Empir Software Eng (2018) 23:2362–2397
https://doi.org/10.1007/s10664-017-9588-z

What are the effects of history length and age on mining software change impact?

Leon Moonen[1] · **Thomas Rolfsnes**[1] · **Dave Binkley**[2] · **Stefano Di Alesio**[1]

Published online: 6 March 2018

Abstract The goal of Software Change Impact Analysis is to identify artifacts (typically source-code files or individual methods therein) potentially affected by a change. Recently, there has been 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 coupling. Specifically, we report on a series of systematic experiments using three state-of-the-art mining algorithms that involve 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

Communicated by: Gabriele Bavota and Michaela Greiler

✉ Leon Moonen
leon.moonen@computer.org

Thomas Rolfsnes
thomgrol@simula.no

Dave Binkley
binkley@cs.loyola.edu

Stefano Di Alesio
stefano@simula.no

[1] Simula Research Laboratory, Oslo, Norway

[2] Loyola University Maryland, Baltimore, MD, USA

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.

Keywords Change impact analysis · Evolutionary coupling · Association rule mining · Parameter tuning

1 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. Aimed at identifying software artifacts (e.g., files, methods, classes) affected by a given change, Change Impact Analysis (Bohner and Arnold 1996) has been proposed as a solution to this problem. Traditionally, techniques for change impact analysis use static or dynamic analysis to identify dependencies, for example, methods calling or called by a changed method (Law and Rothermel 2003; Ren et al. 2004; Jashki et al. 2008). However, static and dynamic analysis are generally language-specific, making them hard to apply to modern heterogeneous software systems (Yazdanshenas and Moonen 2011). In addition, dynamic analysis can involve considerable overhead (e.g., from code instrumentation), while static analysis tends to over-approximate the impact of a change (Podgurski and Clarke 1990).

To address these challenges, alternative techniques have been proposed that identify dependencies through *evolutionary coupling* (Hassan and Holt 2004; Canfora and Cerulo 2005; Zanjani et al. 2014; Rolfsnes et al. 2016a). In essence, evolutionary coupling exploits a developer's inherent knowledge of the dependencies in the system, which manifest themselves through commit comments, bug reports, context switches in IDEs, and so on (Canfora and Cerulo 2005). 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 (Zimmermann et al. 2005; Ying et al. 2004; Kagdi et al. 2006; Rolfsnes et al. 2016a) for mining evolutionary couplings from artifact co-changes build on *association rule mining* (or *association rule learning*) (Agrawal et al. 1993), and differ in the way in which association rules are generated from the history. Nevertheless, key to all such techniques is the *history* learned 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. Thus with history length we explore how much history is considered, while with history age we explore how "out of date" the selected history has become. 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 builds upon our previous work that explored the extent to which history length and age affect the quality of software change impact analysis via association rule mining (Moonen et al. 2016a). We present a series of systematic experiments using

the change histories of two large industrial systems and 17 large open source systems. In particular, this paper extends our previous work in the following key respects: (1) We extend our analysis to include two more state-of-the-art software change mining techniques, CO-CHANGE and ROSE, in addition to the TARMAQ technique covered in our previous work. (2) We refine the change histories used in the analysis from a coarse-grained file-level to a more *practical fine-grained level* consisting of method-level change information for all parseable source-code, and file-level change information for any remaining un-parseable files. (3) We include an additional longitudinal study that considers change histories covering the *complete evolution history* of seven open source systems selected because their histories are significantly longer (up to 540 000 transactions) than the histories considered earlier. (4) We include a new research question that investigates the stability of impact analysis quality throughout the evolution history given a particular history length and history age. (5) We strengthen the power of the statistical analysis used to address all our research questions. (6) Finally, we extend our discussion of background material, threats to validity, and related work. We use the results from these investigations to 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 their specific context.

Overview Section 2 provides background on mining evolutionary coupling. Section 3 presents our research questions. Section 4 describes the setup of our empirical investigation, whose results are presented in Section 5. Finally, Section 6 discusses the threats to validity, Section 7 presents related work, and Section 8 provides some concluding remarks.

2 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 in a project's version control system (Eick et al. 2001), as fixes to a bug in an issue tracking system (Gethers et al. 2011), or by instrumenting the development environment (Robbes et al. 2008). Most techniques that uncover evolutionary coupling from co-change data build on *association rule mining*, an unsupervised learning technique that discovers relations between artifacts (referred to as *items* in the more general context) of a dataset (Agrawal et al. 1993).

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 historical co-change data we consider two artifacts: first, the files and methods of a system[1] and, second, the sequence (history) of transactions, $\mathscr{T}$. This history is made up of *commits* to the versioning system, where a

[1]Note that various granularity choices are possible since the 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 more coarse-grained files. In this paper we consider a *practical* fine-grained level that uses method-level information where possible (i.e., for source files that can be parsed, as discussed later in the paper), and file-level information otherwise (e.g., for test plans, build files, and configuration files).

transaction $T \in \mathcal{T}$ is the set of artifacts that were either changed or added while addressing a given bug fix or feature addition, hence creating a *logical dependence* between them (Gall et al. 1998).

As originally defined (Agrawal et al. 1993), association rule mining generates rules that express patterns in a complete data set. However, some applications can exploit a more focused set of rules. *Targeted association rule mining* (Srikant et al. 1997) 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 (Srikant et al. 1997).

When performing change impact analysis, rule generation is constrained based on a *change set*, also known as a *query*. For example, the set of modified artifacts since the last commit. In this case, only rules with at least one changed artifact in the antecedent are created. The resulting impacted artifacts are those found in the rule consequents. Thus, the output of change impact analysis (the *impact set*) is the set of artifacts that are historically changed alongside the elements of the change set.

Only a few targeted association rule mining algorithms have been considered in the context of change impact analysis: the ROSE algorithm by Zimmermann et al. (2005), the FP-TREE algorithm by Ying et al. (2004), the CO-CHANGE algorithm by Kagdi et al. (2006), and the TARMAQ algorithm introduced in our earlier work (Rolfsnes et al. 2016a). These algorithms differ in terms of constraints on how the query is matched against the transactions of the history. For example, given a change set $\{a, b, c\}$, ROSE and FP-TREE only uncover those artifacts that ever change in the history together with *all* artifacts in the query $\{a, b, c\}$. This strict matching constraint is aimed at obtaining a more precise impact set, but an analysis of the algorithm's *applicability* showed that the constraint also prevents the algorithms from producing an answer more often than not (Rolfsnes et al. 2016a). In contrast, CO-CHANGE uncovers artifacts that ever change in the history together with *any* of a, or b, or c. This more lenient constraint is aimed at giving more answers, which are, however, potentially noisy; since the answers are only based on one matching element, they can have little relation to the full query. Finally, TARMAQ reports the artifacts that have co-changed with largest possible subset of the query, a constraint aimed at dynamically balancing the precision of a complete match with improved applicability (Rolfsnes et al. 2016a). In cases where a match of the complete query is possible, ROSE, FP-TREE, and TARMAQ will give the exact same result. In cases where only a subset of the query can be matched, ROSE and FP-TREE fail to produce an impact set, while TARMAQ provides the impact set that results from the largest possible match between the query and the change history.

3 Research Questions

It is regularly surmised in the mining literature (Graves et al. 2000; Zimmermann et al. 2005; Hassan 2008) that learning from too short or too long a history (in our case to few or two many commits) results in a suboptimal outcome, respectively because not enough knowledge about the system is uncovered, or because outdated information introduces noise. We aim to better understand the influence of *history length* via the following research question:

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

We refine RQ 1 using the following sub-questions:

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?*

We refine RQ 2 using the following sub-questions:

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 at which impact analysis quality* ceases to deteriorate *as history age increases?*

Next, we investigate the possibility of providing project-specific advise for setting the 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 transaction size and the number of developers)?*

Finally, we investigate stability by considering the sensitivity of the algorithms using a common history length and history age at different points in the system's evolution history:

RQ 4 *How is impact analysis quality throughout the evolution history affected by choosing a fixed* history length *and* history age?

Scope of Investigation 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. Moreover, based on our initial results (Moonen et al. 2016a) with respect to RQ 1.3, this paper includes an additional *longitudinal* study that considers the *complete evolution history* of seven open source systems. These systems were selected because their available change history is significantly longer than the histories available for other systems. This combination allows us to cover both the width of a substantial set of systems and the depth of a long evolution history.

4 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. To consider

the influence that the mining algorithm has on the study, we experiment with three of the four algorithms introduced in Section 2 (we omit FP-TREE because it produces the same impact sets as ROSE). The goal of our study is to answer the research questions introduced in Section 3 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: Section 4.1 introduces the software systems included in the study. Section 4.2 describes the strategy we use to systematically vary history length and age. Sections 4.3 to 4.5 describe how we use targeted association rule mining to generate change impact sets for a change set (i.e., a *query*) of artifacts. Finally, Section 4.6 introduces the two measures used to evaluate the quality of the generated change impact sets.

4.1 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 artifacts, 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. 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 all well known open-source projects.

Table 1 shows an overview of the systems, which includes various characteristics illustrating their diversity and a summary of the programming languages used as an indication of heterogeneity. For each system, we extracted the 50 000 most recent transactions (*commits*). This extraction 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.

The table shows that the systems vary from medium to large size, with almost 300 000 different files for one system, nearly 768 000 artifacts in another, and almost 3 500 developers contributing to a third. For each system we extracted the following seven demographic characteristics that we expect to be useful for answering RQ 3:

1. *Number of (unique) artifacts* appearing in the history of a system;
2. *Average Commit Size:* the average number of artifacts 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.

4.2 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.

Table 1 Characteristics of the evaluated software systems (based on our extraction of the most recent 50 000 transactions for each)

Software system	History (in yrs)	Unique # files	Languages used*
CPython	12.05	7725	Python (53%), C (36%), 16 other (11%)
Mozilla Gecko	1.08	86650	C++ (37%), C (17%), JavaScript (21%), 34 other (25%)
Git	11.02	3753	C (45%), shell script (35%), Perl (9%), 14 other (11%)
Apache Hadoop	6.91	24607	Java (65%), XML (31%), 10 other (4%)
HTTPD	19.78	10019	XML (56%), C (32%), Forth (8%), 19 other (4%)
IntelliJ IDEA	2.61	62692	Java (71%), Python (17%), XML (5%), 26 other (7%)
Liferay Portal	0.87	144792	Java (71%), XML (23%), 12 other (4%)
Linux Kernel	0.77	26412	C (94%), 16 other (6%)
LLVM	4.54	25600	C++ (71%), Assembly (15%), C (10%), 16 other (6%)
MediaWiki	11.69	12252	PHP (78%), JavaScript (17%), 11 other (5%)
MySQL	10.68	42589	C++ (57%), C (18%), JavaScript (16%), 24 other (9%)
PHP	10.82	21295	C (59%), PHP (13%), XML (8%), 24 other (20%)
Ruby on Rails	11.42	10631	Ruby (98%), 6 other (2%)
RavenDB	8.59	29245	C# (52%), JavaScript (27%), XML (16%), 12 other (5%)
Subversion	14.03	6559	C (61%), Python (19%), C++ (7%), 15 other (13%)
WebKit	3.33	281898	HTML (29%), JavaScript (30%), C++ (26%), 23 other (15%)
Wine	6.60	8234	C (97%), 16 other (3%)
Cisco Norway	2.43	64974	C++, C, C#, Python, Java, XML, other build/config
Kongsberg Maritime	15.97	35111	C++, C, XML, other build/config

Software system	Unique # artifacts	Avg. # artifacts per commit	Nr. of Devs	Mode Inter-Commit	Median Inter-Commit	Avg. Commit Streak	Median Commit Streak
CPython	30090	4.52	159	0	0	5.97	4
Mozilla Gecko	231850	12.28	1047	0	11	2.66	1
Git	17716	3.13	1404	0	0	2.22	1
Apache Hadoop	272902	47.79	126	0	5	2.63	2
HTTPD	29216	6.99	119	0	1	7.85	5
IntelliJ IDEA	343613	12.60	194	0	4	2.58	1
Liferay Portal	767955	29.90	212	0	2	6.26	2
Linux Kernel	161022	5.50	3256	0	0	3.10	1
LLVM	66604	5.91	530	0	6	3.18	2
MediaWiki	12267	3.31	541	0	1	1.65	1
MySQL	136925	10.66	274	0	0	36.90	2
PHP	53510	6.74	471	0	0	3.33	2
Ruby on Rails	10631	2.56	3497	0	1	0.99	0
RavenDB	139415	18.18	259	0	0	2.84	1
Subversion	46136	6.36	91	0	1	5.95	4

Table 1 (continued)

Software system	Unique # artifacts	Avg. # artifacts per commit	Nr. of Devs	Mode Inter-Commit	Median Inter-Commit	Avg. Commit Streak	Median Commit Streak
WebKit	397850	18.12	393	0	12	2.68	2
Wine	126177	6.68	517	0	0	3.15	1
Cisco Norway	251321	13.62	–	–	–	–	–
Kongsberg Maritime	35111	5.08	–	–	–	–	–

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

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, finding 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 35 000 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, in steps of 100 commits; (b) from 0 to 200 commits, in steps of 10 commits; (c) from 0 to 20 commits, in steps of 1 commit. Note that we omit history length zero 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 35 000 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 35 000 and age in [0, 1, 2, ... 20]. To investigate fine-grained variations on a larger scale, we also consider the collection *length35k*, which varies history length from 0 to 20 in steps of 1 commit, from 20 to 200 commits in steps of 10 commits, and then from 200 to 35 000 commits in steps of 100 commits.

We do not consider a similar large 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.

Finally, based on our initial findings (Moonen et al. 2016a), this paper adds a longitudinal study to get a more conclusive answer to RQ 1.3. In this study we analyze the *complete evolution history* of seven systems to better understand if there is an upper-bound on history length where outdated knowledge starts to negatively affect impact analysis quality. Table 2 gives an overview of the characteristics of the seven systems that were used in this investigation, which were selected because of their significantly longer evolution histories. The scenarios considered in the *longitudinal* study fix age at zero and consider history lengths from 10 000 up to the maximum available history for each system (column two of Table 2) in steps of 10 000 commits.

Table 2 Characteristics of the seven software systems selected for the longitudinal study (ordered by increasing number of available transactions for each)

Software system	Available commits	History (in yrs)	Unique # of artifacts	Avg. # artifacts per commit
Wine	110950	22.81	227223	8.02
LLVM	118967	14.88	106185	5.07
IntelliJ IDEA	159020	11.47	1083032	17.23
Liferay Portal	171507	10.01	1581784	26.95
WebKit	171604	14.64	808437	17.34
Mozilla Gecko	430127	18.07	1016343	10.87
Linux Kernel	542098	10.99	953194	6.50

4.3 History Filtering

One challenge faced by association rule mining is that large transactions lead to a combinatorial explosion in number of association rules (Agrawal et al. 1993). Fortunately, as seen in Fig. 1, which provides violin plots of transaction size for the individual systems, transaction sizes are heavily skewed toward smaller transactions. Unfortunately, as also seen in the violin plots, there exist outlier transactions containing 10 000 or more artifacts. To combat the combinatorial explosion challenge raised by such large commits, it is common to filter the history by removing transactions larger than a certain size (Ying et al. 2004; Zimmermann et al. 2005; Alali 2008; Kagdi et al. 2013). Filtering reduces noise by removing large transactions that are likely not relevant for evolutionary coupling, such as mass license updates or version bumps.

In an attempt to reflect *most* change impact analysis scenarios, we employ a quite liberal filtering and remove only those transactions larger than *300* artifacts. The rational behind choosing this cutoff is that for each program at least 99% of all transactions are smaller then 300 artifacts. In most cases, the percentage is well above 99% of the available data.

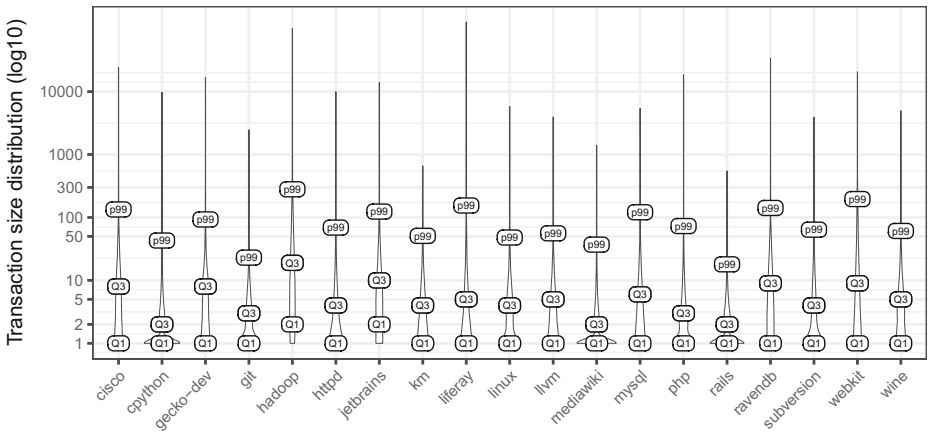

Fig. 1 An overview of the distributions of transactions sizes for each subject system

4.4 Query Generation and Execution

Conceptually, a *query* Q represents a set of artifacts that a developer changed since the last synchronization with the version control system. Recall that the main assumption behind evolutionary coupling is that artifacts 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 then 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.

From each filtered history we take a sample of 1100 recent transactions,[2] with the constraint that the transaction must contain at least two artifacts. This constraint ensures that, at the minimum, a transaction can be split into a query of at least one artifact and an expected outcome of at least one. Each of these transactions is randomly split into a non-empty query and a non-empty expected outcome. The resulting 1100 queries are executed using each of the three algorithms and each of (a) the 24 scenarios in the coarse-grained study, (b) the 123 scenarios in the fine-grained studies, and (c) the 385 scenarios in the *length35k* study. This setup yields a total of $1100 \cdot 3 \cdot (24 + 123 + 386) = 1\,758\,900$ data points for each of the 19 systems, where each data point is the estimated impact set for a given query (33.42 million data points in total). The longitudinal study adds another 151 scenarios for the seven systems considered, yielding an additional $1100 \cdot 3 \cdot 151 = 498\,300$ data points which brings the total close to 34 million. Note that in each of the scenarios, we only mine from transactions that are *older* than the transaction T used to generate the query. This is done to respect the historical time-line for the query and the transactions used to address that query.

4.5 Estimating the Impact of a Change

All queries are executed using each of the three rule mining algorithms. Recall from Section 2 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* (Agrawal et al. 1993). 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 that contain only 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 each mining algorithm to rank rules using support, breaking ties based on confidence. This strategy has been applied in several

[2]For a normally distributed population of 50 000, a minimum of 657 samples is required to attain 99% confidence with a 5% confidence interval that the sampled transactions are representative of the population. Since we do not know the distribution of transactions, we correct the sample size to the number needed for a non-parametric test to have the same ability to reject the null hypothesis. This correction is done using the Asymptotic Relative Efficiency (ARE). As AREs differ for various non-parametric tests, we choose the lowest coefficient, 0.637, yielding a conservative minimum sample size of $657/0.637 = 1032$ transactions. Hence, a sample size of 1100 is more than sufficient to attain 99% confidence with a 5% confidence interval that the samples are representative of the population.

association rule mining approaches for software change impact analysis (Ying et al. 2004; Zimmermann et al. 2005; Alali 2008; Kagdi et al. 2013). 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 (Rolfsnes et al. 2016b).

4.6 Quality Measures

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

Definition 4.1 (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 artifacts in the top k artifacts), and $\triangle r(k)$ is the *change in recall* calculated only on the $k-1$th and kth artifacts (i.e., the number of additional correct items predicted compared to the previous rank) (Baeza-Yates and Ribeiro-Neto 1999).

As an overall performance measure for a scenario (i.e., for a given history length and age) across a system, we use the Mean Average Precision (MAP) computed over all the queries executed for the given 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 a 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 an artifact 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).

4.7 Bootstrapping Procedure

The distribution of AP values is unsurprisingly highly left skewed because these values drop quickly when there is no correct artifact in the first few positions. For example, consider three ranked lists for a query whose expected outcome includes a single artifact. In the first list the correct artifact is first, in the second it is second, and in the third it is third. In this case the AP values are 1.00, 0.50 and 0.33 respectively. AP values can drop even faster if there is more than one relevant artifact. Thus most AP values reside to the left (are closer to 0.0 than to 1.0). However, recommendations also frequently are of *very high quality* meaning that many true positives are found in the very first positions in the recommendation, this results in a right skew in the AP values as well. An investigation of the *residuals* following an ANOVA fit proved that residuals were not normally distributed. The nature of our study also requires looking at various 2-way interactions, which is infeasible in certain cases when using non-parametric methods. In particular, as weights are replaced by ranks in non-parametric methods, it is not possible to investigate interactions which do not involve

sign changes. For example, in relation to our experiment an interaction where a change in *history age* has a *larger negative* effect on one subject system with regards to another can not be studied non-parametrically.

For these reasons we apply bootstrapping to approximate the sampling distribution. Doing so preserves centrality (i.e., the mean does not change), but yields an approximately normal distribution of both the AP values and more importantly of the residuals following an ANOVA.

In order to generate the sampling distribution we sample 100 AP values with replacement and calculate their mean, we do this 1000 times for each unique combination of factors. We found that a sample size of 100 best approximated a normal distribution, where lower sample sizes produced mostly unchanged AP distribution, while higher sample sizes produced a distribution only consisting of the approximate mean of the total population (resulting from the large amount of available data).

4.8 Data Set for Replication Studies

A data set consisting of both the unprocessed results and measures and their bootstrapped counterparts, for all open source systems analyzed in our study, has been archived on Zenodo with DOI: https://doi.org/10.5281/zenodo.1083964.

5 Results and Discussion

This section first presents the results of the coarse-grained and then the fine-grained analysis of history length and age as described in Section 4. In particular, the results of the coarse-grained study motivate the two fine-grained studies: (1) the impact of various history lengths at the fixed age zero and (2) the impact of various history ages for the fixed history length 35 000.

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* artifacts from the query. In such cases TARMAQ, CO-CHANGE, and ROSE are each *not applicable* as they are unable to generate an impact set. While it is possible to assign an AP of zero to such cases, doing so is *harsh* because the algorithm 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 when applicable* (the value of MAP computed using only applicable queries), and *overall MAP* (the value of MAP computed using all queries where AP is assumed to be zero when the algorithm is not applicable).

5.1 Coarse-Grained Study

Table 3 presents the results of an ANOVA explaining AP using the data of the coarse-grained study. For this initial look we include all scenarios, i.e., not only the *applicable scenarios* which will be explored later. In addition to history length and age, which are the main variables of interest, we include subject system and algorithm as explanatory variables to allow the statistical model to account for system or algorithm specific variations. We also include all two-way interactions to account for possible interaction effects.

With extremely small p-values and F-Values substantially larger than one, all four primary explanatory variables are highly statistically significant. This is also reflected by the

Table 3 ANOVA results for the coarse-grained study

Explanatory variable	Partial eta^2	Sum Sq	Df	F-value	p-value
subject system	0.76	2973.47	18	206196.51	<0.00001
algorithm	0.36	536.86	2	335061.64	<0.00001
history length	0.20	236.63	3	98453.51	<0.00001
age	0.63	1574.33	5	393022.32	<0.00001
subject system : algorithm	0.05	49.12	36	1703.01	<0.00001
subject system : history length	0.06	58.89	43	1709.41	<0.00001
subject system : age	0.18	204.58	90	2837.39	<0.00001
algorithm : history length	0.00	3.63	6	755.55	<0.00001
algorithm : age	0.01	9.79	10	1221.59	<0.00001
history length : age	0.01	5.80	15	482.59	<0.00001

effect sizes (shown as *partial eta*^{2}). A QQ-plot of the residuals (not shown) shows minimal deviation from the diagonal, indicating a near normal distribution.

While statistically significant, in general the interactions have only a minor impact, as shown by their relatively small F-values and effect sizes. In particular, there is only a very small interaction between the main variables of interest, history length and age, which is just visible in the different slopes of the lines in the interaction plots shown in Fig. 2. These graphs also show 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. One interaction that is a bit larger is the one between subject system and age, indicating that the impact of history age on impact analysis quality is to some extent system dependent, in contrast the impact of history length on algorithm boarders on negligible.

Table 4 reports Tukey's Honest Significant Difference (HSD) test for the ANOVA of Table 3 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

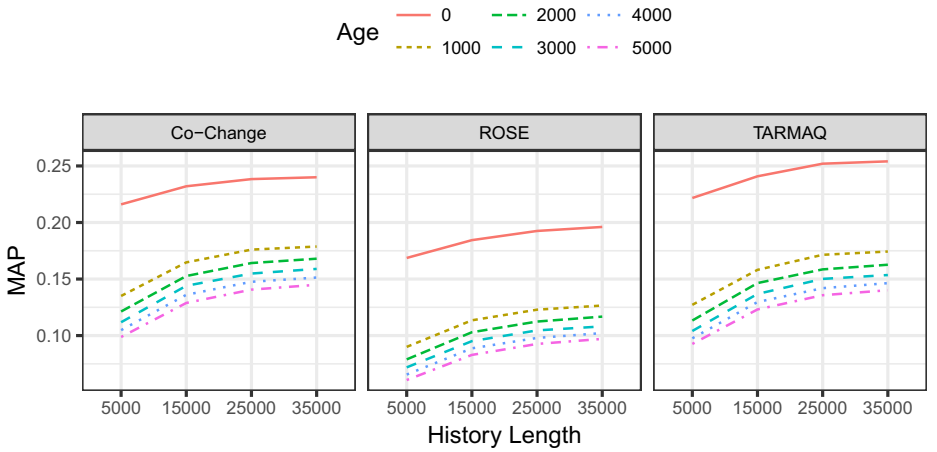

Fig. 2 Interaction plots of Age by History Length for the various algorithms

Table 4 Tukey's HSD for History Length and History Age (each sorted on decreasing *overall MAP* values)

History length			History age		
Length	MAP	Group	Age	MAP	Group
35000	0.1567	a	0	0.2185	a
25000	0.1530	b	1000	0.1430	b
15000	0.1423	c	2000	0.1312	c
5000	0.1155	d	3000	0.1225	d
			4000	0.1155	e
			5000	0.1096	f

significantly different *overall MAP* values. The test suggests three main conclusions: First, there are significant differences between *all* levels of each variable. Second, for history length, the best performance is attained by the largest length value of 35 000. We will use this particular length value later in the fine-grained study of history age (Section 5.2.2). 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 5.2.1). Both the graphs and Tukey's HSDs indicate that *very recent commits have a strong influence on the ability to predict change impacts*. This observation motivated our fine-grained study of small history length and age. In addition, we consider history lengths longer than 35 000 commits in Section 5.4.

Focusing on the age zero data only, Fig. 3 shows the *applicability* of TARMAQ, CO-CHANGE, and ROSE, along with the *overall MAP* and the *MAP when applicable*. The *applicability* of all three 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 when applicable* are very similar. However, the graphs do hint that the difference between the two increases as the history length shortens.

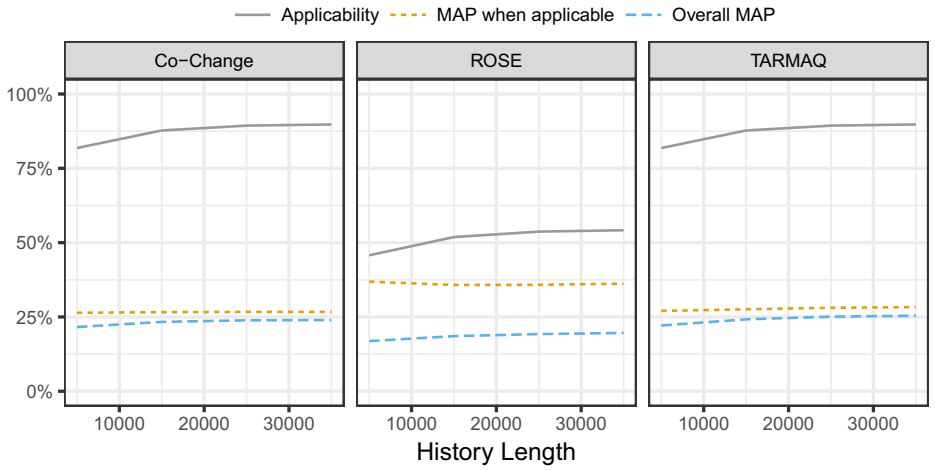

Fig. 3 Trends in the coarse-level study of history length (with *age = 0*) for the various algorithms

Table 5 ANOVA results for the fine-grained study of *length35k* (with $age = 0$)

Explanatory variable	Partial eta^2	Sum Sq	Df	F value	p-value
Subject System	0.68	5641.18	18	418709.25	<0.0001
Algorithm	0.24	854.90	2	571083.84	<0.0001
History Length	0.72	6879.40	55	167110.18	<0.0001
Subject System:Algorithm	0.04	113.01	36	4194.01	<0.0001
Subject System:History Length	0.31	1212.13	990	1635.79	<0.0001
Algorithm:History Length	0.03	93.80	110	1139.22	<0.0001

5.2 Fine-Grained Studies

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

Table 6 Tukey's HSD for *overall MAP* achieved on the fine-grained *length2000*, *length200*, and *length20* collections (each sorted by decreasing MAP values)

length 2000			*length200*			*length20*		
Length	MAP	Group	Length	MAP	Group	Length	MAP	Group
2000	0.1857	a	200	0.1329	a	20	0.0829	a
1900	0.1844	b	190	0.1319	b	19	0.0818	b
1800	0.1832	c	180	0.1314	b	18	0.0805	c
1700	0.1814	d	170	0.1298	c	17	0.0798	d
1600	0.1799	e	160	0.1289	d	16	0.0783	e
1500	0.1786	f	150	0.1271	e	15	0.0769	f
1400	0.1773	g	140	0.1261	f	14	0.0756	g
1300	0.1751	h	130	0.1241	g	13	0.0742	h
1200	0.1737	i	120	0.1224	h	12	0.0727	i
1100	0.1719	j	110	0.1200	i	11	0.0708	j
1000	0.1691	k	100	0.1181	j	10	0.0692	k
900	0.1670	l	90	0.1155	k	9	0.0675	l
800	0.1648	m	80	0.1124	l	8	0.0657	m
700	0.1614	n	70	0.1097	m	7	0.0635	n
600	0.1583	o	60	0.1059	n	6	0.0609	o
500	0.1538	p	50	0.1019	o	5	0.0576	p
400	0.1486	q	40	0.0968	p	4	0.0539	q
300	0.1422	r	30	0.0913	q	3	0.0493	r
200	0.1343	s	20	0.0834	r	2	0.0427	s
100	0.1185	t	10	0.0696	s	1	0.0321	t

5.2.1 History Length

An ANOVA for the fine-grained study of history lengths using age zero, shown in Table 5, finds largely the same patterns as the coarse-grained analysis. However, with the effects of age factored out, the effects of history length become more prominent (e.g., compare the two F-values). Furthermore, from history length's interactions we also see that the differences between individual systems has grown more prominent. Tables 6 and 7 show the results of Tukey's HSD for *length2000*, *length200*, and *length20* for respectively *overall MAP* and *MAP when applicable*. The results show that for *overall MAP*, statistically significantly higher MAP values are produced by long histories, while for *MAP when applicable* the reverse is true. Note that because each column represents a separate sample of commits, the MAP values should not be directly compared between columns. Only the trends are comparable.

Thus, the data suggests two contrasting trends: on the one hand the overlap between the artifacts contained in the history and those of the query progressively increases as history length increases, which leads to longer histories yielding better *overall MAP* values. On the other hand the relatedness of the artifacts contained in the history and those of a query progressively decreases as history length increases, which leads to short histories yielding better values for *MAP when applicable*. An explanation for these contrasting trends is found

Table 7 Tukey's HSD for *MAP when applicable* achieved on the fine-grained *length2000*, *length200*, and *length20* collections (each sorted by decreasing MAP values)

length 2000			*length200*			*length20*		
Length	MAP	Group	Length	MAP	Group	Length	MAP	Group
100	0.3383	a	10	0.3765	a	1	0.4017	a
200	0.3298	b	20	0.3667	b	2	0.3989	b
300	0.3224	c	30	0.3602	c	3	0.3949	c
400	0.3196	d	40	0.3513	d	4	0.3929	d
500	0.3165	e	50	0.3488	e	5	0.3886	e
600	0.3138	f	60	0.3449	f	6	0.3870	f
700	0.3118	g	70	0.3426	g	7	0.3850	g
800	0.3106	h	80	0.3397	h	8	0.3832	h
900	0.3086	i	90	0.3385	i	9	0.3801	i
1000	0.3078	j	100	0.3369	j	11	0.3788	j
1100	0.3074	j	110	0.3350	k	10	0.3787	j
1200	0.3060	k	120	0.3335	l	12	0.3780	j
1400	0.3058	kl	130	0.3321	m	13	0.3758	k
1300	0.3051	lm	140	0.3312	n	14	0.3749	k
1500	0.3047	mn	150	0.3299	o	15	0.3740	l
1800	0.3045	mn	160	0.3292	o	17	0.3733	lm
1600	0.3044	mno	170	0.3277	p	16	0.3730	m
1700	0.3043	no	180	0.3272	pq	18	0.3718	n
1900	0.3041	no	190	0.3265	q	19	0.3716	no
2000	0.3037	o	200	0.3254	r	20	0.3709	o

in the considerably lower *applicability* of the algorithms as the history length shortens. This trade-off can be seen in Fig. 4, which shows the *applicability*, *overall MAP*, and *MAP when applicable* for the three fine-grained history length datasets. In particular, across all granularities, *applicability* and *overall MAP* show an increasing trend, while the *MAP when applicable* shows a decreasing trend. A potential explanation for this trend is found in the observation that with longer histories there is a higher chance that at least one past transaction contains artifacts related to the query, which raises *applicability* and consequently *overall MAP*.

These trends continue with even longer history lengths as shown in Fig. 5, which shows the data for the *length35k* collection. The analysis of this figure, combined with the results of Tukey's HSD (not shown), allow us to answer RQ 1 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

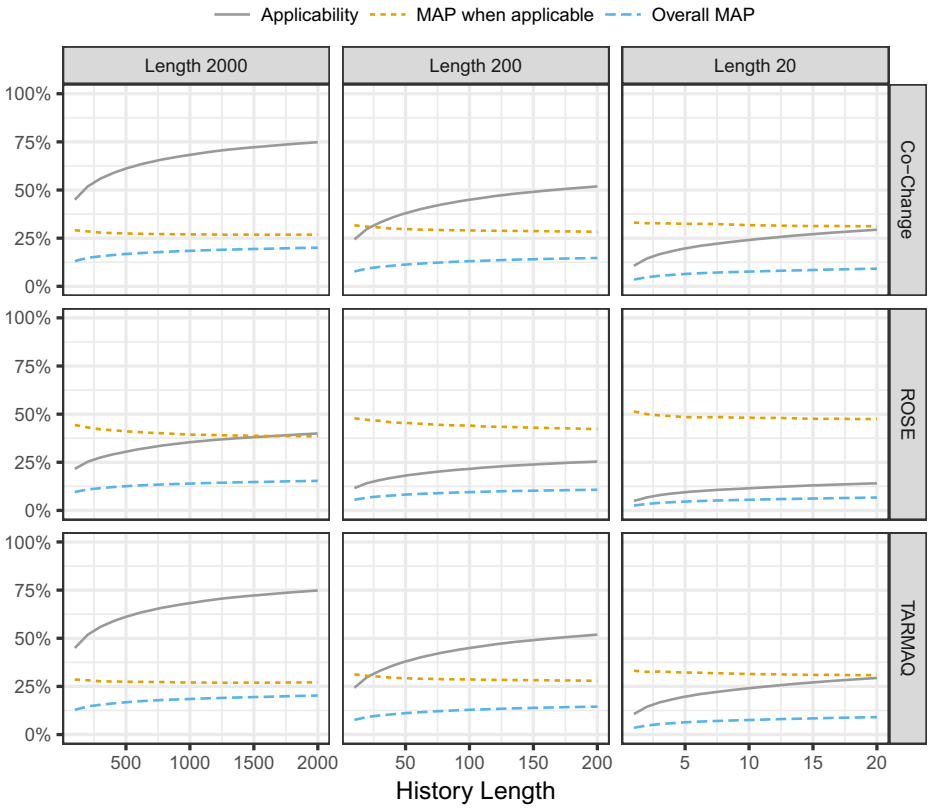

Fig. 4 The fine-grained study's three history-length scenario collections showing the inverse relation between MAP and *applicability*

the system to produce acceptable impact analysis results. To double-check this value, we computed the set of lengths that produced a MAP value within 1% of the highest MAP value and then selected the minimum length from this set. The value produced is 24700, which we round to 25000. Of course those willing to tolerate lower *applicability*, could consider shorter histories.

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

In short, yes. Overall, the trend for the three algorithms is that the performance increases up to around 15 000 commits where, at seen in Fig. 5, it levels off and remains stable for longer histories. Therefore, we set 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?*

In short, no. Our analysis of histories up to 35 000 transactions does not show any evidence of performance degrading because of older outdated commits. In Section 5.4 we will revisit this question using even longer histories.

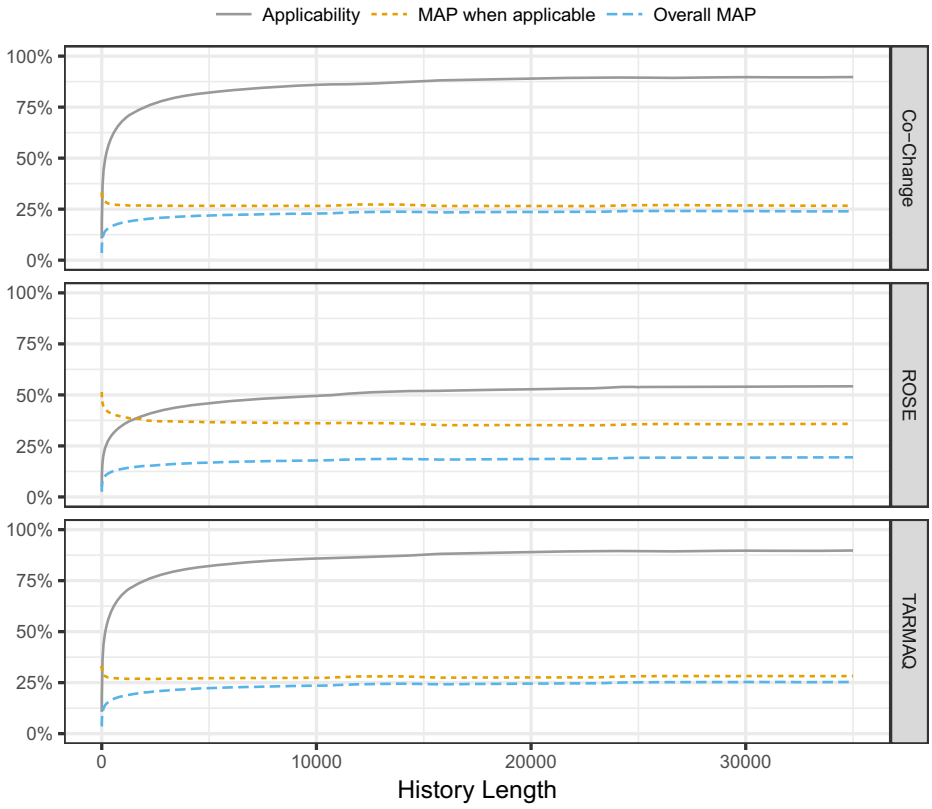

Fig. 5 Results from the large-scale fine-grained investigation of *length35k*

5.2.2 History Age

Parallel to Table 3, the ANOVA for the three age collections (not shown), finds age and subject system to be highly statistically significant. Tables 8 and 9 show the results for Tukey's HSD on the collections *age2000*, *age200*, and *age20* for respectively overall MAP and *MAP when applicable*. In both cases, the scenarios all appear in age order, showing only a few overlapping groups. In all three collections, for both *overall MAP* and *MAP when applicable*, the scenario with age zero performs significantly better than the next greater age. While the gap in MAP values gets smaller as age grows, the differences is significant even when going from an age of one to an age of two.

Figure 6 shows the trends for *applicability*, *overall MAP*, and *MAP when applicable* for the three algorithms. Across the collections, the drop off from age zero is visually evident. However, it gets progressively less prominent from *age2000* to *age200*, and finally to *age20*. Moreover, we see that even though individual values differ, the trends are very similar across the three algorithms. Based on the fast deterioration of impact analysis quality as age increases, we do not consider the study of ages larger than 2000 to be relevant (i.e., we do not include a study comparable to *length35k*).

Similar to RQ 1, the plots in Fig. 6 and Tukey's HSD in Tables 8 and 9 enable us to answer RQ 2.

Table 8 Tukey's HSD for *overall MAP* achieved on the *age2000*, *age200*, and *age20* collections (each sorted by decreasing MAP values)

length 2000			*length200*			*length20*		
Length	MAP	Group	Length	MAP	Group	Length	MAP	Group
0	0.2301	a	0	0.2330	a	0	0.2290	a
100	0.1857	b	10	0.2093	b	1	0.2204	b
200	0.1798	c	20	0.2029	c	2	0.2168	c
300	0.1769	d	30	0.1994	d	3	0.2140	d
400	0.1733	e	40	0.1969	e	4	0.2124	e
500	0.1703	f	50	0.1946	f	5	0.2107	f
600	0.1686	g	60	0.1927	g	6	0.2092	g
700	0.1668	h	70	0.1907	h	7	0.2082	h
800	0.1647	i	80	0.1894	i	8	0.2074	i
900	0.1636	j	90	0.1880	j	9	0.2065	j
1000	0.1619	k	100	0.1875	j	10	0.2053	k
1100	0.1605	l	110	0.1865	k	11	0.2047	k
1200	0.1591	m	120	0.1856	l	12	0.2034	l
1300	0.1580	n	130	0.1850	lm	13	0.2033	lm
1400	0.1565	o	140	0.1844	mn	14	0.2026	mn
1500	0.1554	p	150	0.1838	no	15	0.2021	no
1600	0.1542	q	160	0.1832	op	16	0.2014	o
1700	0.1531	r	170	0.1828	p	17	0.2004	p
1800	0.1521	s	190	0.1817	q	18	0.2000	pq
1900	0.1511	t	180	0.1817	q	19	0.1995	qr
2000	0.1503	u	200	0.1810	q	20	0.1989	r

Table 9 Tukey's HSD for *MAP when applicable* achieved on the *age2000*, *age200*, and *age20* collections (each sorted by decreasing MAP values)

length 2000			*length200*			*length20*		
Length	MAP	Group	Length	MAP	Group	Length	MAP	Group
0	0.2994	a	0	0.3052	a	0	0.2990	a
100	0.2650	b	10	0.2857	b	1	0.2917	b
200	0.2607	c	20	0.2805	c	2	0.2882	c
300	0.2582	d	30	0.2779	d	3	0.2862	d
400	0.2548	e	40	0.2758	e	4	0.2849	e
500	0.2525	f	50	0.2743	f	5	0.2836	f
600	0.2516	g	60	0.2728	g	6	0.2823	g
700	0.2509	g	70	0.2715	h	7	0.2816	gh
800	0.2488	h	80	0.2703	i	8	0.2809	hi
900	0.2485	h	90	0.2699	i	9	0.2801	ij
1000	0.2474	i	100	0.2687	j	10	0.2794	j
1100	0.2467	ij	110	0.2679	jk	11	0.2794	j
1200	0.2462	jk	120	0.2676	k	12	0.2780	k
1300	0.2456	k	130	0.2666	l	13	0.2776	kl
1400	0.2438	l	140	0.2664	l	14	0.2774	kl
1500	0.2438	l	160	0.2662	l	15	0.2771	lm
1600	0.2425	m	170	0.2659	l	16	0.2763	m
1700	0.2423	m	150	0.2659	l	17	0.2754	n
1800	0.2407	n	180	0.2647	m	18	0.2754	n
1900	0.2400	no	190	0.2644	m	19	0.2750	no
2000	0.2398	o	200	0.2641	m	20	0.2744	o

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 one's tolerance for lost precision, which depends on the use that the change impact analysis is put to. Similar to the history length analysis, the falloff in precision is initially quite steep and then tends to gradually narrow. For example, for the *age2000* collection in Table 9, the difference between MAP values for *age*=0 and *age*=100 is 12.9%, 1.6% between *age*=100 and *age*=200, 0.28% between *age*=1000 and *age*=1100, and only 0.08% between *age*=1900 and *age*=2000. Similar trends can be observed in the other two collections. Starting with the highest achieved *MAP when applicable* value and using a 10% tolerance cutoff as an arbitrary maximal acceptable loss, the upper bound for history age is 40 commits when using the *age200* collection. 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 at which 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 value

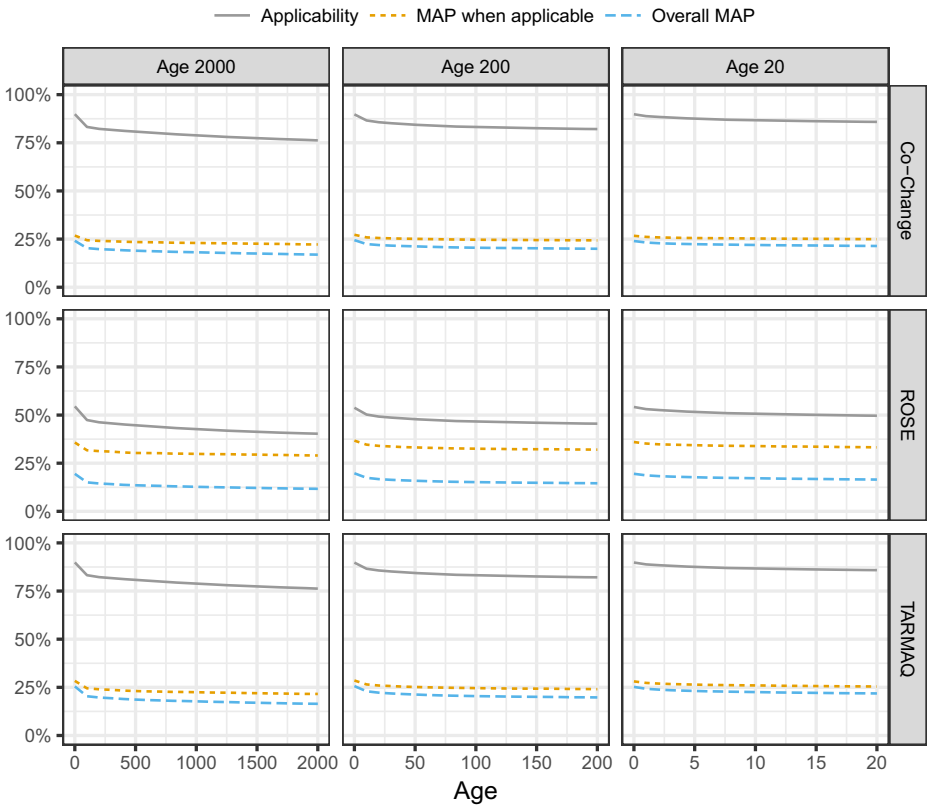

Fig. 6 The fine-grained study's three history-age ranges shown from coarsest to finest

for *MAP when applicable* in Table 9 is 0.2398 for age 2000. Using this value, the target value for *MAP when applicable* is $0.2398 + 10\% = 0.2638$, which is crossed between age 100 and age 200 in the *age2000* collection.[3] Thus, while the performance is monotonically decreasing as age increases, it quickly reaches the point beyond which the remaining deterioration is below our practically acceptable tolerance for deterioration. Said another way, it is possible to find a point beyond which impact analysis quality ceases to deteriorate in a practically significant way as history age increases.

5.3 Project Characteristics

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

[3]Observe that the MAP values in the three subtables of Table 9 were obtained from three different randomly sampled collections, which explains the variation in MAPs for ages repeated in different collections. Although these values are within the 5% confidence interval targeted by our sampling approach, it still means that cutoff values obtained from one collection cannot be used to look up corresponding ages in another collection, as also shown by the values for *age2000* and *age200* in Table 9.

RQ 3 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. Thus in contrast to the previous two research questions, which aim to understand patterns across a wide range of systems, this question considers predicting values for a specific system. To answer this question, we build six separate linear regression models, three that predict the value of history length for TARMAQ, CO-CHANGE, and ROSE, and three that consider age. In both cases, the set of explanatory variables includes the following system demographics (see Table 1), which capture aspects of the development history, team, and process: (1) *number of unique artifacts* in the change history, (2) *average number of artifacts in a commit*, (3) *number of developers* throughout the change history, (4) *mode and median inter-commit time* between two commits by the same developer, (5) *average and median length of commit streak* (a streak is a number of consecutive commits by the same developer). The values of these demographics were obtained by analyzing the change histories for the various systems.

In this analysis, we omit Cisco and KM for which we cannot disclose the demographics. Also, similar to the analysis in Section 5.1, the regression analysis for history length is performed using an age of zero, while the regression analysis for age is performed using a history length of 35 000.

We begin with the three linear regression models for history length. Each requires determining a target history length for each system. A simple selection would be the history length that produced the highest *overall MAP* value. Unfortunately, from the statistical analysis this value is not unique. Thus, from the set of top-performing history lengths, we selected the smallest value under the assumption that requiring less history is preferable (e.g., leads to more efficient impact analysis). In summary, the target value for history length for each system is determined by applying Tukey's HSD test and then selecting from the top group (the 'a' group) the smallest history length.

We then fit a linear model to the data using R's *lm* function starting with all the explanatory variables and then applying backward elimination. The elimination phase removes the least statistically significant variable and then regenerates a new model. For example, in TARMAQ's initial model the variable *median inter-commit streak* has the highest p-value of 0.74. The elimination step removes this variable and then rebuilds the model. This process is repeated until only significant explanatory variables remain.

It is also possible to consider interactions between the explanatory variables. Preliminary work with the demographics made it quite evident that there were interactions among the explanatory variables. Unfortunately, with only 17 systems there is insufficient data to build a model that includes all pair-wise interactions. As a compromise an initial model was generated without any interactions and then interactions were added for each variable having a p-value less then 0.33. The elimination process is then applied to produce the final model. Note that in order to maintain a well-formed model variables with p-values greater than 0.05 are retained when they are part of a significant interaction.

The remainder of this investigation first considers TARMAQ before turning to CO-CHANGE and then finally ROSE. The final model for TARMAQ, shown in Table 10, is statistically significant having a p-value of 0.000934. The model includes three significant explanatory variables and three significant interactions, which makes it challenging to understand the effects of the variables. One standard approach to doing so looks at the pair-wise interactions when the third variable takes its mean value. The resulting interaction graphs, shown in Fig. 7, were generated from the model shown in Table 10 using the following values:

Table 10 Regression model predicting history length for TARMAQ

Explanatory variable	Estimate	Std. Error	t-value	p-value
(Intercept)	8859.2	2356.20	3.76	0.00372
Median Inter Commit Time (MICT)	1342.3	740.50	1.81	0.09995
Number of Unique Artifacts (NUA, in 1000's)	141.5	23.80	5.93	0.00014
Average Number of Artifacts per Commit (ANAC)	−673.8	342.20	−1.97	0.07769
MICT : NUA	−14.9	3.67	−4.21	0.00180
MICT : ANAC	192.3	75.60	2.54	0.02925
NUA : ANAC	−3.1	0.67	−4.59	0.00099

Variable	First Quartile	Mean	Third Quartile
Average Number of Artifacts per Commit (ANAC)	5.5	11.8	12.6
Median Inter Commit Time (MICT)	0.0	2.6	4.0
Number of Unique Artifacts (NUA, in 1000's)	30.0	167.0	232.0

In addition to providing an affirmative answer to the first half of RQ 3, it is interesting to consider the parameter estimates found in the model and discuss possible explanations for the relations between the explanatory variables and the response variable, history length. Starting with the three explanatory variables, the analysis shows that as median inter-commit time (MICT) increases a longer history length is needed. A potential explanation of this relation is working context: it is likely that a developer works on a sequence of related modifications, yielding a sequence of commits from that developer that contain related artifacts. When MICT is zero, the commits of the developer tend to follow each other directly in the history and short history lengths suffice to achieve high-precision impact analysis. As the value of MICT grows there are an increasing number of interleaved commits from other developers who have different working contexts, which means that longer history lengths are needed to include the relevant artifacts and achieve high-precision impact analysis.

The model also shows that as the number of unique artifacts (NUA) increases, so does the predicted history length. Indeed, the more artifacts contained in a system, the further apart commits containing artifacts relevant to a query are likely to be, and hence a larger history is required.

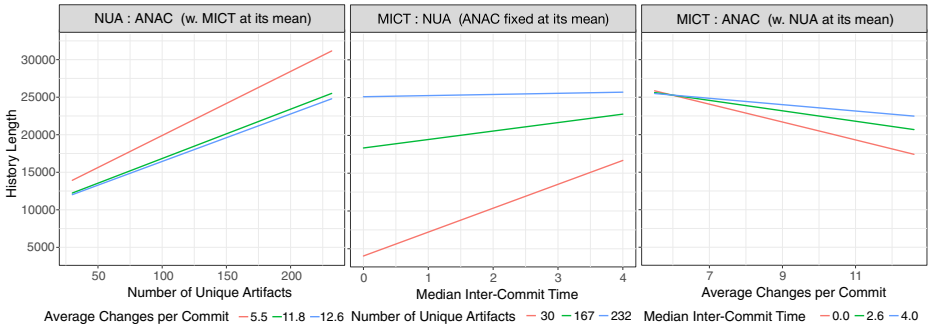

Fig. 7 Interaction plots for the three interaction terms of the regression model shown in Table 10

Finally, the coefficient associated with average number of artifacts per commit, is negative, indicating an inverse relationship. In other words, as the average "size" of a commit increases, the required history length decreases. One explanation for this relation is that larger commits contain 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 $\{a, c\}$. Another possible explanation is the combination of working context and commit practices: developers who tend to create large commits are likely to commit on a task-by-task basis, creating a transaction that contain all the artifacts related to a modification task at once. This behavior makes it less likely that there is information with respect to related artifacts contained in a short history length. Conversely, developers who split a task in several smaller commits effectively decrease the number of artifacts per commit and thus spread out artifacts related to their working context over a larger number of commits, thereby increasing the history length required to achieve high precision impact analysis.

Figure 7 plots the three interactions. The first shows how the impact of the number of unique artifacts decreases as the average number of changes per commit increases. The relative slopes of the lines indicates that this effect is not large, which is also evident from its small parameter estimate (see the last line of Table 10). Looking at the middle graph, the relative impact of the number of unique artifacts is more pronounced as evidenced by the greater difference in the line's slopes. In this case it is interesting that as this value increases, the impact of the median inter-commit time approaches zero. Conversely, for smaller systems the median inter-commit time has a greater influence. Finally, from the third chart, the average number of changes per commit has a greater impact for smaller median inter-commit times. While a higher average number of changes per commit brings a need for less history the reduction is greater them the median inter-commit time is small. The smallest history requirement comes from larger commits without any intervening commits from other developers.

While statistically significant with a p-value of 0.0345, the model for CO-CHANGE omits the median inter-commit time. Perhaps because of its simpler rules, it also includes smaller parameter estimates (for example the coefficient for the number of unique artifacts is 141.5 with TARMAQ, but only 60.7 for CO-CHANGE. Like the TARMAQ model the sign of the average changes per commit and the interaction of the two is negative but again both have smaller magnitudes (-6.6 and -1.7, respectively). Thus overall, the influences of the explanatory variables in the CO-CHANGE model are muted relative to the TARMAQ model.

Finally, the ROSE model, which has a p-value of 0.00114, is similar to the TARMAQ model in that it includes the same explanatory variables and interactions. Furthermore, the coefficients are very similar (for example, consider the average number of changes per commit, which is -673.8 in the TARMAQ model and -673.3 in the ROSE model. The largest difference is the coefficient of median inter-commit time, which in the ROSE model is 1708 compared to 1342 in the TARMAQ model. This difference indicates that ROSE is more susceptible to changes in the median inter-commit time than TARMAQ: while ROSE needs less history when commits are clumped by relevance, as commits become more intermixed, ROSE's need grows to exceed those of TARMAQ. Finally, taken together with more complex rules, their appears to be a greater dependence on median inter-commit time.

The second part of RQ 3 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 use the three MAP target levels of

95%, 90% and 80% of the maximum overall MAP value that can be achieved over the different history ages. We again 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 three response variables and all three algorithms, the elimination process removes all the explanatory variables, failing to produce a regression model. This indicates that variations in the demographic variables is not an effective predictor of the rate at which a model deteriorates.

In summary, this analysis allows us to answer RQ 3 as follows: good values for history length can be predicted. Indeed, a team of developers using TARMAQ can use the regression model found in Table 10, 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.

5.4 Longitudinal Study

This section revisits RQ 1.3, which is repeated here for convenience:

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?

When considering RQ 1.3 in Section 5.2, we studied change histories of up to 35 000 transactions. We found no evidence that older transactions degraded precision. While 35 000 is a reasonably large change history, software systems which have been built over many years by a plethora of developers see much larger histories. In this section, we explore the effects of such long histories, reaching over 540 000 transactions in the case of the *Linux Kernel*.

We begin the analysis visually by plotting *applicability*, *MAP when applicable*, and *overall MAP*. When considering the complete development histories for various systems, we see that they are no longer of all of equal length, which affects the legibility of the plots. To address this, we cluster the seven software systems analyzed in this study in three groups, based on their available history lengths: Fig. 8, shows *LLVM* and *Wine*, which have histories of around 100 000 transactions, Fig. 9 shows *IntelliJ IDEA*, *Liferay Portal*, and *WebKit*, which have histories of around 150 000 transactions, and Fig. 10 shows *Mozilla Gecko* and the *Linux Kernel*, which have histories of just over 400 000 and 500 000 transactions.

Visually, the trends seen in Fig. 5 resurface: initially there is a steady rise in *applicability* and *overall MAP*, and a steady decline in *MAP when applicable*. These eventually level off as history length increases. However, in the case of *IntelliJ IDEA* and the *Linux Kernel* the leveling off seems to take longer.

To analytically investigate whether long histories deteriorate precision, we first pose the *myth hypothesis*, which assumes that they do. As such we compare the AP values produced by TARMAQ when given a relatively short history, against the AP values it produces when given the maximal available history. Throughout our study we have found that history lengths in the neighborhood of 25 000 to 35 000 transactions consistently perform well, we therefore consider 30 000 for the length for our "short histories". While not shown, we collected data for length from 10 000 through 540 000 in increments of 10 000. As a confirmation step, we repeated the analysis presented in this section using 20 000 and 40 000 transactions. The results are virtually identical and in both cases lead to the exact same conclusion.

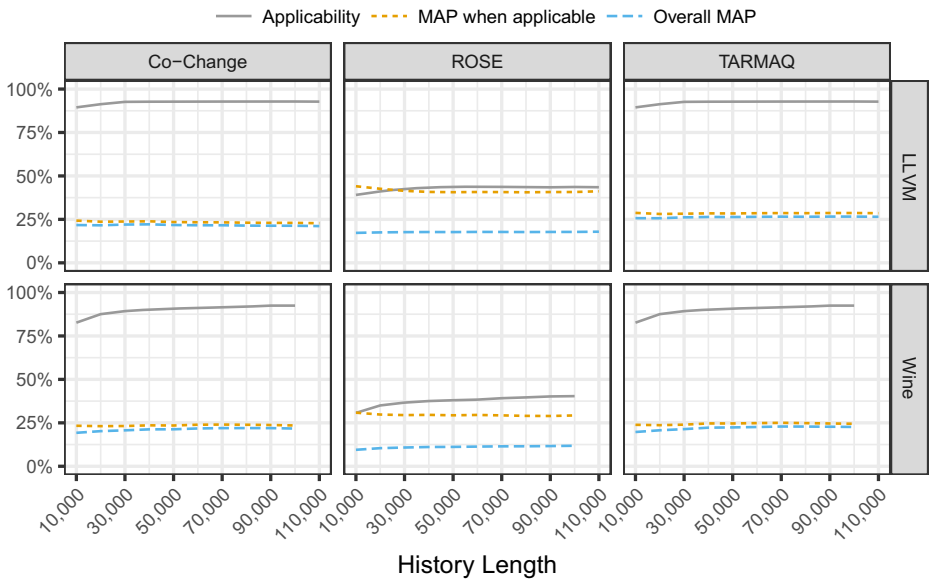

Fig. 8 LLVM and Wine

Formally, the myth hypothesis is

H_0 : The mean AP value produced by TARMAQ when given the maximal history is not less than the mean produced when using a history length of 30 000.

H_1 : The mean AP value produced by TARMAQ when given the maximal history is less than the mean produced when using a history length of 30 000.

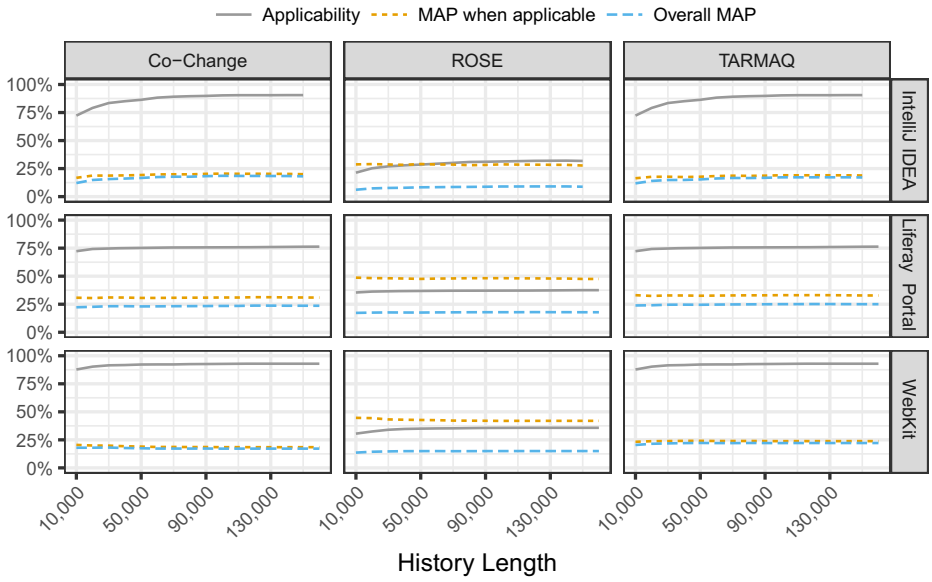

Fig. 9 IntelliJ IDEA, Liferay Portal and WebKit

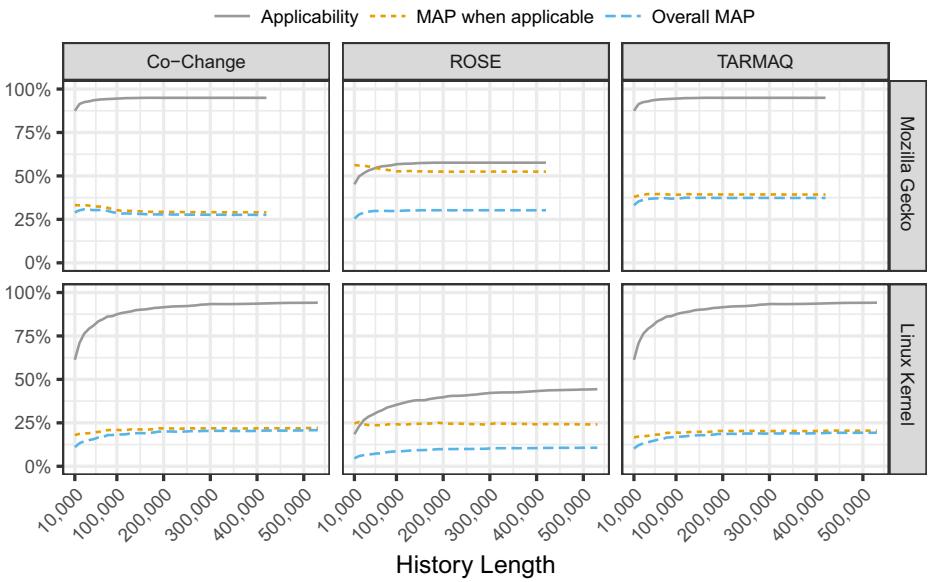

Fig. 10 Mozilla Gecko and Linux Kernel

Perhaps unsurprisingly given our findings so far, we do not obtain significant results for any of the systems (i.e., the mean AP for the maximal length is not significantly less than that for a length of 30 000). On the contrary, based on the figures it seems more likely that maximal histories actually improve precision compared to the shorter 30 000. We therefore ask the opposite question and apply a t-test to the following hypothesis:

H_0 : The mean AP value produced by TARMAQ when given the maximal history is not greater than the mean produced when using a history length of 30 000.

H_1 : The mean AP value produced by TARMAQ when given the maximal history is greater than the mean produced when using a history length of 30 000.

As we here perform a second t-test there is an increased chance that our conclusions become erroneous, to counteract this we apply the *Bonferroni correction* to reduce the alpha level for what is considered a significant result. Thus we set $\alpha = 0.05/2 = 0.025$. With the adjusted alpha level we still obtain significant results for all software systems except *LLVM*, as shown in Table 11. Furthermore, the effect size of the difference as exhibited through *Cliff's delta* supports our earlier visual observation that the *Linux Kernel* and *IntelliJ IDEA* continue to see precision gains beyond 30 000 transactions. Cliff's delta also picks up a small effect for *Wine*. For the remaining systems the effect is negligible.

Combining the evidence of both t-tests with the negligible effect size, we conclude that there is no practical difference between the two history lengths. In summary, even for extremely long history lengths we find no evidence that there is an upper-bound to history length beyond which outdated information negatively affects the precision of impact analysis. Therefore, we are confident to answer RQ 1.3 negatively.

5.5 Stability Study

As a software system evolves passed the point of initial maturity one might expect the quality of change impact analysis to stabilize. For our final analysis we consider this question:

Table 11 Results for second t-test with alternative hypothesis: "The mean AP value produced by TARMAQ when given the maximal history is greater than the mean produced when using a history length of 30 000"

Subject system	p-value	Cliff's delta	Magnitude of effect
Mozilla Gecko	0.00	0.14	Negligible
IntelliJ IDEA	0.00	0.48	Large
Liferay Portal	0.01	0.05	Negligible
Linux Kernel	0.00	0.92	Large
LLVM	0.10	0.03	Negligible
WebKit	0.00	0.08	Negligible
Wine	0.00	0.21	Small

RQ 4 *How is impact analysis quality throughout the evolution history affected by choosing a fixed* history length *and* history age?

To answer RQ 4 we compare the performance of TARMAQ when applied to two data sets: a set of recent queries and a set of older queries. While not presented in this section, we repeated the stability analysis using CO-CHANGE and ROSE and obtained the same results in each case. The design of the stability study is visualized in Fig. 11 where the two periods in time are labeled t_0 and t_1.

It is important to note that in answering RQ 4 we are not concerned with varying *history length/age* as was done when considering our previous research questions. Instead we focus solely on *time* as an explanatory variable. However, we must still set history age and history length. For *history age* we use the value zero because earlier findings in this paper show that non-zero ages significantly degrade TARMAQ's precision. In terms of *history length*, we set this to *15 000* for both t_0 and t_1. This value strikes a balance between *applicability* (see Fig. 5) and having sufficient space between t_0 and t_1. For example, if a longer *history length* were chosen then t_0 would have to be closer to t_1 in order to guarantee that all queries had sufficient previous transactions. The further apart t_0 and t_1 are the stronger we can state that precision has stabilized.

It is unreasonable to expect two samples to have *exactly* the same mean. Therefore equivalence tests employ a threshold value. Naturally, this threshold should be anchored in the variation expressed in the data. We therefore set the threshold for *sufficiently similar* using the standard deviation. Concretely, we first calculate the standard deviation when TARMAQ is applied to the sample queries from t_0 and t_1, and then set the threshold to the minimum of these two values. This process resulted in a threshold of $\varepsilon = 0.034008$. Thus if the difference between the means for t_0 and t_1 differs by less than 0.034008, then we assert that the two are not different.

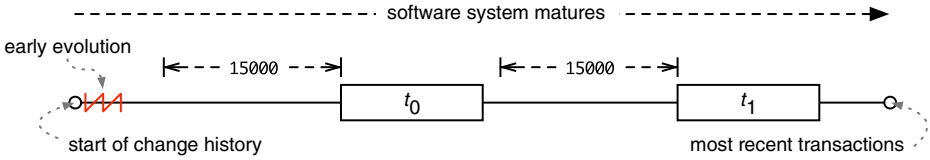

Fig. 11 The stability study compares two separate time periods t_0 and t_1, both outside the early evolution stage, and selected so that queries within each period have access to at least 15 000 previous transactions

Compared to what one might call "standard" hypothesis testing, tests for equivalence reverse the sense of the *null hypothesis*, we therefore state our hypothesis as follows:

H_0 : The mean AP value computed using the transactions of t_0 and t_1 are different with respect to ε

H_1 : The mean AP value computed using the transactions of t_0 and t_1 are equivalent with respect to ε

We use Schuirmann and Westlake's *two one-sided t-test for equivalence* (Schuirmann 1981; Westlake 1981) as implemented in the R package equivalence. In particular, we apply the function tost() to TARMAQ's output for queries randomly chosen from t_0 and t_1 using the equivalence threshold ε.

The resulting p-value, which is <0.0001, captures that the threshold of 0.034008 is well outside the 99% confidence interval of [0.01669, 0.02191]. Thus there is a 1% chance that our confidence interval does not contain the true difference given the population sample. Based on this data, we reject H_0 and accept that the mean AP value computed using the transactions of t_0 and t_1 are equivalent with respect to ε. In summary, for RQ 4 we conclude that after a software system reaches maturity, the precision of change impact analysis remains consistent.

6 Threats to Validity

Realism of Scenarios used in Evaluation We evaluate mining-based change impact analysis by establishing a ground truth from historical transactions, randomly splitting them into a query and an expected outcome of a certain size. However, this approach does not account for the actual order in which changes were made before they were committed together to the versioning system. As a result, it is possible that our queries contain elements that were actually changed later in time than elements of the expected outcome. This cannot be avoided when mining co-change data from a versioning system, because the timing of individual changes is lost in that data. It *can* be addressed by using another source of co-change data, such as a developer interactions with an IDE, but the invasiveness of this kind of data collection means that there are only a limited number of such data-sets available. This lack of aviailibility prevents a study as comprehensive as the one presented here. Moreover, since the evolutionary couplings that are at the basis of our change impact analysis form a bi-directional relation, the actual order in which changes were made before they were committed has no impact on the result. Our goal is not to re-enact the actual timeline of changes, but rather to establish a ground truth w.r.t. related artifacts.

Variation in Software Systems We evaluated the impact of history length and age on mined change impact by studying two industrial systems and 17 large open source systems. These systems vary considerably in size and frequency of transactions (commits), which should provide an comprehensive picture of performance. However, despite our careful choice, we are likely not to have captured all variations.

Commits as Basis for Evolutionary Coupling The evaluation in this paper is grounded in predictions based on the analysis of patterns found in the change histories. The transactions that make up the change histories are however not in any way guaranteed to be "correct" or "complete", in the sense that they represent a coherent unit of work.

Non-related artifacts may be present in the transactions, and related artifacts may be missing from the transactions. However, the included software-systems in our evaluation all (except KM) use Git for version control. As Git provides developers with tools for history-rewriting, we believe that this should lead to more coherent transactions.

Equal Weight for All Commits In our experiment, all transactions from change history are given equal weight while mining change impact. A compelling alternative viewpoint is that more recent transactions are more relevant for current developments and should therefore be given higher weight than older transactions. Similarly, one could argue that, because of their knowledge about the system, transactions committed by code developers should be given higher weight than transactions committed by occasional contributors. For the study described in this paper, we do not include such orthogonal weighing scenarios in our experiments because of their interaction with several of our research questions, such as the length at which considering a longer history would no longer benefit impact analysis quality, or the length at which considering a longer history would start decreasing impact analysis quality due to the inclusion of outdated information. Moreover, the systems considered in this study use a contribution process that includes code reviewing based on pull requests before taking in changes from occasional contributors. We believe this process largely removes the differences between transactions committed by core developers and transactions that originate from occasional contributors but were accepted after review.

Implementation We implemented and thoroughly tested all algorithms studied in this paper in Ruby. However, we can not guarantee the absence of implementation errors which may have affected our evaluation.

7 Related Work

The software repository mining literature (Graves and et al 2000; Zimmermann and et al 2005; Hassan 2008) 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 and et al (2005), 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 and choice of interestingness measure) (Zheng et al. 2001; Lin et al. 2002; Jiang and Gruenwald 2006; Maimon and Rokach 1383), 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 artifacts. For example, Kagdi et al. (2013), Zimmermann and et al (2005) and Ying et al. 2004 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 mined impact analysis quality. In previous work (Moonen et al. 2016b), 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 (Jaafar et al. 2014; Zimmermann et al. 2005; Kagdi et al. 2006). 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 (German 2006). Alali et al. also investigated the total number of lines modified in the files, and the total number of hunks with line changes (Alali et al. 2008). Kolassa et al. performed a similar study on commit frequency, reporting an average inter-commit time of around three days (Kolassa et al. 2013). However, none of these studies investigates how characteristics of the change history affect the quality of change impact analysis.

Characteristics of the Change Set Targeted association rule mining approaches drive the generation of rules by a *query* supplied by the user (Srikant et al. 1997). In general, characteristics of the query can effect the precision of mined impact analysis. 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 (Rolfsnes et al. 2016a). 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 (Hassan and Holt 2004).

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 (Ying et al. 2004), or be continuously updated with respect to the chosen transaction from the test set (Zimmermann and et al 2005). 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).

8 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 *overall MAP* value also increases, although this increase diminishes starting around 15 000 commits. Moreover, the applicability also increases with increased history length, but seems to top out around 25 000 commits. We found that the impact of age on both *overall MAP* and *MAP when applicable* is very significant, as even very little aging yields a strong, basically

exponential decrease. Thus, we find that recent commits are of great value to determine change impact.

Even in our longest studies of up to 540 000 commits, we found no evidence for the commonly held belief that there is an upper-bound to the history that can be used for mining evolutionary coupling before outdated knowledge starts to negatively affect impact analysis quality.

In addition to these studies, to provide a better understanding of the impact of history length and age on the quality of change impact analysis, we also consider prediction models for each algorithm that predict the length of the history that should be used for a given system. These prediction models are functions of system demographics, such as the average number of artifacts in a commit, the median inter-commit time, and the number of unique artifacts in the history. We found no corresponding prediction models for system age, which likely reinforces the rapid detrimental effects of aging.

Finally, we found that the mining algorithms are *stable* with respect to the effects of choosing a particular history length and history age on the impact analysis quality throughout the evolution history. This means that for a system that has matured beyond its chaotic youth, an optimal history length *for that point in time* can be computed using our prediction model. As the system continuous to evolve, that same history length (preferably of at least 15 000 commits) can be used to achieve similar results.

8.1 Future Work

Looking forward, an interesting avenue of investigation would be to assess the impact of rule aggregation (Rolfsnes et al. 2016b) on change impact analysis quality. This analysis was not considered in the current paper to maintain conceptual integrity, and avoid the inclusion of too many orthogonal topics.

Moreover, based on our findings related to the impact of very recent transactions and history age on change impact analysis quality, it would be interesting to experiment with (a) an alternative targeted association rule mining algorithm that does not consider a fixed history length but instead uses an adaptive approach, growing the history until (a number of) applicable transactions are found, and (b) alternative strategies for association rule generation that take age into account, for instance by assigning higher weight to more recent transactions.

Finally, it would be interesting to analyze how aspects of the development process affect the evolutionary coupling and change impacts that are mined from historical co-change data. For example, should transactions from all contributors to a project be considered equally, or would it be beneficial to give higher weight to transactions from core team members. Another aspect to investigate is the relation between the velocity of a development project, or even the typical time between commits, and good values for history size and age.

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

References

Agrawal R, Imielinski T, Swami A (1993) Mining association rules between sets of items in large databases. In: ACM SIGMOD international conference on management of data. ACM, pp 207–216

Alali A (2008) An empirical characterization of commits in software repositories. Ms.c. Kent State University, 53

Alali A, Kagdi H, Maletic JI (2008) What's a typical commit? A characterization of open source software repositories. In: International conference on program comprehension (ICPC). IEEE, pp 182–191

Baeza-Yates R, Ribeiro-Neto B (1999) Modern information retrieval. ACM, p 513

Bohner S, Arnold R (1996) Software change impact analysis. IEEE, USA

Canfora G, Cerulo L (2005) Impact analysis by mining software and change request repositories. In: International software metrics symposium (METRICS). IEEE, pp 29–37

Eick S et al (2001) Does code decay? Assessing the evidence from change management data. IEEE Trans Softw Eng 27(1):1–12

Gall H, Hajek K, Jazayeri M (1998) Detection of logical coupling based on product release history. In: IEEE international conference on software maintenance (ICSM). IEEE, pp 190–198

German DM (2006) An empirical study of fine-grained software modifications. Empir Softw Eng 11(3):369–393

Gethers M et al (2011) An adaptive approach to impact analysis from change requests to source code. In: IEEE/ACM international conference on automated software engineering (ASE). IEEE, pp 540–543

Graves TL et al (2000) Predicting fault incidence using software change history. IEEE Trans Softw Eng 26(7):653–661

Hassan AE (2008) The road ahead for Mining Software Repositories. In: Frontiers of software maintenance. IEEE, pp 48–57

Hassan AE, Holt R (2004) Predicting change propagation in software systems. In: IEEE international conference on software maintenance (ICSM). IEEE, pp 284–293

Jaafar F et al (2014) Detecting asynchrony and dephase change patterns by mining software repositories. J Softw: Evol Process 26(1):77–106

Jashki M-A, Zafarani R, Bagheri E (2008) Towards a more efficient static software change impact analysis method. In: ACM SIGPLAN-SIGSOFT workshop on program analysis for software tools and engineering (PASTE). ACM, pp 84–90

Jiang N, Gruenwald L (2006) Research issues in data stream association rule mining. ACM SIGMOD Rec 35(1):14–19

Kagdi H, Yusuf S, Maletic JI (2006) Mining sequences of changed-files from version histories. In: International workshop on mining software repositories (MSR). ACM, pp 47–53

Kagdi H, Gethers M, Poshyvanyk D (2013) Integrating conceptual and logical couplings for change impact analysis in software. Empir Softw Eng 18(5):933–969

Kolassa C, Riehle D, Salim MA (2013) The empirical commit frequency distribution of open source projects. In: International Symposium On Open Collaboration (WikiSym). ACM, pp 1–8

Law J, Rothermel G (2003) Whole program path-based dynamic impact analysis. In: International conference on software engineering (ICSE). IEEE, pp 308–318

Lin W, Alvarez SA, Ruiz C (2002) Efficient adaptive-support association rule mining for recommender systems. Data Min Knowl Disc 6(1):83–105

Maimon O, Rokach L (1383) In: Maimon O, Rokach L (eds) Data mining and knowledge discovery handbook. Springer, Berlin

Moonen L et al (2016a) Exploring the effects of history length and age on mining software change impact. In: IEEE international working conference on source code analysis and manipulation (SCAM), pp 207–216

Moonen L et al (2016b) Practical guidelines for change recommendation using association rule mining. In: International conference on automated software engineering (ASE). ACM, pp 732–743

Podgurski A, Clarke L (1990) A formal model of program dependences and its implications for software testing, debugging, and maintenance. IEEE Trans Softw Eng 16(9):965–979

Ren X et al (2004) Chianti: a tool for change impact analysis of java programs. In: ACM SIGPLAN conference on object-oriented programming, systems, languages, and applications (OOPSLA), pp 432–448

Robbes R, Pollet D, Lanza M (2008) Logical coupling based on fine- grained change information. In: Working conference on reverse engineering (WCRE). IEEE, pp 42–46

Rolfsnes T et al (2016a) Generalizing the analysis of evolutionary coupling for software change impact analysis. In: International conference on software analysis, evolution, and reengineering (SANER). IEEE, pp 201–212

Rolfsnes T et al (2016b) Improving change recommendation using aggregated association rules. In: International conference on mining software repositories (MSR). ACM, pp 73–84

Schuirmann D (1981) On hypothesis testing to determine if the mean of a normal distribution is contained in a known interval. Biometrics

Srikant R, Vu Q, Agrawal R (1997) Mining association rules with item constraints. In: International conference on knowledge discovery and data mining (KDD). AASI, pp 67–73
Westlake W (1981) Response to T.B.L. Kirkwood: bioequivalence testing—a need to rethink. Biometrics 37:589–594
Yazdanshenas AR, Moonen L (2011) Crossing the boundaries while analyzing heterogeneous component-based software systems. In: IEEE international conference on software maintenance (ICSM). IEEE, pp 193–202
Ying ATT et al (2004) Predicting source code changes by mining change history. IEEE Trans Softw Eng 30(9):574–586
Zanjani MB, Swartzendruber G, Kagdi H (2014) Impact analysis of change requests on source code based on interaction and commit histories. In: International working conference on mining software repositories (MSR), pp 162–171
Zheng Z, Kohavi R, Mason L (2001) Real world performance of association rule algorithms. In: SIGKDD international conference on knowledge discovery and data mining (KDD). ACM, pp 401–406
Zimmermann T et al (2005) Mining version histories to guide software changes. IEEE Trans Softw Eng 31(6):429–445

Leon Moonen is chief research scientist in the Software Engineering department at Simula Research Laboratory, Norway. His research aims at data-driven techniques and tools to support the understanding, assessment and evolution of large industrial software systems. Current projects include recommendation systems for smarter evolution and testing of software-intensive systems, anti-fragile and high integrity software engineering, and software analytics for continuous software quality and maintainability assessments. Dr. Moonen received his MSc (cum laude, Computer Science, 1996) and PhD (Computer Science, 2002) from the University of Amsterdam. He is a member of ACM, IEEE Computer Society, EAPLS and the ERCIM Working Group on Software Evolution.

Thomas Rolfsnes received his PhD (Informatics, Sept. 2017) from the University of Oslo. During his PhD, he has been employed by Simula Research Laboratory where his supervisor, Leon Moonen, is a chief research scientist. His research focused on improving change-recommendation systems for developers, in particular, recommendations based on patterns found in change-histories from sources such as Git. His efforts resulted in the thesis titled "Improving History-Based Change Recommendation Systems for Software Evolution". He has published his work in conferences such as SANER, SCAM, MSR and ASE, and was invited for two EMSE special issues. Dr. Rolfsnes is currently a data scientist at Egmont Publishing.

Dave Binkley is a Professor of Computer Science at Loyola University Maryland where he has worked since earning his doctorate from the University of Wisconsin in 1991. Dr. Binkley has been a visiting faculty researcher at the National Institute of Standards and Technology (NIST), worked with Grammatech Inc. on CodeSurfer development, and was a member of the Crest Centre at Kings' College London. Dr. Binkley's current research, partially funded by NSF, focuses on change recommendation and observational program analysis. He recently completed a sabbatical year working under Fulbright award with the researchers at Simula Research, Oslo Norway.

Stefano Di Alesio is a Chief Expert in the Transaction Monitoring Development department in Nordea Bank AB. He received his Ph.D. (Informatics, March 2015) from the University of Luxembourg. His interests revolve around leveraging machine learning and statistical analysis in order to provide quantitative insights that support business-critical decisions. His research explored the areas of model-driven, search-based, and reverse software engineering. He has published papers on these topics in widely recognized conferences and journals, including MODELS, ISSRE, ACM TOSEM, SANER and ASE. Dr. Di Alesio has also been a reviewer of several acknowledged software engineering journals, such as RESS, SoSyM, and EMSE.