


Aggregating Association Rules to Improve Change Recommendation

Thomas Rolfsnes¹  · Leon Moonen¹ ·
Stefano Di Alesio¹ · Razieh Behjati¹ ·
Dave Binkley²

© Springer Science+Business Media, LLC 2017

Abstract As the complexity of software systems grows, it becomes increasingly difficult for developers to be aware of all the dependencies that exist between artifacts (e.g., files or methods) of a system. Change recommendation has been proposed as a technique to overcome this problem, as it suggests to a developer relevant source-code artifacts related to her changes. Association rule mining has shown promise in deriving such recommendations by uncovering relevant patterns in the system's change history. The strength of the mined association rules is captured using a variety of interestingness measures. However, state-of-the-art recommendation engines typically use only the rule with the highest interestingness value when more than one rule applies. In contrast, we argue that when multiple rules apply, this indicates collective evidence, and aggregating those rules (and their evidence) will lead to more accurate change recommendation. To investigate this hypothesis we conduct a large empirical study of 15 open source software systems and two systems from our industry partners.

Communicated by: Romain Robbes, Christian Bird, and Emily Hill

- ✉ Thomas Rolfsnes
thomgrol@simula.no
- ✉ Leon Moonen
leon.moonen@computer.org
- Stefano Di Alesio
stefano@simula.no
- Razieh Behjati
behjati@simula.no
- Dave Binkley
binkley@cs.loyola.edu

¹ Simula Research Laboratory, Oslo, Norway

² Loyola University Maryland, Baltimore, MD, USA

We evaluate association rule aggregation using four variants of the change history for each system studied, enabling us to compare two different levels of granularity in two different scenarios. Furthermore, we study 40 interestingness measures using the rules produced by two different mining algorithms. The results show that (1) between 13 and 90% of change recommendations can be improved by rule aggregation, (2) rule aggregation almost always improves change recommendation for both algorithms and all measures, and (3) fine-grained histories benefit more from rule aggregation.

Keywords Evolutionary coupling · Targeted association rule mining · Rule aggregation · Interestingness aggregator · Change recommendations · Change impact analysis

1 Introduction

The evolution of a software system is accompanied by continuous growth in the number and complexity of interactions between system artifacts. Thus, over time, it becomes increasingly challenging for developers to manage the impact of changes made to the system. To address this problem, Change Impact Analysis (CIA) has been proposed to identify software artifacts (e.g., files, classes, and methods) affected by a given change (Canfora and Cerulo 2005; Jashki et al. 2008; Ren et al. 2004; Zanjani et al. 2014). CIA is typically used to derive a *software change recommendation* as direct feedback to a developer regarding artifacts that may also be in need of change.

Classical CIA employs static and dynamic dependence analysis (Bohner and Arnold 1996). For example, by identifying the methods that call a changed method. However, both static and dynamic dependence analysis are generally language specific, making them unsuitable for heterogeneous software systems (Yazdanshenas and Moonen 2011). In addition, because of its conservative nature static analysis is known to over-approximate solutions (Podgurski and Clarke 1990), while dynamic analysis can involve considerable run-time overhead (Yong and Horwitz 2002).

These limitations have led to increasing interest in alternative approaches (Bird et al. 2015). Prominent among these is the identification of dependences using *evolutionary coupling*. Such couplings are based on *how* a software system changes over time, something that is missed by static and dynamic dependence analysis. In essence, evolutionary coupling taps into the developers' inherent knowledge of the dependencies in the system. This *co-change* knowledge can manifest itself in several ways: commits, bug-reports, context-switches in an IDE, etc. It can thus be extracted, for example, from the project's version control system (Eick et al. 2001), its issue tracking database, or by instrumenting the development environment (Robbes et al. 2008).

This paper explores the use of co-change information extracted from *git* repositories as the basis for uncovering the evolutionary coupling. Doing so exploits the fact that dependent artifacts are likely to change together. Specifically, we mine evolutionary couplings using *association rule mining* (Agrawal et al. 1993), an unsupervised machine learning approach that reveals relations among items in a data set. The relative importance of mined rules is typically captured by an *interestingness measure* (e.g., *support* or *confidence*), which seeks to capture the relative utility of each mined rule (Kamber and Shinghal 1996; Tan et al. 2004; Geng and Hamilton 2006). In scenarios where multiple rules can be applied, it is common to consider only the single rule with the highest interestingness value (Toivonen et al. 1995).

In contrast, we hypothesize that the aggregation of such rules can be exploited to provide improved recommendations.

Contributions: This article builds upon our previous work with *association rule aggregation* (Rolfesnes et al. 2016). In particular, it extends our previous work in six key respects: (1) we significantly scale up the empirical study of association rule aggregation in the context of change recommendation (2) we include a study of aggregation performance on the level of individual software systems (3) we introduce and study a new aggregation function, *Hyper Cumulative Gain* (4) we study the effect that history granularity (files, methods etc.) has on the precision gained from rule aggregation (5) we formally prove that all studied aggregation functions satisfy our previously proposed aggregation properties, and finally (6) we extend our review of the related work.

Overview: The rest of this article is organized as follows: Section 2 provides background on targeted association rule mining. Section 3 describes limitations of the current state-of-art approaches to association rule mining. Section 4 overviews the interestingness measures used to weigh the mined association rules. Section 5 introduces the aggregation of association rules into hyper-rules, while Section 6 presents the aggregation functions considered in the experiments. Section 7 describes the setup of our empirical evaluation whose results are presented and discussed in Section 8. Section 9 discusses potential threats that could affect the validity of our conclusions, while Section 10 presents related work. Finally, Section 11 concludes the article with final remarks and outlines future work.

2 Association Rule Mining

Agrawal et al. introduced the concept of *association rule mining* as the discipline aimed at inferring relations between *entities* of a data set (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 of entities. 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 historical co-change data, the entities are the files of the system¹ and the sequence (history) \mathcal{T} of transactions, is the sequence of past *commits*. More specifically, a transaction $T \in \mathcal{T}$ is the set of files that were either changed or added while addressing a given bug or feature addition, hence creating a *logical dependence* between the files (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 to those that satisfy a given constraint, e.g., stating that the antecedent of all mined rules has to belong to a particular set of files. Doing so reduces the number of rules generated and thus can significantly improve the rule generation time (Srikant et al. 1997).

¹Other levels of granularity are possible as our algorithms are granularity agnostic. Thus, our initial description at the file level is without loss of generality. Provided suitably co-change data the algorithms can relate methods or variables just as well as files, a fact which will be exploited later on in the paper.

When generating change recommendations, rule constraints are based on a *change set*: the set of modified files since the last commit. In this case, only rules with at least one changed artifact in the antecedent are generated. The output of a change recommendation is the set of files that are historically changed along with the elements of the change set. For example, given the change set $\{a, b, c\}$, a change recommendation would consist of the files that were changed when a , b , and c were changed. The recommended files are those found in the consequent of the mined rules, and these files are typically ranked based on the rule's *interestingness value*.

To the best of our knowledge, only a few targeted association rule mining algorithms have been considered in the context of change recommendation: Zimmermann et al. (2005), Ying et al. (2004), and Rolfsnes et al. (2016) (our previous work). In contrast, simple *co-change* algorithms have been applied in a variety of domains (Ball et al. 1997; Beyer and Noack 2005; Gall et al. 1998; Hassan and Holt 2004). The existing targeted association rule mining algorithms and the simple co-change algorithms differ in terms of the subsets of the change set used to form the rule antecedents. Consider, for example, the subsets of the change-set $C = \{a, b, c, d\}$:

$$\text{powerset}(C) = \{\{\}, \quad (1)$$

$$\{a\}, \{b\}, \{c\}, \{d\}, \quad (2)$$

$$\{a, b\}, \{a, c\}, \{a, d\}, \{b, c\}, \{b, d\}, \{c, d\}, \quad (3)$$

$$\{a, b, c\}, \{a, b, d\}, \{a, c, d\}, \{b, c, d\}, \quad (4)$$

$$\{a, b, c, d\} \quad (5)$$

of C 's subsets, both Zimmerman's and Ying's algorithms only consider rules based on line 5 (i.e., rules of the form $\{a, b, c, d\} \rightarrow X$) because these techniques constrain the antecedent to be equal to the change set. At the other end of the spectrum, co-change algorithms consider rules from the singleton sets in line 2, such as $\{a\} \rightarrow X$ or $\{b\} \rightarrow X$. (As a notational convenience, singleton sets are often written without brackets as in " $a \rightarrow X$ ".) In previous work, we introduced TARMAQ, the most versatile among these algorithms (Rolfsnes et al. 2016). TARMAQ uses the set from any one of lines 2, 3, 4, or 5. The particular line used is dynamically chosen based on the maximal overlap with the change set (Rolfsnes et al. 2016).

In this paper we study only CO-CHANGE and TARMAQ. Zimmerman's ROSE algorithm is not included because its behavior is subsumed by TARMAQ (i.e., whenever ROSE is able to make a recommendation, TARMAQ makes the same recommendation, but TARMAQ is able to generate recommendations when ROSE is not (Rolfsnes et al. 2016)). To provide some intuition on the behavior of CO-CHANGE and TARMAQ we provide the following example:

Example 1 (CO-CHANGE and TARMAQ) Given a query Q of changed artifacts, CO-CHANGE and TARMAQ work as follows:

CO-CHANGE: for each artifact $q \in Q$, find the set of artifacts C not in Q that have changed with q in the past, then for each $c \in C$ create rules such that the left hand side is equal to q , and the right hand side is equal to c .

TARMAQ: find the transactions with the largest intersection with Q , then create rules such that the left hand size is equal to the intersection, and the right hand side is equal to the respective difference ("leftover" artifacts of each transaction).

For example, assume that artifacts a , b , and f have been changed by a developer, yielding the query $Q = \{a, b, f\}$, then given the following change history, CO-CHANGE and TARMAQ will output the following rules:

<i>ChangeHistory</i>	CO-CHANGE	TARMAQ
$TXIDArtifacts$	$a \rightarrow Y$	$a, b \rightarrow X$
$TX_1\{a, Y\}$	$a \rightarrow X$	$b, f \rightarrow X$
$TX_2\{a, b, X\}$	$b \rightarrow X$	
$TX_3\{b, f, X\}$	$f \rightarrow X$	

For CO-CHANGE, a has changed with both Y and X , resulting in the two first rules, while b and f has changed with X , resulting in the two last rules. For TARMAQ, the largest intersections (of size two) can be found in TX_2 , and TX_3 , and the difference with each respective intersection and transaction is X , resulting in the two rules.

3 Problem Description

In complex software systems, as well as systems with many active developers, it can be challenging for each individual developer to be aware of all dependencies that exist in the software. To aid a developer, a *change recommendation* can be made based on recent changes. The application of association rule mining to change recommendation involves looking for the *evolutionary coupling* between artifacts (files, methods, etc.) of a system. This search considers artifacts coupled if and only if they have changed together in the past. Furthermore, the *strength* of a coupling is given by an *interestingness value*. For example, how frequently the rule's artifacts change together.

In our previous work (Rolfnes et al. 2016), which sought to find evolutionary couplings through association rule mining of a system's version history, we noticed that there are often rules with different *antecedents*, but the same *consequent*. For example, consider the following rules involving artifacts a , b , and c :

$$\begin{aligned} r_1 &= \{a\} \rightarrow \{c\} \\ r_2 &= \{b\} \rightarrow \{c\} \end{aligned}$$

which can be interpreted as “if you change a , consider changing c ,” and “if you change b consider changing c .” Given the change set $\{a, b\}$, existing recommendations systems will select *one* of the two rules, that recommend c . However, we conjecture that doing so may be a mistake. For example, having multiple applicable rules for the same consequent potentially provides increased evidence that the consequent is relevant. We hypothesize that this increased evidence can be captured by the *aggregation* of rules into *hyper-rules*, whose use will lead to more accurate recommendations. In terms of the example, we seek to combine rules r_1 and r_2 into the *hyper-rule* r_3 that captures the cumulative evidence that c should be recommended for change when a and b are changed.

A concrete example will help illustrate our goal and also provide a better intuition into the value of association rule aggregation. The example involves a sequence of past transactions that each include a set of artifacts that changed together. The example also motivates the need to aggregate the interestingness values (defined in Section 4) of the rules to produce an interestingness value for the resulting hyper-rule. The example does this using, as a simple interestingness value, the percentage of the transactions that give rise to the rule.

Example 2 Consider the following (historic) sequence of transactions:

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

and the change set $C = \{a, b, c, d\}$ where, based on \mathcal{T} and C , the following rules have been mined (the interestingness of each is given in parentheses):

$$\begin{aligned} a &\rightarrow x && (40\%) \\ b &\rightarrow y && (20\%) \\ c &\rightarrow y && (20\%) \\ d &\rightarrow y && (20\%) \end{aligned}$$

In these rules all the artifacts that occur in an antecedent are part of change set C while all artifacts that occur in a consequent are potentially impacted by the change with a certainty reflected by the rule's interestingness value.

Clearly, without aggregation, x is recommended above y , because it has changed two times with an item in the change set (a), while y had changed at most once with *any individual* item of the change set. However, y has changed more times with *at least one* item of the change set. Therefore, there is combined evidence that y should be recommended above x .

Generalizing this example, our goal is to aggregate mined association rules into hyper-rules, which combine evidence and ultimately provide more accurate recommendations. To this end, the remainder of this paper investigates the impact of three aggregation techniques on the performance of two association rule mining algorithms using a collection of 40 interestingness-measures.

4 Interestingness Measures

The relative value of the rules mined via targeted association rule mining is given by an *interestingness measure*. In Agrawal et al.'s seminal paper on association rule mining (Agrawal et al. 1993), two interestingness measures were introduced, *support* and *confidence*.

Definition 1 (Support) Given a sequence of transactions \mathcal{T} , the *support* of the rule $A \rightarrow B$ is defined as the number of transactions where the union of the antecedent and consequent is a subset, divided by the total number of transactions. Therefore, support represents the probability of $A \cup B$ being a subset in a transaction:

$$\text{support}(A \rightarrow B) \stackrel{\text{def}}{=} \frac{|\{T \in \mathcal{T} : \{A \cup B\} \subseteq T\}|}{|\mathcal{T}|}$$

Intuitively, the higher the support, the more likely the rule is to hold, while rules with low support identify weaker relations. For this reason, a minimum threshold on support is often used to filter out uninteresting rules.

Definition 2 (Confidence) Given a sequence of transactions \mathcal{T} , the *confidence* of the rule $A \rightarrow B$ is defined as the number of transactions with the union of A and B as a subset, divided by the number of transactions where A is a subset. Therefore, the confidence represents

Table 1 Overview of probabilistic building blocks used to define the interestingness measures of Table 2

Probability	Definition	Conditional Probabilities	Definition
$P(A)$	$\frac{ \{T \in \mathcal{T} : A \subseteq T\} }{ \mathcal{T} }$	$P(A B)$	$\frac{P(A, B)}{P(B)}$
$P(B)$	$\frac{ \{T \in \mathcal{T} : B \subseteq T\} }{ \mathcal{T} }$	$P(B A)$	$\frac{P(B, A)}{P(A)}$
$P(A, B)$	$\frac{ \{T \in \mathcal{T} : \{A \cup B\} \subseteq T\} }{ \mathcal{T} }$	$P(\neg A B)$	$\frac{P(\neg A, B)}{P(B)}$
$P(\neg A)$	$1 - P(A)$	$P(\neg B A)$	$\frac{P(\neg B, A)}{P(A)}$
$P(\neg B)$	$1 - P(B)$	$P(A \neg B)$	$\frac{P(A, \neg B)}{P(\neg B)}$
$P(\neg A, \neg B)$	$1 - P(A) - P(B) + P(A, B)$	$P(B \neg A)$	$\frac{P(B, \neg A)}{P(\neg A)}$
$P(\neg A, B)$	$P(B) - P(A, B)$	$P(\neg A \neg B)$	$\frac{P(\neg A, \neg B)}{P(\neg B)}$
$P(A, \neg B)$	$P(A) - P(A, B)$	$P(\neg A \neg B)$	$\frac{P(\neg A, \neg B)}{P(\neg B)}$

the conditional probability of B being a subset in a transaction, given that A is a subset of that transaction:

$$confidence(A \rightarrow B) \stackrel{def}{=} \frac{|\{T \in \mathcal{T} : \{A \cup B\} \subseteq T\}|}{|\{T \in \mathcal{T} : A \subseteq T\}|}$$

Since the introduction of targeted association rule mining, a large number of alternative interestingness measures have been proposed. However, all these measures can be defined using the same set of basic probabilistic quantities. Indeed, the interestingness measures of a rule $A \rightarrow B$ build upon the following probabilities:

- $P(A)$: the likelihood of A changing in the history.
- $P(B)$: the likelihood of B changing in the history.
- $P(A, B)$: the likelihood of A and B changing together in the history.

As shown in Table 1, the other probabilities used in the measure definitions can be inferred from these three. For example, *support* is the probability $P(A, B)$, which is the probability that a transaction includes both A and B . Likewise, *confidence* is the probability $P(B|A)$, which is the conditional probability that B is in a transaction given that A is in the same transaction. Several measures also account for the non-occurrence of the antecedent or consequent. For example, the *causal support* is defined as $P(A, B) + P(\neg A, \neg B)$. A complete list of the interestingness measures used in our study and their definitions is given in Table 2.

There is one final detail related to the interestingness measures that is relevant to our discussion: the *range* of a measure, and specifically its ability to measure either negative, positive, or no correlation between a rule’s antecedent and consequent. Early measures such as support and confidence focus on positive correlations. However, it is also possible to consider negative correlations. These would concern rules that capture, for example, “*if a changes, then it is unlikely that you need to change b*”.

The range of most measures falls into one of a few categories. Most existing measures (e.g., *support*) range between 0 and 1. This [0..1] range is also the easiest to interpret as a correlation, where 0 naturally indicates no correlation and any higher value the degree of positive correlation. Another common range is [-1..0..1], where 0 again indicates no correlation, but negative correlation is also possible. In addition, there also exist ranges such as [0..1..∞), where 1 indicates no correlation and the maximum value is unbounded.

Table 2 Overview of the 40 interestingness measures considered in our study (continued on next page)

Interestingness measure	Range	Definition
Added Value (Tan et al. 2004)	$[-0.5..0..1]$	$P(B A) - P(B)$
Causal Confidence (Kodratoff 2001)	$[0..1]$	$\frac{1}{2} * (P(B A) + P(\neg A \neg B))$
Causal Support (Kodratoff 2001)	$[0..1]$	$P(A, B) + P(\neg A, \neg B)$
Collective Strength (Aggarwal and Yu 1998)	$[0..1..∞)$	$\frac{P(A, B) + P(\neg A \neg B)}{P(A) * P(B) + P(\neg A) * P(\neg B)} * \frac{1 - P(A) * P(B) - P(\neg A) * P(\neg B)}{1 - P(A, B) - P(\neg A, \neg B)}$
Confidence (Agrawal et al. 1993)	$[0..1]$	$P(B A)$
Conviction (Brin et al. 1997)	$[0..∞)$	$\frac{P(A) * P(\neg B)}{P(A, \neg B)}$
Cosine (Tan et al. 2004)	$[0..1]$	$\frac{P(A, B)}{\sqrt{P(A) * P(B)}}$
Coverage (Geng and Hamilton 2006)	$[0..1]$	$P(A)$
Descriptive Confirmed Confidence (Kodratoff 2001)	$[-1..0..1]$	$P(B A) - P(\neg B A)$
Difference Of Confidence (Hofmann and Wilhelm 2001)	$[-1..0..1]$	$P(B A) - P(B \neg A)$
Example and Counterexample Rate (Vaillant et al. 2004)	$(-\infty..0..1]$	$\frac{P(A, B) - P(A, \neg B)}{P(A, B)}$
Gini Index (Breiman et al. 1984)	$[0..1]$	$P(A) * (P(B A)^2 + P(\neg B A)^2) + P(\neg A) * (P(B \neg A)^2 + P(\neg B \neg A)^2) - P(B)^2 - P(\neg B)^2$
Imbalance Ratio (Wu et al. 2010)	$[0..1]$	$\frac{P(A B) - P(B A)}{P(A B) + P(B A) - P(A B) * P(B A)}$
Interestingness Weighting Dependency (with parameters $k=2, m=2$) (Gray and Orłowska 1998)	$[0..1]$	$\left(\frac{P(B A)}{P(B)}\right)^{k-1} * (P(A, B))^m$
J Measure (Smyth and Goodman 1992)	$[0..1]$	$P(A, B) * \log\left(\frac{P(B A)}{P(B)}\right) + P(A, \neg B) * \log\left(\frac{P(\neg B A)}{P(\neg B)}\right)$
Jaccard (Van Rijsbergen 1979)	$[0..1]$	$\frac{P(A, B)}{P(A) + P(B) - P(A, B)}$

Table 2 (continued)

Interestingness measure	Range	Definition
Kappa (Cohen 1960)	$[-1..0..1]$	$\frac{P(A,B)+P(\neg A,\neg B)-P(A)*P(B)-P(\neg A)*P(\neg B)}{1-P(A)*P(B)-P(\neg A)*P(\neg B)}$
Klöggen (Klöggen 1992)	$[-1..0..1]$	$\sqrt{P(A,B) * \max(P(B A) - P(B), P(A B) - P(A))}$
Kulezynski (Kulezynski 1928)	$[0..1]$	$\frac{P(A,B)}{2} * \left(\frac{1}{P(A)} + \frac{1}{P(B)} \right)$
Laplace Corrected Confidence (Good 1966)	$[0..1]$	$\frac{P(A,B)+1}{P(B)+2}$
Least Contradiction (Azé and Kodratoff 2002)	$(-\infty..0..1]$	$\frac{P(A,B)-P(A,\neg B)}{P(B)}$
Leverage(Piatetsky-Shapiro 1991)	$[-1..0..1]$	$P(B A) - P(A) * P(B)$
Lift (Brin et al. 1997)	$[0..1..∞)$	$\frac{P(A,B)}{P(A)*P(B)}$
Linear Correlation Coefficient (Pearson 1896)	$[-1..0..1]$	$\frac{P(A,B)-P(A)*P(B)}{\sqrt{P(A)*P(B)*P(\neg A)*P(\neg B)}}$
Loevinger (Loevinger 1947)	$[-1..0..1]$	$1 - \frac{P(A,\neg B)}{P(A)*P(\neg B)}$
Odd Multiplier (Vaillant et al. 2004)	$[0..∞)$	$\frac{P(A,B)*P(\neg B)}{P(B)*P(A,\neg B)}$
Odds Ratio (Mosteller 1968)	$[0..1..∞)$	$\frac{P(A,B)*P(\neg A,\neg B)}{P(\neg A,\neg B)*P(\neg A,B)}$
One Way Support (Yao and Zhong 1999)	$[-1..0..∞)$	$P(B A) * \log_2 \left(\frac{P(A,B)}{P(A)*P(B)} \right)$
Piatetsky-Shapiro (Piatetsky-Shapiro 1991)	$[-0.25..0.0.25]$	$P(A,B) - P(A) * P(B)$
Prevalence (Geng and Hamilton 2006)	$[0..1]$	$P(B)$
Recall (Geng and Hamilton 2006)	$[0..1]$	$P(A B)$
Relative Risk (Geng and Hamilton 2006)	$[0..∞)$	$\frac{P(B A)}{P(B \neg A)}$
Sebag Schoenauer (Sebag and Schoenauer 1988)	$[0..∞)$	$\frac{P(A,B)}{P(A,\neg B)}$
Specificity (Geng and Hamilton 2006)	$[0..1]$	$P(\neg B \neg A)$
Support (Agrawal et al. 1993)	$[0..1]$	$P(A,B)$

Table 2 (continued)

Interestingness measure	Range	Definition
Two Way Support (Yao and Zhong 1999)	$[-1..0..1]$	$P(A, B) * \log_2 \left(\frac{P(A, B)}{P(A)*P(B)} \right)$
Varying Rates Liaison (Bernard and Charron 1996)	$[-1..0..\infty)$	$\frac{P(A, B)}{P(A)*P(B)} - 1$
Yules Q (Yule 1900)	$[-1..0..1]$	$\frac{\text{odds ratio} - 1}{\text{odds ratio} + 1}$
Yules Y (Yule 1912)	$[-1..0..1]$	$\frac{\sqrt{\text{odds ratio} - 1}}{\sqrt{\text{odds ratio} + 1}}$
Zhang (Zhang 2000)	$[-1..0..1]$	$\frac{P(A, B) - P(A)*P(B)}{\max(P(A, B)*P(-B), P(B)*P(A, -B))}$

The notation $[min..mid..max]$ is used to provide the range of each interestingness measure. min/max provides the minimum and maximum value respectively, mid indicates the point of no correlation. If only $[min..max]$ is used, the point of no correlation is given by min

Piatetsky and Shapiro formalize this notion using four properties that they assert all interestingness measures should satisfy (Piatetsky-Shapiro 1991). The four properties use V to denote the value produced by an interestingness measure:

- No correlation:** $V = 0$ when $P(A)$ and $P(B)$ are statistically independent (i.e., when $P(A, B) = P(A) \cdot P(B)$).
- Positive correlation:** When $P(A)$ and $P(B)$ remain unchanged, but $P(A, B)$ monotonically increase, V should also monotonically increase.
- Negative correlation:** When $P(B)$ and $P(A, B)$ remain unchanged, but $P(A)$ monotonically decrease, V should also monotonically decrease.
- Negative correlation:** When $P(A)$ and $P(A, B)$ remain unchanged, but $P(B)$ monotonically decrease, V should also monotonically decrease.

Existing interestingness measures satisfy these properties to a varying degrees, especially with respect to the way positive and negative correlation are captured (Tan et al. 2004). For example, 23 of the 40 measures shown in Table 2, capture only positive correlations. Furthermore, based on the empirical data we collected, only three of the remaining 17 measures actually produced negative correlations in practice.²

5 Association Rule Aggregation

To assess the value of aggregating the evidence provided by a collection of conventional association rules, we introduce the concept of a *hyper-rule*, which provides an effective summary of a set of constituent rules. When forming hyper-rules, we have to answer two questions: (1) What constitutes a hyper-rule? In other words, how do we select the rules that form a hyper-rule? We address this question in Section 5.1. (2) How do we rank hyper-rules within a set of rules (either conventional or hyper)? We address this question in Section 5.2.

5.1 Hyper-Rule Formation

While in general any set of rules can be aggregated to form a hyper-rule, the focus of this paper is on the aggregation of rules that share a common consequent. These rules represent the collective impact of a change on the respective consequent, which then naturally forms the basis for a change recommendation.

Example 3 Consider the set $R = \{\{a, b\} \rightarrow \{c, f\}, \{a, b\} \rightarrow \{c\}, \{d\} \rightarrow \{c\}\}$. For the purpose of change recommendation, we aggregate the last two rules in set R , since they share the same consequent. The antecedent of the resulting hyper-rule is the set of antecedents of all the constituent rules.

Another potentially interesting application of rule aggregation is to aggregate rules with the same antecedent. Doing so facilitates determining the overall impact of a change. As an example, aggregating the first two rules in set R , introduced in Example 3, results in a hyper-rule that summarizes the impact of changing a and b together. The consequent of such a hyper-rule is the set of consequents of all the constituent rules.

²The three measures are: *descriptive confirmed confidence*, *example and counterexample rate*, and *least contradictions*. Other able measures also sometimes produced negative values, although quite rarely.

Beyond these two, other problem domains may require still other ways of selecting rules for aggregation. In general, a hyper-rule, which intuitively summarizes a set of conventional rules, is defined as follows:

Definition 3 (Hyper-Rule) Given a set of rules $R = \{A_1 \rightarrow C_1, \dots, A_n \rightarrow C_n\}$ we define hyper-rule $\mathcal{H}(R)$ as

$$\mathcal{H}(R) = \bigcup_{i=1}^n \{A_i\} \Rightarrow \bigcup_{i=1}^n \{C_i\}$$

Note that the antecedent and the consequent of a hyper-rule are sets of sets of entities rather than being sets of entities as found in conventional rules. To help distinguish hyper-rules and conventional rules, we use a double-arrow \Rightarrow in a hyper-rule rather than the single arrow \rightarrow used with conventional rules.

Example 4 Given R , the set of rules introduced in Example 3, let $R' \subset R$ be the set of rules that share the same consequent (i.e., $R' = \{\{a, b\} \rightarrow \{c\}, \{d\} \rightarrow \{c\}\}$). Then the hyper-rule generated from R' is:

$$\mathcal{H}(R') = \{\{a, b\}, \{d\}\} \Rightarrow \{\{c\}\}$$

Notice that the definition for a hyper-rule concerns only the association rules and not the originating transactions. This opens up the possibility of identifying *cross transactional patterns*. In Example 4, the hyper-rule simply states that when a and b change, c changes, and when d changes, c also changes. We do not require that all of a , b and d change together with c in the same transaction, rather we are only concerned with combining the evidence found in the individual association rules. Our hypothesis is that combining the evidence for c into a single *hyper-rule* will better capture the collective evidence. The challenge here is to quantify the importance of a hyper-rule, this we discuss in the next section.

5.2 Interestingness Measure Aggregation

In the same manner that an association rule has an interestingness measure a hyper-rule has an aggregated interestingness measure, which summarizes the interestingness values of all its constituent rules into a single value. Aggregated interestingness measures allow ranking hyper-rules; within a set of rules that may potentially contain both hyper-rules and conventional association rules. While an interestingness measure implies a total order over a set of conventional rules, an aggregated interestingness measure extends that total order to sets of rules that contain both hyper-rules and conventional association rules.

Although it is possible to define aggregated interestingness measures by extending each interestingness measure to be applicable to hyper-rules, a more scalable approach is to provide measure-agnostic aggregators that simply aggregate a number of interestingness values into a single value. Such aggregators should conform to a set of properties that are described in Definition 4.

Definition 4 (Properties of a measure aggregator) Let M be an interestingness measure defined over conventional rules, and R be a set of conventional rules. Let \oplus denote a measure aggregator that maps $\mathcal{H}(R)$ and M to a single value representing the aggregated interestingness value (i.e., $\oplus(\mathcal{H}(R), M)$). Then the following properties should hold:

1. Let r_1 and r_2 be two conventional rules, then

$$M(r_1) \geq M(r_2) \implies \oplus(\mathcal{H}(\{r_1\}), M) \geq \oplus(\mathcal{H}(\{r_2\}), M)$$

2. For each set of conventional rules R , if $|R| > 1$, then for each $r \in R$ the following holds:

$$\oplus(\mathcal{H}(R), M) \begin{cases} > \oplus(\mathcal{H}(R - \{r\}), M) \text{ if } M(r) > 0 \\ = \oplus(\mathcal{H}(R - \{r\}), M) \text{ if } M(r) = 0 \\ < \oplus(\mathcal{H}(R - \{r\}), M) \text{ if } M(r) < 0 \end{cases}$$

The first property ensures that applying a measure aggregator, \oplus , on single rules retains their original ordering. The second property ensures monotonicity. For example, it requires that the aggregation function is strictly increasing when rules with positive measure values are aggregated. Note that our theoretical framework for ranking hyper-rules is agnostic with regards to the interestingness measure used.

6 Aggregation Functions

In this section we present three aggregation functions that satisfy the properties of Definition 4 for *non-negative values*.³ Thus this initial exploration of rule aggregators focus on aggregation of *positive correlation*, as we assess the inclusion of *negative correlation* to be a topic in it self. We briefly discuss potential strategies in Section 6.5.

The first two aggregators, Cumulative Gain (CG) and Discounted Cumulative Gain (DCG), are adapted from well known performance measures in Information Retrieval. They are typically used to evaluate search results by evaluating a target list against an ideal (oracle) list (Järvelin and Kekäläinen 2002). In addition to these two, we introduce an additional aggregation function, Hyper Cumulative Gain (HCG). All aggregators are empirically evaluated in Section 8.

6.1 Cumulative Gain

The first aggregation function, Cumulative Gain, comes from Information Retrieval (Järvelin and Kekäläinen 2002).

Definition 5 (Cumulative Gain) Given an interestingness measure M and a set of rules $R = \text{rules}$, where $\forall r \in R: M(r) \geq 0$, the *Cumulative Gain of the hyper-rule $\mathcal{H}(R)$* is defined as follows:

$$CG(\mathcal{H}(R), M) = \sum_{i=1}^n M(r_i)$$

Example 5 (Cumulative Gain) Given the following set of rules R , and an interestingness measure M :

$$R = \{r_1, r_2, r_3\}$$

$$M = \{(r_1, 0.7), (r_2, 0.3), (r_3, 0.3)\}$$

the *Cumulative Gain* of $\mathcal{H}(R)$ is computed as follows:

$$CG(\mathcal{H}(R)) = 0.7 + 0.3 + 0.3 = 1.3$$

³Formal proofs for the three aggregator functions are provided in the [Appendix](#).

6.2 Discounted Cumulative Gain

While similar in nature to CG, DCG adds a coefficient that reduces the impact of subsequent values. This enables DCG to give greater weights to those values that have the largest impact (Järvelin and Kekäläinen 2002). Note that for DCG, internal ordering matters, in the following definition we therefore assume that rules are sorted from high to low according to their interestingness value.

Definition 6 (Discounted Cumulative Gain) Given an interestingness measure M and a set of sorted rules $R = \text{rules}$, where $\forall r \in R: M(r) \geq 0$, the DCG of the hyper-rule $\mathcal{H}(R)$ is defined as follows:

$$DCG(\mathcal{H}(R), M) = \sum_{i=1}^n \frac{M(r_i)}{\log_2(i+1)}$$

Notice that $\frac{M(r_i)}{\log_2(i+1)}$ is monotonically decreasing because $\log_2(i+1)$ is monotonically increasing while the rules values are decreasing. In Appendix A we provide a formal proof that DCG satisfies the properties of Definition 4 for non-negative values of M .

Example 6 (Discounted Cumulative Gain) Consider the following set of rules R , with M giving the corresponding interestingness values:

$$\begin{aligned} R &= \{r_1, r_2, r_3\} \\ M &= \{(r_1, 0.7), (r_2, 0.3), (r_3, 0.3)\} \end{aligned}$$

DCG of $\mathcal{H}(R)$ for M is given by:

$$DCG(\mathcal{H}(R), M) = \frac{0.7}{\log_2(2)} + \frac{0.3}{\log_2(3)} + \frac{0.3}{\log_2(4)} \approx 1.04$$

6.3 Hyper Cumulative Gain

The last aggregator function, *Hyper Cumulative Gain* (HCG), incorporates two properties, which makes it different from CG and DCG. First, aggregation through HCG respects the bounds of the source interestingness measure. Second, it incorporates the *number of rules* that were aggregated to produce the hyper-rule. In order to achieve this, HCG outputs a *pair* rather than a single value. We will talk about each element of the pair in order, and refer to them as HCG_1 and HCG_2 .

6.3.1 HCG_1 : Aggregating the Interestingness Measure Values

For any given measure M , the values that HCG_1 generates are guaranteed to be within the range of M . This is as opposed to CG and DCG that do not necessarily preserve the original range of the respective interestingness measure. For example, the support measure has the range $[0..1]$. Given two support values of 0.8 and 0.7, the CG aggregator results in 1.5, which is greater than the upper bound 1. However, HCG_1 is 0.94, which is within the range $[0..1]$. Consequently, compared to CG and DCG, the aggregated values produced by HCG_1 put the hyper-rules on a more level playing field with the conventional rules, which are naturally constrained by their interestingness measure's range.

Apart from satisfying the range constraint, HCG_1 also has a probabilistic interpretation. From this perspective, HCG_1 provides the likelihood that at least one of the rules in a hyper-rule is relevant.

Example 7 Let C be a change-set; $r_1, r_2,$ and r_3 be three rules derived from C ; and M be a measure that indicates the probability that a rule is relevant to its respective change-set. Then, the probability that at least one of $r_1, r_2,$ or r_3 is relevant to C is calculated using the following formula:

$$M(r_1) + (1 - M(r_1)) * M(r_2) + (1 - M(r_1)) * (1 - M(r_2)) * M(r_3) \\ = 1 - (1 - M(r_1)) * (1 - M(r_2)) * (1 - M(r_3))$$

If a hyper-rule were composed of $r_1, r_2,$ and $r_3,$ its HCG_1 for measure M would be calculated using the formula above. Generalizing this formula to an arbitrary number of rules, HCG_1 is defined as follows:

Definition 7 (HCG_1) Given an interestingness measure $M,$ with upper bound $b,$ and a set of rules $R = rules,$ where $\forall r \in R: M(r) \geq 0,$ the HCG_1 of $\mathcal{H}(R)$ for M is defined as:

$$HCG_1(\mathcal{H}(R), M) = M(r_1) + \sum_{i=2}^n \left(M(r_i) \cdot \prod_{j=1}^{i-1} \left(1 - \frac{M(r_j)}{b} \right) \right)$$

which, for finite values of $b,$ is equivalent to

$$HCG_1(\mathcal{H}(R), M) = b - \prod_{i=1}^n \left(1 - \frac{M(r_i)}{b} \right)$$

For measures without a finite upper bound ($b = \infty$), the term $\frac{M(r_j)}{b}$ is defined to be zero. In these cases, HCG_1 behaves the same as CG.

6.3.2 HCG_2 : the Number of Rules

The second part of the HCG pair, $HCG_2,$ captures the number of rules that were used to construct a hyper-rule. From Definition 4 we know that a rule which expresses no correlation through its interestingness measure value should not affect the aggregated value, HCG_2 satisfies this property by filtering out these rules. Following the filtering, HCG_2 is simply the cardinality of this filtered set:

Definition 8 (HCG_2) Given a set of rules $R = rules,$ HCG_2 of R is defined as:

$$HCG_2(R, M) = |\{r \in R | M(r) > 0\}|$$

6.3.3 HCG: Combining HCG_1 and HCG_2

With HCG_1 and HCG_2 defined, HCG can simply be expressed as their ordered pair:

Definition 9 (Hyper Cumulative Gain) Given an interestingness measure $M,$ with upper bound $b,$ and a set of rules $R = rules,$ where $\forall r \in R: M(r) \geq 0,$ the HCG of $\mathcal{H}(R)$ for M is defined as:

$$HCG(\mathcal{H}(R), M) = (HCG_1(\mathcal{H}(R), M), HCG_2(R, M))$$

To enable ranking hyper-rules based on their HCG values, we define the following total order relation over its value-count pairs.

Definition 10 (Total order relation over pairs) For the pairs (V_1, c_1) and (V_2, c_2) the total order relation \geq is defined as:

$$(V_1, c_1) \geq (V_2, c_2) \equiv V_1 > V_2 \vee (V_1 = V_2 \wedge c_1 \geq c_2)$$

Note here that HCG_1 takes precedence over HCG_2 .

Since HCG incorporates the range in its definition, we provide two examples, one for a measure that has a finite range, and one for a measure that has an infinite range.

Example 8 (Hyper Cumulative Gain over finite range) Consider a set of rules R and an interestingness measure M with range $[0, 1]$:

$$\begin{aligned} R &= \{r_1, r_2, r_3\} \\ M &= \{(r_1, 0.7), (r_2, 0.3), (r_3, 0.3)\} \end{aligned}$$

The HCG of $\mathcal{H}(R)$ for M is given by:

$$HCG(\mathcal{H}(R), M) = (1 - (1 - 0.7) * (1 - 0.3) * (1 - 0.3), 3) = (0.853, 3)$$

Example 9 (Hyper Cumulative Gain over infinite range) Consider a set of rules R and an interestingness measure M with range $[0, \infty)$:

$$\begin{aligned} R &= \{r_1, r_2, r_3\} \\ M &= \{(r_1, 0.7), (r_2, 0.3), (r_3, 0.3)\} \end{aligned}$$

Recall that the term $\frac{M(r_i)}{b}$ is defined to be zero when the upper bound, b , is infinity; thus, HCG of $\mathcal{H}(R)$ for M is simply:

$$HCG(\mathcal{H}(R), M) = (0.7 + 0.3 * (1 - 0) + 0.3 * (1 - 0)(1 - 0), 3) = (1.3, 3)$$

Notice that the HCG therefore is exactly equal to CG for interestingness measures with infinite max bound.

6.4 Ensuring Additive Identity Through Centering

In Definition 4 we introduced several properties that should hold when association rules are aggregated. One of these properties is the *additive identity*, which says that aggregation with a measure having value 0 should not increase the aggregated value. However, not all measures have 0 as their point of *no correlation* between the antecedent and consequent of a rule, for these measures we must perform a simple *centering* to preserve semantics.

Most interestingness measures use the value 0 to indicate no correlation, one example is the original *support* measure. When the *support* of a rule, say $A \rightarrow B$, is equal to 0, the artifacts $a \in A$ and $b \in B$ have never all changed together in a single transaction. From the view of the *support* measure this is interpreted as there being no correlation between A and B . However, interestingness measures do not strictly need to use 0 as the point of no correlation, for the measures included in our study, *collective strength*, *lift*, and *odds ratio* are defined in such a way that l is the point of no correlation between the antecedent and consequent of a rule. Interestingness measures such as these must be re-centered.

Definition 11 (Centered Interestingness Measure) An interestingness measure is *centered* if (1) its range contains the value 0; and (2) the value 0 indicates no correlation between the rule and the change set.

To motivate the need for this centering, consider the following example:

Example 10 (Centering) The lift interestingness measure has a range of $[0..1..\infty)$ with 1 indicating no correlation. Consider the following set of association rules, where the *lift* of each rule is also given:

$$\begin{aligned} \{a\} \rightarrow \{X\} & \text{ lift:1.5} \\ \{b\} \rightarrow \{Y\} & \text{ lift:1} \\ \{c\} \rightarrow \{Y\} & \text{ lift:1} \end{aligned}$$

Two of the rules share the same consequent and can therefore be aggregated. We do this twice, with and without centering.

$$\begin{array}{ccc} \text{not centered} & & \text{centered} \\ \{\{b\}, \{c\}\} \rightarrow \{Y\} & CG([1, 1]) = 2 & \{a\} \rightarrow \{X\} \quad CG(0.5) = 0.5 \\ \{a\} \rightarrow \{X\} & CG(1.5) = 1.5 & \{\{b\}, \{c\}\} \rightarrow \{Y\} \quad CG([0, 0]) = 0 \end{array}$$

For the aggregated rule-set where *lift* was not centered before aggregation, the two rules with *Y* as a consequent now rank higher than the *X*-rule, *even though there was no correlation between b, c, and Y*. However, if we center before aggregation, the *non-correlation is preserved after aggregation*.

6.5 Aggregation of Negative Values

As stated earlier, our proposed aggregators are defined under the assumption that measure values are positive. Including negative values, correlations, in the aggregation complicates the situation in two ways:

1. Negative values may not have the same range as positive values for the same interestingness measure.
2. Depending on the aggregator, the order in which negative and positive values are mixed can have significant implications.

One possible remedy for (1) may be some sort of normalization, while for (2) we envision that negative and positive values may be aggregated separately, using the absolute value of the negative value. However, given that our study incorporates few interestingness measures that empirically produce negative values, we have left this exploration for future work.

7 Experiment Design

To assess the viability of association rule aggregation and especially the measurement aggregation functions proposed in Section 6, we perform a large-scale empirical study. While we believe that association rule aggregation will be useful in a variety of problem domains, our study focuses on change recommendation. In other words, we focus on aggregating rules that share the same consequent. This is because, as discussed in Section 5, only hyper-rules of this form establish the basis for a change recommendation.

The evaluation investigates the performance of association rule aggregation when using different aggregation functions, in the context of various software-systems and various

interestingness measures. Furthermore, it investigates if and how the *granularity* of the underlying change history affects the result of association rule aggregation. Two granularity levels are considered: file level and method level. Specifically, the following questions are investigated:

- RQ1** How frequently can change recommendation be improved by association rule aggregation?
- RQ2** What is the effect of aggregating association rules for change recommendation?
- RQ3** What is the effect of aggregating association rules within each studied software-system?
- RQ4** How much does a change in granularity impact the precision gain from rule aggregation?

In total, we generated approximately 21.8 million data points to answer our four research questions. The remainder of this section will describe our study in detail. To start, Fig. 1 provides a high-level overview of the experiment design.

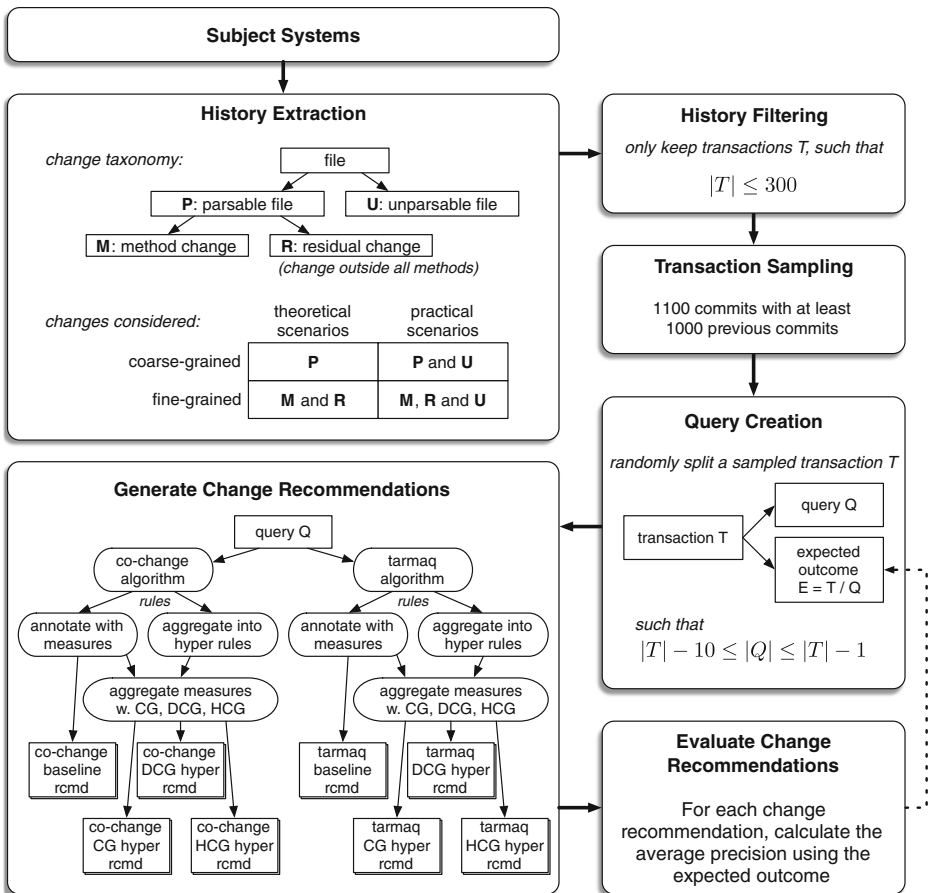


Fig. 1 The high level flow of our experiment design

7.1 Subject Systems

To assess the impact of association rule aggregation under a range of conditions, we study 17 large systems with varying characteristics, such as frequency of changes, number of file and method changes, and average number of changes per commit. Two of the 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 analyze 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 analyze the common software platform that KM uses across various systems in the maritime and energy domain.

The other 15 systems include the well known open-source projects shown in the first column of Table 3. In addition to information regarding the extracted histories (discussed in Section 7.2), the table shows that the systems vary from medium to large size, ranging up to just over 280 000 unique files in the largest system. Finally, the lower subtable shows the programming languages used in each system, as an indication of heterogeneity.

7.2 History Extraction

From each of the 17 subject systems, we extract *four* different histories based on up to the 50 000 most recent transactions (*commits*). This choice is motivated by our previous work in the area of software repository mining (Moonen et al. 2016), which suggests that considering such a number of transactions does not include outdated co-change information. The four differ in terms of *granularity* and *parsability*. Here granularity is either file level or method level, and parsability either includes or excludes files that can be parsed into methods. Parsability is tied back to our history extractor’s use of SrcML (Collard et al. 2013), which supports method-level parsing of C, C++, C#, and Java code. In addition to these languages the change histories of our subject systems contain code written in a multitude of other languages such as Python, Ruby, and JavaScript (as well as build/configuration files in XML, etc.).

From the point of view of a developer in need of a change recommendation, the more fine-grained the response the better. For example, it is easier to act on the recommendation “consider changing method *M*”, than the recommendation “consider changing file *F*”. On the other hand, there exist evolutionary couplings between files where method-level change information is unavailable. In such cases file-level recommendations are of more use than no recommendations at all.

We explore the impact of granularity under two different scenarios: *the practical scenario* and *the theoretical scenario*. Under the practical scenario we acknowledge that there are many files for which we do not have method-level information and include file-level changes in such cases. In contrast, under the theoretical scenario we study the hypothetical situation in which we have method-level data for all changes. Because we, in fact, do not have such information, we approximate this situation by removing from the analysis files for which we don’t have method-level data. Thus, we extract four change histories divided into two distinct classes, practical and theoretical. These histories are formalized as follows:

Practical Scenarios The two practical scenarios capture the reality that parsing all files is infeasible. In this case we compare a pure file-level history, with a mixed history that incorporates as much method-level information as possible. The two histories used for the practical study are:

Table 3 Characteristics of the evaluated software systems (based on our extraction of the last 50 000 transactions for each of he systems)

Software System	History (in yrs)	Unique # files	Unique # artifacts	Avg. # artifacts in commit
CPython	12.05	7725	30090	4.52
Mozilla Gecko	1.08	86650	231850	12.28
Git	11.02	3753	17716	3.13
Apache Hadoop	6.91	24607	272902	47.79
HTTPD	19.78	10019	29216	6.99
Liferay Portal	0.87	144792	767955	29.9
Linux Kernel	0.77	26412	161022	5.5
MediaWiki	9.87	12252	12252	5.43
MySQL	10.68	42589	136925	10.66
PHP	10.82	21295	53510	6.74
Ruby on Rails	11.42	10631	10631	2.56
RavenDB	8.59	29245	47403	8.27
Subversion	14.03	6559	46136	6.36
WebKit	3.33	281898	397850	18.12
Wine	6.6	8234	126177	6.68
Cisco Norway	2.43	64974	251321	13.62
Kongsberg Maritime	15.97	35111	35111	5.08
Software System	Languages used*			
CPython	Python (53%), C (36%), 16 other (11%)			
Mozilla Gecko	C++ (37%), C (17%), JavaScript (21%), 34other (25%)			
Git	C (45%), shell script (35%), Perl (9%), 14 other (11%)			
Apache Hadoop	Java (65%), XML (31%), 10 other (4%)			
HTTPD	XML (56%), C (32%), Forth (8%), 19 other (4%)			
Liferay Portal	Java (71%), XML (23%), 12 other (4%)			
Linux Kernel	C (94%), 16 other (6%)			
MediaWiki	PHP (78%), JavaScript (17%), 11 other (5%)			
MySQL	C++ (57%), C (18%), JavaScript (16%),			
PHP	C (59%), PHP (13%), XML (8%), 24 other (20%) 24 other (9%)			
Ruby on Rails	Ruby (98%), 6 other (2%)			
RavenDB	C# (52%), JavaScript (27%), XML (16%), 12 other (5%)			
Subversion	C (61%), Python (19%), C++ (7%), 15 other (13%)			
WebKit	HTML (29%), JavaScript (30%), C++ (26%), 23 other (15%)			
Wine	C (97%), 16 other (3%)			
Cisco Norway	C++, C, C#, Python, Java, XML, other build/config			
Kongsberg Maritime	C++, C, XML, other build/config			

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

practical coarse-grained: A history where each transaction includes the files changed in the respective commit. The *practical coarse-grained* histories receive the least processing. They are essentially what is returned by `'git log'` after filtering as described in the next section.

practical fine-grained: A history where each *parseable* file of *practical coarse-grained* is replaced by the changed methods of the file together with the file's "residual," which is included only if there are changes to source-code outside all of the file's methods.

Theoretical Scenarios The theoretical scenario is used to explore the question "what would happen if all files were parseable?". As discussed above, this theoretical ideal is achieved by pruning the data. It considers the following histories:

theoretical fine-grained: A history that discards from *practical coarse-grained* all unparseable files.

theoretical coarse-grained: A history that discards from *practical coarse-grained* all unparseable files (thus *theoretical coarse-grained* is *theoretical fine-grained* projected back to the file level).

Table 3 reports statistics related to the extracted change histories, which illustrate the diversity of the systems studied. For example, the transactions cover vastly different time spans, 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 also shows the number of unique files changed in the *practical coarse-grained* data set, as well as the number of unique artifacts changed in the *practical fine-grained* data set.

We now provide an example of how a single commit is interpreted in the four different histories:

Example 11 (History Parsing) Consider the following changes to the files `A.c`, `B.cpp` and `C.yaml`, which were all added to the same commit. First, in the C file `A.c`, a line was changed in the method `m1(int p)`. Secondly, in the C++ file `B.cpp`, a line was changed in the method `m2(int p)` and a public variable was changed in the parent class. Lastly, in the configuration file `C.yaml`, a single line was changed.

<code>A.c</code>	<code>B.cpp</code>	<code>C.yaml</code>
<pre> int m1(int p) { - old_line + new_line } </pre>	<pre> class Class1 { - public var_old + public var_new int m2(int p) { - old_line + new_line } } </pre>	<pre> - old_config: X + new_config: Y </pre>

The changes of these three files, found in the same commit, are interpreted in four different ways to construct four different transactions. We will now list how each of the four types of histories studied in this paper, represent the changes found in the commit. Note that changes which occur outside of all methods are tagged as `file:@residuals`.

practical coarse-grained {`A.c`, `B.cpp`, `C.yaml`}

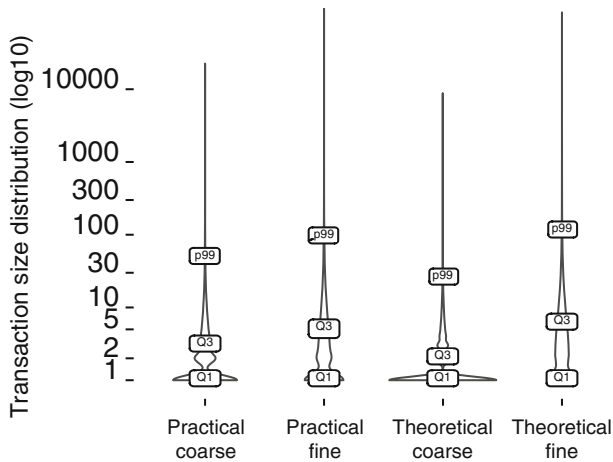


Fig. 2 An overview of the overall distributions across all subject systems for each granularity. Each distribution is shown annotated with the three quartiles, as well as the 99% percentile (for some distributions the percentiles overlap, in those cases the larger percentile is on top)

All changed files found in the commit are included as is in the transaction.

practical fine-grained {A.c:m1, B.cpp:m2, B.cpp:@residuals, C.yaml}

C and C++ is supported for method level parsing, so fine-grained information is included for those files. The yaml file is not supported and is included as is.

theoretical coarse-grained {A.c, B.cpp}

All changed files which are supported for fine-grained parsing are included.

theoretical fine-grained {A.c:m1, B.cpp:m2, B.cpp:@residuals}

Fine-grained information from the parseable files is included, the yaml file hence is discarded.

Note that class and parameter information also is encoded on artifacts to deal with name overloading, but this information is left out of the example above for readability.

7.3 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. 2, which provides violin plots of transaction size for the four data sets, transaction sizes are heavily skewed toward smaller transactions. This pattern is consistent across the individual systems. For example, using the *practical fine-grained* histories Fig. 3 provides separate violin plots for each system.

Unfortunately, as also seen in the violin plots, there exist outlier transactions containing 10 000 or more artifacts. To combat the combinatorial explosion problem raised by such large commits, it is common to filter the history. In an attempt to reflect *most* change recommendation 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 than 300 artifacts. In most cases, the percentage is well above 99% of the available data.

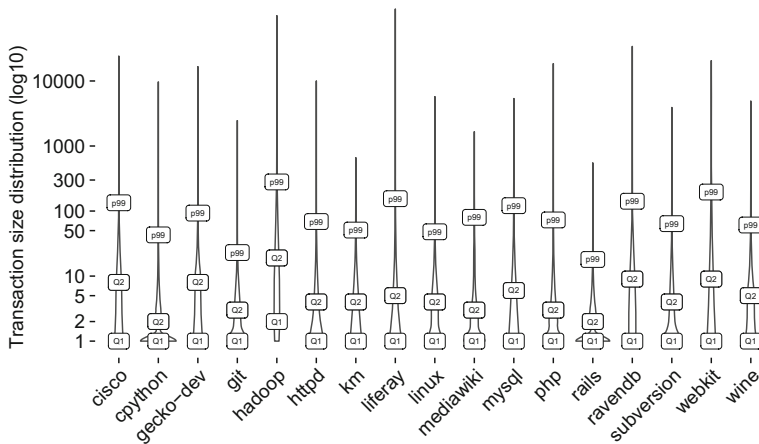


Fig. 3 An overview of the distributions of transactions sizes for each subject system (*practical fine-grained history*)

7.4 Transaction Sampling and Query Creation

Conceptually, a *query* Q represents a set of files that a developer changed since the last synchronization with the version control system. The key idea behind our evaluation is to generate from each sampled transaction T , a query that emulates a developer errantly forgetting to update some subset of T . To this end, we partition each transaction T into a non-empty query Q and a non-empty expected outcome $E \stackrel{\text{def}}{=} T \setminus Q$. In this way, we can evaluate the ability of a recommendation tool to infer E from Q .

From each filtered history we take a representative sample of 1100 transactions,⁴ with the following two constraints:

- The transaction must contain at least three artifacts. This constraint ensures that, at the minimum, a transaction can be split into a query of at least two artifacts and an expected outcome of at least one. Two artifacts are the minimum for there to be the possibility that a recommendation will contain at least two rules that can be aggregated.
- The transaction must have a previous history of at least 1000 transactions. This constraint ensures that there is a minimum number of transactions in the training set for the mining algorithms.

7.5 Generate Change Recommendations

All queries are executed using two different targeted association rule mining algorithms, namely TARMAQ and CO-CHANGE (introduced in Section 2). Executing a query Q , created from a transaction T , creates a set of association rules. The rules often differ based on the

⁴For a normally distributed population of 50000, 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.

algorithm used. Moving from a set of rules to a change-recommendation with respect to Q , requires giving weight to the rules such that they can be sorted. In this paper we experiment with the 40 interestingness measures shown in Table 2.

Central to this paper, and as first envisioned in Section 3, there exists a potential to improve the recommendation by combining the evidence captured in the individual association rules. We explore this potential by aggregating rules that share the same consequent into a hyper-rule, and weighing it using the measure aggregators presented in Section 6. For any *one* query, we therefore create four different recommendations: the original recommendation, and three recommendations produced by aggregating the rules of the original recommendation using the aggregators CD, DCG, and HCG.

7.6 Evaluate Change Recommendations

To evaluate each recommendation we compute its *average precision* (AP). This value captures the precision computed at each relevant document (i.e., each expected outcome) and thus favors recommendations where relevant documents are toward the beginning of the list. Furthermore, we capture the performance over a set of queries (e.g., when using one of the two rule-generation algorithms with a given interestingness measure) using the *mean average precision* (MAP). Formally average precision is defined as follows:

Definition 12 (Average Precision) Given a recommendation R , and an expected outcome E , the average precision of R is given by:

$$AP(R) \stackrel{\text{def}}{=} \sum_{k=1}^{|R|} P(k) * \Delta r(k)$$

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 $\Delta r(k)$ is the *change in recall* calculated only on the $k - 1^{\text{th}}$ and k^{th} files (i.e., how many more correct files were predicted compared to the previous rank).

Note that since we consider only rules with singleton consequents, $\Delta r(k)$ will always be equal to either zero or $1/|E|$ (i.e., a rank either does not contain a file from the expected outcome, or it contains exactly one file from the expected outcome). Table 4 illustrates the computation of AP , $P(k)$, and $\Delta r(k)$ given the ranked list $[c, a, f, g, d]$ and the expected outcome $\{c, d, f\}$.

8 Results and Discussion

This section presents the results of the study described in Section 7, and is structured according to our four research questions: We first discuss RQ1 in Section 8.1 on how often our

Table 4 Calculation of average precision, based on ranked list $[c, a, f, g, d]$ and expected outcome $\{c, d, f\}$

	Rank (k)	Artifact	$P(k)$	$\Delta r(k)$
	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
<i>average precision</i> (AP) = $1/1 * 1/3 + 1/2 * 0 + 2/3 * 1/3 + 2/4 * 0 + 3/5 * 1/3 \approx 0.75$				

technique for rule aggregation can be applied. This is followed by RQ2 in Section 8.2 on how aggregation may improve change recommendation. In Section 8.3 we discuss RQ3, which considers aggregation performance over individual software systems, and finally in Section 8.4 we discuss RQ4, which explores the effect of artifact granularity in the context of aggregation. The first three research questions consider patterns *within* a single history, here the history is used because it captures the largest number of changed artifacts at the finest level of granularity, making it the most useful in real world change recommendation scenarios. However, we did repeat the analysis of RQs 1 to 3 using the other histories and found effectively the same patterns. The final research question considers all four histories. It explores the relative performance of association rule aggregation as a function of granularity and parsability.

8.1 Applicability of Hyper-Rules (RQ1)

As discussed in Section 3, a recommendation can be aggregated if there are at least two rules which share the same consequent, in this section we investigate how often this scenario occurs in practice.

As explained in Section 2, the two algorithms used in our study, CO-CHANGE and TARMAQ, sit at opposing ends with respect to their approach to rule generation. CO-CHANGE on the one hand, splits the input query into its individual artifacts, and generates all possible singleton rules for each artifact. Doing so increases the odds that multiple rules will share the same consequent. On the other hand, the antecedents found in rules mined by TARMAQ are dynamically determined by searching for the largest subset of the query that has some support in the history. More often than not, these subsets are close to the query, resulting in less variation in unique antecedents, and therefore also less possibility for rules which share the same consequent.

To analyze the effect that the choice of association rule mining algorithm has on applicability of hyper-rules, we count the number of recommendations that contain aggregable rules. The expectation here is that CO-CHANGE, by virtue of its creating the maximal amount of rules given a query, also will produce recommendations which are frequently aggregable. The experiment bears out this expectation. For the *practical fine-grained* history the recommendations generated by CO-CHANGE were aggregable 84% of the time, while the recommendations generated by TARMAQ were aggregable only 15% of the time.

8.2 Ability to Improve Precision (RQ2)

The results for RQ1 show that there are ample of opportunities for association rule aggregation. RQ2 considers the impact aggregation has on the quality of the resulting recommendation. To address RQ2, the evaluation considers three dimensions: the rule generation algorithm used (CO-CHANGE or TARMAQ), the interestingness measure used to rank the rules, and the aggregation function used to form the hyper-rules. Hereafter we refer to an (algorithm, measure) combination as a *case*. We report the result of a statistical comparison of the mean average precision and two measures of effect size. To test for statistically significant differences between original and aggregated recommendations we use a one-tailed, paired Wilcoxon signed rank tests. For each case we compare the recommendations without aggregation against each of the aggregated recommendations.

The first of the two effect-size measures is the standardized effect size, which is calculated by dividing the Wilcoxon p -value's corresponding z -statistic by the square root of the number of observations to obtain *Pearson's r* (Rosenthal 1991).

$$(Pearson)r = \frac{z}{\sqrt{N}}$$

Note that r in this case is a *non parametric* effect size not related to central tendencies such as the mean which can be biased by outliers. Rather, r is here a *rank correlation* which intuitively can be interpreted as the probability that an aggregated recommendation has a higher average precision than the corresponding un-aggregated recommendation. The second measure of effect size is the non-standardized effect size, which is defined as the *percentage change in mean average precision (CiMAP)*:

$$(\%)CiMAP = \frac{MAP_{aggregated} - MAP_{original}}{MAP_{original}} * 100$$

The p -values and standardized effect sizes are shown in Fig. 4 while the non-standardized effect sizes are shown in Fig. 6. In both figures the interestingness measures are ordered based on the respective y -measure (r or CiMAP) for CG over CO-CHANGE recommendations, note that this choice is incidental and is done purely to ease later comparisons.

Figure 4 shows both the p -values and the corresponding effect size measured with Pearson's r . The results of the Wilcoxon tests are shown using *hollow circles* and *crosses* where a circle designates a significant result ($p < 0.05$) and a cross a non-significant result. To ease the effect size interpretation, vertical black bars have been added at $r = 0.1, 0.3$ and 0.5 , corresponding to what typically are considered small, medium, and large effects (Cohen 1992). The effect of aggregating CO-CHANGE recommendations is shown with solid lines, while TARMAQ is shown with dotted lines. Furthermore, color is used to differentiate the three aggregators with CG shown in red, DCG in green, and HCG in blue.

In all cases aggregation has a positive effect on the change recommendation, meaning that the average precision of the aggregated recommendations tends to be higher than that of their non-aggregated counterparts. This effect is significant in all but two cases: *example and counterexample rate* and *descriptive confirmed confidence* using CO-CHANGE. In terms of the size of the effect, aggregation of TARMAQ recommendations results in relatively stable low to medium positive effects across most interestingness measures. In the case of CO-CHANGE there is larger spread in effect sizes with a few measures experiencing no to low effect, while the remaining measures are split between experiencing a low to medium or a medium to large effect.

As an overall positive effect has been found, we now turn to a direct comparison between the aggregators. First, observe that CG and DCG closely follow each other across CO-CHANGE and TARMAQ and across all the measures, this indicates that their recommendations tend to be quite similar. Thus in practice either one can be used, although we would recommend CG as its aggregated values are easier to interpret.

The more interesting comparison is between CG/DCG and HCG, where HCG seems to perform significantly better on a handful of interestingness measures. To investigate the differences more closely we applied the Wilcoxon test to just CG and HCG. The results are shown in Fig. 5. Here the interestingness measures have been partitioned based on their range. Recall that HCG incorporates the range into its definition. In Fig. 5, the bottom three partitions contain the interestingness measures with finite ranges, while the top three contain those with infinite ranges. For the interestingness measures with finite ranges, HCG was not

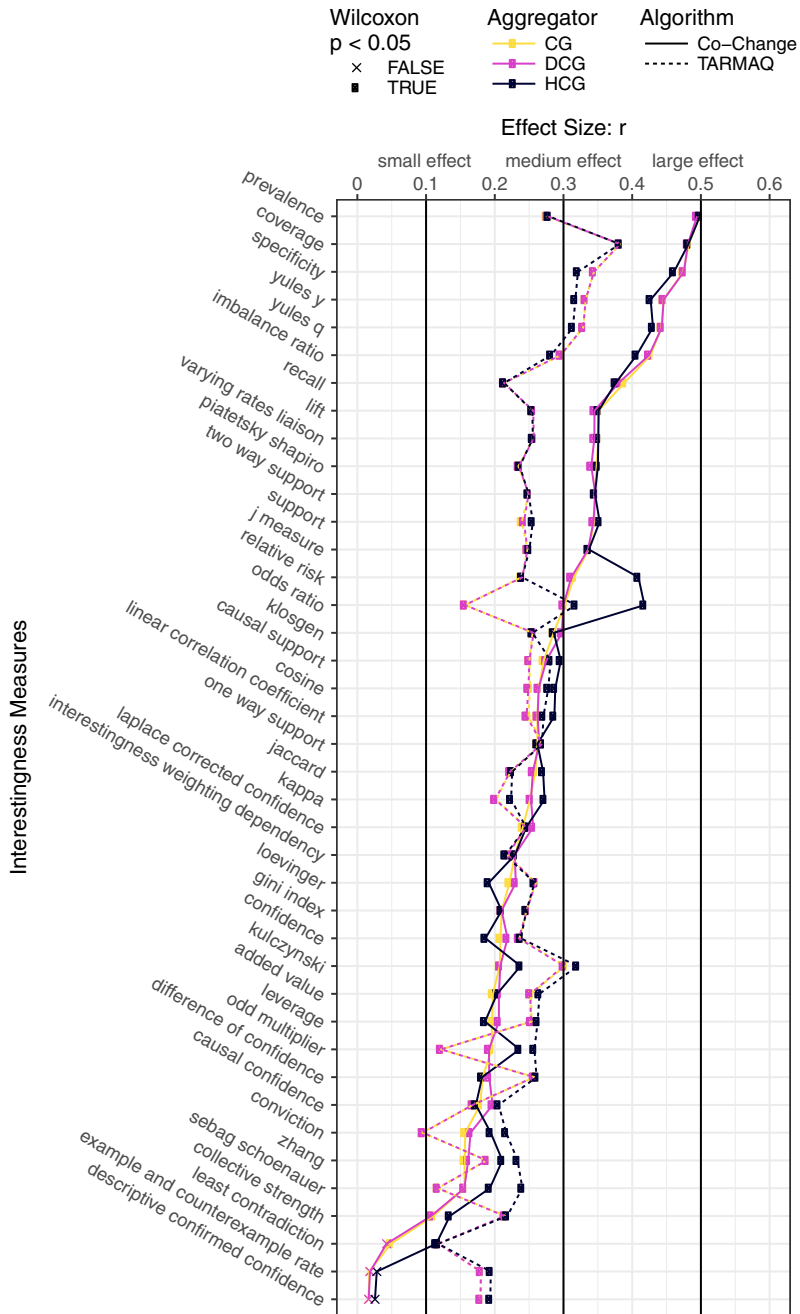


Fig. 4 Effect of aggregating change recommendations using various combinations of interestingness measures, mining algorithms and aggregation functions. Effect size given by Pearson's r

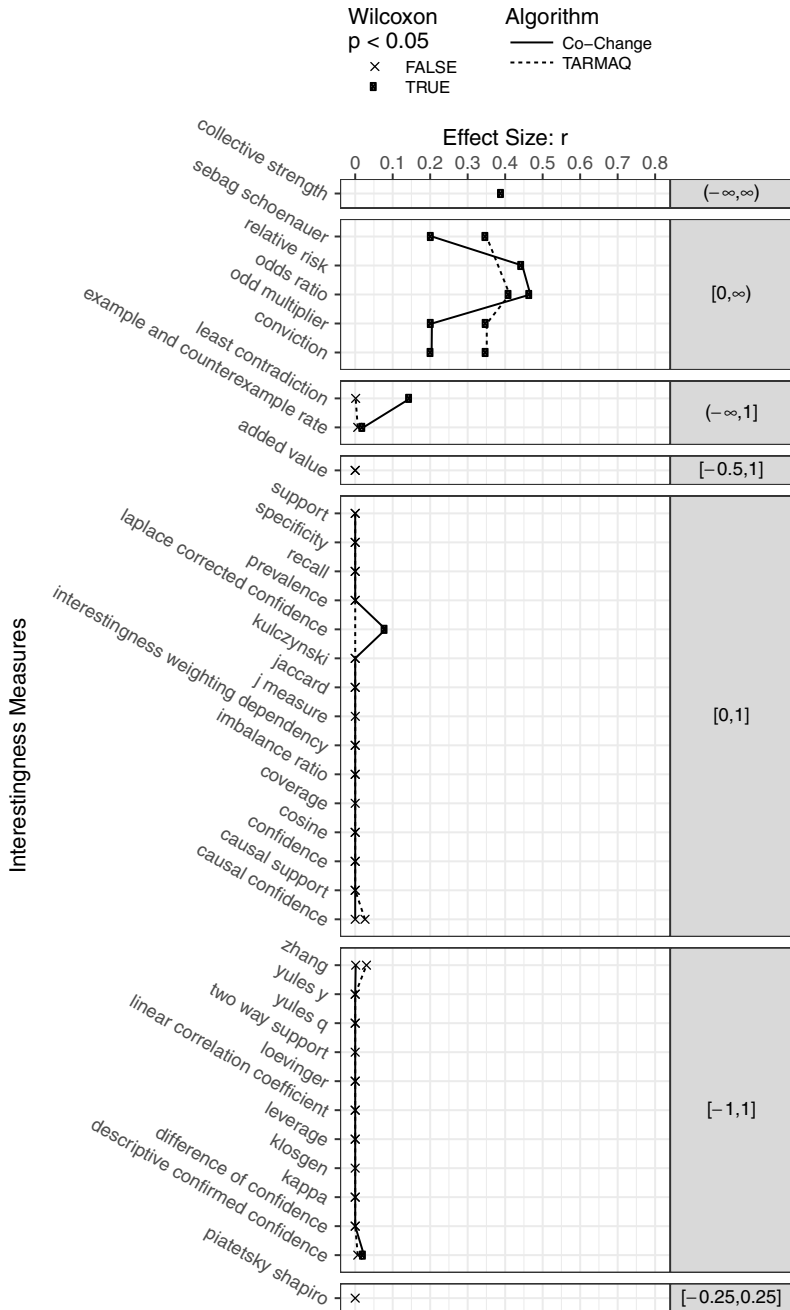


Fig. 5 Positive effect of aggregating rules using the HCG aggregator compared to using the CG aggregator. Interestingness measures are grouped based on their range

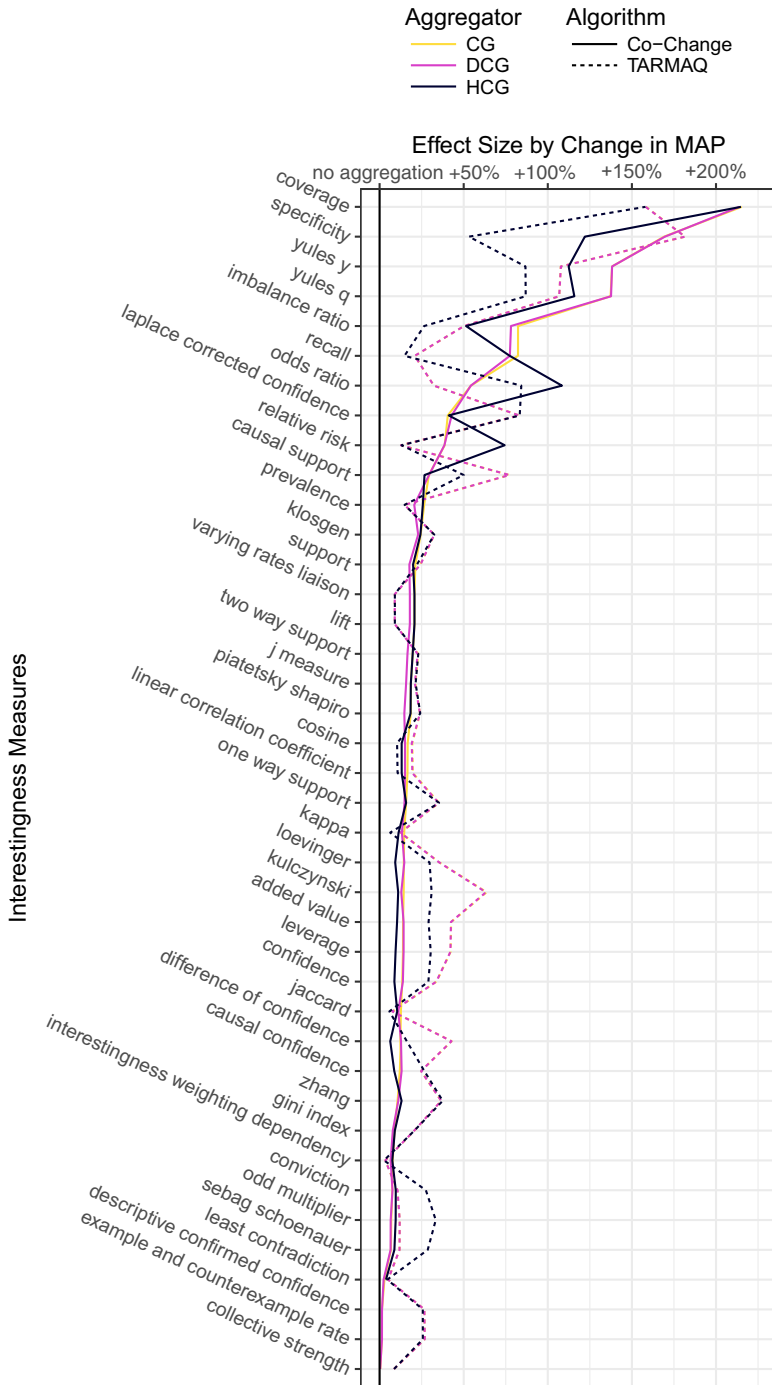


Fig. 6 Effect of aggregating change recommendations using various combinations of interestingness measures, mining algorithms and aggregation functions. Effect size given by *change in mean average precision (CiMAP)*

significantly better than CG with the exception of two cases (*laplace corrected confidence* and *descriptive confirmed confidence*). Furthermore, for these cases the effect was small. On the other hand, HCG is much better for interestingness measures with infinite range. Recall that the way HCG is defined, aggregated values for CG and HCG are the same when the range is infinite; however, HCG also incorporates a tie breaking mechanism based on the number of rules aggregated to create a hyper-rule. Thus the only difference between HCG and CG for the infinite range measures is this tie breaker. It is evident that tie breaking brings a positive effect on aggregation. Given these findings we would recommend the CG aggregator in combination with the tie breaking mechanism of HCG. In simpler terms, a promising rule aggregator is simply one that *sums the respective interestingness measure values* and breaks ties by *the number of values in the sum*.

8.2.1 Improvement Measured by Change in MAP

So far we have discussed the effect of aggregation in terms of a standardized measure of effect size, however, this comes at the cost of distancing the measure from the original measurement unit which is *average precision*. As a complementary view of the data we consider CiMAP, which captures differences in MAP when recommendations are aggregated. The data is shown in Fig. 6, which again has the interestingness measures sorted based on CG performance using the CO-CHANGE recommendation.

Comparing the two, the top five measures when using the standardized measure (Fig. 4) were *prevalence*, *coverage*, *specificity*, *yules y* and *yules q*. There is a large overlap with the non-standardized measure: four of five measures are the same. The only difference is that *prevalence* is ranked lower when using CiMAP. In Fig. 4 we found that *coverage* experienced a medium to large effect of aggregation for TARMAQ and a large effect for CO-CHANGE. We can now see how this effect size translates to one using average precision; the MAP of aggregated TARMAQ recommendations improves by a factor of approximately 2.5 (a 150% increase), while the MAP of aggregated CO-CHANGE recommendations improves by a factor of approximately 3 (a 200% increase).

8.2.2 Implications of Results

Looking beyond RQ2, our results also support several other interesting observations. First, we found that both CO-CHANGE and TARMAQ produced recommendations which responded well to aggregation. This should be emphasized, as the two algorithms significantly differ in their way of rule generation; CO-CHANGE can be thought of as generating the maximum number of rules, while TARMAQ can be thought of as generating the minimum number of highly relevant rules. From this we posit that there is a high likelihood that also other association rule mining algorithms will benefit from rule aggregation.

Turning to the three studied aggregation functions, our results can be used to select an appropriate aggregation function given the algorithm and interestingness measures which

Table 5 The mean number of rules used to form each hyper-rule

	[2,3)	[3,4)	[4,5)	[5,6)	[6,7)	[7,8)	[8,9)	[9,10)	10+
CO-CHANGE	71.69%	11.61%	5.22%	2.65%	1.88%	1.02%	0.98%	0.72%	4.20%
TARMAQ	99.6%	0.32%	4+ = 0.08%						

one is desirable to use. However, from a practical standpoint, more often than not the differences between aggregators are minimal. As an explanation of this consider Table 5. As we can observe, over 99% of all hyper-rules are created from two to three rules in the case of TARMAQ, while the same number for CO-CHANGE is approximately 72%. As the aggregators of Section 6 essentially only differ in their coefficients in a sum, the more rules that are aggregated, the larger the differences are between the aggregated values. So, as typically only 2 or 3 rules are used to form hyper-rules, differences between the values produced by the different aggregators are minimal. This explains why for example CG and DCG often have the same effect on a recommendation. Looking forward, an interesting avenue might be to consider aggregation functions that maximize the benefit when only aggregating a few rules.

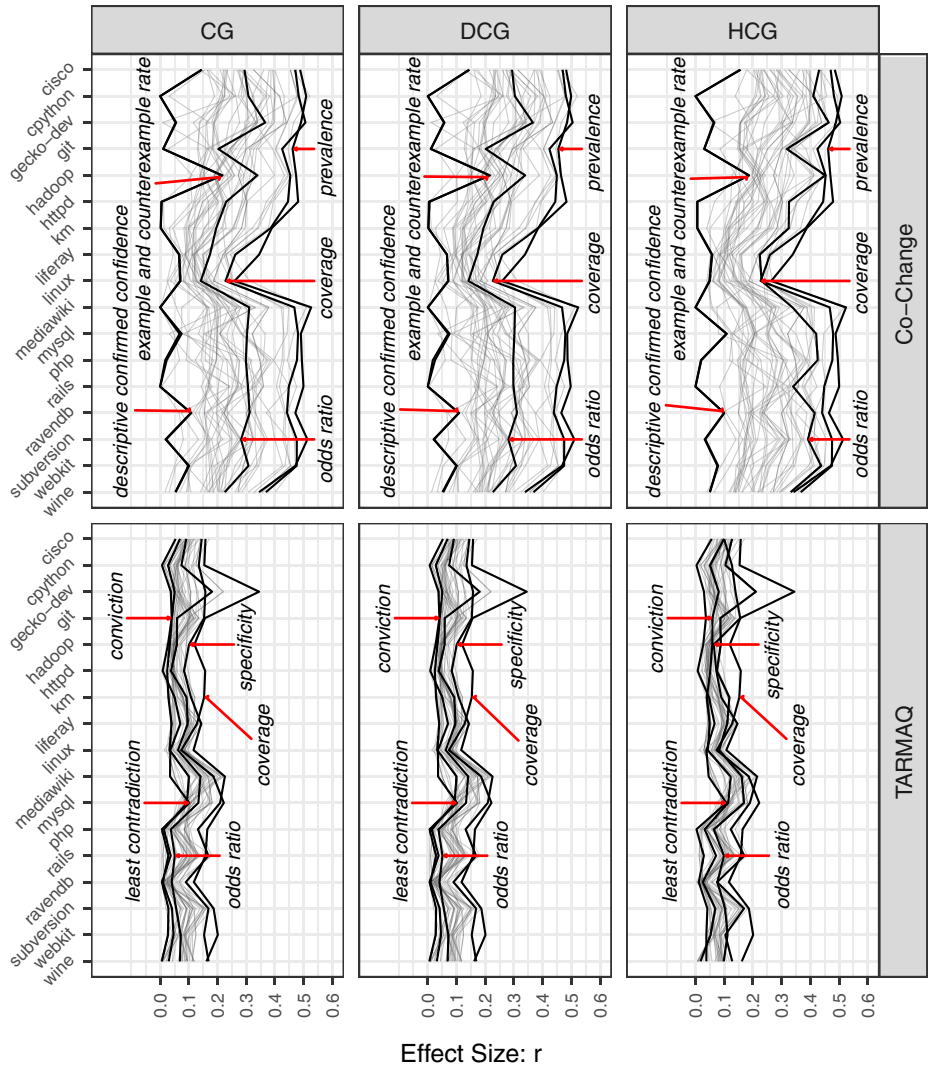


Fig. 7 Effect of aggregating interestingness measures across software systems. A selection of high and low performing interestingness measures from the system-agnostic analysis are highlighted (see Fig. 4)

8.3 System Specific Analysis (RQ3)

In prior sections we have explored overall trends which emerge when association rules are aggregated. However, as the studied software systems (see Table 3) differ on several aspects such as languages used, frequency/size of commits, etc., we consider the possibility that the aggregators might exhibit system-specific differences. To investigate this potential we analyze the effect of aggregating recommendations generated for each individual software system using the same method (Wilcoxon and Pearson's r) as used with the system-agnostic analysis in RQ2. The results can be found in the six plots shown in Fig. 7, where each plot provides the result for one combination of algorithm and aggregator. As reference points we have highlighted a selection of interestingness measures from the system-agnostic analysis. We first consider the observed range of effect sizes for the system-specific analysis. In our system-agnostic analysis we found that aggregation of CO-CHANGE recommendations resulted in a wide range of effect sizes from essentially *no effect* to a *large effect*. In the system-specific analysis we find the same pattern. For TARMAQ we also see the same effect size range. However, there clearly also exist differences between software systems. In particular, *linux* and *liferay* experience less benefit from aggregation with CO-CHANGE recommendations. For interestingness measures that saw very little effect in the system-agnostic study, there also exist specific software systems where these result in medium to large effects, one example is *example and counterexample rate* for *hadoop*. In the case of TARMAQ, the *coverage* interestingness measure experiences a large effect for *gecko-dev* but only a small effect with the other software systems.

We will now turn to the *labeled* interestingness measures of Fig. 7. We first consider the interestingness measures that showed little effect from aggregation in the system-agnostic study. For these interestingness measures we can observe the same pattern for the system-specific results; for CO-CHANGE the *descriptive confirmed confidence* and *example and counterexample rate* consistently has the least effect, while for TARMAQ the *least contradiction* also consistently has the least effect across all systems. Furthermore, TARMAQ with *conviction*, for which we previously saw a considerably larger effect using HCG compared to CG/DCG, can now be connected to the specific software systems where HCG outperforms the other aggregators (e.g., *cisco*, *rails*, *git*). If we turn to the interestingness measures where aggregation had a large effect, those same measures also see the largest effect of aggregation when observed on the software-system level (*prevalence*, *coverage*, *specificity*). In addition to the extremities discussed so far, we have also highlighted *odds ratio* because of its diverse results in the system-agnostic analysis. For CO-CHANGE and CG/DCG, odds ratio saw average effect of aggregation relative to other measures, while it had one of the largest effects for HCG. Interestingly we see the same pattern reflected in the system-specific results of Fig. 7. A similar pattern is reflected in the results for TARMAQ.

In summary the effect of aggregation on interestingness measures to a large degree are consistent across software systems. However, some interestingness measures are more prone to system specific deviations and should therefore be evaluated on a case by case basis.

8.4 Effect of Granularity (RQ4)

So far we have considered histories that consist of a mix of methods and files. To gain a deeper understanding of rule aggregation this section investigates aggregation's behavior across all four histories. As discussed in Section 7.2, we differentiate between *theoretical* and *practical* histories, which can be summarized as follows:

Table 6 Applicability for the theoretical histories

	<i>theoretical coarse-grained</i>	<i>theoretical fine-grained</i>
CO-CHANGE	90%	80%
TARMAQ	14%	13%

Theoretical histories where only parseable artifacts are considered.

Practical histories where all available artifacts are considered.

The next two subsections present results for these two classes of histories.

8.4.1 Theoretical Histories

For the theoretical histories, all artifacts that could not be parsed have been removed. We can therefore focus purely on the effect of granularity without the blurring effect caused when histories share unparseable files. To begin with, Table 6 shows the percentage of recommendations that were aggregable given histories containing only file level changes, and then separately only method changes. While there is little change for TARMAQ, there is nearly a ten percent increase in applicability for CO-CHANGE when moving from the *theoretical fine-grained* history to the *theoretical coarse-grained* history. At first, this may seem paradoxical, as the number of artifacts increases with finer granularity; however, the opportunities for aggregation may indeed decrease with finer granularity. To see this, consider the visualization shown in Fig. 8. Here, the association rules at the method level have different consequents, while if we lift the rules to the file level, the consequent now becomes shared. Thus, the rules at the method level are non-aggregable (by virtue of having different consequents), while the rules on the file level are aggregable. The converse can not happen however; if two file-level rules are non-aggregable, then all the corresponding method-level rules will be non-aggregable.

While there are more opportunities for aggregation at the file level, both levels provide ample of opportunity; thus we turn to RQ2, and consider the impact of aggregation using the recommendation based on the non-aggregated rules as a baseline. This involves three steps:

- Starting with the *theoretical coarse-grained* history, for each combination of algorithm, aggregator, and measure, calculate CiMAP (see Section 8.2)
- Also calculate CiMAP for the *theoretical fine-grained* history.

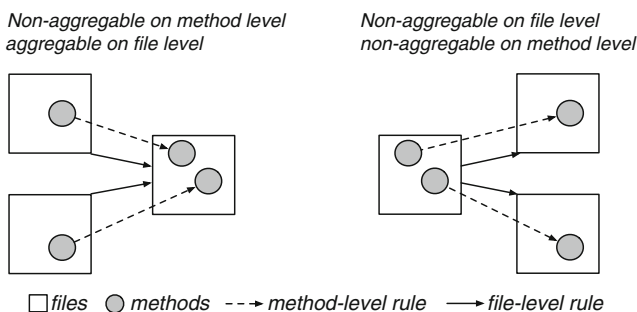


Fig. 8 Effect of changing granularity level on whether association rules are aggregable or not

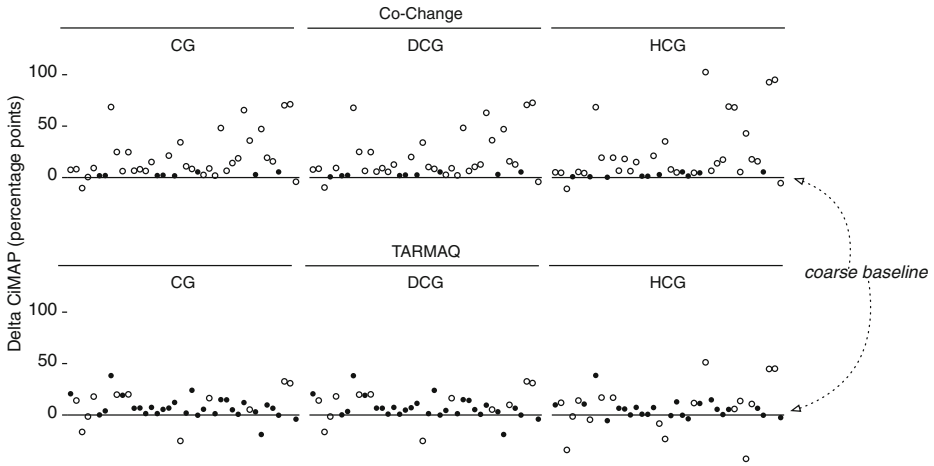


Fig. 9 Change in the effect of aggregation when moving from *theoretical coarse-grained* to *theoretical fine-grained* histories

- Finally, compute the difference between *theoretical coarse-grained* CiMAP and *theoretical fine-grained* CiMAP.

Say that for an interestingness measure, aggregation with the coarse data improved the recommendation by 50% (CiMAP), for that same measure with the fine-grained data, the recommendation improved by 60%. The *delta CiMAP*, given in percentage points, is then $60\% - 50\% = 10$ percentage points.

The results are shown in Fig. 9. In the plot, the *theoretical coarse-grained* history forms the baseline at 0% and the data-set has been split into six sub-plots one for each pairing of algorithm and aggregator. In a sub-plot, each circle gives the result for a single interestingness measure. If a circle lies above the line at $y = 0$, aggregation achieved a higher CiMAP

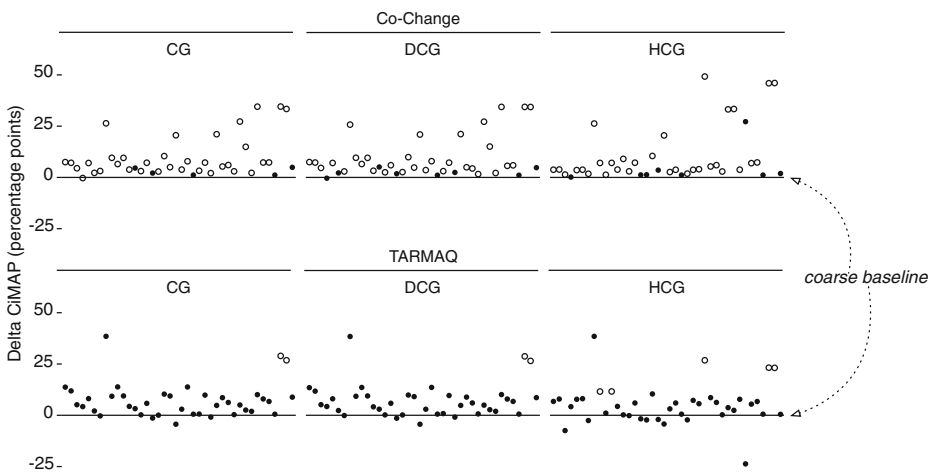


Fig. 10 Change in the effect of aggregation when moving from *practical coarse-grained* to *practical fine-grained* granular histories

with the *theoretical fine-grained* history than with the *theoretical coarse-grained* history. If the circle lies below the line the converse is true. Note that the scatter-plot nature of Fig. 9 (and Fig. 10) is intentional; the focus here is not on individual interestingness measures but rather on overall trends that emerge when granularity is altered.

Lastly, we performed a Wilcoxon test for each algorithm+measure+aggregator to test for differences in CiMAP of the two granularities. The result are captured by either hollow or filled circles in Fig. 9, where a hollow circle indicates that there was a significant difference. Note that the location of the circle on the y-axis reflects the mean, while the Wilcoxon test compares the distribution of the underlying values, as such it is possible for a circle whose mean is near zero to still show a significant difference. Overall, Fig. 9 shows that granularity has a stronger effect on the recommendations generated by CO-CHANGE than those generated by TARMAQ: the mean improvement for the *theoretical fine-grained* over the *theoretical coarse-grained* histories is 18.1 percentage points for CO-CHANGE and 7.5 percentage points for TARMAQ. To conclude, the positive effect of aggregation is almost universally more pronounced when using the finer-grained histories.

8.4.2 Practical Histories

In the previous section, only files that could be parsed were considered. However, in practice, transactions often include files that cannot be parsed for various reasons. Examples of unparseable files include configuration files, binary files, documentation, or simply those of unsupported programming languages. Being able to recommend these types of files is a key advantage of evolutionary coupling when compared to approaches that use static or dynamic source-code analysis. Thus this section repeats the analysis of the previous section and also compares results from the theoretical versus practical histories.

To begin with, Table 7 presents aggregation's applicability for CO-CHANGE and TARMAQ. Overall the values are slightly higher than those of Table 6. In addition, the differences between the fine and coarse grained histories are clearly muted when compared to those seen using the theoretical histories. Thus the data shows the expected damping of the differences caused by the shared unparseable files.

Finally, we consider precision breakdown shown in Fig. 10, which parallels that shown in Fig. 9. Compared to the theoretical histories, there is less difference between the granularity levels (note the difference in the scale used on the y-axis). Thus while granularity again has a stronger effect on the recommendations generated by CO-CHANGE than those generated by TARMAQ. The effect is not as large: the mean improvement for the *practical fine-grained* over the *practical coarse-grained* histories is 9.5 percentage points for CO-CHANGE and 6.5 percentage points for TARMAQ. The damping effect is also evident in that the Wilcoxon tests find fewer significant differences.

To conclude, aggregation provides less improvement on the practical histories compared to the theoretical histories. Still, the trend for both CO-CHANGE and TARMAQ mirror the theoretical histories: change recommendation based on fine-grained histories benefits more from aggregation than that based on the coarse-grained histories.

Table 7 The percentage of recommendations which were aggregable given histories containing only a mix of method and file changes, or only file changes

	<i>practical coarse-grained</i>	<i>practical fine-grained</i>
CO-CHANGE	89%	84%
TARMAQ	15%	15%

Table 8 Execution time for rule aggregation in the context of our experiment

	algorithm	minimum	q1	median	mean	q3	maximum
1	Co-Change	0.00596	0.56860	2.83200	25.9400	13.1300	8185.0
2	TARMAQ	0.00572	0.03004	0.09465	0.4525	0.3395	499.7

8.5 Addendum I: Time Complexity

In our experimental design, rule aggregation is implemented as a post-processing step which is not optimal with respect to execution time. In production the generation of hyper-rules would be incorporated into an existing rule mining algorithm. Given our non-optimal setup we still found aggregation to have low overhead. In Table 8, we have summarized the observed execution times for the rule aggregation step,

the numbers are given in *milliseconds*. The median execution time for CO-CHANGE was 2.83 ms, while the median for TARMAQ was 0.09 ms. Aggregation over CO-CHANGE recommendations naturally see higher execution times compared to TARMAQ as the algorithm tends to generate more rules. However, only 0.001% of aggregations took more than 1 second.

8.6 Addendum II: Absolute Performance

In earlier sections, we have shown that rule aggregation significantly improves the average precision across all interestingness measures. In doing so we have only considered the *relative difference* between aggregated and non-aggregated recommendations. However, it may also be of interest to see the absolute, non-relative, performance of aggregated recommendations. We provide this data in Fig. 11 using the MAP. Here the interestingness measures are ordered based on non-aggregated MAP (e.g., non-aggregated *difference of confidence* achieved the highest MAP for the CO-CHANGE algorithm).

As seen in the figure, the aggregated recommendations always achieve a higher MAP score compared to non-aggregated recommendations.⁵ In particular, the highest achieving interestingness measures are further improved through aggregation; referring back to Fig. 4, *difference of confidence* for CO-CHANGE obtained a significant effect size of $r = 0.2$, while for TARMAQ, *gini index* obtained a significant effect size of $r = 0.25$. Perhaps of more interest, a large number of aggregated interestingness measures achieve higher MAP than the single non-aggregated interestingness measure with highest MAP.

9 Threats to Validity

Problem Domain used in Evaluation: We evaluated hyper-rules in the context of change recommendations. However, the different interestingness measures studied might not fit well into all problem domains (Tan et al. 2004). Still, since we evaluated hyper-rules by looking at the *difference in precision* compared to not using hyper-rules, rather than looking at the actual precision, we believe that the effect of the problem domain is minimized.

⁵Exceptions are the *descriptive confirmed confidence* and *example and counterexample rate*, where aggregation was also found to have a non-significant effect in Fig. 4.

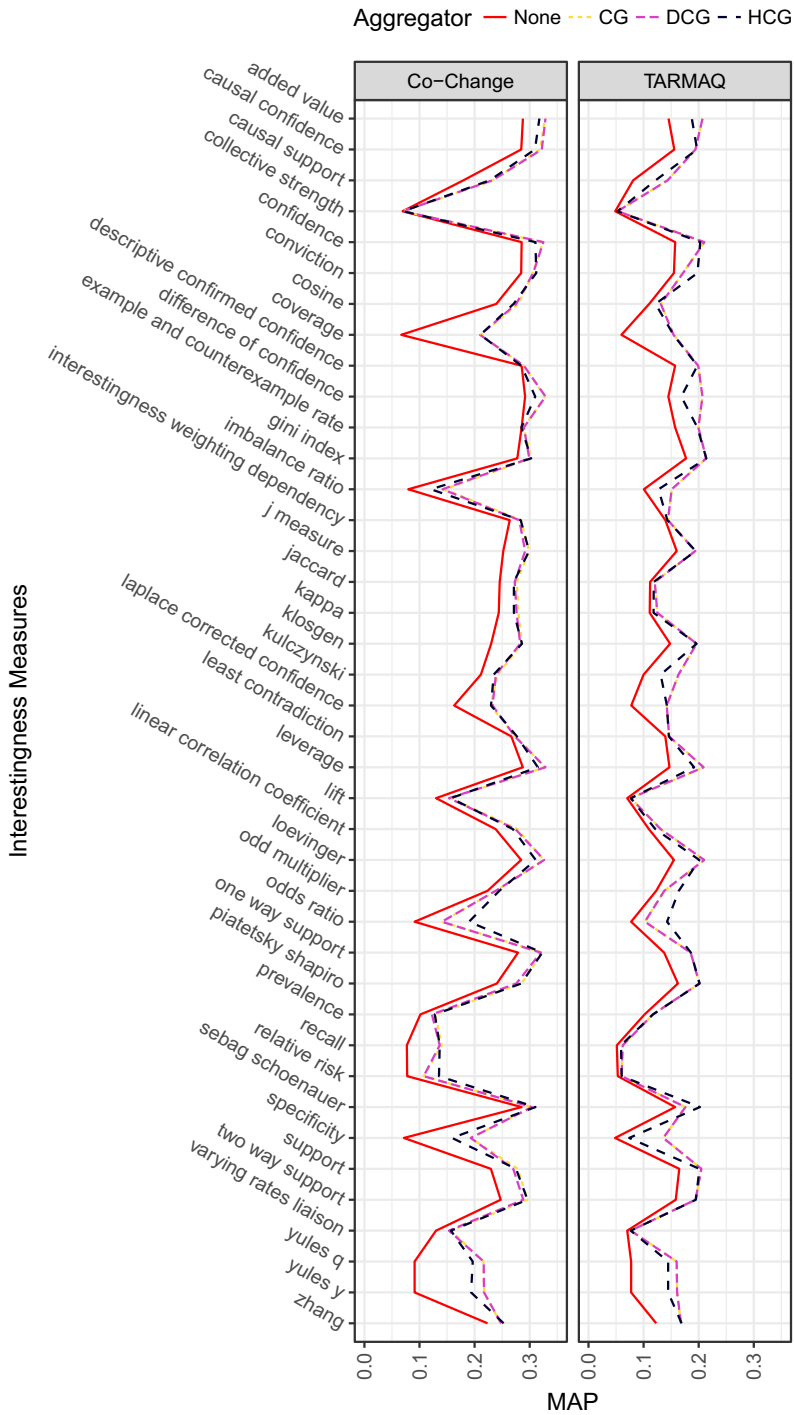


Fig. 11 MAP of both aggregated and non-aggregated recommendations across different interestingness measures and mining algorithms

Aggregation of only Positive Measures: As discussed in Section 4, interestingness measures typically capture either only positive, or both positive and negative correlations between the antecedent and consequent of a rule. In our evaluation however, we only consider positive correlations. While the exact interpretation of negative correlation may differ from measure to measure, the overall interpretation in the context of change recommendations would be: “if this artifact is changed, what artifacts are typically *not* changed?”, while we are more interested in the question: “if this artifact is changed, what other artifacts are also typically changed?”. Moreover, existing targeted association rule mining algorithms show a clear bias toward mining only positive rules (typically based on artifacts that have changed together in the past). Thus the interestingness measures that have the ability to measure both negative and positive correlations will be heavily skewed toward the positive correlations. Looking forward however, we plan to explore possible beneficial ways of handling the aggregation of both positive and negative correlation.

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

Variation in software systems: We evaluated hyper-rules on two industrial systems and 15 large open source systems. These systems vary considerably in size and frequency of transactions (commits), which should provide an accurate picture of the performance of hyper-rules in various contexts. 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 files may be present in the transactions, and related files 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 do believe that this might cause more coherent transactions.

10 Related Work

We distinguish related work on aggregating association rules, clustering and pruning association rules, and on comparing interestingness measures. Furthermore, we specifically discuss the relation between hyper-rules, and the more familiar techniques of closed and maximal rule mining.

Aggregating Association Rules: To the best of our knowledge, no other work has investigated the aggregation of association rules with the goal of combining all available evidence provided by individual association rules for a specific interestingness measure. However, the existing work explores independent and complimentary techniques. Lucia et al. (2014) and Wang et al. (2011) aggregate (“fuse”) *different* association measures in the context of fault localization using program traces. This work is complementary to ours as the challenge here is to find optimal combinations of different measures, compared to our work which seeks to find optimal combinations of association rules for a single measure. Thus a process which incorporates both approaches could first do rule aggregation, followed by aggregating the various measures calculated. Jorge and Azevedo use a *post bagging* technique where multiple bags (sampled subsets) are created from the original set of association rules. Each bag then votes for the most relevant artifact (Jorge and Azevedo 2005). As association rules

which share a consequent are still discarded within each bag, we believe the rule aggregation technique presented in this paper can be used to improve the contribution of each bag in the ensemble.

Also related, Massoud et al. address the challenge of mining multi-dimensional association rules that aim to combine and relate association rules generated from two or more different sets of transactions (Messaoud et al. 2006). They do not aggregate rules that combine evidence for the same conclusion but aim to create aggregate rules that span the dimensions of all transactions.

Clustering and Pruning Association Rules: Several authors investigate methods to discover the most informative or useful rules in a large collection of mined association rules, for example by clustering rules that convey redundant information, or by pruning *non-interesting* rules. Thus, while our method aims to aggregate rules to combine all existing evidence, this work tries to keep (or only generate) the “most important” rules. Toivonen et al. present *association rule covers* as a method to reduce the number of redundant rules (Toivonen et al. 1995). Their method first groups rules which shared the same consequent, and then filters this set by considering the size of the antecedent in combination with the interestingness measures of the rules. No association rules or interestingness measures are aggregated. Kannan and Bhaskaran build upon Toivonen et al.’s work, and instead consider only rules with high interestingness values when generating clusters of rules sharing the same consequent (Kannan and Bhaskaran 2009). In particular, they conclude that extracting clusters from the half of rules with highest interestingness value yields a minimal loss of information. Zaki introduces the *closed* frequent itemset as an alternative association rule mining technique that only generate non-redundant association rules (Zaki 2000). The number of redundant rules produced by the new approach is dramatically smaller than the rule set from the traditional approach but this is achieved at generation time, i.e., no association rules or interestingness measures are aggregated. Baralis et al. investigate an association rule mining technique that combines *schema constraints* (i.e., rule constraints) and rule taxonomies to filter out redundant rules (Baralis et al. 2012). As with Zaki’s approach, this is achieved at generation time, and no association rules or interestingness measures are aggregated. Liu et al. introduce *direction setting rules* as a method of summarizing the set of rules for a human user (Liu et al. 1999). Essentially, direction setting rules are simple rules which capture part of the same relationships also captured in larger rules, i.e., they are more concise.

Selecting and Comparing Interestingness Measures: Tan et al. presents a technique, which given a set of desired properties, can be used to select an appropriate interestingness measure for that context (Tan et al. 2004). In this paper, we have not distinguished between interestingness measures in that regard, as such we went for completeness rather than strictly limiting the set of measures to those appropriate for change recommendation. We attempted to control for the potential domain mismatch by only considering the relative improvement between the original and aggregated recommendations. Closely related to Tan et al., Geng and Hamilton survey a range of interestingness measures and discuss properties such as surprisingness and conciseness (Geng and Hamilton 2006). In a complementary paper, Vaillant et al. clusters interestingness measures based on empirical performance, rather than theoretical properties (Vaillant et al. 2004). McGarry surveys a range of interestingness measures, not only relating to association rules, but patterns for knowledge discovery in general (McGarry 2005). Of particular relevance here is his discussion of understandability of patterns, for example, the support measure is objectively easier to understand for an end-user compared to the collective strength. When hyper-rules are constructed, we also increase complexity, and perhaps lower understandability of patterns in the change recommendation.

As the understanding of patterns is subjective, future work may want to qualitatively study how hyper-rules are interpreted and understood by end-users. Le and Lo evaluate 38 interestingness measures in the context of specification mining, typically; “if File.close has been called, File.open must have been called earlier” (Le and Lo 2015). We believe that such temporal, sequence rules, also can benefit from association rule aggregation. In particular, association rule aggregation should be applicable when multiple sequences overlap at at least one point.

Relation to closed and maximal rules: As the number of association rules often grows unwieldy in conventional association rule mining, techniques such as closed and maximal rule mining has been proposed. Both techniques can be used to effectively reduce the number of generated rules, while still being left with the most relevant. A closed rule is a rule for which no superseding rules have the same support (Zaki and Hsiao 1999), while a maximal rule is a rule for which there are no superseding rules that are frequent (Bayardo 1998; Lin and Kedem 1998). In relating closed and maximal rules to hyper-rules, it is most accurate to think about techniques to identify closed and maximal rules as rule mining algorithms, in the same manner that CO-CHANGE and TARMAQ are rule mining algorithms. In this paper we have considered CO-CHANGE and TARMAQ as they are targeted association rule mining algorithms, which fit the problem domain of change recommendation. In doing so, we explored hyper-rules which aggregate rules with the same consequent, as previously stated, hyper-rules could also be formed based on other selection criteria. Association rule aggregation is agnostic to the origin of the generated rules, it is up to the user to define rule clusters which could benefit from aggregation. We therefore strongly believe that there is potential for association rule aggregation *over closed or maximal rules*. Not only could this further reduce the number of rules, but also improve relevance of the resulting rule set.

11 Concluding Remarks

Association rules capture knowledge found in the relationships between artifacts. In this paper we present a technique for *rule aggregation*. The resulting hyper-rules combine knowledge from sets of conventional association rules in a beneficial way. This paper extends and complements our initial work on this topic, which introduced the notion of hyper-rules (Rolfnes et al. 2016); however, this paper is self contained and makes the following contributions: (1) We identify an opportunity, missed by traditional recommendation systems, to increase accuracy using the evidence of multiple applicable rules in support of a particular conclusion. (2) We provide a theoretical foundation for rule aggregation through the concept of hyper-rules. (3) We present three aggregation strategies for forming hyper-rules, where two are adapted from Information Retrieval and one is introduced in this paper. (4) We provide formal proofs that all studied aggregators satisfy the set of desirable properties given in Definition 4 for non-negative values. (5) We perform a large empirical study where hyper-rules are evaluated in the context of change recommendation. We include systems from our two industry partners, Cisco and Kongsberg Maritime, as well as 15 open source systems. Furthermore, for each system, four different histories are studied. The histories vary in terms of their *granularity* (file level versus method level) and *parsability*. Our findings are as follows: (RQ1) We find that, depending on the underlying history and rule mining algorithm, between approximately 13% and 90% of the generated change recommendations are candidates for association rule aggregation. (RQ2) Of the 40 studied interestingness measures, we find that rule aggregation significantly improves the precision of change recommendation for all but two interestingness measures when used with

CO-CHANGE and for all measures when used with TARMAQ. **(RQ3)** We find that aggregation performance varies across software-systems, but the effect is consistently positive. **(RQ4)** In our study of history granularity, we find that typically, finer grained histories benefit more from rule aggregation than the coarse histories. Furthermore, histories where all artifacts are parseable, also see more benefit from rule aggregation than histories that contain a mix of parseable and unparseable artifacts. We conjecture that this is the case because those histories only contain source-code artifacts, which are more likely to show relevant co-change patterns.

Directions for Future Work: In the future we would like to address the following. **(1)** Generally, different association rules may be generated from the same transactions. When aggregating such rules it might be beneficial to account for these overlaps. **(2)** While we found that the studied aggregators did not benefit from negative correlations, we plan to explore alternative aggregation possibilities in the future. **(3)** We also plan to consider how the tie-breaking mechanism used by HCG can be used to create more effective variants of other aggregation functions. **(4)** The experiments studied the effect of aggregation using *all* the rules generated by CO-CHANGE and TARMAQ. A natural extension would be to explore interestingness constraints. For example, the use of a minimum support value to produce *frequent hyper-rules*, where only frequent rules are aggregated. **(5)** We would like to investigate the effect of rule aggregation on other transaction definitions, e.g., sliding windows. **(6)** We also plan to investigate the behavior of hyper-rules when using other association rule mining algorithms from the change-recommendation domain. **(7)** This will be complemented by the exploration of uses for hyper-rules in other domains. **(8)** Finally, with regards to weighting hyper-rules, rather than calculating aggregated measures, we conjecture that new interestingness measures can be defined directly in terms of the hyper-rules.

Acknowledgements 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.

Appendix

A Proofs

Within this section we formally prove that DCG and HCG satisfies the properties of Definition 4. As expressed earlier, we have limited our study of aggregation functions to positive values. We leave out the proof for CG as it is simply the algebraic sum and therefore naturally satisfies all properties of Definition 4.

A.1 Proof for Discounted Cumulative Gain

Theorem 1 *DCG (Definition 6) satisfies the properties in Definition 4 for non-negative values of an interestingness measure M .*

Proof Let M be an interestingness measure, R be a set of rules with non-negative interestingness values, and $V = [v_1, \dots, v_n]$ be an ordered list of interestingness values of rules in R for measure M , such that $\forall i < j. v_i \geq v_j$. Let r be an arbitrary rule in R , and R' be equal to $R \setminus \{r\}$. Then, there exists a $v_k \in V$ such that $M(r) = v_k$. We define $U = [u_1, \dots, u_{n-1}]$

as the ordered list of interestingness values of rules in R' for measure M . U is equal to V except that v_k is removed from it, and we have $\forall i < j. u_i \geq u_j$. Then:

$$\forall 1 \leq i < k. v_i = u_i \tag{6}$$

$$\forall k < i \leq n. v_i = u_{i-1} \tag{7}$$

The first property in Definition 4, which concerns R s of size one, is trivial. To prove that the second property in Definition 4 holds for DCG, we compute the difference between the DCG of R and the DCG of R' and show that for $v_k > 0$, this difference is positive, and for $v_k = 0$, this difference is equal to zero. Let $DCG(R, M)$ and $DCG(R', M)$ denote the DCGs of R and R' for the interestingness measure M , respectively.

$$\begin{aligned} DCG(R, M) - DCG(R', M) &= \sum_{i=1}^n \frac{v_i}{\log_2(i+1)} - \sum_{i=1}^{n-1} \frac{u_i}{\log_2(i+1)} \\ &= \sum_{i=1}^{k-1} \frac{v_i - v_i}{\log_2(i+1)} \\ &\quad + \sum_{i=k}^n \frac{v_i}{\log_2(i+1)} - \sum_{i=k}^{n-1} \frac{u_i}{\log_2(i+1)} \\ &\stackrel{Eq6,7}{=} \sum_{i=k}^{n-1} \frac{v_i - v_{i+1}}{\log_2(i+1)} + \frac{v_n}{\log_2(n+1)} \end{aligned} \tag{8}$$

Note that both terms in the last line are always non-negative. Now, we consider two cases based on the value of v_k :

$v_k > 0$ – Since $v_k > 0$, the two terms in (8) cannot be zero simultaneously. Because this requires $v_n = 0$, and at the same time $\forall i \geq k, v_i = v_{i+1}$. The latter implies $v_n = v_k > 0$, which contradicts with the former. Therefore, in this case, there is always at least one positive term in (8). Thus:

$$DCG(R, M) - DCG(R', M) > 0 \tag{9}$$

This proves that the second property in Definition 4 holds when $M(r) = v_k$ is positive.

$v_k = 0$ – In this case $DCG(R, M) - DCG(R', M) = 0$. This follows from the original assumption that the rules in V are ordered according to their absolute values. Therefore, in (8) all v_i s are equal to zero. \square

A.2 Proof for Hyper Cumulative Gain

We now prove that HCG satisfies the monotonicity properties of Definition 4 for non-negative values of an interestingness measure M . We start by introducing a new operator, and two lemmas that are used in the proof.

Definition 13 (Correlative sum) Let a_1 and a_2 be real numbers. For any nonzero real number b , we define the operator S_b as

$$a_1 S_b a_2 = a_1 + \frac{b - a_1}{b} \cdot a_2.$$

Lemma 1 (Properties of correlative sum) For any nonzero real number b , the correlative sum S_b is commutative, and associative.

Proof S_b is commutative:

$$\begin{aligned} a_1 S_b a_2 &= a_1 + \frac{b - a_1}{b} \cdot a_2 \\ &= \frac{a_1 b + a_2 b - a_1 a_2}{b} \\ &= a_2 + \frac{b - a_2}{b} \cdot a_1 \\ &= a_2 S_b a_1 \end{aligned}$$

S_b is associative:

$$\begin{aligned} (a_1 S_b a_2) S_b a_3 &= a_1 + \frac{b - a_1}{b} \cdot a_2 + \frac{b - (a_1 + \frac{b - a_1}{b} \cdot a_2)}{b} \cdot a_3 \\ &= \frac{a_1 b + a_2 b - a_1 a_2}{b} + \frac{b - (\frac{a_1 b + a_2 b - a_1 a_2}{b})}{b} \cdot a_3 \\ &= \frac{a_1 b + a_2 b - a_1 a_2}{b} + \frac{b^2 - (a_1 b + a_2 b - a_1 a_2)}{b^2} \cdot a_3 \\ &= \frac{a_1 b^2 + a_2 b^2 - a_1 a_2 b + a_3 b^2 - a_1 a_3 b - a_2 a_3 b + a_1 a_2 a_3}{b^2} \\ &= a_1 + \frac{b - a_1}{b} \cdot \frac{a_2 b + a_3 b - a_2 a_3}{b} \\ &= a_1 + \frac{b - a_1}{b} \cdot (a_2 + \frac{b - a_2}{b} \cdot a_3) \\ &= a_1 S_b (a_2 S_b a_3) \end{aligned}$$

□

An important implication of this lemma is that S_b can be applied to a sequence of numbers independent from the ordering of the elements in the sequence.

Lemma 2 For any nonzero real number b , let $L = \{l_1, l_2, \dots, l_n\}$ be a sequence of real numbers. Let $S_b(L)$ denote $l_1 S_b l_2 \cdots l_{n-1} S_b l_n$. Then

$$S_b(L) = l_1 + \sum_{i=2}^n \left(l_i \cdot \prod_{j=1}^{i-1} \left(1 - \frac{l_j}{b} \right) \right) \tag{10}$$

and for any given real number l , we have:

$$S_b(L \cup \{l\}) = S_b(L) + l \cdot \prod_{l_j \in L} \left(1 - \frac{l_j}{b} \right). \tag{11}$$

Proof The proof for both parts is straightforward after expanding the polynomials. □

An implication of (11) is that, for any sequence of real numbers $L = \{l_1, l_2, \dots, l_n\}$, and any arbitrary real number l in L , we have:

$$S_b(L) - S_b(L \setminus \{l\}) = l \cdot \prod_{l_j \in L \setminus \{l\}} \left(1 - \frac{l_j}{b} \right) \tag{12}$$

Theorem 2 HCG (Definition 9) satisfies the properties of Definition 4 for non-negative values of an interestingness measure M .

Proof Let M be a normalized interestingness measure with upper bound $b \in \mathbb{R} \setminus \{0\}$, $R = \text{rules}$ be a set of rules, and $L_M = \langle M(r_1), \dots, M(r_n) \rangle$ be the sequence of non-negative interestingness values for the rules in R . HCG of $\mathcal{H}(R)$ for M defined in Definition 9 can be rewritten as follows:

$$HCG(\mathcal{H}(R), M) = (S_b(L_M), m) \quad (13)$$

where m is given by

$$m = |\{r \in R \mid M(r) > 0\}|$$

The first property in Definition 4, which concerns R s of size one, is again trivial. Let r be an arbitrary rule in R , and l denote $M(r)$. To prove that the second property in Definition 4 holds for HCG, we compare the HCGs of R and $R \setminus \{r\}$, and consider three cases based on the value of $M(r)$:

$M(r) > 0$ – In this case, we need to show that the HCG of R is greater than the HCG of $R \setminus \{r\}$.

$$\begin{aligned} HCG(\mathcal{H}(R), M) > HCG(\mathcal{H}(R \setminus \{r\}), M) &\stackrel{Eq13}{=} (S_b(L_M), m) > (S_b(L_M \setminus \{l\}), m - 1) \\ &\stackrel{Def10}{=} S_b(L_M) \geq S_b(L_M \setminus \{l\}) \end{aligned} \quad (14)$$

To show that inequality (14) holds we show that $S_b(L_M) - S_b(L_M \setminus \{l\})$ is non-negative, which, according to (11), is equivalent to showing that

$$l \cdot \prod_{l_j \in L_M \setminus \{l\}} \left(1 - \frac{l_j}{b}\right) \geq 0$$

This inequality holds because $l = M(r) > 0$, and for all $l_j \in L_M \setminus \{l\}$ the term $1 - \frac{l_j}{b}$ is non-negative (because l_j is at most b). Since $m > m - 1$, inequality (14) holds, completing the case for $M(r) > 0$.

$M(r) = 0$ – In this case, we have to show that the HCGs of R and $R \setminus \{r\}$ are equal.

$$HCG(\mathcal{H}(R), M) = HCG(\mathcal{H}(R \setminus \{r\}), M) \stackrel{Eq13}{=} (S_b(L_M), m) = (S_b(L_M \setminus \{l\}), m)$$

To do so, we have to show that $S_b(L_M) = S_b(L_M \setminus \{l\})$. This is proven by forming $S_b(L) - S_b(L_M \setminus \{l\})$, and replacing l with zero in the right-hand side of (11). Therefore, completing the proof.

So far, we have proven that HCG satisfies the properties in Definition 4 for interestingness measures that have a finite upper bound. If the upper bound of an interestingness measures is infinity, then like CG, HCG becomes the sum of interestingness values of all rules in R . Therefore, it satisfies all the properties in Definition 4. \square

References

- Aggarwal CC, Yu PS (1998) A new framework for itemset generation. In: ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS), 2. ACM, pp 18–24. <https://doi.org/10.1145/275487.275490>
- 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. <https://doi.org/10.1145/170035.170072>
- Az e J, Kodratoff Y (2002) Evaluation de la r esistance au bruit de quelques mesures d'extraction de r egles d'association. In: Extraction et gestion des connaissances (EGC), vol 1. Hermes Science Publications, pp 143–154

- Ball T, Kim J, Siy HP (1997) If your version control system could talk. In: Workshop on Process Modelling and Empirical Studies of Software Engineering, ICSE. 10.1.1.48.910
- Baralis E, Cagliero L, Cerquitelli T, Garza P (2012) Generalized association rule mining with constraints. *Inf Sci* 194:68–84. <https://doi.org/10.1016/j.ins.2011.05.016>
- Bayardo RJ (1998) Efficiently mining long patterns from databases. *ACM SIGMOD Record* 27(2):85–93. <https://doi.org/10.1145/276305.276313>
- Bernard JM, Charron C (1996) Bayesian implicative analysis, a method for the study of oriented dependencies. *Mathématiques. Informatique et Sci Humaines* 135:5–18
- Beyer D, Noack A (2005) Clustering software artifacts based on frequent common changes. In: International Workshop on Program Comprehension (IWPC). IEEE, pp 259–268. <https://doi.org/10.1109/WPC.2005.12>
- Bird C, Menzies T, Zimmermann T (2015) Past, present, and future of analyzing software data. In: The Art and Science of Analyzing Software Data, pp 1–13. <https://doi.org/10.1016/B978-0-12-411519-4.00001-X>
- Bohner S, Arnold R (1996) Software change impact analysis. IEEE, CA, USA
- Breiman L, Friedman J, Stone CJ, Olshen RA (1984) Classification and Regression Trees, vol. 19
- Brin S, Motwani R, Ullman JD, Tsur S (1997) Dynamic itemset counting and implication rules for market basket data. In: ACM SIGMOD International Conference on Management of Data (SIGMOD), vol 26. ACM, pp 255–264. <https://doi.org/10.1145/253260.253325>
- Canfora G, Cerulo L (2005) Impact analysis by mining software and change request repositories. In: International Software Metrics Symposium (METRICS). IEEE, pp 29–37x. <https://doi.org/10.1109/METRICS.2005.28>
- Cohen J (1960) A coefficient of agreement for nominal scales. *Educ Psychol Meas* 20(1):37–46. <https://doi.org/10.1177/001316446002000104>
- Cohen J (1992) A power primer. *Psychol Bull* 112(1):155–159. <https://doi.org/10.1037/0033-2909.112.1.155>
- Collard ML, Decker MJ, Maletic JI (2013) srcML: an infrastructure for the exploration, analysis, and manipulation of source code: a tool demonstration. In: IEEE International conference on software maintenance (ICSM). IEEE, pp 516–519. <https://doi.org/10.1109/ICSM.2013.85>
- Eick S, Graves TL, Karr A, Marron J, Mockus A (2001) Does code decay? Assessing the evidence from change management data. *IEEE Trans Softw Eng* 27(1):1–12. 10.1109/32.895984
- 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. <https://doi.org/10.1109/ICSM.1998.738508>
- Geng L, Hamilton HJ (2006) Interestingness measures for data mining. *ACM Computing Surveys* 38(3). <https://doi.org/10.1145/1132960.1132963>
- Good IJ (1966) The estimation of probabilities: an essay on modern Bayesian methods. MIT Press
- Gray B, Orłowska ME (1998) CCAIA: Clustering categorical attributes into interesting association rules. In: Lecture Notes in Computer Science (LNCS), vol 1394, pp 132–143. https://doi.org/10.1007/3-540-64383-4_12
- Hassan AE, Holt R (2004) Predicting change propagation in software systems. In: IEEE International conference on software maintenance (ICSM). IEEE, pp 284–293. <https://doi.org/10.1109/ICSM.2004.1357812>
- Hofmann H, Wilhelm A (2001) Visual comparison of association rules. *Comput Stat* 16(3):399–415. <https://doi.org/10.1007/s001800100075>
- Järvelin K, Kekäläinen J (2002) Cumulated gain-based evaluation of IR techniques. *ACM Trans Inf Syst* 20(4):422–446. <https://doi.org/10.1145/582415.582418>
- Jashki MA, 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. <https://doi.org/10.1145/1512475.1512493>
- Jorge AM, Azevedo PJ (2005) An experiment with association rules and classification: post-bagging and conviction. In: Hoffmann A, Motoda H, Scheffer T (eds) Proceedings of the 8th International Conference on Discovery Science DS 2005, Lecture Notes in Computer Science, vol 3735. Springer, Berlin, pp 137–149. https://doi.org/10.1007/11563983_13
- Kamber M, Shinghal R (1996) Evaluating the interestingness of characteristic rules. In: SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pp 263–266
- Kannan S, Bhaskaran R (2009) Association rule pruning based on interestingness measures with clustering. *J Comput Sci* 6(1):35–43
- Klößgen W (1992) Problems for knowledge discovery in databases and their treatment in the statistics interpreter explora. *Int J Intell Syst* 7(7):649–673. <https://doi.org/10.1002/int.4550070707>
- Kodratoff Y (2001) Comparing machine learning and knowledge discovery in databases: an application to knowledge discovery in texts. In: Machine Learning and Its Applications, LNAI 2049, chap. 1. Springer, pp 1–21. https://doi.org/10.1007/3-540-44673-7_1

- Kulczyński S (1928) Die Pflanzenassoziationen der Pieninen Imprimerie de l'université
- Le TDB, Lo D (2015) Beyond support and confidence: exploring interestingness measures for rule-based specification mining. In: International Conference on Software Analysis, Evolution, and Reengineering (SANER). IEEE, pp 331–340. <https://doi.org/10.1109/SANER.2015.7081843>
- Lin DI, Kedem ZM (1998) Pincer-search: a new algorithm for discovering the maximum frequent set. pp 103–119. <https://doi.org/10.1007/BFb0100980>
- Liu B, Hsu W, Ma Y (1999) Pruning and summarizing the discovered associations. In: SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD). ACM, pp 125–134. <https://doi.org/10.1145/312129.312216>
- Loevinger J (1947) A systematic approach to the construction and evaluation of tests of ability, vol 61. <https://doi.org/10.1037/h0093565>
- Lucia, Lo D, Xia X (2014) Fusion fault localizers. In: Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering - ASE '14. ACM Press, New York, pp 127–138. <https://doi.org/10.1145/2642937.2642983>
- McGarry K (2005) A survey of interestingness measures for knowledge discovery. *Knowl Eng Rev* 20(01):39. <https://doi.org/10.1017/S0269888905000408>
- Messaoud RB, Rabaséda SL, Boussaid O, Missaoui R (2006) Enhanced mining of association rules from data cubes. In: International Workshop on Data Warehousing and OLAP (DOLAP). ACM, p 11. <https://doi.org/10.1145/1183512.1183517>
- Moonen L, Di Alesio S, Rolfnes T, Binkley DW (2016) 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. <https://doi.org/10.1109/SCAM.2016.9>
- Mosteller F (1968) Association and estimation in contingency tables. *J Am Stat Assoc* 63(321):1–28. <https://doi.org/10.1080/01621459.1968.11009219>
- Pearson K (1896) Mathematical contributions to the theory of evolution. III. Regression, Heredity, and Panmixia. *Philosophical Transactions of the Royal Society A: Mathematical, Phys Eng Sci* 187:253–318. <https://doi.org/10.1098/rsta.1896.0007>
- Piatetsky-Shapiro G (1991) Discovery, analysis, and presentation of strong rules. *Knowledge discovery in databases* pp 229—238
- 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. <https://doi.org/10.1109/32.58784>
- Ren X, Shah F, Tip F, Ryder BG, Chesley O (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. <https://doi.org/10.1145/1035292.1029012>
- 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. <https://doi.org/10.1109/WCRE.2008.47>
- Rolfnes T, Di Alesio S, Behjati R, Moonen L, Binkley DW (2016) 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. <https://doi.org/10.1109/SANER.2016.101>
- Rolfnes T, Moonen L, Di Alesio S, Behjati R, Binkley DW (2016) Improving change recommendation using aggregated association rules. In: International Conference on Mining Software Repositories (MSR). ACM, pp 73–84. <https://doi.org/10.1145/2901739.2901756>
- Rosenthal R (1991) *Meta-analytic procedures for social research*. SAGE
- Sebag M, Schoenauer M (1988) Generation of rules with certainty and confidence factors from incomplete and incoherent learning bases. In: Proceedings of the european knowledge acquisition workshop (EKAW), p 28
- Wang S, Lo D, Jiang L, Lucia, Lau HC (2011) Search-based fault localization. In: 2011 26Th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011). IEEE, pp 556–559. <https://doi.org/10.1109/ASE.2011.6100124>
- Smyth P, Goodman R (1992) An information theoretic approach to rule induction from databases. *IEEE Trans Knowl Data Eng* 4(4):301–316. <https://doi.org/10.1109/69.149926>
- 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
- Tan PN, Kumar V, Srivastava J (2004) Selecting the right objective measure for association analysis. *Inf Syst* 29(4):293–313. [https://doi.org/10.1016/S0306-4379\(03\)00072-3](https://doi.org/10.1016/S0306-4379(03)00072-3)
- Toivonen H, Klemettinen M, Ronkainen P, Hätönen K, Mannila H (1995) Pruning and grouping discovered association rules. In: Workshop on Statistics, Machine Learning, and Knowledge Discovery in Databases, pp 47–52

- Vaillant B, Lenca P, Lallich S (2004) A Clustering of Interestingness Measures. In: Lecture Notes in Artificial Intelligence (LNAI), vol 3245, pp 290–297. https://doi.org/10.1007/978-3-540-30214-8_23
- Van Rijsbergen CJ (1979) Information retrieval. Butterworth-Heinemann
- Wu T, Chen Y, Han J (2010) Re-examination of interestingness measures in pattern mining: a unified framework. *Data Min Knowl Disc* 21(3):371–397. <https://doi.org/10.1007/s10618-009-0161-2>
- Yao YY, Zhong N (1999) An analysis of quantitative measures associated with rules. In: Methodologies for Knowledge Discovery and Data Mining (LNCS 1574). Springer, pp 479–488. https://doi.org/10.1007/3-540-48912-6_64
- 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. <https://doi.org/10.1109/ICSM.2011.6080786>
- Ying ATT, Murphy G, Ng RT, Chu-Carroll M (2004) Predicting source code changes by mining change history. *IEEE Trans Softw Eng* 30(9):574–586. <https://doi.org/10.1109/TSE.2004.52>
- Yong SH, Horwitz S (2002) Reducing the overhead of dynamic analysis. *Electron Notes Theor Comput Sci* 70(4):158–178. [https://doi.org/10.1016/S1571-0661\(04\)80583-8](https://doi.org/10.1016/S1571-0661(04)80583-8)
- Yule GU (1900) On the association of attributes in statistics. *Philos Trans R Soc Lond* 194:257–319
- Yule GU (1912) On the methods of measuring association between two attributes. *J R Stat Soc LXXXV*:579–652. <https://doi.org/10.2307/2340126>
- Zaki MJ (2000) Generating non-redundant association rules SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD). ACM, pp 34–43. <https://doi.org/10.1145/347090.347101>
- Zaki MJ, Hsiao CJ (1999) CHARM: an efficient algorithm for closed association rule mining. In: 2nd SIAM International Conference on Data Mining, pp 457–473. <https://doi.org/10.1137/1.9781611972726.27>
- 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. <https://doi.org/10.1145/2597073.2597096>
- Zhang T (2000) Association rules. In: Knowledge Discovery and Data Mining. Current Issues and New Applications, c, pp 245–256. https://doi.org/10.1007/3-540-45571-X_31
- Zimmermann T, Zeller A, Weissgerber P, Diehl S (2005) Mining version histories to guide software changes. *IEEE Trans Softw Eng* 31(6):429–445. <https://doi.org/10.1109/TSE.2005.72>



Thomas Rolfesnes Dr. Rolfesnes recently received his PhD (Informatics, Sept. 2017) from the University of Oslo. During his PhD he has been employed by the Simula Research Laboratory where his supervisor, Leon Moonen, is a chief research scientist. His research has 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.



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.



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. While carrying out his research, Dr. Di Alesio explored the areas of model-driven, search-based, and reverse software engineering. He has published papers on these topics on 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.



Razieh Behjati Dr. Behjati is a senior test automation engineer. She received her PhD from University of Oslo, and Simula Research Laboratory in 2012. Since 2009 she has been involved in various industrial and research projects related to testing, verification and validation of embedded software systems and software product families. Her research is focused on studying the application of logic and constraint programming as well as machine learning techniques in automated software testing.



Dave Binkley Dr. 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. He 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.