Understanding Changes in Use Cases: A Case Study

Mohammad R. Basirati, Henning Femmer, Sebastian Eder
Technische Universität München,
Germany
m.basirati@tum.de, {femmer, eders}@in.tum.de

Martin Fritzsche, Alexander Widera
Munich Re,
Germany
{MFritzsche, AWidera}@munichre.com

***Abstract*—Requirements change and so (should) do requirements artifacts, such as use cases. However, we have little knowledge about which changes requirements engineers actually perform on use cases. We do not know *what* is changing, *at which locations* use cases change and need a deeper understanding of which changes are *problematic* in terms of difficult or risky.**

To explore these challenges from an industrial point of view, we conducted a mixed methods case study in which we analyze 15 month of changes in use cases in an industrial software project. The study provided interesting observations for both practitioners and researchers involved: First, the most frequently changing use cases had an issue in their structuring. Second, alternative flows (i.e., variations or extensions of the main flow) were especially prone to changes. Third, changes in content (semantic changes) and in presentation of the content (syntactic changes) happen similarly frequently. Last, a qualitative and quantitative analysis aiming at a deeper understanding of problematic changes identified taxonomy changes, as well as locally or temporally dispersed changes as particularly difficult and risky.

In this paper, we contribute a first empirical inquiry for understanding the maintainability of use cases: The presented study provides empirical evidence that there are particular maintenance risks and suggests to continuously analyze local and temporal dispersion.

***Index Terms*—Requirements Engineering, Artifacts, Use Cases, Change, Maintainability**

I. Introduction

Requirements Engineering (RE) artifacts play a central role in many software development projects [12]: RE, Customers and other stakeholders use it to communicate stakeholder's needs, developers create software and testers create systems tests based on these artifacts, to name only a few roles. One common way to document requirements, are use cases (see, e.g., [16]).

In long living software systems, the requirements to the system change over time. With changing requirements, changing the use case artifacts is inevitable: If the use cases are not maintained over time, they become useless to stakeholders, since they do not reflect the (currently intended) system behavior [2].

Changes to requirements artifacts can be of different nature: Some changes target the functionality (the semantics) of the system, leading to changes in other artifacts, such as test cases or source code. Other changes alter only the presentation (on a syntactical level) of a requirement, for example restructuring use cases or improving the understandability of the written text.

In both cases, changing a use case can require a high effort. The reason for this are interconnected use cases, where a change in one use case leads to changes in various other use cases. For example, changing the user interaction for storing certain data might require changes in use cases for creating, reading, updating and deleting this data and, furthermore, in other downstream artifacts such as test cases or source code.

Additionally, changing use cases is risky, since a change might introduce wrong requirements or requirements that are hard to understand. The latter leads to difficulties in subsequent software engineering activities, such as testing or implementing the system. In the case of interconnected requirements artifacts, changes might introduce inconsistencies between these artifacts, which leads to possibly contradicting requirements.

How difficult it is to keep the use cases up-to-date with the changing requirements, depends on the use case's *maintainability*: The risk and the efforts should decrease with an increasing maintainability of the requirements documents.

Problem: There is a lack of knowledge how use cases change over time. In order to be able to manage requirement artifacts change and foster an understanding of maintainability of requirements, we need to empirically investigate requirements change in real world projects.

Approach: This paper aims at exploring maintenance in use cases. In an industrial case study, we manually inspect more than 400 changes in a corpus of 32 use cases over the time of 15 months. In order to validate our results and gain additional insights we pair with industry practitioners through interviews.

Our approach is split into three steps: We first identify change hotspots in the form of often changing use cases and use case parts and analyze the reason for these changes. Afterwards, we inspect the changes in depth by developing a taxonomy of use case changes, and by classifying all changes in an industrial project of the reinsurance company Munich Re. We furthermore investigate, which of the changes impose higher risk or effort.

Contributions: The main contribution of this study are: a) the identification of use cases and parts of use cases, where maintenance happens most frequently, b) a taxonomy of concrete changes in use cases, and c) an elaboration of which changes to use cases lead to problems in our study object.

Impact: The taxonomy presented in this paper enables a more structured investigation of the maintenance activities in use cases, which can be helpful for researchers in order to understand maintenance and can be helpful for practitioners to understand the state of their project. Furthermore the

978-1-4673-6905-3/15 © 2015 IEEE

RE 2015, Ottawa, ON, Canada
Industry Paper

identification of problematic changes points to systematic problems that have to be investigated in detail. The study shows cases in which maintainability of use cases can have a severe impact on the risks and efforts imposed by changes in use cases.

Structure of this work: The paper organized as follows: After discussing related work, we present the design of our study and its results. Then, we describe the threats to our study's validity and conclude with a summary and future research directions.

II. Related Work

Change in software has been studied as a software maintenance and evolution issue for many years. The work of Swanson [8] and later Briand et al. [4] built the foundation for further researchers. Chapin et al. [9] propose a comprehensive redefinition of the change types in software maintenance and evolution, and Buckley et al. propose a taxonomy of software change based on characterizing the mechanisms of change and its influencing factors [7]. Moreover Benestad et al. [3] present a systematic literature review on software maintenance research based on analyzing individual changes which gives us a comprehensive picture of this perspective.

In the domain of RE, most researchers agree that also change in software requirements is a constant phenomenon: Harker et al. recommend to consider the characterization of changes and their nature to improve our insight on their impact [15]. Following this view, researchers analyze changes based on different attributes to reach a better characterization of requirement changes.

There are many studies that analyze changes based on addition, deletion, and modification of the software requirement [14, 24, 21, 10]. Most of the studies focus on classifying the reasons (sources) which trigger changes in requirements [14, 21, 22, 10, 18, 20, 19]. McGee and Greer provide a generic change source taxonomy for better managing requirement changes in a series of studies [18, 20, 19]. They distinguish between uncertainty and trigger as reasons for change and introduce five generic sources of change: Market, Customer Organization, Project Vision, Requirement Specification, and Solution.

A few studies go further and distinguish between the reason of a change and its origin [15, 21, 22]. Nurmuliani et al. identify the reasons of a change as rationales behind the proposed changes, such as Design Improvement, and the origin of a change as where it is originated, such as Design Review [21].

Type of Requirement: Some researches classify changes based on the requirement, on which the changes are applied to [14, 24, 15]. Ghosh et al. classify changes based on five requirement categories: Non Functional, two groups of Functional, User Interface, and Deliverable Requirements [14].

Summarized, a wide range of studies analyze different factors of requirement changes. However, our goal is to understand maintainability of the artifact itself. Therefore, we focus on change within the artifacts from a requirements engineer's point of view instead of its reasons and origins, or its effects and consequences later in the software development process.

III. Study Design

To close the aforementioned gap, we designed a case study investigating on the nature of use case changes in practice.

A. Goal and research questions

The goal of this study (formally defined in Table I) is to understand changes in use case artifacts and the modifications that are performed on them. To accomplish the stated goal, we aim at finding out what is changed particularly often (which of use cases and in which part), what these changes are and what types of changes are problematic to requirements engineers.

TABLE I: Research goal

Analyze changes in contents of use case artifacts **with respect to** frequency, location, change types, and level of risk **from the point of view of** requirements engineers in practice **in the context of** a large business information system in maintenance.

From this goal definition, we conduct an exploratory study based on the following research questions:

RQ1. Which use cases change and in which part? To understand the changes in the project, we first analyze the distribution of changes *over use cases* individually: Are there single use cases that change more than others? If so, why?

We furthermore inspect the relation of changes to the *structure* of use cases in general. In consistency with, e.g., [13], we refer to the use case structural elements as content items (e.g., basic flow, preconditions, etc.): Do some content items form hotspots in use cases that are particularly prone to changes? If we understand these hotspots, they might point at certain types of bad maintenance. In addition, for parts of use cases that change particularly often, maintenance might be especially important.

RQ2. Which types of changes exist and occur in use cases? After understanding which use cases change and in which parts, we can then investigate into the contents of the changes, i.e., we need to create a *taxonomy* of these changes. This enables us to understand whether certain *types of changes occur more frequently* than other.

RQ3. Which changes are the most problematic? Lastly, our ultimate goal is to understand and handle problematic changes. We want to provide a first understanding what are problematic changes from a practitioner's perspective and analyze their nature according to the classifications in RQ2. In the long term, this might enable us foresee and prevent some of these problematic changes.

B. Case and Subjects Selection

The case was selected with the goal to study realistic models under realistic conditions, i.e., a real project from industry; Out of these, the selection was performed opportunistically. We invite other researchers to reproduce the study in different contexts.

For the subject selection, we carefully selected those practitioners who performed the analyzed changes and thus worked or work with the use cases.

C. Data Collection and Analysis

We retrieved the use cases of each iteration from the companies versioning system. For this, we extracted the major version of each iteration.

We identified a change as *a block of added, removed or modified words*. We compared the textual contents between the iterations, based on the built-in diff function of Microsoft Word. Even though this adds manual effort for extracting the changes out of Microsoft Word for quantitative statistics, it eases classification, since changes are displayed in the context of changes in the whole use case. When relying on the built-in diff was not technically possible, e.g., due to a difference in layout, we manually compared the text. When use cases changed their name, we identified the corresponding use case and compared them nevertheless. The first version of newly appearing use cases is not counted as a change, but only the following changes, since our study focuses on maintenance.

We filtered graphical changes, layout changes, as well as changes in the document meta data (table of contents, authors, etc.), since we focused on the changes of the textual content in use case artifacts. We manually inspected the changes to filter out single, extreme outliers (e.g., in change size) that would otherwise skew the data.

To answer **RQ1**, we proceeded as follows (more detailed explanation and definitions of the metrics can be found in Table II):

1) Frequency of changes can be understood in three ways: The total number of changes (modified text blocks), $size_{count}$, the number of words changing (often called $churn$ in source code metrics), $total\_churn$, or the number of iterations in which a use cases changes, $size_{iter}$. To identify frequently changing use cases, we aggregate $size_{count}$, $total\_churn$ and $size_{iter}$ per use case.
2) To identify frequently changing content items, we aggregate, for each content item, the average number of changes over all iterations, $avg\_changes$. However, since the content items vary strongly in size (e.g., a list of actors section versus the basic flow section), we also normalize the number of changes by the size of the content item, $relative\_churn$.
3) We discuss the three most frequently changing use cases as well as the most frequently changing content items according to all aforementioned metrics with practitioners, in order to understand why these use cases and content items might change more often.

To answer **RQ2** we proceeded as follows:

4) We perform iterative, open coding on the changes and translate the codes into a taxonomy.
5) We classify all changes according to the created taxonomy.
6) We calculate the $size_{count}$ as well as the $size_{words}$ for each taxonomy item.
7) We discuss the taxonomy, as well as its distribution with practitioners.

To answer **RQ3** we proceeded as follows:

TABLE II: Used Definitions and Metrics

$change$	A block of added, removed or modified words
$changes(uc, it)$	The set of changes for a use case in a given iteration
$size_{count}(change)$	The count-size of a single change is 1
$size_{count}(changes)$	The number of changed text blocks
$size_{words}(change)$	The number of words removed, added or modified within a single change
$size_{words}(changes)$	$\sum_{\forall c \in changes} size_{words}(c)$
$size_{iter}(changes)$	The number of distinct iterations of the changes
$churn(uc, it)$	The number of words changed in a use case in an iteration: $size_{words}(changes(uc, it))$
$total\_churn(uc)$	$\sum_{\forall it \in iterations} churn(uc, it)$
$relative\_churn(ci)$	$\frac{\sum_{\forall it \in iterations} \frac{\sum_{\forall c \in changes(ci,it)} size_{words}(c)}{size_{words}(ci, it)}}{n(iterations)}$
$changes(ci, it)$	The set of changes in a given content item in a given iteration
$avg\_changes(ci))$	The average $size_{count}(changes(ci, it))$ of a content item over all iterations

8) We interview practitioners about problematic changes in use cases from their perspective.
9) We group syntactically similar (*coupled*) changes and calculate dispersion of a group over use cases as well as the dispersion over time (number of different iterations).
10) We compare the largest groups with the categories of RQ2 and discuss results with practitioners.

D. Validity Procedure

To control threats to internal validity due to errors or mistakes, all data analysis and interpretation is executed by at least two researchers, and validated with practitioners involved in the case study.

In order to control the risk that misunderstanding or ambiguity of the taxonomy impacts the distribution for Step 5-7, two researchers independently classify a random subset of 10% of the changes and calculate the inter-rater agreement (Cohen's kappa).

IV. Results

In the following, we report on the results of a first execution of the study at Munich Re.

A. Case and Subjects Description

We performed the study at Munich Re, which is one of the world's leading reinsurance companies with more than 43,000 employees in reinsurance and primary insurance worldwide. For their insurance business, they develop a variety of custom software systems. To elicit the changes of a regular Munich Re project, we inspected the development of a medium-sized industrial software project (around 102,000 SLoC in about 1,900 files), which went live 5 years ago and is thus currently in the maintenance phase. The system forms a complex interface to the Munich Re calculation of life insurance probabilities and conditions. The system behavior is currently described in 32 use cases with a total size of around 35,000 words. The system is furthermore described through supplementary requirements artifacts, such as business rules, business specifications, security

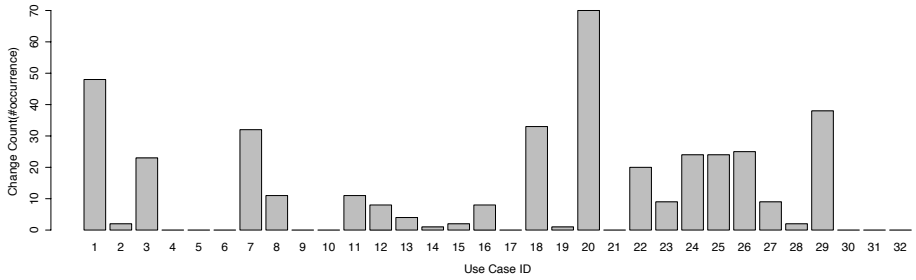

Fig. 1: Total Number of Changes in Use Cases

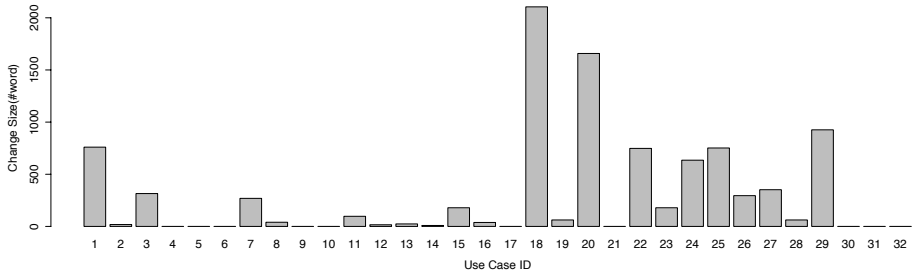

Fig. 2: Total Size of Changes in Use Cases

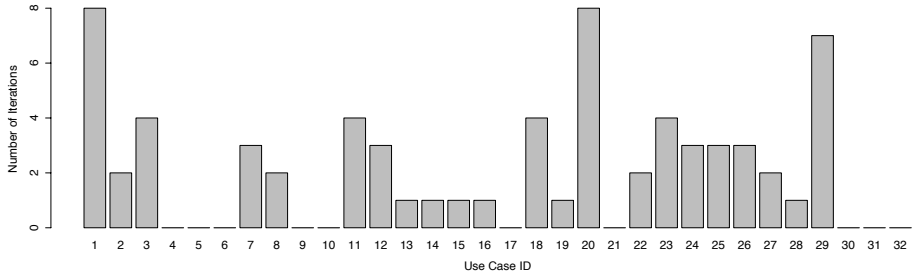

Fig. 3: Number of Iterations Use Case Changed in

plans and guidelines, as well as UI specifications, all of which were not subject of this analysis.

The analyzed use cases were tracked in twelve iterations over the time of 15 months from February 2012 until May 2013. All changes were performed by the RE team of the project. We consider the Munich Re use cases a very common interpretation of the established Cockburn template (e.g. [11]). They contain the following content items: A *brief description* over the contents of the use case, the *actors* and their authorizations in this use case, *pre-* and *post conditions*, as well as the *basic flow* ("main success scenario" in Cockburn [11]) and a set of *alternative flows* ("extensions" or "subvariations" [11]).

In total, our change set included 485 changes. Of these, we filtered 33 changes in the document history, 23 graphical changes, and one outlier, which extracted all business rules into another separated document, and which would have skewed our size metrics. Hence, the following analysis is based on the remaining 405 changes.

Due to non-disclosure agreements and sensible information, in the following, we removed some details of the software system under analysis, focusing on the research questions.

In the following, we answer each research question, by providing our results, explaining practitioners feedback and validation and finally interpreting the results.

RQ1: Which use cases change and where?

We first analyze which use cases change and then analyze the content items changing.

Which use cases change most often? As explained in the study design, the size of a set of changes can be understood in three different ways: Either by taking each change as an atomic unit and simply counting the number of changes for each use case, or by counting the number of words that are

changed within each single change, or, lastly, by counting the number of iterations in which the use cases change.

Figure 1 depicts changes in use cases in terms of count. Use Cases 1, 20 and 29 have the largest number of changes, with between 38 and 70 changes over all iterations. Figure 2 depicts changes in use cases in terms of number of words. Use Cases 18, 20 and 29 have the largest amount of changes with between 920 and 2100 words changed. Lastly, we want to know which use cases were the most volatile over time and changed during larger number of iterations. Figure 3 depicts the number of different iterations in which each use case has changed. The most volatile ones are Use Cases 1 and 20 with changes in 8 iterations and Use Cases 29 with changes in 7 iterations.

Practitioners Feedback: Practitioners explained that these results were not unexpected, since there is an inherent dependency between Use Cases 1, 20, and 29 due to the business workflow. Practitioners told us that this is a structural problem based on the slicing of use cases and, if they would have to write the use cases again, they would put these three use cases into a single use case. Regarding the fact that these use cases changed among 7 or 8 iterations, they told us that this represents Munich Re's process of sometimes incrementally adding functionality. In one case the use cases were changed over two to three iterations until they were implemented into code. The explanation for the large changes in Use Case 18 was a major extension that was executed in this year in an incremental style over 4 iterations.

Interpretation: The analysis revealed three problematic use cases that are constituents of one business flow. In other words, there was one challenging volatile business flow that needed the most attention to take care of. This analysis shows that in our case study the analysis of most changing use cases was indicating at a dependency between use cases.

Which content items of use cases change most often?

In a second part of this research question, we inspect the change within different content items of use cases, measured by the *churn* (see Table II): Figure 4 depicts the average number of changes occurring in each content item per iteration. Absolutely, most changes (avg. of 24 changes per iteration) appear in the basic flow, followed by the alternative flows (avg. of 7 changes per iteration). The other content items hardly change at all.

However, assuming that change is statistically dependent with size, we normalize the change frequency through the size in words, as the *relative churn*, resulting in the distribution shown in Figure 5. This figure shows a different view: The alternative flow is, normalized by the size, changing more than the basic flow.

When inspecting the changes in alternative flows in more depth, we could see that most of changes in this section are adding new flows from scratch or moving concepts from basic flow to alternative flows.

Practitioners Feedback: Practitioners were surprised to see that the alternative flow has a greater relative churn than the basic flow. They saw two reasons for this: First, it could be caused due to common flow restructuring in

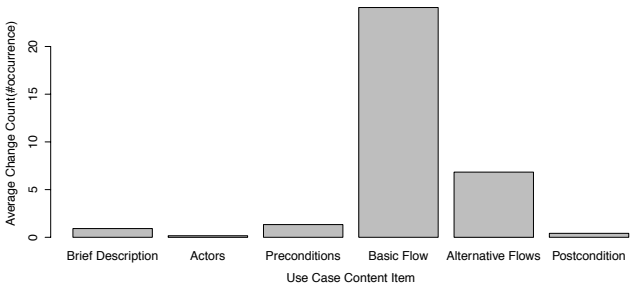

Fig. 4: Average Number of Changes in Use Case Content Items per Iteration

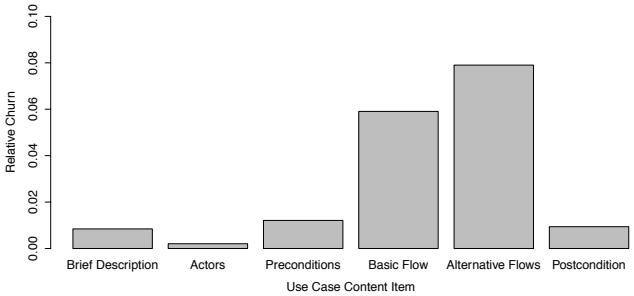

Fig. 5: Change Probability of Use Case Content Item per Word

alternative flows. Second, since this is a project with a long life span, new requirements often are extensions to existing use cases. This could lead to the fact that new requirements manifest themselves as new alternative flows. Regarding the non-changing content items, they agreed, since changes in actors (including authorizations) for such a mature project is very rare.

Interpretation: In this case study, the basic flow is changed most often. However, looking at the content per word, the alternative flow has a higher change rate. Based on the feedback from practitioners we interpret the data as follows: Alternative flows of a use case are not completely clear at the beginning; there are alternative flows which initially cannot be recognized neither by customer nor by requirement engineers and it takes time until they get exposed. Another reason for high relative churn in alternative flows can be lack of attention on alternative flows at first, caused by the fact that basic flow matters more than alternative flows.

In summary, we interpret these results as showing, that the maintenance of basic and alternative flows should be watched carefully, since these content items will change more often over time. This implies for practitioners, that proper slicing of use cases early on can potentially save effort later in the process.

RQ2: Which types of changes exist and occur?

To answer this research question, we developed a taxonomy of use case changes. In contrast to related work, we focused

on concrete changes in use cases without considering the high level reason or source of a change.

Which types of changes exist? In order to create a taxonomy that adequately represents the changes under analysis, and not limit ourselves with any constraints or assumptions, we applied a bottom-up approach, i.e., an open classification of a subset of changes. We iteratively extended and groomed this taxonomy, until we resulted in the categories given in Table III. In the following we will not explain each category in detail, but the structure and motivation of the categories.

The taxonomy is based on the differentiation between semantic and syntactic changes: When a semantic change is executed, the system that is described in the use case changes. This implies, that semantic changes necessarily lead to subsequent changes in other artifacts. In contrast, syntactic changes only change how the system is described in the use cases. Syntactic changes are similar to refactorings, as known from source code.[1]

All semantic changes are, from an RE perspective, to add, modify or remove requirements. All differentiation here would lead to an analysis of the reasons for use case changes, which is out of scope of this research. In contrast, the syntactic changes scatter over various types of use case refactorings. Please refer to Table III for more details.

Practitioners Feedback: The taxonomy was successfully validated with practitioners: Practitioners were not aware of further categories, and advised against removing any of the introduced categories.

Which types of changes occur? To understand which changes occur, we made a closed classification of all 405 changes, applying the bottom-up taxonomy discussed before. Figure 6 shows the total number of changes in each category. `Adding Requirement`, `Modifying Requirement`, and `Clarifying Taxonomy` have the largest number of changes among all change categories respectively. They are followed by `Flow Management`, `UI`, and `Sentence Enhancement`.

Figure 7 depicts a box plot for change sizes in each category, ordered by total size in terms of words of changes in each category. `Adding Requirement`, `Modifying Requirement`, and `Flow Management` have the largest total size of change, while `Document Management`, `Flow Management`, and `Adding Requirement` have the greatest mean size of change respectively. The results show that after `Adding Requirement` and `Modifying Requirement`, `Flow Management` changes were the most common on use case documents. Furthermore striking is the large number of `Clarifying Taxonomy` changes occurred, all of which had very small sizes.

[1]We use the terms 'semantic' and 'syntactic' here in the context of system behavior, in contrast to the semantics of natural language. Obviously, nearly every addition, modification or deletion of words changes the semantics of the natural language text, but not each change also changes the semantics of the described system. For example, if we clarify the meaning of a term appearing in the use case, we still describe the same system behavior, but with more precise language (thus the semantics of the text changes).

Practitioners Feedback: Practitioners were surprised by the high number of `Adding Requirement` and `Modifying Requirement` changes. They were unsure whether this was typical for their projects. They expected the high number of `Clarifying Taxonomy` changes, due to major changes that we will describe in detail in the RQ3.

Interpretation: In our case study, `Adding Requirement` and `Modifying Requirement`, as well as `Clarifying Taxonomy` are the most common change types. Future work needs to investigate whether this is also common for other projects or a case specific result.

How are semantic and syntactic changes distributed? Besides the detailed analysis of each category by itself, we also analyze how changes are distributed over the categories of semantic and syntactic changes. To identify semantic and syntactic changes we used the classification based on our detailed categories of changes, in which we classify `Adding Requirement`, `Modifying Requirement`, or `Removing Requirement` as semantic changes and other categories as syntactic changes.

TABLE IV: Syntactic vs Semantic Changes

	Semantic	Syntactic
$size_{count}$	47%	53%
$size_{words}$	71%	29%

The results in Table IV show that 53 percent of all changes are syntactic changes and 47 percent are semantics. When taking the size in words into account 29 percent of the total size of all changes are syntactic and 71 percent are semantic. Hence, semantic changes are larger than syntactic changes (avg. size of 13 words for syntactic changes vs. avg. size of 35 words for semantic changes).

In more depth, Figure 8 and Figure 9 show the count and size percentage of semantic and syntactic changes in use case content items respectively. Actors had only syntactic changes, however only few changes occurred in this content item. Actors, Preconditions, and Postconditions are more prone to syntactic changes than other content items. Figure 11 depicts that the Brief Description had been changed mostly because of a semantic change in use cases. The relation between semantic and syntactic changes in Basic Flow and Alternative Flows represents roughly average distribution (see Table IV).

Figure 10 shows the distribution of semantic vs syntactic changes over all 12 iterations. We consider Iteration 7 an outlier, since it only contained 4 changes. Iteration 1, 5 and 6 show the highest percentage of syntactic changes, with around 60%. Other iterations show around 20% to 40% of syntactic changes.

Practitioners Feedback: Practitioners agreed with classifying changes into semantic and syntactic and immediately had an intuitive understanding. However, from their practical standpoint, they could not relate to the relative numbers from Tbl. IV, since the size and count of changes does not necessarily represent effort spent. Future work should look deeper into a change-effort relation for RE. However, they considered inspecting the trend of the relation over time reasonable: In

TABLE III: Use Case Change Taxonomy

Semantic Changes	Adding Requirement	Adding a text which adds something new to the system	"System sets the validation status to invalid"
	Modifying Requirement	Adding or modifying a text which leads to a change in the system	"five different" to "the maximum number is 12"
	Removing Requirement	Removing a text in order to delete a part of or even a complete requirement	"In this case the latter action will be performed without changing the status"
Syntactic Changes	Flow Management	Reorganizing the basic or alternative flows of use case in which the actions are taking place	Dividing, merging, moving a part or whole of a step
	Document Management	Extracting a text or figures into another document or vice versa	Extracting GUI figures to another document.
	Sentence Enhancement	Enhancing the structure of a sentence to make it more clear without changing its meaning	"the permitted" to "those that the user is authorized for"
	Typos	A typographical error	"fore" to "for"
	Clarifying Taxonomy	Adding or modifying some words in a phrase to distinguish it from a changed or new concept in a clearer way. If this rephrasing also changes a requirement it is classified as modifying requirement	"business" to "entity X"
	Formatting	Changing the presentation style of a requirement by adding, modifying or removing text	Representing some rules in a table instead of lines of text.
	Adding Supplementary Reference	Adding a reference to a context to make it more clear or connect it to other required documents	"(see BR_XX)"
	Updating Reference	Updating a reference to new version	"(apply UC_BRXX)" to "(apply LRXX_BRXX)"
	Adding Details	Adding a text to make something more clear without adding or changing the context of a requirement	"both fields" to "the text field and the drop down box"
	Removing Useless	Removing a useless or outdated text	Deletion of an outdated version of a rule
	UI	Adding, modifying, or removing a text or figure which exclusively speaks on user interface	"If an element has the status X is highlighted" design and features

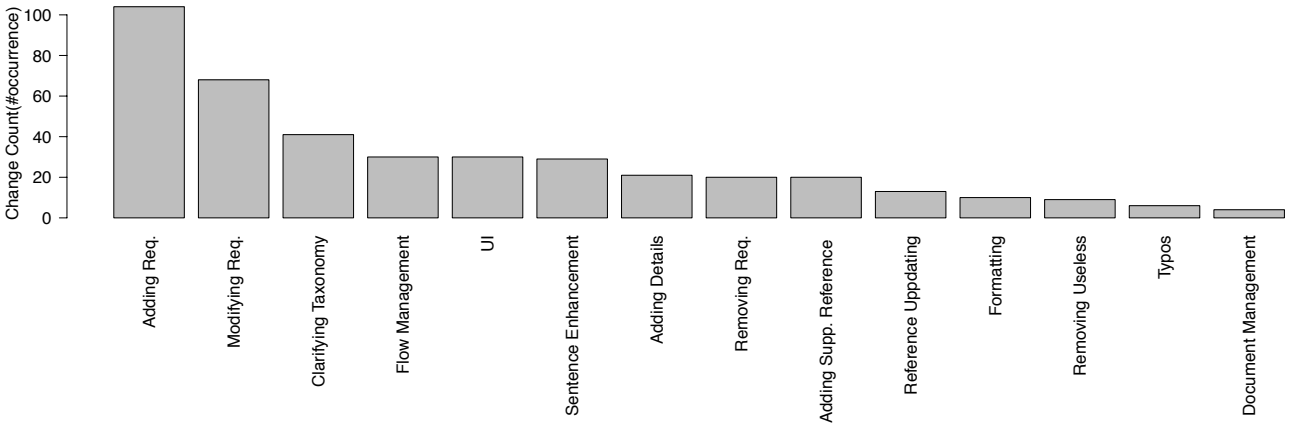

Fig. 6: Total Number of Changes per Category

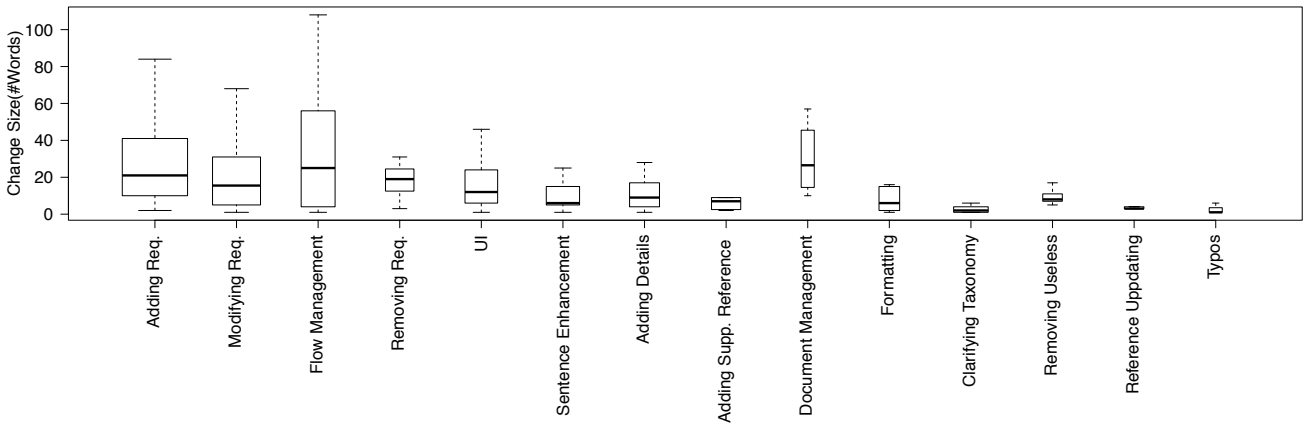

Fig. 7: Word-Size of Changes per Category, Ordered by Total Size, Width∼Sample Size

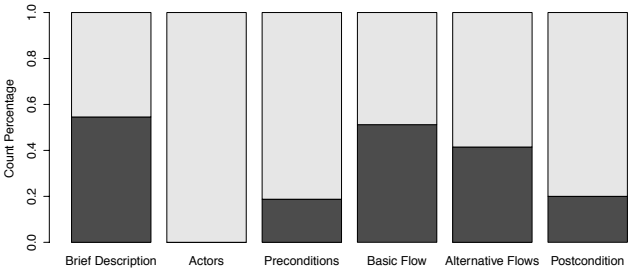

Fig. 8: Syntactic (Light Gray) VS Semantic (Dark Gray) Change Size Count Percentage in Content Items

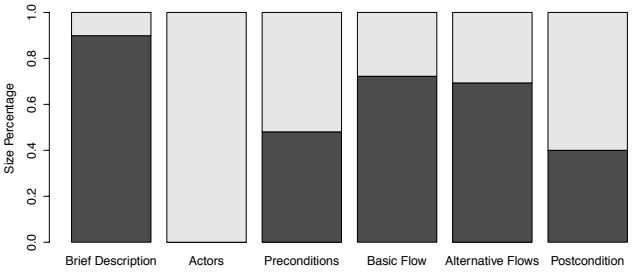

Fig. 9: Syntactic (Light Gray) VS Semantic (Dark Gray) Change Size Word Percentage in Content Items

the first sprints, requirements engineers had time to deal with syntactic refactorings in the use cases. The same holds for Iteration 6.

Interpretation: This case study showed a proportion of 53% of syntactic changes on RE artifacts, which, due to their smaller size, account for 29% of the size of changes. For source code, studies show that, e.g., after reviews only around 25% (e.g. [1]) of changes are semantic changes. Even though this provides us with a rough benchmark, unfortunately, due to the high granularity of iterations in our case study, we cannot detect which changes were triggered through reviews. Future work should reproduce our results with review changes.

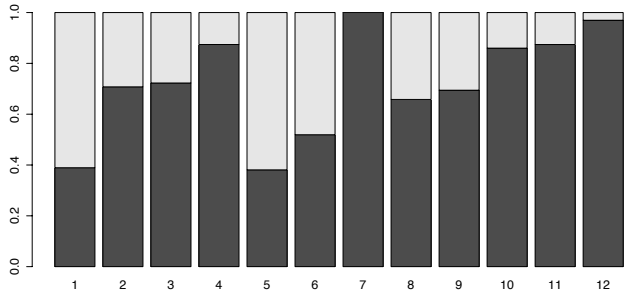

Fig. 10: Syntactic (Light Gray) VS Semantic (Dark Gray) Changes Size Word Over Iterations

TABLE V: Five largest change groups

		#Occurrence	# Iterations	# Use Cases
Group 1	Clarifying Taxonomy	18	3	6
	Modifying Requirement	1		
	Removing Requirement	1		
Group 2	Clarifying Taxonomy	3	3	4
	UI	9		
Group 3	Clarifying Taxonomy	2	1	1
	Modifying Requirement	5		
	Adding Requirement	5		
Group 4	Modifying Requirement	1	1	2
	Adding Requirement	5		
Group 5	Clarifying Taxonomy	7	1	1

In our study preconditions and postconditions showed a higher proportion of syntactic changes. It remained unclear where this difference results from.

We could furthermore observe that the percentage of syntactic changes reflected rather calm periods in the project. We imagine that tracking the distribution of syntactic vs semantic changes could indicate for some sort of requirements technical debt (cf. e.g. [6]), indicating periods of pressure and refactoring in the project.

RQ3: What are problematic types of changes?

In our last research question, we wanted to understand which changes are difficult to conduct within the requirements artifact[2].

What are problematic types of requirements artifact changes from an industry perspective? In order to receive open, unrestricted feedback, we openly asked the practitioners what they considered problematic changes for use cases. This resulted in two types of problematic changes:

The first group of problematic changes are semantic changes that are inherently complex (*essential complexity* in the words of Brooks [5]), due to the nature of the domain. This is especially common in the insurance domain, since the systems build on a complex mathematical and legal background.

The second group of problematic changes are changes that are not inherently complex, but particularly risky, since there is a dependency between multiple changes. We would consider these changes *accidentally complex*, since the complexity results from the RE instead of the domain. The practitioners took our taxonomy as reference and identified `Clarifying Taxonomy`, `Flow Management`, `Adding Details` or also locally dispersed semantic changes as particularly risky, since these changes might have a higher probability of introducing inconsistencies or unintentionally introduce other incorrect behavior (c.f. the risk of dispersed software clones leading to inconsistencies in code [17]).

[2]We only consider the maintainability of the requirements artifact itself, not the impact that the change triggers. This has been studied by previous works, cf. Section II.

Can we find problematic changes through syntactically coupled changes? In order to detect risky changes from dependencies, we searched for *locally dispersed changes*, i.e., changes that are syntactically similar (or *coupled*), but appear in different locations, either within a document, between documents, or even over time (*temporally dispersed*). Syntactically coupled changes are a group of changes which can be recognized explicitly by their terms and phrases as the same change in different spots in use cases and iterations. Table V shows the five largest change groups in terms of highest number of occurrences, including their change types and number of occurrence, number of dispersed iterations and use cases. In the following, we provide details and explain practitioners feedback on the rationale behind each group.

Group 1 was a fundamental taxonomy change due to a change in the requirements: At first, the whole system was built around a single type of entity, the *model point*, which is added, changed, calculated etc. However, at a certain iteration in the middle of the project, the system needed to be changed so that it could handle two different types of model points. This lead to fundamental changes over 6 use cases, which needed three iterations until finished (one change was conducted several iterations later, a fact that could hint at an inconsistent/forgotten change). This group showed a risky temporally and locally dispersed change.

Group 2 contained a set of changes in the taxonomy, which lead to changes in multiple use cases and iterations. The problem here was that the use cases also contained UI references, which needed to be updated subsequently. Practitioners told us that they considered this bad practice and, if possible, would move UI design specification to a separate artifact. This indicated bad requirements artifact maintainability resulting from UI details in use cases.

Group 3 was a set of reference to an enumeration that needed to be continuously updated when a new item is added to the list, e.g., B1, B2, etc. They stated that these types of numbered references are very hard to maintain and can easily lead to wrong references.

Group 4 was an essentially complex requirements change from the business domain.

Group 5 was a clarification of taxonomy that was always implicitly clear for insiders, but became obvious when new people joined the team.

Interpretation: 49% of changes in the top five change groups were in the category `Clarifying Taxonomy`. It's an inherent property of Taxonomy Changes to spread among different use cases, since terms are usually used orthogonally through all use cases. We interpret these change groups as advice to have the terms clear at the start of writing use cases, since changes in taxonomy in late phases causes dispersed and thus potentially problematic changes. Furthermore, we found evidence that UI details as well as enumerated references can cause dispersed changes and thus decrease the maintainability of use cases.

V. Threats to Validity

Regarding our answers to the research questions, two major threats could constrain our internal validity: First, elaborating our change taxonomy was a creative process, hence, it could be ambiguous, incorrect or incomplete. We analyzed the ambiguity through independent reclassification of a subset of 10% of the changes, leading to a substantial inter-rater agreement (Cohan's kappa: 0.65). We therefore consider this threat negligible. Second, our change taxonomy could also be incorrect (internal validity) or incomplete (external validity). We control these threats, as well as threats regarding potential bias in interpretation of practitioner's feedback, through validation with practitioners.

Regarding external validity, case study research inherently comes with advantages, but limited generalizability, since it always answers research questions for a limited set of cases [23]. Our study intentionally focused on an industrial project in the maintenance phase, hence, the results might not be generalizable for requirements artifacts in elaboration. Furthermore, we intentionally analyzed changes per iteration instead of more fine-grained changes. This could also create a different picture, such as more typos. Lastly, our results show that changes depend on the maintainability of the use cases. Therefore, we are expecting to see different results for use cases in different quality.

In addition, our study intentionally focused on expert opinions, which can only provide some facets of (bad) maintainability. Thus, we invite other researchers to reproduce this case study in order to confirm or refute our observations and extend the validity onto other change granularities, project settings, and maintainability facets.

VI. Conclusion and Outlook

Changes in requirements artifacts are common in software projects, since outdated artifacts are not useful to stakeholders. However, there is little existing knowledge on maintainability of use cases.

This paper presents an analysis of use case changes based on a case study in an industrial software project in maintenance. Applying qualitative and quantitative methods to more than 400 changes and discussing the results with practitioners, we answer our research questions in this case as following:

RQ1: Which use cases change and where? Our analysis revealed that the most frequently changing use cases in our case, are in a strong dependency with each other (belonging to the same workflow). We also found that although most of changes occur in the basic flow, alternative flows are most prone to change relative to their size. We observe that improper slicing of use cases forms one way of bad maintainability and that most maintenance changes go into alternative and basic flows. This indicates that these two content items have a stronger need for use case maintainability.

RQ2: Which types of changes exist and occur? We developed a detailed change taxonomy for use cases, with which we found out 50% number of all changes and 30% of the total size of all changes are syntactic. Over time we

see that phases with higher proportion of syntactic changes coincide with rather calm periods in the project. We conclude that tracking the proportion of syntactic and semantic changes over time can indicate the effort going into quality assurance of use cases. Therefore, future work should analyze the potential of this metric for project and QA monitoring.

RQ3: What are problematic types of changes? Practitioners report that problematic changes origin from essential complexity, i.e., complexity in the domain, and accidental complexity, i.e., complexity in the requirements artifacts themselves. For the latter, we identified dispersed changes as particularly risky. This study has shown the particular difficulty of changes in the domain taxonomy. We furthermore identified UI details and improper referencing as other causes for risky, dispersed changes. This motivates to monitor dispersed changes, but also provides first empirical evidence towards factors for bad maintainability, such as UI details or improper references.

Future work: We found these factors for bad maintainability in our case study. Future work should dig deeper This study was performed on major versions of use cases. An analysis on minor versions of use cases can gives us deeper insight on the way use cases change, including which changes are triggered by use case reviews in particular.

ACKNOWLEDGMENTS

This work was performed within the project Q-Effekt; it was funded by the German Federal Ministry of Education and Research (BMBF) under grant no. 01IS15003 A-B. The authors assume responsibility for the content.

REFERENCES

[1] Moritz Beller, Alberto Bacchelli, Andy Zaidman, and Elmar Juergens. Modern code reviews in open-source projects: which problems do they fix? In *MSR*, 2014.

[2] E. Ben Charrada, A. Koziolek, and M. Glinz. Identifying outdated requirements based on source code changes. In *RE*, 2012.

[3] Hans Christian Benestad, Bente, and Erik Arisholm. Understanding software maintenance and evolution by analyzing individual changes: a literature review. *Journal of Software Maintenance and Evolution: Research and Practice*, 2009.

[4] Lionel C. Briand, Victor R. Basili, and Yong-Mi Kim. A change analysis process to characterize software maintenance projects. In *ICSM*, 1994.

[5] Frederick P. Brooks, Jr. No silver bullet: Essence and accidents of software engineering. *IEEE Computer*, 1987.

[6] Nanette Brown et al. Managing technical debt in software-reliant systems. In *FSE/SDP workshop on Future of software engineering research*, 2010.

[7] Jim Buckley, Tom Mens, Matthias Zenger, Rashid Awais, and Gúnter Kniesel. Towards a taxonomy of software change. *Journal of Software Maintenance and Evolution: Research and Practice*, 2005.

[8] Swanson E. Burton. The dimensions of maintenance. In *ICSE*, 1976.

[9] Ned Chapin, Joanne E Hale, Khaled Md Khan, Juan F Ramil, and Wui-Gee Tan. Types of software evolution and software maintenance. *Journal of Software Maintenance: Research and Practice*, 2001.

[10] Bee Bee Chua and June Verner. Examining requirements change rework effort: A study. *International Journal of Software Engineering & Applications*, 2010.

[11] Alistair Cockburn. Basic use case template. *Humans and Technology, Technical Report*, 1998.

[12] Henning Femmer, Jakob Mund, and Daniel Méndez Fernández. It's the Activities, Stupid! A New Perspective on RE Quality. In *RET workshop at ICSE*, 2015.

[13] Daniel Méndez Fernández, Birgit Penzenstadler, Marco Kuhrmann, and Manfred Broy. A meta model for artefact-orientation: fundamentals and lessons learned in requirements engineering. In *Model Driven Engineering Languages and Systems*. 2010.

[14] Sanjay Ghosh, Srini Ramaswamy, and Raoul Praful Jetley. Towards requirements change decision support. In *APSEC*, 2013.

[15] S.D.P. Harker, K.D. Eason, and J.E. Dobson. The change and evolution of requirements as a challenge to the practice of software engineering. In *RE*, 1993.

[16] Ivar Jacobson, Grady Booch, and James Rumbaugh. *The unified software development process*. Addison-Wesley Reading, 1999.

[17] Elmar Juergens, Florian Deissenboeck, Benjamin Hummel, and Stefan Wagner. Do code clones matter? In *ICSE*, 2009.

[18] Sharon McGee and Des Greer. A software requirements change source taxonomy. In *ICSEA*, 2009.

[19] Sharon McGee and Des Greer. Sources of software requirements change from the perspectives of development and maintenance. *International Journal on Advances in Software*, 2010.

[20] Sharon McGee and Des Greer. Software requirements change taxonomy: Evaluation by case study. In *RE*, 2011.

[21] Nur Nurmuliani, Didar Zowghi, and Sue Fowell. Analysis of requirements volatility during software development life cycle. In *ASWEC*, 2004.

[22] Nur Nurmuliani, Didar Zowghi, and Susan P. Williams. Requirements volatility and its impact on change effort: Evidence-based research in software development projects. In *ASWEC*, 2006.

[23] P. Runeson and M. Höst. Guidelines for Conducting and Reporting Case Study Research in Software Engineering. *EMSE*, 2009.

[24] George E. Stark, Paul Oman, Alan Skillicorn, and Alan Ameele. An examination of the effects of requirements changes on software maintenance releases. *Journal of Software Maintenance: Research and Practice*, 1999.