Exploring the Effects of History Length and Age on Mining Software Change Impact

Leon Moonen*, Stefano Di Alesio*, Thomas Rolfsnes* and Dave W. Binkley§
*Simula Research Laboratory, Oslo, Norway §Loyola University Maryland, Baltimore, Maryland, USA
leon.moonen@computer.org, {stefano,thomgrol}@simula.no, binkley@cs.loyola.edu

***Abstract*—The goal of Software Change Impact Analysis is to identify artifacts (typically source-code files) potentially affected by a change. Recently, there is an increased interest in *mining* software change impact based on evolutionary coupling. A particularly promising approach uses association rule mining to uncover potentially affected artifacts from patterns in the system's change history. Two main considerations when using this approach are the *history length*, the number of transactions from the change history used to identify the impact of a change, and *history age*, the number of transactions that have occurred since patterns were last mined from the history. Although history length and age can significantly affect the quality of mining results, few guidelines exist on how to best select appropriate values for these two parameters.**

In this paper, we empirically investigate the effects of history length and age on the quality of change impact analysis using mined evolutionary couplings. Specifically, we report on a series of systematic experiments involving the change histories of two large industrial systems and 17 large open source systems. In these experiments, we vary the length and age of the history used to mine software change impact, and assess how this affects precision and applicability. Results from the study are used to derive practical guidelines for choosing history length and age when applying association rule mining to conduct software change impact analysis.

***Index Terms*—change impact analysis, evolutionary coupling, association rule mining, parameter tuning.**

I. Introduction

When software systems evolve, the interactions in the source code grow in number and complexity. As a result, it becomes increasingly challenging for developers to predict the overall effect of making a change to the system. Change Impact Analysis [1] has been proposed as a solution to this problem, aimed at identifying software artifacts (e.g., files, methods, classes) affected by a given change. Traditionally, techniques for change impact analysis are based on static or dynamic analysis, which identify dependencies, for example, methods calling or called by a changed method [2, 3, 4]. However, static and dynamic analysis are generally language-specific, making them hard to apply to modern heterogeneous software systems [5]. In addition, dynamic analysis can involve considerable overhead (e.g., from code instrumentation), while static analysis tends to over-approximate the impact of changes [6].

To address these challenges, alternative techniques have been proposed that identify dependencies through *evolutionary coupling* [7, 8, 9, 10]. In essence, evolutionary coupling exploits a developers inherent knowledge of the dependencies in the system, which manifest themselves through commit comments, bug reports, context switches in IDEs, and so on [8]. These couplings differ from those found through static and dynamic analysis, because they are based on how the software system has evolved over time, rather than how system components are interconnected.

This paper considers *historical co-change* between artifacts as the basis for uncovering evolutionary coupling. Known techniques [10, 11, 12, 13] for mining evolutionary couplings from artifact co-changes build on *association rule mining* (or *association rule learning*) [14], and differ in the way that the association rules are generated from the history. Nevertheless, key to all such techniques is the *history* used to learn from. There are two main factors related to the history that impact the mined rules: (1) the *history length*, the number of transactions in the history considered while mining co-change patterns, and (2) the *history age*, the number of transactions that have occurred since these patterns were mined. The resulting rules directly affect the quality of any change impact analysis based on mined evolutionary coupling. However, while reviewing the literature, we found that the effects of history length and age on mined association rules have not been systematically studied. We address this shortcoming.

Contributions: This paper presents a series of systematic experiments using the change histories of two large industrial systems and 17 large open source systems. The study makes two key contributions: (1) we investigate the extent to which history length and age affect the quality (formalized in Section IV-F) of the change impact sets derived via association rule mining, and (2) we derive practical guidelines for selecting an appropriate system-specific value for history length and for determining at what age a model has sufficiently deteriorated to benefit from rebuilding. The guidelines enable a team of engineers to best exploit association rule mining for change impact analysis in the context of their project.

Overview: The remainder of this paper is organized as follows: Section II provides background on mining evolutionary coupling. Section III presents our research questions. Section IV describes the setup of our empirical investigation, whose results are presented in Section V. Finally, Section VI presents related work, and then Section VII provides some concluding remarks.

II. Mining Software Change Impact

We use historical co-change between artifacts to uncover evolutionary coupling. Such co-change data can, for example, be found as revisions a project's version control system [15], as fixes to a bug in an issue tracking system [16], or by instrumenting the development environment [17]. Most techniques that uncover evolutionary coupling from co-change data build on *association rule mining*, an unsupervised learning technique to discover relations between *entities* of a dataset [14].

Association rules are implications of the form $A \rightarrow B$, where A is referred to as the *antecedent*, B as the *consequent*, and A and B are disjoint sets. For example, consider the classic application of analyzing shopping cart data: if multiple transactions include bread and butter then a potential association rule is *bread* $\rightarrow$ *butter*, which can be read as *"if you buy bread, then you are likely to buy butter."*

While mining evolutionary coupling from co-change data, we consider the entities to be the files of the system[1] and the collection (history) $\mathcal{T}$ of transactions, to be a list of past *commits* from a versioning system. More specifically, a transaction $T \in \mathcal{T}$ is the set of files that were either changed or added while addressing a given bug fix or feature addition, hence creating a *logical dependence* between the files [18].

As originally defined [14], association rule mining generates rules that express patterns in the complete data set. However, some applications can exploit a more focused set of rules. *Targeted association rule mining* [19] focuses the generation of rules by applying a constraint. An example constraint specifies that the antecedent of all mined rules belongs to a particular set of files, which effectively reduces the number of rules that need to be created. This reduction drastically improves the execution time of rule generation [19].

When performing change impact analysis, rule generation is constrained based on a *change set*, also known as the *query*. For example, the set of modified files since the last commit. In this case, only rules with at least one changed entity in the antecedent are created. The resulting impacted files are those found in the rule consequents. Thus, the output of change impact analysis (the *impact set*) is the set of files that are historically changed alongside the elements of the change set. Given a change set $\{a, b, c\}$, the algorithms by Zimmermann [11] and Ying [12] only uncover those files that are changed together with *all* files in the query $\{a, b, c\}$. This strict constraint tends to yield a more precise impact set, but fails to produce an answer more often than not [10]. In contrast, Kagdi's algorithm [13] uncovers any file that was co-changed with a, or b, or c. This lenient constraint enables giving more answers, which are, however, potentially noisy. Finally, in earlier work we introduced TARMAQ, which reports the files that have co-changed with largest possible subset of the query, thereby dynamically balancing the precision of a complete match with the ability to give more answers [10].

[1] Other levels of granularity are possible, and our consideration of the file level is without loss of generality as our algorithms are granularity agnostic: if fine-grained co-change data is available (or computable), the same algorithms will relate methods or variables just as well as files.

III. Research Questions

It is regularly surmised in mining literature [11, 20, 21] that learning from too short or too long a history results in a suboptimal outcome, respectively because not enough knowledge about the system can be uncovered, or because outdated information introduces noise. We aim to better understand these effects via the following research questions:

RQ 1 *What influence does* history length *have on impact analysis quality?*

This is refined in the following subquestions:

RQ 1.1 *Can we identify a* lower bound *on the history length that is needed to learn enough about the system to produce acceptable impact analysis results?*

RQ 1.2 *Do we see a* diminishing return *in impact analysis quality as history length increases?*

RQ 1.3 *Can we identify an* upper bound *on history length where outdated knowledge starts to negatively affect our analysis causing quality to decrease below acceptable levels?*

A closely related aspect is *history age*, which we define as the number of transactions that have occurred since the most recent transaction of the history used to conduct the analysis. History age basically tells us how long a model can successfully be used to make predictions regarding a system. Knowledge about the quality of impact analysis based on older histories gives valuable input regarding the feasibility of an incremental approach that reuses older association rules.

RQ 2 *What influence does* history age *have on impact analysis quality?*

Which is refined using the following subquestions:

RQ 2.1 *Can we identify an* upper bound *on the history age beyond which the generated model has grown too old and can no longer produce acceptable impact analysis results?*

RQ 2.2 *Is there a point where impact analysis quality* ceases to deteriorate *as history age increases?*

Finally, we investigate the possibility of providing project-specific advise for values of history length and history age:

RQ 3 *Can we predict good values for history length and age for a given software-system based on characteristics of its change-history (such as the average size of transactions and the number of developers)?*

To ensure a complete understanding, we will initially investigate the effects of history length and age at a *coarse* level, and progressively zoom in at *finer* levels of granularity for areas of interest indicated by the coarse study.

IV. Empirical Study

We perform a comprehensive empirical study to assess the effects of history length and age on the quality of change impact analysis through mined evolutionary coupling. As a reference mining technique we use our previous work, TARMAQ, which has proven to perform consistently better than the state-of-the-art alternatives for change impact analysis [10]. The goal of our study is to answer the research questions introduced

in Section III by controlling the history length and age while mining change impact on several large software systems.

The remainder of this section details the design of our empirical study and is organized as follows: in Section IV-A we introduce the software systems included in the study. Section IV-B describes the strategy we use to systematically vary history length and age. In Sections IV-C to IV-E we describe how we use targeted association rule mining to generate change impact sets for a change set (referred to as the *query*) of files. Finally, in Section IV-F we introduce the two measures used to evaluate the quality of the change impact sets generated via targeted association rule mining.

A. Subject Systems

To assess targeted association rule mining in a variety of conditions, we selected 19 large systems having varying characteristics, such as size and frequency of transactions, number of files, and number of developers. Two of these systems come from our industry partners, Cisco Norway and Kongsberg Maritime (KM). Cisco Norway is the Norwegian division of Cisco Systems, a worldwide leader in the production of networking equipment. We consider their software product line for professional video conferencing systems, developed by Cisco Norway. KM is a leader in the production of systems for positioning, surveying, navigation, and automation of merchant vessels and offshore installations. We consider a common software platform KM uses across various systems in the maritime and energy domain.

The other 17 systems are well known open-source projects, and are reported in Table I along demographics showing their diversity. The table shows that the systems vary from medium to large size, with up to 300 000 different files for one system, and nearly 3 500 developers contributing to another. For each system, we extracted the 50 000 most recent transactions (*commits*). This number of transactions covers vastly different time spans across the systems, ranging from almost 20 years in the case of HTTPD, to a little over 10 months in the case of the Linux kernel. We also report on six characteristics that we expect may be useful for answering RQ3.

1) *Number of Files* in the history of the system;
2) *Average Commit Size:* the average number of files appearing in a transaction in the history of a system;
3) *Number of Developers* who committed at least one transaction in the history of a system;
4) *Mode and Median Inter-Commit Time:* the inter-commit time is the time between two commits by the same developer, measured as a number of commits;
5) *Average and Median Commit Streaks:* a commit streak is the number of consecutive commits by the same developer in the history of a system.

Finally, the last column shows the programming languages used in each system, as an indication of heterogeneity.

B. History Length and Age

Given that the time span covered by 50 000 commits varies considerably across the systems in our study, we choose to express history length and age as a *number of transactions*, rather than using calendar time. This sets the same baseline for each system, enabling a meaningful comparison of the effects of history length and age across systems.

We refer to a fixed combination of history length and age as a *scenario*. In our coarse-grained study, we examine 24 scenarios pairing history lengths of 5 000, 15 000, 25 000, and 35 000 commits with ages of zero (no age), 1 000, 2 000, 3 000, 4 000 and 5 000 commits.

Preliminary results of the coarse-grained study highlighted large variations in change impact analysis quality for small length and age values, showing that the quality rapidly decreases with history aging, while increasing with longer histories. To zoom in on these areas, we conduct two additional fine-grained studies in which we respectively investigate small history lengths (for a fixed age of zero), and small ages (for a fixed history length of 25 000 commits). In each of the studies, we examine three intervals of progressively finer granularity for the variable of interest: (a) from 0 to 2 000 commits by every 100 commits; (b) from 0 to 200 commits by every 10 commits; (c) from 0 to 20 commits by every single commit. Note that we skip history lengths of zero commits because, trivially, no association rules can be mined from an empty history Thus, we consider 60 scenarios for small history lengths and age zero, and 63 scenarios for small ages and a history of 25 000 commits. We refer to the fine-grained collections of scenarios characterized by each of these ranges as *lengthX* and *ageX*, where *length* and *age* specify the context where the range is used, and *X* specifies the upper bound of the range. For example, *age20* represents the fine-grained collection containing the scenarios with history length 25 000 and age in $[0, 1, 2, \ldots 20]$.

Finally, to investigate fine-grained variations on a larger scale, we consider the collection *length35k*, which varies history length from 100 to 35 000 commits in steps of 100 commits. We do not consider a similar interval for history age, as the preliminary coarse-grained study did not show significant variations in change impact analysis quality for age values larger than about 2 000 commits.

C. Transaction Filtering

It is a common practice in association rule mining to filter the history to remove transactions larger than a certain size [11, 12, 22, 23]. Filtering reduces noise by removing large transactions that are likely not relevant for evolutionary coupling, such as mass license updates or version bumps.

In previous work, we considered the effect of transaction filtering size on the quality of change impact analysis using association rule mining [24]. That study was conducted in a similar setting as this paper, and found that filtering transactions larger than eight items gave the best results. Therefore, for each scenario, we mine association rules from a filtered history containing transactions with at most eight items.

D. Query Generation and Execution

Conceptually, a *query* Q represents a set of files that a developer changed since the last synchronization with the

TABLE I
CHARACTERISTICS OF THE EVALUATED SOFTWARE SYSTEMS (BASED ON OUR EXTRACTION OF THE LAST 50 000 TRANSACTIONS FOR EACH).

Software System	Nr. of Files	Avg. Commit Size	History (in yrs)	Nr. of Devs	Mode Inter-Commit	Median Inter-Commit	Avg. Commit Streak	Median Commit Streak	Languages used*
CPython	7725	2.70	11.97	159	0	0	5.97	4	Python (53%), C (36%), 16 other (11%)
Mozilla Gecko	8665	7.16	1.08	1047	0	11	2.67	1	C++ (37%), C (17%), JavaScript (21%), 34 other (25%)
Git	3753	1.96	11.02	1404	0	0	2.22	1	C (45%), shell script (35%), Perl (9%), 14 other (11%)
Apache Hadoop	2460	8.27	6.91	126	0	5	2.63	2	Java (65%), XML (31%), 10 other (4%)
HTTPD	1001	4.94	19.78	119	0	1	7.85	5	XML (56%), C (32%), Forth (8%), 19 other (4%)
IntelliJ IDEA	6269	3.83	2.6	194	0	4	2.58	1	Java (71%), Python (17%), XML (5%), 26 other (7%)
Liferay Portal	14479	8.95	0.86	212	0	2	6.25	2	Java (71%), XML (23%), 12 other (4%)
Linux Kernel	2641	2.15	0.76	3256	0	0	3.10	1	C (94%), 16 other (6%)
LLVM	2560	4.08	4.15	530	0	6	3.17	2	C++ (71%), Assembly (15%), C (10%), 16 other (6%)
MediaWiki	1225	5.43	9.88	541	0	1	1.65	1	PHP (78%), JavaScript (17%), 11 other (5%)
MySQL	4258	6.18	10.35	274	0	0	36.90	2	C++ (57%), C (18%), JavaScript (16%), 24 other (9%)
PHP	2129	4.03	10.8	471	0	0	3.35	2	C (59%), PHP (13%), XML (8%), 24 other (20%)
Ruby on Rails	1063	2.55	11.42	3497	0	1	0.99	0	Ruby (98%), 6 other (2%)
RavenDB	2924	6.55	8.59	259	0	0	2.84	1	C# (52%), JavaScript (27%), XML (16%), 12 other (5%)
Subversion	6559	2.96	13.99	91	0	1	5.95	4	C (61%), Python (19%), C++ (7%), 15 other (13%)
WebKit	28189	13.80	3.27	393	0	12	2.68	2	HTML (29%), JavaScript (30%), C++ (26%), 23 other (15%)
Wine	8234	2.53	6.56	517	0	0	3.15	1	C (97%), 16 other (3%)
Cisco Norway	6497	6.44	2.35	-	-	-	-	-	C++, C, C#, Python, Java, XML, other build/config
Kongsberg Maritime	3511	5.08	15.97	-	-	-	-	-	C++, C, XML, other build/config

* languages used by open source systems are from http://www.openhub.net, percentages and demographic data for the industrial systems are not disclosed.

version control system. Recall that the main assumption behind evolutionary coupling is that files that frequently change together are likely to depend on each other. The key idea behind our evaluation is to sample a transaction T from the history, and randomly partition it into a non-empty query Q and a non-empty *expected outcome* $E \stackrel{\text{def}}{=} T \setminus Q$. This allows us to evaluate to what extent our change impact analysis technique is able to estimate E from Q for a given scenario.

We take a representative sample of 660 transactions[2] from each system history, and for each of those transactions we randomly generate a single query. Note that, in order to generate non-empty queries, we only sample from transactions with at least two files. The resulting 660 queries are executed once for each of (a) the 24 scenarios in the coarse-grained study, (b) the 123 scenarios in the fine-grained studies, and (c) the 350 scenarios in the *length35k* study. This setup yields a total of $660 \cdot (24 + 123 + 350) = 328\,020$ data points for each of the 19 systems, where each data point is the estimated impact set for a given query (6.23 million data points in total).

E. Estimating the Impact of a Change

All queries are executed using TARMAQ, the rule mining algorithm we introduced in previous work [10]. Recall from Section II that, in the context of targeted association rule mining, executing a query Q entails the generation of a set of association rules. The *impact set* of Q is the list of consequents of the rules generated for Q, where such rules are ranked according to their *interestingness*. While a number of interestingness measures have been defined over the years, in our study we rank association rules based on *support* and *confidence* [14]. The support of a rule is the percentage of transactions in the history containing both the antecedent and the consequent of a rule. Intuitively, high support suggests that a rule is more likely to hold because there is more historical evidence for it. On the other hand, the confidence of a rule is the number of historical transactions containing both the antecedent and the consequent divided by the number of transactions only containing the antecedent. Intuitively, the higher the confidence, the higher the chance that when items in the antecedent of a rule change, the items in the consequent also change. We configure TARMAQ to rank rules based on support, breaking ties based on confidence. This strategy has been applied in several association rule mining approaches for software change impact analysis [11, 12, 22, 23]. Note that we consider only the largest interestingness score for each consequent. This means that, for the purpose of this study, we do not consider rule aggregation strategies, such as those we proposed in previous work [25].

F. Quality Measures

We empirically assess the quality of the change impact sets generated using two measures, Average Precision (AP) and Applicability.

Definition 1 (Average Precision) *Given a query Q, its impact set I_Q, and expected outcome E_Q, the average precision AP of I_Q is given by*

$$AP(I_Q) \stackrel{\text{def}}{=} \sum_{k=1}^{|I_Q|} P(k) * \triangle r(k) \tag{1}$$

where $P(k)$ is the precision *calculated on the first k items in the list (i.e., the fraction of correct files in the top k files), and $\triangle r(k)$ is the* change in recall *calculated only on the $k-1^{th}$ and k^{th} files (i.e., the number of additional correct items predicted compared to the previous rank) [26].*

As an overall performance measure for a scenario (i.e., for a given history length and age) across all the systems, we use the

[2] Sampling 660 transactions from a population of 50 000 transactions is sufficient to have a 99% confidence level with a 5% confidence interval.

Mean Average Precision (MAP) computed over all the queries executed using the scenario.

Average Precision is a standard measure commonly used in Information Retrieval to assess the extent to which a list of retrieved documents includes the relevant documents for a query. However, when using association rule mining, it is not always possible to generate such list. This can happen for example when there are no transactions in the history whose items changed at least once with an item in the query. While this scenario is unlikely for long histories, the chance of finding a previous transaction involving a file from the query decreases as the history length shortens. Therefore, we define *Applicability* as the percentage of queries for which an impact set can be generated (i.e., where the history contains transactions involving items from the query).

V. Results and Discussion

This section presents the results of the coarse-grained and fine-grained analysis of history length and age described in Section IV. In particular, the results of the coarse-grained study motivate two fine-grained studies: (1) the impact of history length at the fixed age zero, and (2) the impact of history age for the fixed history length $25\,000$.

Note that one challenge that shorter history lengths bring is a higher likelihood that for a given query no other commit from the history includes *any* files from the query. In such cases TARMAQ is not applicable as it is unable to generate an impact set. While it is possible to assign an AP of zero to such cases, doing so is *harsh* because TARMAQ can correctly inform the user that it is not applicable. From a user perspective, this is substantially better than an incorrect impact set (where AP is truly zero). To account for this, we report three things: *applicability*, *MAP for applicable queries*, the value of MAP for only applicable queries, and *overall MAP*, the value of MAP computed using all queries including those to which TARMAQ is not applicable. In the following, we perform statistical analysis on the MAP for applicable queries (simply referred to as *MAP* in the tables).

A. Coarse-grained Study

Table II presents the results of an ANOVA explaining the MAP values using the data of the coarse-grained study. In addition to history length and age, we include as explanatory variables their interaction (*age : history length*) and the subject system. The latter explanatory variable allows the statistical model to account for inter-system variations.

TABLE II
ANOVA results for the coarse-grained study

Explanatory Variable	F-value	p-value
age	611.36	< 0.0001
history length	178.61	< 0.0001
subject system	929.64	< 0.0001
age : history length	4.44	< 0.0001

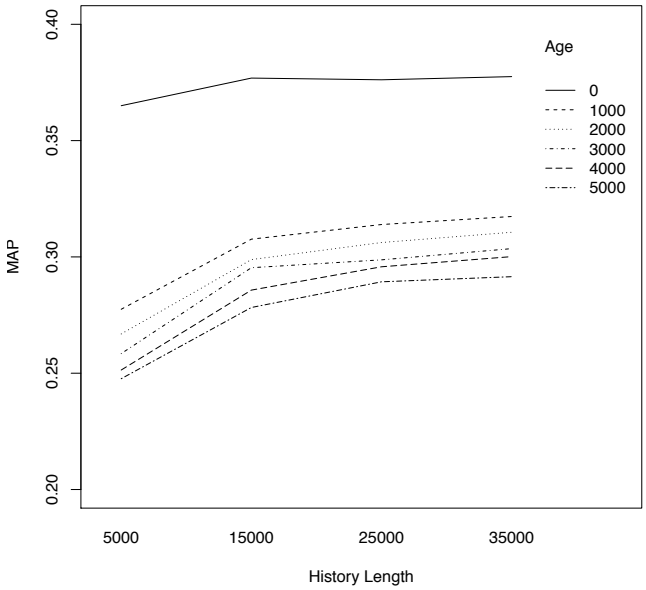


Fig. 1. Interaction plot of Age by History Length.

With extremely small p-values and F-Values substantially larger than one, all four explanatory variables are highly statistically significant. The residuals (not shown) are reasonably normal with slightly elongated tails. Recall from Section IV-D that we collect over $300\,000$ data points for each subject system. This large sample size is sufficient to overcome the minor difference from a normal distribution, especially considering ANOVA's resilience to non-normality given a large data set. As a precaution, Kruskal-Wallis' nonparametric test was used to confirm the significance findings of the ANOVA. Of the four variables, the interaction has the weakest impact, as shown the relatively small F-value. This means that there is a small variation in the impact of history length depending on the age, which is visible as the different slopes of the lines in the interaction plot shown in Figure 1. This graph also shows that the MAP values for age zero are considerably higher than those of the other ages, which are bunched relatively close together. The gap going from age zero to age 1000 is the motivation for the fine-grained study zooming in on the smaller ages.

Table III reports Tukey's Honest Significant Difference (HSD) test for the ANOVA of Table II applied to history length and age. Tukey's test partitions values of history length and age in groups in such a way that values belonging to the same group do not yield statistically significantly different

TABLE III
Tukey's HSD for History Length and Age

History length		
Length	MAP	Group
35000	0.322	a
25000	0.318	a
15000	0.312	b
5000	0.285	c

History Age		
Age	MAP	Group
0	0.372	a
1000	0.298	b
2000	0.289	c
3000	0.283	cd
4000	0.276	de
5000	0.270	e

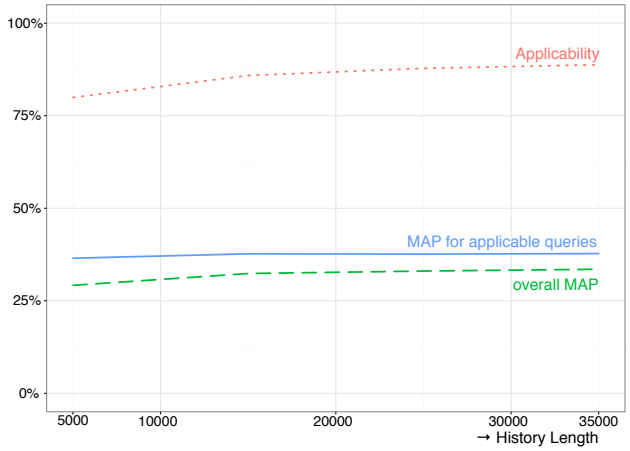


Fig. 2. Coarse-level study of history length

MAP values. The test suggests three main conclusions. First, there are significant differences between many of the levels of each variable. Second, for history length, the best performance is attained by the two larger length values, 35 000 and 25 000. We use the shorter of these, 25 000, in the fine-grained age study (Section V-B2) because it allows us to sample queries from the larger space of 25 000 commits out of the 50 000 extracted for each subject system (Section IV-A). Finally, for history age, the best performance is attained by age zero, which we hence use in the fine-grained study of history length (Section V-B1). Both the graphs and Tukey's HSD indicate that *very recent commits have a strong influence on the ability to predict change impacts*. These three findings motivated our fine-grained study of small history length and age.

Focusing on the age zero data only, Figure 2 shows the applicability of TARMAQ, along with the overall MAP and the MAP for applicable queries. The applicability of TARMAQ follows the expected trend: the algorithm grows more applicable as the history length increases. The results of the coarse-grained study also suggest that the overall MAP and the MAP for applicable queries are very similar. However, intuition suggests that the difference between the two will increase as the history length shortens.

B. Fine-grained Study

The coarse-grained analysis motivates the study of small history lengths and ages. The results of these studies are presented and discussed in the following two subsections.

1) History Length: An ANOVA for the fine-grained study of history lengths using age zero finds substantially the same results as the coarse-grained analysis. Table IV shows the results of Tukey's HSD for the scenario collections *length2000*, *length200*, and *length20*. The results show that statistically significantly higher MAP values are produced by short histories. In particular, in the first column the scenario with a history length of 100 yields statistically significantly higher MAP than all the other scenarios. The same trend is seen in the second column, where longer history yields a strictly decreasing MAP value. Finally, the third column shows that histories of lengths as short as 1 or 2 commits are very effective in estimating the change impact set, while longer histories up to 20 commits are progressively less effective. Because each column represents a separate sample of the commits, the MAP values in each column should not be directly compared, only the trends are comparable.

TABLE IV
TUKEY'S HSD FOR *length2000*, *length200*, AND *length20*

length2000			*length200*			*length20*		
Len	MAP	Grp	Len	MAP	Group	Len	MAP	Group
100	0.408	a	10	0.481	a	1	0.601	a
200	0.389	b	20	0.448	b	2	0.571	a
300	0.382	bc	30	0.424	c	3	0.540	b
400	0.379	bc	40	0.413	cd	4	0.530	b
600	0.376	bc	50	0.407	cde	5	0.516	bc
500	0.375	bc	60	0.400	def	6	0.505	cd
700	0.373	c	70	0.399	defg	7	0.498	cde
1000	0.373	c	80	0.396	defgh	8	0.489	def
2000	0.372	c	90	0.392	efghi	9	0.483	defg
800	0.372	c	100	0.388	fghij	10	0.476	efgh
900	0.372	c	110	0.385	fghij	11	0.472	fghi
1100	0.371	c	120	0.384	fghij	12	0.470	fghi
1600	0.371	c	130	0.384	ghij	13	0.468	fghi
1700	0.371	c	140	0.382	hij	14	0.465	ghi
1900	0.370	c	150	0.380	ij	15	0.464	ghi
1500	0.370	c	160	0.379	ij	16	0.459	hi
1400	0.370	c	170	0.377	ij	17	0.458	hi
1300	0.370	c	180	0.377	ij	18	0.457	hi
1800	0.370	c	190	0.376	j	19	0.455	hi
1200	0.370	c	200	0.375	j	20	0.452	i

This data suggests that very short histories yield the best results and furthermore that the extent to which files contained in past transactions are related to the files in a query progressively decreases as history length increases. In other words older transactions are more likely to contain files unrelated to a query. However, an explanation for this seemingly odd behavior is found in the considerably lower applicability of TARMAQ as the history length shortens. This trade-off can be seen in Figure 3, which reports the applicability, MAP, and MAP for applicable queries in the three fine-grained studies on the history length. In particular, across all granularities, applicability and MAP show an increasing trend, while the MAP for applicable queries shows a decreasing trend. This is expected because, the longer the history, the higher the chance that at least one past transaction contains files related to the query, which raises applicability and (overall) MAP.

The trends shown in Figure 3 led to the collection *length35k*, which is shown in Figure 4. The analysis of this figure, combined with the results of Tukey's HSD, allow us to answer RQ1 as follows.

RQ 1 *What influence does* history length *have on impact analysis quality?*

RQ 1.1 *Can we identify a* lower bound *on the history length that is needed to learn enough about the system to produce acceptable impact analysis results?*

Given the leveling off of applicability as history length grows, our analysis suggests that 25 000 commits is the point at which there is sufficient history to learn enough about the system to produce acceptable impact analysis results. Of course those willing to tolerate lower applicability, could consider shorter histories.

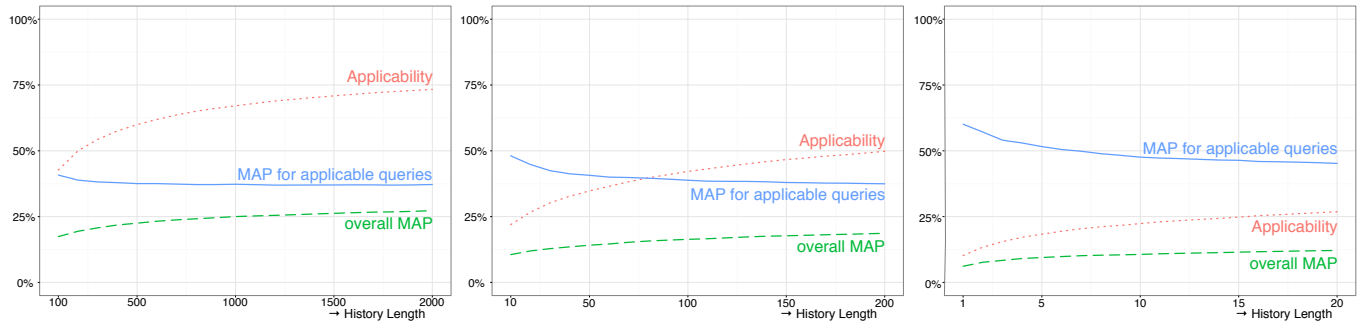


Fig. 3. The fine-grained study's three history-length scenario collections showing the inverse relation between MAP and applicability.

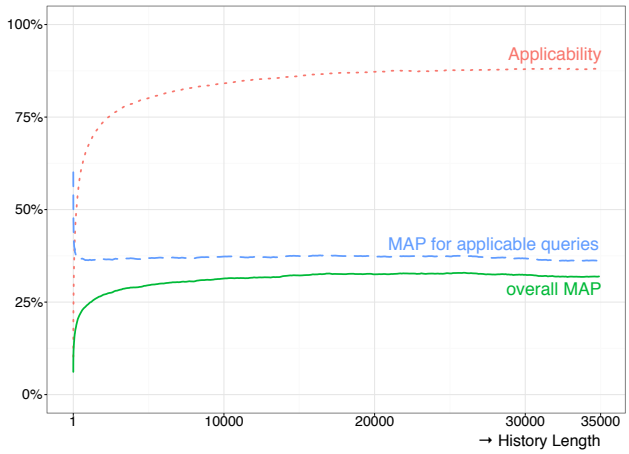


Fig. 4. Results from the large-scale fine-grained investigation of *length35k*.

RQ 1.2 *Do we see a* diminishing return *in impact analysis quality as history length increases?*

In short, Yes. Our analysis shows that TARMAQ's performance consistently increases for histories up to 15 000 commits, and then levels off and remains stable for longer histories (Figure 4). Therefore, we identify the point of diminishing return as 15 000 commits.

RQ 1.3 *Can we identify an* upper bound *on history length where outdated knowledge starts to negatively affect our analysis causing quality to decrease below acceptable levels?*

No, our analysis of histories up to 35 000 transactions does not show any evidence of performance degrading because of older outdated commits. Future work will consider histories longer than 35 000 transactions.

2) History Age: Parallel to Table II, the ANOVA for the three age collections (not shown), finds age and subject system to be highly statistically significant. Table V shows the top entries of Tukey's HSD for MAP in the collections *age2000*, *age200*, and *age20*. The remaining scenarios all appear in age order, and show considerable group overlap in the middle, similar to the center column of Table IV. Note that, in all three collections, the scenario with age zero performs statistically better than that of the next smallest age. While the gap in MAP gets smaller as the ages in the scenarios get closer, the MAP difference is significant even when going from an age of one commit to an age of four commits in the *age20* collection.

Figure 5 shows the trends for applicability, overall MAP, and MAP for applicable queries. Across the collections, the drop off from the scenario characterized by age zero to the next age is visually evident, although it clearly gets less prominent from *age2000* to *age200*, and finally to *age20*. Similar to RQ1, these charts and Tukey's HSD enable us to answer RQ2.

RQ 2 *What influence does* history age *have on impact analysis quality?*

RQ 2.1 *Can we identify an* upper bound *on the history age beyond which the generated model has grown too old and can no longer produce acceptable impact analysis results?*

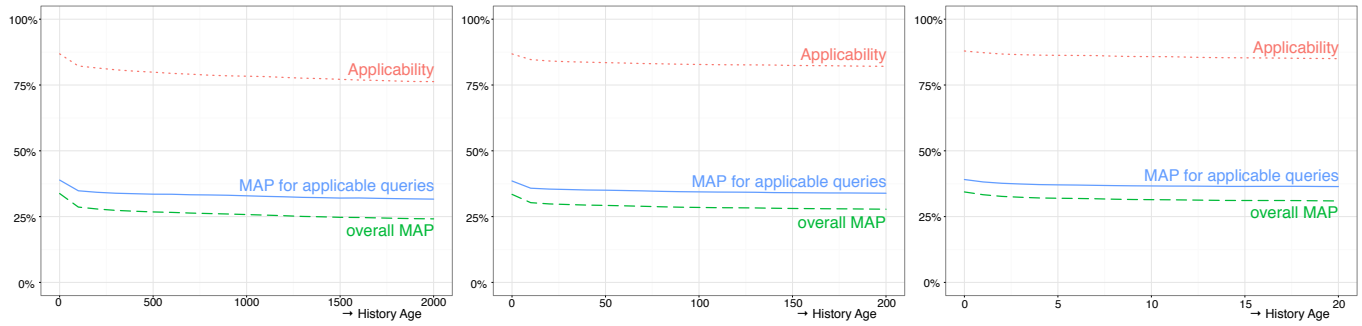
Such a bound is a function of ones tolerance for lost precision, which depends on the use and application of targeted association rule mining for change impact analysis. Similar to the length analysis (Section V-B1), the falloff in precision tends to gradually narrow. However, the falloff is initially quite steep. For example, in Table V the second entry shows a reduction of 12.5% for *age2000*, 8% for *age200* and 3% for *age20*. When using a 10% tolerance cutoff of the maximum achievable MAP as an arbitrary maximal acceptable loss, the upper bound for history age is 30 commits. In summary, we conclude there is a bound on age for RQ 2.1, where the actual value for this bound is a function of the user's tolerance and experience.

RQ 2.2 *Is there a point where impact analysis quality* ceases to deteriorate *as history age increases?*

Similar to RQ 2.1, the point of diminishing deterioration is subjective, as it depends on the cutoff for the MAP values. Following RQ 2.1, we again use a 10% cutoff. The lowest MAP is 0.270, which is for age 5000 as shown in Table III. Using this value, the target MAP value is $0.270 + 10\% = 0.297$, which, according to the study of *age2000* (not shown in Table V), is crossed at age 1200. Thus, while the performance is monotonically decreasing as age increases, it does reach a point after which the remaining deterioration is not significant.

TABLE V
TOP ENTRIES (THROUGH THE FIRST 'c') FROM TUKEY'S HSD FOR *age2000*, *age200*, AND *age20*

age2000			*age200*			*age20*		
Age	MAP	Grp	Age	MAP	Group	Age	MAP	Group
0	0.376	a	0	0.379	a	0	0.384	a
100	0.329	b	10	0.348	b	1	0.371	b
200	0.322	c	20	0.344	bc	2	0.365	bc



age2000 *age200* *age20*

Fig. 5. The fine-grained study's three history-age ranges shown from coarsest to finest

Therefore, it is possible to find a point beyond which impact analysis quality ceases to deteriorate significantly as history age increases.

C. Project Characteristics

RQ3 aims to support a team of developers working on a specific system by providing practical guidelines for selecting an appropriate value for history length and for predicting at what age a model has sufficiently deteriorated to need rebuilding. To answer this research question, we predict the values of history length and age by building two separate linear regression models. In both cases, the set of explanatory variables used are the system demographics reported in Table I. In this way, the resulting regression models characterize aspects of a system that affect change impact analysis using evolutionary coupling. In both analyses, we omit the three systems Hadoop, which has insufficient history, and Cisco and KM for which we lack complete demographics. Similar to the analysis in Section V-A, the regression analysis for history length is performed using an age of zero, while the regression analysis for age is performed with a history length of 25 000.

First, we consider the linear regression model for history length, for which we compute the highest MAP for applicable queries for each system. Note that, in most cases, several history lengths produce MAP values that show no statistical difference. To simplify our analysis, rather than modeling a set of MAP values, we use the single history length yielding the highest MAP value as the explanatory variable in the model.

We then fit a linear model to the data using R's *lm* function starting with all explanatory variables and applying backward elimination. The elimination phase removes the least statistically significant term, and then regenerates a new model. For example, in the initial model constructed the median inter-commit time has the highest p-value of 0.63. The elimination step removes this variable and then rebuilds the model. This process is repeated until only significant variables remain. The final model for history length is shown in Table VI, which yields the following prediction:

$$\begin{aligned} \textit{History Length} &= 33974.1 \\ &+ 208.4 \times \textit{Number of Files (in 1000's)} \\ &- 3958.9 \times \textit{Average Commit Size} \end{aligned}$$

TABLE VI
FINAL LINEAR REGRESSION MODEL PREDICTING HISTORY LENGTH

Explanatory Variable	Estimate	p-value
Intercept	33974.1	< 0.001
Number of Files (in 1000's)	208.4	0.012
Average Commit Size	-3958.9	0.034
Model p-value: 0.0319		

It would be interesting to understand *why* the other explanatory variables have no significant impact. Unfortunately statistical modeling fails to uncover causation. We test the reasonableness of this model using the two systems with the *smallest* and the *largest* demographic values, namely Git and WebKit (other systems fall in between these two). For Git, the model recommends a history length of 26 902, while for WebKit, the model recommends a history length of 37 414. These values are consistent with our analysis in Section V-A, which finds that lengths of 25 000 and 35 000 are the best performing.

Finally, it is interesting to consider likely causal relations that connect the explanatory variables found in the final model to the response variable (i.e., history length). First, we find that as the number of files increases, so does the predicted history length. Indeed, the more files a system has, the further apart commits containing files relevant to a query are likely to be, and hence a larger history is required. However, commits are not uniformly spaced because engineers tend to work on different areas of the code at different points in time, and thus this effect is not as strong as it might be. Nonetheless, the effect has a statistically and practically significant impact. Second, the coefficient estimate associated with average commit size, is negative, thus indicating an inverse relationship. In other words, as the average commit size increases, the required history length decreases. A likely explanation for this relation is that larger commits include more information per commit. For example, to conclude that changing a impacts b and c can be derived from the single size-three commit $\{a, b, c\}$, but requires two size-two commits: $\{a, b\}$ and $\{a, c\}$.

The second part of RQ3 looks at predicting how old the history can grow before it needs to be updated with the latest commits. In order to do so, we first define a set of MAP levels indicating that the history is too old. Specifically, we define the four MAP levels of 99%, 95%, 90% and 80% of the maximum MAP for applicable queries that can be achieved over the different history ages. We perform linear regression with backward elimination, using each of these percentages

separately as the response variables, and the demographic information as explanatory variables. For all four response variables, the elimination process removes all the explanatory variables, failing to produce a regression model. This indicates that variations in the demographic variables are not effective predictors of the rate at which a model deteriorates.

In summary, this analysis allows us to answer RQ3:

RQ 3 *Can we predict good values for history length and age for a given software-system based on characteristics of its change-history (such as the average size of transactions and the number of developers)?*

Good values for history length can be predicted. Indeed, a team of developers can use the regression model in Table VI to predict the amount of system's history that should be used. In contrast, no similar correlation exists for predicting the deterioration of change impact analysis quality based on history age. In an attempt to better predict age deterioration, future work will consider additional demographic information such as some measure of the development process.

VI. Related Work

Software repository mining literature [11, 20, 21] frequently alludes to the notion that learning from a too short, or an overly long history harms the outcome, either because not enough knowledge can be uncovered, or because outdated information introduces noise. However, except for some smaller experiments by Zimmermann [11], the impact of these effects has not been systematically investigated.

Similarly, authors in the field of association rule mining have stated the need to investigate sensitivity to algorithm parameters (e.g., transaction filter size used, choice of interestingness measure) [27, 28, 29, 30], but we have not found work that discusses sensitivity to the number of transactions used for mining (i.e., our history length), or to aging of transactions.

Parameters in Mining Change Impact: In the context of software change impact analysis, several studies remark on the importance of discarding from the history large change sets which are likely to contain unrelated files. For example, Kagdi et al. [23], Zimmermann et al. [11] and Ying et al. [12] propose to filter out transactions larger than 10, 30, and 100 items, respectively. However, none of this work reports how the threshold was chosen nor does it discuss the impact of different values on recommendation quality. In previous work [24], we systematically explored the effect of filtering size on the quality of change impact analysis, and found that filtering transactions larger than eight items yields the best result for similar systems as considered in this paper.

Characteristics of the Change History: Over the years, several studies proposed strategies to group transactions in the revision history of software projects [11, 13, 31]. The reason for doing so is that a developer might accidentally commit an incomplete transaction, and modify the remaining files related to the same change in a subsequent transaction. As a consequence, a single change set might be scattered across several transactions in the change history. Nevertheless, in modern version control systems, transactions are stashed in the user local repository and finalized at a later stage, reducing the risk of committing incomplete transactions.

In contrast, whether properties such as average commit size and frequency affect the quality of software recommendations is a relatively less studied subject. In this direction, German carried out an empirical study on several open source projects, finding that the revision history of most systems contains mostly small commits [32]. Alali et al. also investigated the total number of lines modified in the files, and the total number of hunks with line changes [33]. Kolassa et al. performed a similar study on commit frequency, reporting an average inter-commit time around three days [34]. However, none of these studies investigates how characteristics of the change history affect the quality of change recommendations.

Characteristics of the Change Set: Targeted association rule mining approaches drive the generation of rules by a *query* supplied by the user [19]. In general, characteristics of the query can effect the precision of recommendations. For example, Rolfsnes et al. found a particular class of queries, strongly related to query size, for which the most common targeted association rule mining approaches cannot generate recommendations [10]. In other work, Hassan and Holt investigated the effectiveness of evolutionary coupling in predicting change propagation effects resulting from source code changes, but did not evaluate whether the size of transactions in the history affects the quality of the predictions generated [7].

Aged histories in evaluation: For the purpose of evaluating change impact analysis or change recommendation techniques, it is common practice to split the change history into training and test sets. The training set can either be treated as a static prediction model [12], or be continuously updated with respect to the chosen transaction from the test set [11]. If treated as a static model this means that the model will be aged differently with respect to each transactions in the test set, and as we have seen in RQ2 aging affects impact analysis quality. Therefore, any study involving history splitting should take aging into consideration. Since we have seen that aging can only lead to deterioration of impact analysis quality, we suggest evaluation setups where the prediction model is updated for every transaction in the test set (i.e., an age of zero).

VII. Concluding Remarks

This paper presents a systematic study of the effects of history length and age on 19 different software systems. Key findings include that as history length increases, the MAP also increases, although this increase diminishes around 15 000 commits. Moreover, the applicability also increases with increased history length, but seems to top out around 15 000 to 25 000 commits. We found that the impact of age on MAP is very significant, as even very little aging yields a strong, basically exponential decrease.

Finally, in addition to the study providing a better understanding of the impact of history length and age on the quality of change impact analysis, we also derive a prediction

model for the length of the history that should be used with a given system. This prediction model is a function of system demographics, specifically the average commit size and the number of files.

Looking forward, we did not find any evidence of the folkloric belief that large histories contain outdated information that degrades change impact analysis quality. Thus, as part of future work, we plan to investigate significantly larger histories. For example, the complete development history of the Linux kernel includes over 650 000 commits. Other possibilities include studying the impact of aggregation [25] on the recommendations and considering the impact of history length and age on alternative algorithms [11, 12, 23] where we expect the impact to be more dramatic because these algorithms are more restricted in their use of the history. Finally, based on our findings on the impact of history age, we plan to experiment with alternative strategies for association rules generation that take age into account, for instance by assigning higher weight to more recent transactions.

Acknowledgement: This work is supported by the Research Council of Norway through the EvolveIT project (#221751/F20) and the Certus SFI (#203461/030). Dr. Binkley is supported by NSF grant IIA-1360707 and a J. William Fulbright award.

References

[1] S. Bohner and R. Arnold. *Software Change Impact Analysis*. CA, USA: IEEE, 1996.

[2] J. Law and G. Rothermel. "Whole Program Path-Based Dynamic Impact Analysis". In: *Int'l Conf. Softw. Engineering*. IEEE, 2003, pp. 308–318.

[3] X. Ren et al. "Chianti: a tool for change impact analysis of java programs". In: *Conf. Object-oriented Programming, Systems, Languages, and Applications*. 2004, pp. 432–448.

[4] M.-A. Jashki, R. Zafarani, and E. Bagheri. "Towards a more efficient static software change impact analysis method". In: *Ws. Program Analysis for Softw. Tools and Engineering (PASTE)*. ACM, 2008, pp. 84–90.

[5] A. R. Yazdanshenas and L. Moonen. "Crossing the boundaries while analyzing heterogeneous component-based software systems". In: *Int'l Conf. Softw. Maintenance*. IEEE, 2011, pp. 193–202.

[6] A. Podgurski and L. Clarke. "A formal model of program dependences and its implications for software testing, debugging, and maintenance". In: *IEEE TSE* 16.9 (1990), pp. 965–979.

[7] A. Hassan and R. Holt. "Predicting change propagation in software systems". In: *Int'l Conf. Softw. Maintenance*. IEEE, 2004, pp. 284–293.

[8] G. Canfora and L. Cerulo. "Impact Analysis by Mining Software and Change Request Repositories". In: *Int'l Softw. Metrics Symp.* IEEE, 2005, pp. 29–37.

[9] M. B. Zanjani, G. Swartzendruber, and H. Kagdi. "Impact analysis of change requests on source code based on interaction and commit histories". In: *Int'l Working Conf. Mining Softw. Repositories*. 2014, pp. 162–171.

[10] T. Rolfsnes et al. "Generalizing the Analysis of Evolutionary Coupling for Software Change Impact Analysis". In: *Int'l Conf. Softw. Analysis, Evolution, and Reengineering*. IEEE, 2016, pp. 201–212.

[11] T. Zimmermann et al. "Mining version histories to guide software changes". In: *IEEE TSE* 31.6 (2005), pp. 429–445.

[12] A. Ying et al. "Predicting source code changes by mining change history". In: *IEEE TSE* 30.9 (2004), pp. 574–586.

[13] H. Kagdi, S. Yusuf, and J. I. Maletic. "Mining sequences of changed-files from version histories". In: *Int'l Ws. Mining Softw. Repositories*. ACM, 2006, pp. 47–53.

[14] R. Agrawal, T. Imielinski, and A. Swami. "Mining association rules between sets of items in large databases". In: *Int'l Conf. Management of Data*. ACM, 1993, pp. 207–216.

[15] S. Eick et al. "Does code decay? Assessing the evidence from change management data". In: *IEEE TSE* 27.1 (2001), pp. 1–12.

[16] M. Gethers et al. "An adaptive approach to impact analysis from change requests to source code". In: *Int'l Conf. Automated Softw. Engineering*. IEEE, 2011, pp. 540–543.

[17] R. Robbes, D. Pollet, and M. Lanza. "Logical Coupling Based on Fine-Grained Change Information". In: *Working Conf. Reverse Engineering*. IEEE, 2008, pp. 42–46.

[18] H. Gall, K. Hajek, and M. Jazayeri. "Detection of logical coupling based on product release history". In: *Int'l Conf. Softw. Maintenance*. IEEE, 1998, pp. 190–198.

[19] R. Srikant, Q. Vu, and R. Agrawal. "Mining Association Rules with Item Constraints". In: *Int'l Conf. Knowledge Discovery and Data Mining (KDD)*. AASI, 1997, pp. 67–73.

[20] T. Graves et al. "Predicting fault incidence using software change history". In: *IEEE TSE* 26.7 (2000), pp. 653–661.

[21] A. E. Hassan. "The road ahead for Mining Software Repositories". In: *Frontiers of Softw. Maintenance*. IEEE, 2008, pp. 48–57.

[22] A. Alali. "An Empirical Characterization of Commits in Software Repositories". Ms.c. Kent State University, 2008, p. 53.

[23] H. Kagdi, M. Gethers, and D. Poshyvanyk. "Integrating conceptual and logical couplings for change impact analysis in software". In: *Empirical Software Engineering* 18.5 (2013), pp. 933–969.

[24] L. Moonen et al. "Practical Guidelines for Change Recommendation using Association Rule Mining". In: *Int'l Conf. Automated Softw. Engineering*. Singapore: IEEE, 2016.

[25] T. Rolfsnes et al. "Improving change recommendation using aggregated association rules". In: *Int'l Conf. Mining Softw. Repositories*. ACM, 2016, pp. 73–84.

[26] R. (Baeza-Yates and B. Ribeiro-Neto. *Modern information retrieval*. ACM, 1999, p. 513.

[27] Z. Zheng, R. Kohavi, and L. Mason. "Real world performance of association rule algorithms". In: *Int'l Conf. Knowledge discovery and data mining (KDD)*. ACM, 2001, pp. 401–406.

[28] W. Lin, S. A. Alvarez, and C. Ruiz. "No Title". In: *Data Mining and Knowledge Discovery* 6.1 (2002), pp. 83–105.

[29] N. Jiang and L. Gruenwald. "Research issues in data stream association rule mining". In: *ACM SIGMOD Record* 35.1 (2006), pp. 14–19.

[30] O. Maimon and L. Rokach. *Data Mining and Knowledge Discovery Handbook*. Ed. by O. Maimon and L. Rokach. Springer, 2010, p. 1383.

[31] F. Jaafar et al. "Detecting asynchrony and dephase change patterns by mining software repositories". In: *Journal of Software: Evolution and Process* 26.1 (2014), pp. 77–106.

[32] D. M. German. "An empirical study of fine-grained software modifications". In: *Empirical Software Engineering* 11.3 (2006), pp. 369–393.

[33] A. Alali, H. Kagdi, and J. Maletic. "What's a Typical Commit? A Characterization of Open Source Software Repositories". In: *Int'l Conf. Program Comprehension*. IEEE, 2008, pp. 182–191.

[34] C. Kolassa, D. Riehle, and M. A. Salim. "The empirical commit frequency distribution of open source projects". In: *Int'l Symp. Open Collaboration (WikiSym)*. ACM, 2013, pp. 1–8.