian-NEtherlands software eVOLution workshop (BENEVOL 2007) (2007)
27. Meyers, B., Mannadiar, R., Vangheluwe, H.: Evolution of Modelling Languages. In: 8th BElgian-NEtherlands software eVOLution seminar (BENEVOL) (2009)
28. Meyers, B., Vangheluwe, H.: A framework for evolution of modelling languages. Science of Computer Programming, Special Issue on Software Evolution, Adaptability and Variability **76**(12), 1223 – 1246 (2011)
29. Mohagheghi, P., Gilani, W., Stefanescu, A., Fernandez, M., Nordmoen, B., Fritzsche, M.: Where does model-driven engineering help? Experiences from three industrial cases. Software and Systems Modeling (2013)
30. Noy, N.F., Klein, M.: Ontology evolution: Not the same as schema evolution. Knowledge and information systems **6**(4), 428–440 (2004)
31. OMG: UML 2.0 Superstructure Specification. (2004)
32. Roddick, J.F.: Schema Evolution in Database Systems - An Annotated Bibliography. SIGMOD record **21**(4), 35–40 (1992)
33. Sadovykh, A., Vigier, L., Gomez, E., Hoffmann, A., Grossmann, J., Estekhin, O.: On Study Results: Round Trip Engineering of Space Systems. In: R. Paige, A. Hartman, A. Rensink (eds.) Model Driven Architecture - Foundations and Applications, *LNCS*, vol. 5562, pp. 265–276. Springer (2009)
34. Seaman, C.: Qualitative methods in empirical studies of software engineering. Software Engineering, IEEE Transactions on, title=Qualitative methods in empirical studies of software engineering **25**(4), 557–572 (1999)
35. Selic, B.: The Pragmatics of Model-Driven Development. IEEE Software **20**(5), 19–25 (2003). URL http://csdl.computer.org/dl/mags/so/2003/05/s5019.pdf
36. Shirtz, D., Kazakov, M., Shaham-Gafni, Y.: Adopting model driven development in a large financial organization. In: Proceedings of the 3rd European conference on Model driven architecture-foundations and applications, ECMDA-FA'07, pp. 172–183. Springer-Verlag, Berlin, Heidelberg (2007)
37. Stammel, J., Durdik, Z., Krogmann, K., Weiss, R., Koziolek, H.: Software Evolution for Industrial Automation Systems: Literature Overview. Tech. rep., KIT, Fakultät für Informatik (2011)
38. Vogel, R.: Practical case study of MDD infusion in a SME: Final Results. In: D. Tamzalit, D. Deridder, B. Schätz (eds.) Models and Evolution Joint MODELS09 Workshop on Model-Driven Software Evolution (MoDSE) and Model Co-Evolution and Consistency Management (MCCM),, pp. 68–78 (2009)
39. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M., Regnell, B., Wesslén, A.: Experimentation in Software Engineering : An Introduction. Kluwer Academic Publishers (1999)
40. Yie, A., Casallas, R., Wagelaar, D., Deridder, D.: An Approach for Evolving Transformation Chains. In: A. Schürr, B. Selic (eds.) Model Driven Engineering Languages and Systems, *LNCS*, vol. 5795, pp. 551–555. Springer (2009)