# Practical Guidelines for Change Recommendation using Association Rule Mining

Leon Moonen* leon.moonen@computer.org
Stefano Di Alesio* stefano@simula.no
David Binkley‡ binkley@cs.loyola.edu
Thomas Rolfsnes* thomgrol@simula.no

* Simula Research Laboratory, Oslo, Norway
‡ Loyola University Maryland, Baltimore, Maryland, USA

## ABSTRACT

Association rule mining is an unsupervised learning technique that infers relationships among items in a data set. This technique has been successfully used to analyze a system's change history and uncover *evolutionary coupling* between system artifacts. Evolutionary coupling can, in turn, be used to *recommend* artifacts that are potentially *affected* by a given *set of changes* to the system. In general, the quality of such recommendations is affected by (1) the values selected for various parameters of the mining algorithm, (2) characteristics of the set of changes used to derive a recommendation, and (3) characteristics of the system's change history for which recommendations are generated.

In this paper, we empirically investigate the extent to which certain choices for these factors affect change recommendation. Specifically, we conduct a series of systematic experiments on the change histories of two large industrial systems and eight large open source systems, in which we control the size of the change set for which to derive a recommendation, the measure used to assess the strength of the evolutionary coupling, and the maximum size of historical changes taken into account when inferring these couplings. We use the results from our study to derive a number of practical guidelines for applying association rule mining for change recommendation.

## CCS Concepts

•**Software and its engineering** → **Software evolution;** *Software reverse engineering;* •**Information systems** → **Recommender systems;** *Association rules;*

## Keywords

Evolutionary coupling, association rule mining, parameter tuning, change recommendations, change impact analysis.



*ASE'16*, September 3–7, 2016, Singapore, Singapore
© 2016 ACM. 978-1-4503-3845-5/16/09...$15.00
http://dx.doi.org/10.1145/2970276.2970327

## 1. INTRODUCTION

A well-know effect of the continued evolution of a software system is the increasing disorder or entropy in the system: as a result of repeated changes, the number and complexity of dependencies between parts of the code grows, making it increasingly difficult for developers to foresee and reason about the effects of changes they make to a system.

Automated *change impact analysis* techniques [1, 2, 3, 4] aim to support a developer during system evolution by identifying the artifacts (e.g., files, methods, or classes) affected by a given change. Traditionally, these change impact analysis techniques are based on static or dynamic dependency analysis [5] (for example, by identifying the methods that call a changed method). More recently, promising alternative techniques have been proposed that identify dependencies by means of *evolutionary coupling*. These alternative approaches avoid certain limitations in existing techniques. For example, static and dynamic dependency analysis are generally language-specific, making them unsuitable for the analysis of heterogeneous software systems [6]. In addition, they can involve considerable overhead (e.g., dynamic analysis' need for code-instrumentation), and tend to over-approximate the impact of a change [7].

Evolutionary couplings differ from the ones found through static and dynamic dependency analysis, in that they are based on *how* the software was changed over time. In essence, evolutionary coupling aims to build on the developer's inherent knowledge of the dependencies in the system, which can manifest themselves by means of commit-comments, bug-reports, context-switches in an IDE, etc. In this paper, we consider *co-change* as the basis for uncovering evolutionary coupling. Co-change information can, for example, be extracted from a project's version control system [8], from its issue tracking system, or by instrumenting the development environment [9].

The most frequently used method for mining evolutionary coupling from co-change data is *association rule mining* (also called *association rule learning*) [10]. Various variants on the approach have been described in the software engineering literature [11, 12, 13, 14]. All of these approaches have in common that the technique is tuned with a number of parameters. While studying the literature, we found that there is little to no practical guidance on the tuning of these parameters, nor has there been a systematic evaluation of their impact on change recommendation quality. Moreover, we conjecture that, in addition to parameters of the min-

ing algorithm, the recommendation quality is also affected by characteristics of the change set that is used to derive the recommendation, and by characteristics of the system's change history from which the recommendation is generated.

In this paper, we empirically investigate the extent to which these factors affect change recommendation. Specifically, we conduct a series of systematic experiments on the change histories of two large industrial systems and eight large open source systems. In these experiments we control the size of change set used to derive recommendations, the measure used to assess the strength of the evolutionary coupling, and the maximum size of historical changes taken into account when inferring association rules. We use the results from our study to derive practical guidelines for applying association rule mining to construct software change recommendations.

**Contributions:** This paper presents three key contributions: (1) we investigate a previously unexplored area of tuning association rule mining parameters for software change recommendation; (2) we evaluate how recommendation quality is impacted by both characteristics of the change set used to derive a recommendation, and characteristics of change history used for learning; (3) we derive practical guidelines for improving the application of association rule mining in the derivation of software change recommendations.

**Overview:** The remainder of this paper is organized as follows: Section 2 provides background on targeted association rule mining. Section 3 describes limitations of classical approaches. Section 4 describes the setup of our empirical investigation whose results are presented in Section 5. Finally, Section 6 presents the related work and then Section 7 provides some concluding remarks.

## 2. ASSOCIATION RULE MINING

Agrawal et al. introduced the concept of *association rule mining* as a discipline aimed at inferring relations between *entities* of a dataset [10]. *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*. This rule can be read as *"if you buy bread, then you are also likely to buy butter."*

In the context of mining evolutionary coupling from co-change information, the entities involved are the files of the system[1] and a collection of transactions, denoted $\mathcal{T}$ (e.g., a history of commits to the version control system). A transaction $T \in \mathcal{T}$ is the set of files that were either changed or added while addressing a given bug or feature request, hence creating a *logical dependence* between the files [15].

As originally defined [10], 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* [16] focuses the generation of rules by applying a constraint. One 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 can drastically improve rule generation time [16].

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

When performing change impact analysis, rule constraints are based on a *change set*, for example, the set of files that were modified since the last commit. This set is also referred to as a *query* for which to search for potential impact. In the constrained case, only rules with at least one changed entity in the antecedent are created. The output of change impact analysis is the set of files from the system historically changed alongside the elements of the change set. For example, given the change set $\{a, b, c\}$, change impact analysis would uncover files that were changed when $a$, $b$, and $c$ were changed. The resulting impacted files are those found in the rule consequents. These files can be ranked based on the rule's *interestingness measure*.

To our knowledge, only a few targeted association rule mining algorithms have been considered in the context of change impact analysis: Zimmerman et al. [11], Ying et al. [12], Kagdi et al. [13], and our previous work [14]. In contrast, simpler *co-change* algorithms have been well studied in a variety of contexts [15, 17, 18, 19]. Existing targeted association rule mining algorithms and the co-change algorithms differ in terms of which subsets of the change set are allowed in the antecedent of generated rules. Consider, for example, the subsets of the change set $C = \{a, b, c, d\}$:

$$
\begin{aligned}
powerset(C) = \{&\{\}, && (1)\\
&\{a\}, \{b\}, \{c\}, \{d\}, && (2)\\
&\{a, b\}, \{a, c\}, \{a, d\}, \{b, c\}, \{b, d\}, \{c, d\}, && (3)\\
&\{a, b, c\}, \{a, b, d\}, \{a, c, d\}, \{b, c, d\}, && (4)\\
&\{a, b, c, d\}\} && (5)
\end{aligned}
$$

Both Zimmerman's and Ying's algorithms constrain the antecedent of rules to be equal to the change set, and hence only generate rules based on Line 5 (i.e., rules of the form $\{a, b, c, d\} \rightarrow X$). At the other end of the spectrum, co-change algorithms only generate rules from the singleton sets in Line 2, such as $\{a\} \rightarrow X$ or $\{b\} \rightarrow X$. In our previous work, we introduced TARMAQ, the most versatile among the existing algorithms [14]. TARMAQ generates rules whose antecedent can be from any of Lines 2, 3, 4, or 5. The particular line used is dynamically chosen based on the maximal overlap between the change set $C$ and the transactions that make up the history.

## 3. PROBLEM DESCRIPTION

Change impact analysis takes as input a set of changed entities (e.g., the files changed in a system), referred to as a *change set* or *query*, and outputs a set of potentially impacted entities. A common strategy for change impact analysis is to use association rule mining to capture the *evolutionary couplings* between such entities. In particular, entities are considered to be coupled if they have changed together in the past. Furthermore, the *strength* of a coupling depends on how frequently the entities have changed together. The stronger the coupling between two entities, the more likely it is that the entities are related to each other, and hence that one is impacted by changes to the other.

The strength of an evolutionary coupling is usually assessed using an *interestingness measure* applied to the association rules. The literature reports over 40 of these measures, which have been applied in a variety of domains [20]. It has been shown that the particular measure chosen has a significant impact over the ranking of rules, and consequently on the quality of the recommendations generated [21, 22, 23].

The following four examples illustrate how performance is affected by the two configuration parameters interestingness measure and the filter size, the size of the query and expected outcome, and finally the system characteristics.

**Example 1** *Consider the following history $\mathcal{T}$ of transactions:*

$$\mathcal{T} = \big[\{a,c\},\{b,c\},\{a,b,c\},\{a,b,x,y,z\},\{y,z\},\{x,y,z\}\big]$$

*and the change set $C = \{a,b\}$ where, based on $\mathcal{T}$ and $C$, the following four rules have been mined.*[2] *For each rule three measures,* confidence $\kappa$, prevalence $\phi$, *and* recall $\rho$ *[21] are given in parentheses. The confidence (recall) measures the ratio between the number of transactions where the antecedent and consequent changed, and the number of transactions where only the antecedent (consequent) changed. Prevalence measures the percentage of transactions in the history where the consequent changed.*

$$\begin{aligned} a,b \to c \quad & (\kappa = 1/2,\ \phi = 3/6,\ \rho = 1/3) \\ a,b \to x \quad & (\kappa = 1/2,\ \phi = 2/6,\ \rho = 1/2) \\ a,b \to y \quad & (\kappa = 1/2,\ \phi = 3/6,\ \rho = 1/3) \\ a,b \to z \quad & (\kappa = 1/2,\ \phi = 3/6,\ \rho = 1/3) \end{aligned}$$

In the example, the confidence values suggest that the four rules are all equally likely to hold, while prevalence suggests that $a,b \to x$ is inferior to the others, because $x$ is changed only twice in $\mathcal{T}$. On the other hand, recall suggests the opposite, that $a,b \to x$ is more likely than the others, because the co-occurrence of $\{a,b\}$ and $x$ in $\{a,b,x,y,z\}$ is more significant than the co-occurrence of $\{a,b\}$ and other files that changed more often without both $a$ and $b$. However, one's intuition matches the suggestion made by $\phi$ since $a$ and $b$ have a stronger relation with $c$ than with $x$. This is because $c$ appears also singularly with $a$ and $b$, while $x$ tends to change together with $y$ and $z$.

In general, the particular interestingness measure (or combination thereof) used to rank the rules is a parameter of the change recommendation algorithm. There exist a number of such parameters that affect how rules are ranked, and recommendations are generated.

**Example 2** *Consider again the history $\mathcal{T}$ of transactions:*

$$\mathcal{T} = \big[\{a,c\},\{b,c\},\{a,b,c\},\{a,b,x,y,z\},\{y,z\},\{x,y,z\}\big]$$

*and the change set $C = \{a,b\}$. However, assume that transactions larger than three files are discarded from the history when generating rules and calculating their interestingness. Based on $C$ and the filtered history, only the rule $a,b \to c$ is mined.*

In this second example, $x$, $y$, and $z$ will not be recommended, because $\{a,b,x,y,z\}$, the change set containing evidence for their recommendation, has been filtered out of $\mathcal{T}$. Intuitively, filtering out larger transactions from the history avoids generating rules from change sets that contain potentially unrelated files, and is a strategy used by most approaches [11, 12, 14]. However, filtering too aggressively may remove legitimate evidence of evolutionary coupling.

Values for configurable parameters are not the only criteria that can affect the recommendations generated.

**Example 3** *Consider the same history $\mathcal{T}$ of transactions:*

$$\mathcal{T} = \big[\{a,c\},\{b,c\},\{a,b,c\},\{a,b,x,y,z\},\{y,z\},\{x,y,z\}\big]$$

*and the change sets $C_1 = \{a\}$ and $C_2 = \{a,b\}$. The following rules are mined for $C_1$ and $C_2$.*

| *rules for $C_1$* | *rules for $C_2$* |
|---|---|
| $a \to b$ | $a,b \to c$ |
| $a \to c$ | $a,b \to x$ |
| $a \to x$ | $a,b \to y$ |
| $a \to y$ | $a,b \to z$ |
| $a \to z$ | |

$C_1$ and $C_2$ have different sizes, and hence, when used as queries they provide different amounts of evidence from which to make a recommendation. For example, consider the recommendation of $c$. In this case, $C_1$ contains only $a$ in support of the recommendation while $C_2$ contains both $a$ and $b$ and thus likely provides stronger support. While intuitively, larger queries can be expected to lead to stronger recommendations, queries that are too large may contain potentially unrelated files, degrading the quality of the recommendation.

An inverse argument applies to the *expected outcome*, the set of files that we seek to recommend. First consider the goal of predicting any one file from the expected outcome. Clearly, the larger the expected outcome, the easier the task of predicting one of them correctly. However, our goal is to recommend *all* of the files in the expected outcome. In that case, intuitively, a *smaller* expected outcome should be easier to recommend, as it is easier for a query to provide the necessary supporting evidence.

Finally, characteristics of the change history can affect the precision of a recommendation.

**Example 4** *Consider two transaction histories $\mathcal{T}_1$ and $\mathcal{T}_2$ from systems $\mathcal{S}_1$ and $\mathcal{S}_2$, respectively. Assume that the transactions of $\mathcal{T}_1$ are on average larger than those of $\mathcal{T}_2$ because of differences in development processes. For example, $\mathcal{S}_1$ might be developed with an agile process, where small change sets are committed frequently while $\mathcal{S}_2$ was developed with a less nimble process, where large change sets are committed less often.*

In this last example, queries and expected outcomes for $\mathcal{T}_1$ are likely to be smaller than those for $\mathcal{T}_2$. Thus entailing the implications discussed in Example 3, and also potentially differing impacts from transaction filtering which was described in Example 2.

# 4. EMPIRICAL STUDY

We perform a large empirical study to assess the extent to which the quality of change recommendations generated using targeted association rule mining is affected by (1) the values of various parameters of the mining algorithm, (2) characteristics of the change set used to derive the recommendation, and (3) characteristics of the system's change history. We use as a reference mining approach from our previous work, which has proven to perform consistently better than the previous state-of the art for software change impact analysis [14]. Our study investigates the performance of targeted association rule mining in the context of several software-systems, and several parameters configurations.

Specifically, we investigate the following four research questions:

**RQ 1** *To what extent does the interestingness measure affect the precision of change recommendation?*

**RQ 2** *To what extent does the size limit used to filter the transactions of the history affect the precision of change recommendation?*

[2] As a notational convenience, we write association rules without the set designation, for example, $a \to X$ in place of $\{a\} \to X$.

*Table 1: Characteristics of the last 30 000 transactions of the evaluated software systems*

| Software System | Nr. of files | Avg. commit size | History covered (in years) | Languages used* |
|---|---|---|---|---|
| Cisco Norway | 41701 | 6.20 | 1.07 | C++, C, C#, Python, Java, XML, other build/config (% undisclosed) |
| Kongsberg Maritime | 35111 | 5.08 | 15.97 | C++, C, XML, other build/config (% undisclosed) |
| Git | 3574 | 1.95 | 10.42 | C (45%), shell script (35%), Perl (9%), 14 other (11%) |
| HTTPD | 10021 | 4.80 | 19.78 | XML (56%), C (32%), Forth (8%), 19 other (4%) |
| Linux Kernel | 19768 | 2.14 | 0.48 | C (94%), 16 other (6%) |
| LLVM | 20745 | 4.41 | 2.15 | C++ (71%), Assembly (15%), C (10%), 16 other (4%) |
| JetBrains IntelliJ | 36162 | 3.38 | 1.58 | Java (71%), Python (17%), XML (5%), 26 other (7%) |
| Ruby on Rails | 5346 | 2.25 | 5.78 | Ruby (98%), 6 other (2%) |
| Subversion | 2915 | 2.76 | 7.61 | C (61%), Python (19%), C++ (7%), 15 other (13%) |
| Wine | 6679 | 2.47 | 4.61 | C (97%), 16 other (3%) |

* data on the languages used by the open source systems obtained from http://www.openhub.net.

**RQ 3** *To what extent do query size and expected outcome size affect the precision of change recommendation?*

**RQ 4** *To what extent does the average commit (transaction) size of the history affect the precision of change recommendation?*

The remainder of this section details our evaluation setup, and is organized as follows: in Section 4.1 we describe the software-systems included in the study. Sections 4.2 and 4.3 describe the interestingness measures and the history filtering. Section 4.4 describes two central concepts for the evaluation, *query generation* and *query execution*. Section 4.5 explains the generation of change recommendations. Finally, Section 4.6 explains how we measure performance.

## 4.1 Subject Systems

To assess change recommendation in a variety of conditions, we selected ten large systems having varying size and transaction frequency. 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. In particular, we consider a software product line for professional video conferencing systems developed by Cisco Norway. KM is a leading company in the production of systems for positioning, surveying, navigation, and automation of merchant vessels and offshore installations. Specifically, we consider a common software platform KM uses across various systems in the maritime and energy domain.

The other eight systems are well known open-source projects. Table 1 summarizes descriptive characteristics of the software systems used in the evaluation. The table shows that the systems vary from medium to large in size, with up to forty thousand different files committed in the transaction history. For each system, we considered the 30 000 most recent transactions (*commits*). This value represents a balance between too short a history, which would lack sufficient connections, and too long a history, which is hard to process efficiently and can contain outdated couplings caused by, for example, architectural changes. Across all ten systems, 30 000 transactions covers a significantly different development time span, ranging from almost 20 years in the case of HTTPD, to 6 months in the case of the Linux kernel. Most of the systems are heterogeneous, being developed in more than one programming language. Finally, we note that the median commit size for all the selected systems is one.

## 4.2 Interestingness Measures

Mining of change recommendations often uses or combines the *support* and *confidence* measures from the data mining community to separate interesting from uninteresting rules [11, 12, 14, 24]. However, as introduced in Section 3, over 40 interestingness measures have been defined in the literature to measure the strength of the evolutionary couplings mined using association rules. These measures are usually defined based on a probabilistic interpretation of the occurrence in the history of the rules antecedent and consequent. For example, given the rule $A \rightarrow B$ the probability $P(A)$ is the percentage of transactions from the history that include $A$, while the probability $P(A, B)$ is the percentage of transactions in the history that contain both $A$ and $B$. Therefore, the interestingness of the rule $A \rightarrow B$ is usually defined as a function of $P(A)$, $P(B)$, and various combinations thereof obtained through negations, fractions, and conditional operators. In this paper, we consider 39 interestingness measures commonly used in several data mining and machine learning applications. Due to space limitations, we only report their names in Table 2, and refer the reader to the original sources for the definitions [25, 26].

## 4.3 History Filtering

Several approaches for mining change recommendations start by filtering the history to remove transactions larger than a given size. This common heuristic seeks to avoid mining from transactions that do not contain relevant information on the evolutionary coupling of files, such as in the case of license updates or refactoring [11, 12, 24, 27]. Removing transactions from the history also has the effect of significantly speeding up the rule generation process.

This paper considers seven different transaction filtering sizes: 2, 4, 6, 8, 10, 20, and 30. Note that 30 is the threshold used in the work of Zimmermann et al. [11]. Starting from this value, we progressively consider more restrictive filtering. For each filter size $s$, we generate a filtered history $H_s$, from which we mine the association rules used to generate recommendations. In addition, we also consider the unfiltered history $H$. Since $H$ can be thought of as filtered with an infinite filter size, we simply refer to these as *eight* transaction filtering sizes in the rest of the paper.

## 4.4 Query Generation and Execution

A challenge in evaluating change recommendation techniques is what "*gold standard*" can be used to compare the generated recommendation against. A common strategy [11, 12, 13] is to take a transaction $T$ from the change history and randomly partition it into a non-empty query $Q$ and a non-empty expected outcome $E \stackrel{\text{def}}{=} T \setminus Q$. Since the files in a transaction are considered to be logically coupled [15], this

*Table 2: Overview of the 39 interestingness measures considered in our study*

| # | Interestingness Measure | # | Interestingness Measure | # | Interestingness Measure |
|---|---|---|---|---|---|
| 1 | Added Value | 14 | Interestingness Weighting Dependency | 27 | Odds Ratio |
| 2 | Casual Confidence | 15 | J Measure | 28 | One Way Support |
| 3 | Casual Support | 16 | Jaccard | 29 | Prevalence |
| 4 | Collective Strength | 17 | Kappa | 30 | Recall |
| 5 | Confidence | 18 | Klosgen | 31 | Relative Risk |
| 6 | Conviction | 19 | Kulczynski | 32 | Sebag Schoenauer |
| 7 | Cosine | 20 | Laplace Corrected Confidence | 33 | Specificity |
| 8 | Coverage | 21 | Least Contradiction | 34 | Support |
| 9 | Descriptive Confirmed Confidence | 22 | Leverage | 35 | Two Way Support |
| 10 | Difference Of Confidence | 23 | Lift | 36 | Varying Rates Liaison |
| 11 | Example and Counterexample Rate | 24 | Linear Correlation Coefficient | 37 | Yules Q |
| 12 | Gini Index | 25 | Loevinger | 38 | Yules Y |
| 13 | Imbalance Ratio | 26 | Odd Multiplier | 39 | Zhang |

stategy provides a suitable gold standard. The evaluation then assesses how well a change recommendation technique can recover expected outcome $E$ from query $Q$ using the transactions that came before $T$ in the change history.

From the history of each system, we randomly sample 1500 transactions,[3] which are used to generate 1500 queries (and their related expected outcomes). Each query is executed for each of the 39 interestingness measures reported in Table 2, using each of the eight filtering sizes introduced in Section 4.3. This setup yields a total of $1500 \cdot 39 \cdot 8 = 468\,000$ data points for each system, where each data point is a recommendation for a query.

## 4.5 Generating Change Recommendations

All queries are executed using TARMAQ, the targeted association rule mining algorithm we introduced in previous work [14]. Recall from Section 2 that executing a query $Q$ creates a set of association rules. In order to efficiently generate recommendations, TARMAQ only considers rules whose consequent contains a single file [14]. Generating a change recommendation for $Q$ consists of sorting the rules generated for $Q$ according to their interestingness score, and returning the ranked list of the consequents of the rules. We only consider the largest interestingness score for each consequent, i.e., we do not use rule aggregation strategies, such as the ones we proposed in previous work [26].

## 4.6 Performance Measure

To evaluate a recommendation we use *average precision* (AP), which is commonly used in Information Retrieval to assess the quality of a ranked list [28]:

**Definition 1 (Average Precision)** *Given a recommendation* R*, and an expected outcome* E*, the average precision of* R *is given by:*

$$AP(R) \stackrel{def}{=} \sum_{k=1}^{|R|} P(k) * \triangle r(k) \quad (6)$$

[3] For a normally distributed population of 30 000, a minimum of 651 samples is required to achieve a 99% confidence level with a 5% confidence interval. 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. The correcting 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 minimum sample size of $651/0.637 = 1022$ transactions. Hence, sampling 1500 transactions is more than sufficient to achieve a 99% confidence level with a 5% confidence interval.

*Table 3: Example of average precision calculation*

Consider $\{c, d, f\}$ as expected outcome in the following list:

| Rank ($k$) | File | $P(k)$ | $\triangle r(k)$ |
|---|---|---|---|
| 1 | c | 1/1 | 1/3 |
| 2 | a | 1/2 | 0 |
| 3 | f | 2/3 | 1/3 |
| 4 | g | 2/4 | 0 |
| 5 | d | 3/5 | 1/3 |

$$AP = 1/1 \cdot 1/3 + 1/2 \cdot 0 + 2/3 \cdot 1/3 + 2/4 \cdot 0 + 3/5 \cdot 1/3 \approx 0.75$$

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

Since we consider only rules with a single consequent, $\triangle r(k)$ will always be equal to either zero or $1/|E|$, because a rank either does not contain a file from expected outcome $E$, or it contains exactly one file from $E$. Table 3 illustrates the computation of $AP$, $P(k)$, and $\triangle r(k)$ given the ranked list $[c, a, f, g, d]$ and the expected outcome $\{c, d, f\}$.

As an overall performance measure for a group of factors (e.g., a given filtering size and interestingness measure) we use the *mean average precision* (MAP) computed over all the queries executed using the given factor combination.

# 5. RESULTS

This section presents the results of the study described in Section 4. A replication package is provided online.[4]

We begin by analyzing a key descriptive statistic, the commit size distribution, which is shown in Table 4. Because the majority (90.4%) of the commits contains less than six files, we focus the remainder of our analysis on this dominant subset of the data.

## 5.1 Analysis of Explanatory Variables

Central to our analysis is an ANOVA, used to explore the significance of the five explanatory variables program, filter size, expected outcome size, interestingness measure (or measure for short), and query size, together with all ten of their pairwise interactions. The Q-Q Plot of the residuals (left out) finds that these residuals follow a sufficiently normal distribution, especially considering ANOVA's performance in the presence of large data sets such as the nearly five million data points considered here.

[4] http://evolveit.bitbucket.org/publications/ase2016/replication/

*Table 4:* **Commit size** *frequency and cumulative percentage of all transactions considered.*

| commit size | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | (10,20] | (20,30] | >30 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| frequency | 157636 | 48903 | 23801 | 13767 | 8561 | 5546 | 3829 | 2814 | 2103 | 1611 | 6536 | 1778 | 2717 |
| cumulative % | 56.4% | 73.9% | 82.4% | 87.3% | 90.4% | 92.4% | 93.7% | 94.7% | 95.5% | 96.1% | 98.4% | 99.0% | 100.0% |

***Table 5: ANOVA Model - Explanatory variables and their pair-wise interactions, ordered by F-value, which provides a measure of the significance of the variable/interaction.***

| Explanatory Variable | Df | Mean Sq | F-value | *p*-value |
|---|---|---|---|---|
| expected outcome size | 1 | 2244 | 15930 | <0.0001 |
| program | 9 | 1643 | 11667 | <0.0001 |
| filter size | 1 | 1411 | 10018 | <0.0001 |
| measure | 38 | 452 | 3212 | <0.0001 |
| query size | 1 | 130 | 923 | <0.0001 |
| expected outcome sz:filter size | 1 | 122 | 864 | <0.0001 |
| program:query size | 9 | 55 | 390 | <0.0001 |
| program:expected outcome size | 9 | 47 | 332 | <0.0001 |
| program:filter size | 9 | 27 | 194 | <0.0001 |
| query size:filter size | 1 | 21 | 149 | <0.0001 |
| query size:measure | 38 | 21 | 145 | <0.0001 |
| query size:expected outcome sz | 1 | 20 | 142 | <0.0001 |
| expected outcome size:measure | 38 | 5 | 38 | <0.0001 |
| filter size:measure | 38 | 5 | 33 | <0.0001 |
| program:measure | 342 | 4 | 28 | <0.0001 |

The model is shown in Table 5, with terms ordered by their F-value. Even though all fifteen terms are highly statistically significant, the five variables make a larger contribution than the interaction terms (i.e., have larger F-values).

## 5.2 Impact of Interestingness Measures

Using the ANOVA, we continue our analysis by considering the first research question:

**RQ 1** *To what extent does the interestingness measure affect the precision of change recommendation?*

To visually explore the influence of interestingness measures, Figures 1–4 show interaction plots of measure with respectively program, query size, expected outcome size, and filter size (where *inf* indicates no filtering). The overall pattern, evident in these graphs, is that measure is largely independent of

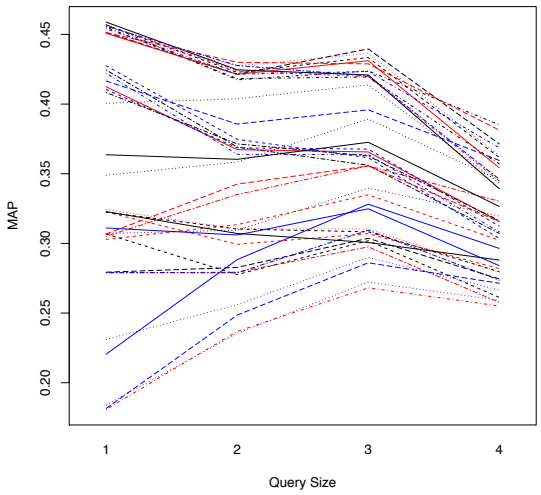


***Figure 2: Interaction plot of* measure *and* query size. *The* measures *are shown unlabeled to avoid cluttering the plot.***

program, query size, expected outcome size, and filter size. Statistically, the interactions are significant primarily due to a few interestingness measures that "buck the trend" (producing line crosses in the plots). This visually evident "mostly independent" observation is supported by the comparatively low F-values for the four interactions involving "measure" (they are four of the five smallest in the model shown in Table 5).

The low F-values and the interactions plots support the notion that there are consistent best measures for change recommendation based on evolutionary coupling. Not surprisingly, there is no single best measure. Tukey's honestly significant difference (HSD) test finds that eleven measures populate the top equivalence class. These are shown in Table 6. Note that even though their MAP values are slightly

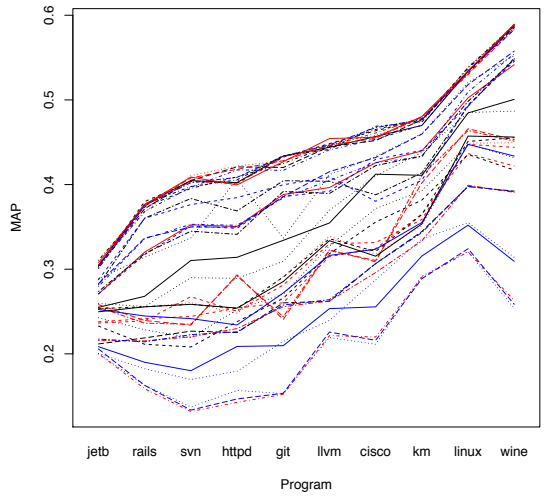


***Figure 1: Interaction plot of* measure *and* program. *The* measures *are shown unlabeled to avoid cluttering the plot.***

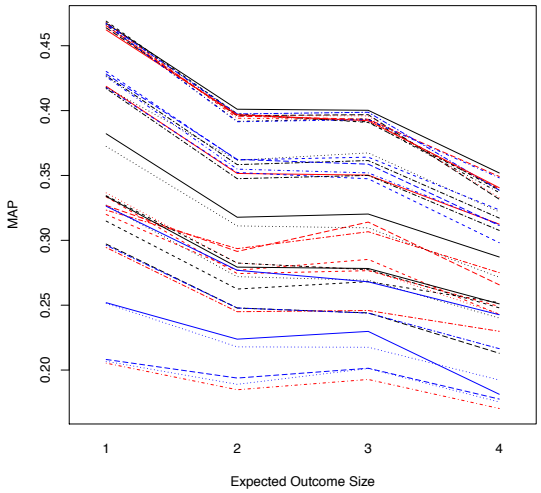


***Figure 3: Interaction plot of* measure *and* expected outcome size. *The* measures *are shown unlabeled to avoid clutter.***

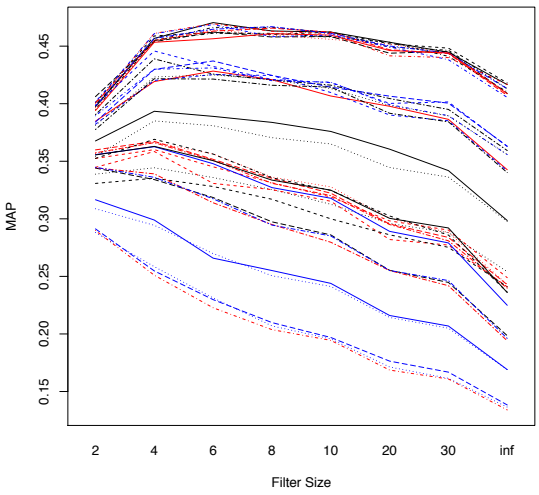


***Figure 4: Interaction plot of* measure *and* filter size. *The* measures *are unlabeled to avoid cluttering the plot.***

different, the measures in this class are not statistically different from each other. Also observe that Table 6 includes the classic measures confidence and support. Their presence reinforces a result from the recent work of Le and Lo [25]. While their work considers a different problem (the effect of different interestingness measures in rule-based specification mining), they too conclude that the standard measures work well. Thus in summary for RQ1, we find that to a large extent measure's influence on average precision is consistent across differences in other variables and that the traditional measures are top performers.

## 5.3 Impact of Transaction Filtering

Next, we turn to answering our second research question:

**RQ 2** *To what extent does the size limit used to filter the transactions of the history affect the precision of change recommendation?*

The results for this question are rather surprising. When filtering the history it is common to remove large commits as they are assumed to reflect licencing changes and alike. Prior work has typically used 30 as a cut off, while some experiments have used values as high as 100 [11, 12].

One might wonder if filter sizes 10, 20, or 30 yield any differences, because the average commit sizes for the ten systems studied are smaller than these filter sizes. The answer can be in Figure 4, which shows that no filtering (filter size *inf*) performs worse than filtering commits larger than 30, which in turn performs worse than filtering commits larger than 20, and similarly for filter size 10. Analysis using Tukey's HSD also shows that the MAPs continue to significantly decrease for filter sizes 10, 20, 30, and no filtering.

***Table 6: Top group of Tukey's HSD for* measure**

| interestingness measure | MAP | group |
|---|---|---|
| casual confidence | 0.446 | a |
| klosgen | 0.446 | a |
| descriptive confirmed confidence | 0.446 | a |
| added value | 0.445 | a |
| collective strength | 0.445 | a |
| loevinger | 0.445 | a |
| confidence | 0.444 | a |
| leverage | 0.443 | a |
| example and counterexample rate | 0.443 | a |
| difference of confidence | 0.443 | a |
| support | 0.442 | a |

***Table 7: Tukey's HSD for* filter size *(for the top* measures*).***

| filter size | | |
|---|---|---|
| size | mean | group |
| 6 | 0.464 | a |
| 8 | 0.463 | ab |
| 10 | 0.459 | bc |
| 4 | 0.457 | c |

For the eight filter sizes studied, Figure 5 shows the interaction between filter size and program for two subsets of the data. On the left, the data for all measures is shown, while on the right, the data for the top group of measures identified in Table 6 is shown. It is clear from this figure that smaller commits contain much more useful information than larger commits as the best value for most systems occurs at a filter size of only four or six when considering all measures, and at a filter size of six to ten for the top group of measures.

Considering only the top group of measures, Tukey's HSD test, shown in Table 7, groups filter size six and eight in the top equivalence class. Thus, in answer to RQ2, aggressive history filtering appears to retain only high value commits that support the creation of high quality association rules.

This is a noteworthy finding, as it suggests that filtering should be applied much more aggressively. Although small commits capture stronger couplings, prior work has not exploited this fact. For example, as discussed in Section 2, ROSE [11] only generates association rules whose antecedent is equal to the query. In particular, this means that the algorithm can not generate rules from transactions smaller than the query. In contrast, the more recent algorithm Tarmaq [14] can exploit partial matches between the query and historical transactions, and can thus be used when considering only small high-focus commits.

## 5.4 Impact of Query Size and Expected Outcome Size

Our third research question considers the impact of the query and the expected outcome:

**RQ 3** *To what extent do query size and expected outcome size affect the precision of change recommendation?*

Because all the explanatory variables *and* all their interactions are statistically significant, the resulting coefficient equation is very complex (it includes almost 100 terms!). Rather than state this equation, we zoom in on a few of its key terms. Specifically those involving query size, expected outcome size, and commit size, which is the sum of query size and expected outcome size. These three include three significant interactions and thus the interpretation of even just this subset is complex.

Fortunately the practical range of these explanatory variables is limited (as shown in Table 4, where 90% of the data is covered by commit sizes less than or equal to 5). The use of a small range of values makes it is viable to enumerate all possible combinations. Table 8 unravels the interactions for the various combinations of query size and expected outcome size and uses color to indicate entries with the same commit size. For example, the hardest case (where the MAP value gets the lowest contribution) is for a query size of 1 and an expected outcome size of 4. This high-challenge level is expected as this

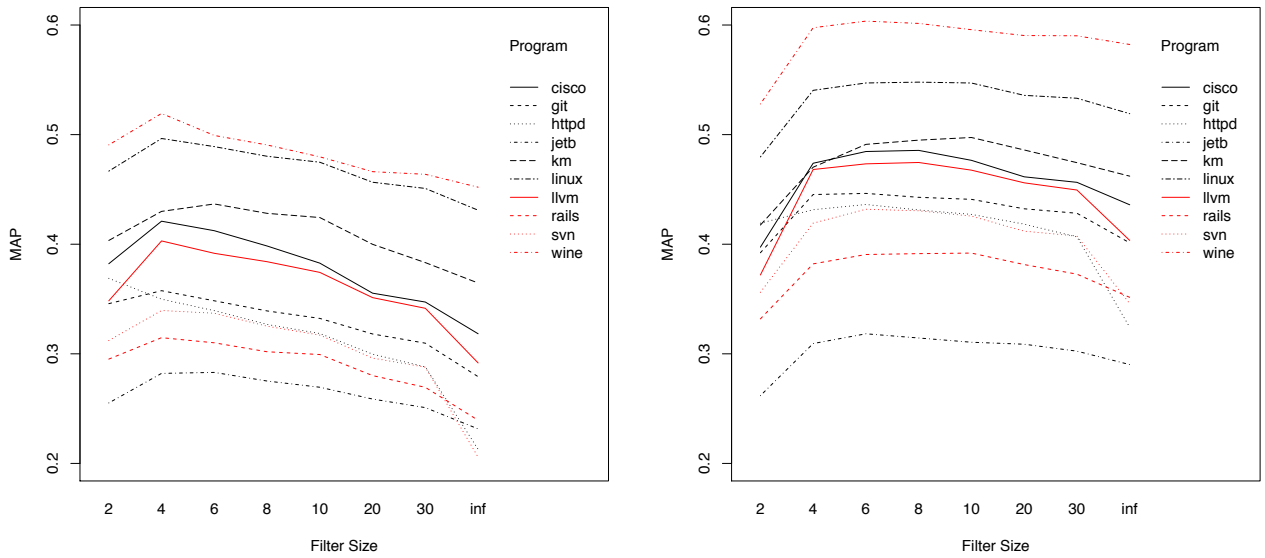


***Figure 5: Interactions of* filter size *and* program. *To the left for all* measures, *and to the right for the top group of* measures.**

case aims to predict the largest of the outcomes based on minimal information.

The table makes it possible to look at trends. Starting with commit size, following the diagonal with green cells, which have a commit size of five, we see that the prediction gets easier (the contribution to MAP increases) as the expected outcome size decreases, and the query size increases. This same pattern is seen with the other commit sizes (other colors). A similar pattern is also clear for a fixed query size (any given column of the table). In this case, the prediction gets more difficult (the contribution to MAP decreases) as expected outcome size grows.

Finally, for a fixed expected outcome size (a given row in the table) the trend is less clear: one might expect the prediction to get easier as query size increases, because there is more information to build on. We can indeed see this pattern in the row for expected outcome size of three. However, the row where expected outcome size is two is approximately flat, and most surprisingly, the top row where expected outcome size is one shows a clearly decreasing trend. Thus, the data shows that the prediction of a single result gets more difficult (the contribution to MAP decreases) as query size grows.

We can not completely explain this decrease, but conjecture that it is an effect related to small commits (such as those shown in yellow and blue) being more focussed than larger commits. In other words, larger commits are more likely to introduce noise into the recommendation. It may also be an artifact of using coarse-grained (file level) change data, and thus the expected trend would be visible with more fine-grained (method level) change data. We plan to investigate this phenomena in more detail in our future work.

***Table 8: Relative impact of* query size *and* expected outcome size *and their interactions. The colors indicate the commit size, as shown in the legend on the right.***

| expected outcome size | query size 1 | 2 | 3 | 4 | commit size |
|---|---|---|---|---|---|
| 1 | 10 | 9 | 8 | 5 | 2 |
| 2 | 5 | 5 | 4 | | 3 |
| 3 | 2 | 3 | | | 4 |
| 4 | 1 | | | | 5 |

## 5.5 Impact of Average Transaction Size

Finally, we consider our last research question:

**RQ 4** *To what extent does the average commit (transaction) size of the history affect the precision of change recommendation?*

To analyze this question we perform a linear regression using the average commit size to predict the MAP value for each system. This data is shown in Table 9. The regression finds no significant correlation between the average commit size and MAP. Thus in answer to RQ4: the average commit size in the change history has no impact on the precision of change recommendation. In retrospect, considering how Table 4 shows that the frequency of commit sizes is rather skewed, the finding that the average commit size is not a good predictor of the algorithm's precision should come as no surprise.

## 5.6 Threats to Validity

**Commits as a basis for evolutionary coupling:** The evaluation presented in this paper is grounded in predictions made from analyzing patterns in change histories. However, as pointed out by Herzig et al. [29, 30], the transactions in the change histories could contain unrelated files, or could miss related files added in subsequent transactions. In our case, the software systems studied except KM use Git for version

***Table 9: Average* commit size *and MAP for each* program.**

| program | average commit size | MAP |
|---|---|---|
| cisco | 6.20 | 0.375 |
| git | 1.96 | 0.328 |
| httpd | 4.80 | 0.311 |
| jetbrains | 3.39 | 0.263 |
| km | 5.08 | 0.408 |
| linux | 2.15 | 0.468 |
| llvm | 4.42 | 0.360 |
| rails | 2.25 | 0.288 |
| subversion | 2.77 | 0.302 |
| wine | 2.47 | 0.482 |

control, which provides developers with tools for amending commits and rewriting history. Therefore, we argue that the impact of incomplete or entangled transactions is less significant in our case than it would be using other version control systems such as SVN or CVS. In our related work, we also discuss methods for grouping related commits that are orthogonal to our approach, and could be thus used to refine the change history.

**Variation in software systems:** We conducted our experiments on two industrial systems and eight large open source systems. These systems vary considerably in size and frequency of transactions (commits), which should provide an accurate picture of the performance in various contexts (see Table 1). However, despite our careful choice, we are likely not to have captured all possible variations.

**Random sampling errors:** Our experiment is based on taking a large number of random samples from the change history of each system. Although we use uniform random sampling, there is the possibility that our samples do not accurately represent the actual change history, for example the distribution of transaction sizes considered in the sample may be different full history. We address this particular issue applying a chi-squared test to validate that our samples are representative of the population (the change histories). An alternative approach would be to use a stratified sampling strategy where one extracts samples of each transaction size in proportion to their frequency in the complete history. The results of the chi-squared test indicate that such an alternative strategy is not necessary.

**Implementation:** We have implemented the experiments using Ruby and thoroughly tested all algorithms and measures studied in this paper. We also performed the statistical analysis using standard methods provided by R. However, we can not guarantee the absence of implementation errors, which may affect our results.

# 6. RELATED WORK

Recent research highlighted that the performance of data mining algorithms is impacted by their configuration parameters [31]. A common strategy for tuning these parameters consists in optimizing their values over half of the test set (*inner* validation), and then using the other half to assess the performance of the optimal parameters (*outer* validation) [32]. While this strategy is useful to fine-tune the parameters for a particular type of data, it does not provide insights on how the algorithm's performance is affected by the parameters [33]. In the context of association rule mining, several authors have noted the need to investigate the impact of parameter settings on the quality of the generated rules [34, 35, 36].

Therefore, in this paper, we investigate the extent to which the quality of change recommendation via association rule mining is affected by three factors (1) the values selected for various parameters of the mining algorithm, (2) characteristics of the change set used to derive the recommendation, and (3) characteristics of the system's change history for which recommendations are generated. In the rest of this Section, we distinguish related work on these three aspects, focusing on the area of software maintenance and evolution.

**Parameters in Association Rule Mining:** In general, association rule mining algorithms differ from each other in the data structures used to represent transactions, and the strategy used to select transactions relevant to a given query [37]. However, the majority of such algorithms are characterized by similar parameters. Among those, this paper focuses on the maximum size of transactions used to generate rules (*transaction filtering size*), and on the metric measuring the strength of evolutionary couplings inferred by those rules (*interestingness measure*). In the context of software change impact analysis, several studies remark on the importance of discarding from the history large change sets that are likely to contain unrelated files. For example, Kagdi et al [24], Zimmermann et al. [11] and Ying et al. [12] propose filtering transactions larger than 10, 30, and 100 items, respectively. However, these authors generally do not report how such threshold values have been chosen, nor do they explore the impact of alternative values on recommendation precision. Several authors have also remarked that selecting the right interestingness measure for a problem domain can significantly affect recommendation accuracy [20, 22, 23, 25]. In particular, Le and Lo compared 38 measures in the context of rule-based specification mining, highlighting the need to look beyond standard support and confidence to find interesting rules [25]. We are not aware of similar studies carried out in the context of software change impact analysis.

**Characteristics of the Change Set:** Targeted association rule mining uses the query to drive the generation of rules [16]. In general, particular characteristics of the query can effect the quality of recommendations. For example, in the context of software change impact analysis, we identified in previous work a particular class of queries for which the most common targeted association rule mining approaches cannot generate recommendations [14]. Note that it is common practice to validate targeted association rule mining approaches by sampling random queries from the change history [11, 12]. While this strategy ensures that the approach is evaluated on a variety of change sets from the history, authors in general do not consider how query size might affect the quality of the recommendations. Among others, 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 affected the quality of the predictions generated [19].

**Characteristics of the Change History:** Several past studies have proposed strategies to group transactions in the change history [11, 13, 38]. The reason for doing so is that a developer might (accidentally) commit an incomplete transaction, and update the remaining files related to the same change in a separate transaction. As a consequence, entities that are logically coupled via the same change might become spread across several transactions in the change history. Nevertheless, in modern version control systems transactions are stashed in the user's local repository and can be amended before they are finalized at a later stage, thereby reducing the risk of committing incomplete transactions. In contrast, the question whether properties of the change history, such as average commit size and frequency, affect the quality of change recommendation is less studied. In this direction, German carried out an empirical study on several open source projects, finding that the revision histories of most systems contains mostly small commits [39]. Alali et al. also investigated the total number of lines modified in the files and the total number of hunks (continuous groups of lines) that were changed [40]. Kolassa et al. performed

a similar study on commit frequency, reporting an average inter-commit time of about three days [41]. However, none of these studies investigate how characteristics of the change history affect the quality of the change recommendation.

# 7. CONCLUDING REMARKS

**Conclusions:** Association rule mining is an automated, unsupervised, learning technique that has been successfully used to analyze a system's change history and uncover *evolutionary coupling* between its artifacts. These evolutionary couplings can be used to *recommend* artifacts that are potentially *impacted by a given set of changes* to the system. Doing so can help developers address the increasing entropy caused by repeated software maintenance and evolution.

In this paper, we present a series of experiments using the change histories of two large industrial systems and eight large open source systems. For each system, we randomly extract a representative sample of the transactions from the change history, randomly split each of these transactions in two parts, a query and an expected outcome, and analyzed how several parameters affect the prediction of the expected outcome based on the query.

We draw the following conclusions: first, our analysis shows that the impact of interestingness measure is largely independent of the particular program, query size, and expected outcome size. This means that it is possible to identify the best measures for change recommendations based on evolutionary coupling. To do so, we clustered measures using Tukey's HSD where the measures in each cluster are not statistically different from each other. We find that the standard measures confidence and support are in the top equivalence class; thus statistically no other measure provides better performance than these two. This result is in line with the earlier findings by Lo and Le [25], who conducted a related study of interestingness measures for rule-based specification mining.

Second, learning from a change history that is filtered to remove transactions larger than six to eight files yields the best results, regardless of program, query size, and expected outcome size. Furthermore, the average precision produced by higher thresholds is significant worse.

Third, for smaller commits (up to a transaction size of five, which includes just over 90% of all transactions), the smaller the expected outcome, the higher the average precision. This suggests that for relatively small commits the less information you want to predict, the better the results. When we extend the analysis to larger commits (which are the minority), larger expected outcomes lead to higher precision. Combined these two observations suggest that predicting a few missed files is easy, while predicting a large number of missed files from limited information is hard. A similar pattern is seen when comparing smaller and larger commit sizes.

Fourth, for the smaller commits one would expect that larger queries would produce better average precision. Our data only partially supports this expectation. Looking at the rows of Table 8 two effects are competing with each other. A simplified version of the coefficient equation used to fill in this table includes both $+\text{query size}$ and $-\text{query size}^2$. The data in the table reflects these two factors. Initially the linear term dominates and increased query size increases the average precision. However, for larger values of query size the negative quadratic term dominates and the average precision is reduced. While there is insufficient data to establish a trend, the peak value appears to be a function of the expected outcome size. In the first row the value is less than or equal to one, for the second row it is two while for the third and fourth rows it is greater than two. Clearly, this effect warrants future research consideration.

**Guidelines:** From our conclusions, we derive the following practical guidelines for applying association rule mining in the context of software change recommendation: (1) stick with default interestingness measures, as they are in the top perform class and have a long proven track record, (2) learn from a subset of the change history that includes only the smaller transaction sizes to avoid noise. We have good experience with transaction sizes up to about eight, but these values may depend on the actual transaction sizes in the particular history being analyzed.

**Future Work:** We consider the following directions for future extension of this work: (1) *Use of fine-grained co-change information.* Although our algorithms are granularity agnostic, we expect interesting differences in the *change patterns* used to mine evolutionary coupling at different levels of granularity. For instance, consider how multiple method-level changes in the same file get abstracted into one change at the file-level. These differences will likely have an effect on parameter values, such as the transaction filter size. (2) *Language-specific change recommendations.* The change recommendation mining in this paper is independent of the programming language of the artifacts in the change history. This has advantages in the context of heterogeneous systems, but it also potentially increases noise. Consider a software development project that consists of a front-end written in Java and a back-end written in C. In this case, front-end Java developers are not likely to benefit from recommendations involving the C code. In contrast, *full-stack integrators*, responsible for connecting the front and back end, would benefit from multi-language recommendations. In this context, there is value in being able to provide *language-specific change recommendations*, i.e., recommendations that are based on filtering the change history for artifacts of a particular type. (3) *Impact of software development styles.* The third area of related work will involve a deeper analysis of the impact of various types of software systems and their software development styles on the quality of change recommendations. Aspects that may play a role here include frequency of commits, tendency to piggyback/tangle commits, and code ownership. The question is how to classify systems based on these factors. An initial study might compare industrial software systems with open source systems, as there are often clear differences between the two styles. With that comparison in mind, it is of interest to observe that the two industrial systems that were considered in this study could not be distinguished from the open source systems based on the interaction plots (Figures 1 to 5), nor could be distinguished in terms of MAP or using Tukey's HSD analysis. However, this observation is based on a very small sample size. A more complete comparison should consider a larger sample of industrial systems, unfortunately acquiring such may not be an easy task. We welcome suggestions for tackling this challenge.

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

## References


[1] G. Canfora and L. Cerulo. "Impact Analysis by Mining Software and Change Request Repositories". In: *International Software Metrics Symposium (METRICS)*. IEEE, 2005, pp. 29–37.

[2] M.-A. Jashki, R. Zafarani, and E. Bagheri. "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, 2008, pp. 84–90.

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

[4] M. B. Zanjani, G. Swartzendruber, and H. Kagdi. "Impact analysis of change requests on source code based on interaction and commit histories". In: *International Working Conference on Mining Software Repositories (MSR)*. 2014, pp. 162–171.

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

[6] A. R. Yazdanshenas and L. Moonen. "Crossing the boundaries while analyzing heterogeneous component-based software systems". In: *International Conference on Software Maintenance (ICSM)*. IEEE, 2011, pp. 193–202.

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

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

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

[10] R. Agrawal, T. Imielinski, and A. Swami. "Mining association rules between sets of items in large databases". In: *ACM SIGMOD International Conference on Management of Data*. ACM, 1993, pp. 207–216.

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

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

[13] H. Kagdi, S. Yusuf, and J. I. Maletic. "Mining sequences of changed-files from version histories". In: *International Workshop on Mining Software Repositories (MSR)*. ACM, 2006, pp. 47–53.

[14] T. Rolfsnes et al. "Generalizing the Analysis of Evolutionary Coupling for Software Change Impact Analysis". In: *International Conference on Software Analysis, Evolution, and Reengineering (SANER)*. IEEE, 2016, pp. 201–212.

[15] H. Gall, K. Hajek, and M. Jazayeri. "Detection of logical coupling based on product release history". In: *International Conference on Software Maintenance (ICSM)*. IEEE, 1998, pp. 190–198.

[16] R. Srikant, Q. Vu, and R. Agrawal. "Mining Association Rules with Item Constraints". In: *International Conference on Knowledge Discovery and Data Mining (KDD)*. AASI, 1997, pp. 67–73.

[17] T. Ball, J. Kim, and H. P. Siy. "If your version control system could talk". In: *ICSE Workshop on Process Modelling and Empirical Studies of Software Engineering*. 1997.

[18] D. Beyer and A. Noack. "Clustering Software Artifacts Based on Frequent Common Changes". In: *International Workshop on Program Comprehension (IWPC)*. IEEE, 2005, pp. 259–268.

[19] A. Hassan and R. Holt. "Predicting change propagation in software systems". In: *International Conference on Software Maintenance (ICSM)*. IEEE, 2004, pp. 284–293.

[20] L. Geng and H. J. Hamilton. "Interestingness measures for data mining". In: *ACM Computing Surveys* 38.3 (2006).

[21] R. Kamber, Micheline and Shinghal. "Evaluating the Interestingness of Characteristic Rules". In: *KDD*. 1996, pp. 263–266.

[22] P.-N. Tan, V. Kumar, and J. Srivastava. "Selecting the right objective measure for association analysis". In: *Information Systems* 29.4 (2004), pp. 293–313.

[23] K. Mcgarry. "A survey of interestingness measures for knowledge discovery". In: *The Knowledge Engineering Review* 20.01 (2005), p. 39.

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

[25] T.-d. B. Le and D. Lo. "Beyond support and confidence: Exploring interestingness measures for rule-based specification mining". In: *International Conference on Software Analysis, Evolution, and Reengineering (SANER)*. IEEE, 2015, pp. 331–340.

[26] T. Rolfsnes et al. "Improving change recommendation using aggregated association rules". In: *International Conference on Mining Software Repositories (MSR)*. ACM, 2016, pp. 73–84.

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

[28] H. J. Hilderman, Robert and Hamilton. *Knowledge discovery and measures of interest*. Springer Science & Business Media, 2013.

[29] K. Herzig and A. Zeller. "The impact of tangled code changes". In: *Working Conference on Mining Software Repositories (MSR)*. IEEE, 2013, pp. 121–130.

[30] K. Herzig, S. Just, and A. Zeller. "The impact of tangled code changes on defect prediction models". In: *Empirical Software Engineering* 21.2 (2016), pp. 303–336.

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

[32] I. H. Witten, E. Frank, and M. A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2011.

[33] E. Keogh, S. Lonardi, and C. A. Ratanamahatana. "Towards parameter-free data mining". In: *Sigkdd* (2004), pp. 206–215.

[34] Z. Zheng, R. Kohavi, and L. Mason. "Real world performance of association rule algorithms". In: *SIGKDD international conference on Knowledge discovery and data mining (KDD)*. ACM, 2001, pp. 401–406.

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

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

[37] A. Silva and C. Antunes. "Constrained pattern mining in the new era". In: *Knowledge and Information Systems* 47.3 (2016), pp. 489–516.

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

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

[40] A. Alali, H. Kagdi, and J. Maletic. "What's a Typical Commit? A Characterization of Open Source Software Repositories". In: *International Conference on Program Comprehension (ICPC)*. IEEE, 2008, pp. 182–191.

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