An Eclectic Approach for Change Impact Analysis

Michele Ceccarelli, Luigi Cerulo
Dept. of Biological and Environmental Studies – RCOST, Univ. of Sannio, Italy
ceccarelli@unisannio.it, lcerulo@unisannio.it

Gerardo Canfora, Massimiliano Di Penta
Dept. of Engineering – RCOST, Univ. of Sannio, Italy
dipenta@unisannio.it, canfora@unisannio.it

ABSTRACT

Change impact analysis aims at identifying software artifacts being affected by a change. In the past, this problem has been addressed by approaches relying on static, dynamic, and textual analysis. Recently, techniques based on historical analysis and association rules have been explored.

This paper proposes a novel change impact analysis method based on the idea that the mutual relationships between software objects can be inferred with a statistical learning approach. We use the bivariate Granger causality test, a multivariate time series forecasting approach used to verify whether past values of a time series are useful for predicting future values of another time series.

Results of a preliminary study performed on the Samba daemon show that change impact relationships inferred with the Granger causality test are complementary to those inferred with association rules. This opens the road towards the development of an eclectic impact analysis approach conceived by combining different techniques.

Categories and Subject Descriptors

D.2.7 [**Software Engineering**]: Distribution, Maintenance, and Enhancement—*Version control*

General Terms

Measurements

Keywords

Mining Software Repositories, Change Impact Analysis

1. INTRODUCTION

Change impact analysis is the process of determining the set of software artifacts likely to be changed when (or after) one artifact has been changed. Impact analysis is crucial to make informed decisions among different alternatives to implement a change, to plan regression testing activities, to anticipate future maintenance tasks, and to re-factor/re-design portions of a system to make it more resilient to changes.

Existing approaches for impact analysis are based on static analysis (control, data flow and dependency) [2], dynamic analysis [7], and on textual analysis of change requests and the code being maintained [3]. Recently, the availability of data from versioning systems and bug tracking systems opened the road towards a new software analysis perspective, namely *historical analysis*, which adds complementary information to what available with static and dynamic analysis, i.e., information about changes occurring to software artifacts over time, about artifacts changing together, about who performed a change, and why the change was made.

Ying *et al.* [9] proposed a technique to determine the impact of changes based on association rules. Such technique learns change impact relations based on the co-changes of software artifacts. The underlying idea is that a change occurring to a software artifact a is likely to impact another artifact b if in the past they changed together. This kind of historical analysis is often able to capture change coupling that might not be captured by means of traditional (static and dynamic) analysis: two artifacts might be related—and thus a change to one influences the other—while not having an explicit control or data dependency. Examples are change coupling between high level artifacts (e.g., requirements) and source code, but also changes between source code files belonging to different features sharing something (e.g., GUI style, licenses, database access) that might not necessarily be captured through data and control-flow dependencies. While such technique has proven successful in capturing change relations among artifacts co-changing within a single change set [10], it fails to capture change relations having a consequentiality spread over a larger interval of time.

This paper proposes the use of a Vector Auto–Regression (VAR) model, a generalization of univariate Auto–Regression (AR) model, to capture the evolution and the interdependencies between multiple time series. In particular, we use the bivariate Granger causality test [4] to identify if the changes of a software artifact are useful to forecasting the changes of another software artifact. For a couple of software artifacts, a and b, an equation explains the change evolution of a based on its past changes and the past changes of b; then, a statistical test shows whether the past changes of a are useful for predicting changes to b.

The bivariate Granger causality test has been successfully used in bioinformatics for identifying gene regulatory relationships [8], while to the best of our knowledge it has

ICSE '10, May 2-8 2010, Cape Town, South Africa

not been applied yet to study software evolution phenomena where, up to now, univariate time series models have been used to predict software metrics and changes [6].

Preliminary results obtained over 10531 snapshots of Samba suggest that Granger causality helps to capture change impacts in a complementary way with respect to association rules: while the latter captures relations between source code files changing together, the former identifies temporal causality relations among changing artifacts.

The paper is organized as follows. Section 2 provides an overview of existing historical approaches, based on association rules, to identify the impact of changes [9]. Then, it introduces the proposed change impact analysis approach, based on the bivariate Granger causality test. Section 3 reports and discusses results from a preliminary empirical study performed on Samba. Finally, Section 4 concludes the paper and overviews challenges and potential applications of the technique.

2. CHANGE IMPACT PREDICTION WITH HISTORICAL ANALYSIS

The evolution of a software system, $S = \{f_1, f_2, \ldots, f_K\}$, under the control of a versioning system can be viewed as a sequence of source code *Snapshots* generated by a sequence of *Change Sets*, $\Delta_1, \Delta_2, \ldots, \Delta_T$. A change set, Δ_t, represents the changes performed by a developer on a set of source files, $\Delta_t \subseteq S$, at time t. Change sets can be extracted from a versioning (e.g., CVS or SVN) history log by using for example a time-windowing approach, that considers a change set as a sequence of file revisions that share the same author, branch, and commit notes, and such that the difference between the timestamps of two subsequent commits is less than or equal to 200 seconds [10]. Historical analysis of a software system consists of extracting useful information from change set data.

In this section we recall a consolidated historical analysis approach for predicting change impacts, i.e., the use of association rules, and then we show how the bivariate Granger causality test can be used as a novel approach to predict change impacts. We assume the following definition of change impact relation from source file f_i to f_j, with $i \neq j$:

> *f_j is impacted by a change in f_i ($f_i \rightarrowtail f_j$) $\Rightarrow$ a change performed in f_i will generate a change in f_j in at least K, with $K \geq 0$, ahead change sets.*

2.1 Association Rules

Association rule discovery is an unsupervised learning technique used for local pattern detection showing attribute value conditions that occur together in a given dataset [1]. The application of association rule mining to change impact prediction assumes a dataset composed by a sequence of change sets, e.g., source files, that have been committed together within an interval of time into a versioning repository [10, 9]. An association rule, $F_{left} \Rightarrow F_{right}$, between two disjoint source file sets means that if a change occurs in each $f_i \in F_{left}$, then another change should happens in each $f_j \in F_{right}$ within the same change set. The strength of an association rule is determined by its confidence and support [1], defined as:

$$Support = \frac{|F_{left} \cup F_{right}|}{T}; \quad Confidence = \frac{|F_{left} \cup F_{right}|}{|F_{left}|}$$

where T is the total number of change sets extracted from the versioning repository and $|F|$ is the number of change sets, Δ_t, where $F \subseteq \Delta_t$, i.e., the source files in F changed together.

To find the pair of source files, f_i and f_j, such that $f_i \rightarrowtail f_j$ (i.e. f_j is impacted by a change in f_i), we compute the confidence and support for each $f_i, f_j \in S$, with $i \neq j$, and consider the top N of the list of source file pairs ranked by their confidence and support in decreasing order.

2.2 Granger Causality

Granger causality test is a technique for determining whether one time series is useful in forecasting another [4]. It has become popular in recent years in bioinformatics to discover gene and metabolic pathways of an organism [8]. The basic idea is that a cause cannot come after the effect, then if a variable x affects a variable z, the former should help improving the predictions of the latter variable. We use this notion to capture whether changes performed in a source file could "*cause*", in a Granger sense, a change in another source file. Let $f_k(t)$, $t = 1, \ldots, T$ be the change time series of the source file f_k defined as:

$$f_k(t) = \begin{cases} 1, & f_k \in \Delta_t \\ 0, & f_k \notin \Delta_t \end{cases}$$

i.e., $f_k(t)$ is one if the file f_k changes in snapshot Δ_t, zero otherwise. The simplest bivariate Granger test between two time series $f_1(t)$ and $f_2(t)$ uses the autoregressive specification [5], which consists of the following bivariate and univariate autoregression, solved by estimating the ordinary least squares:

$$f_2(t) = c_1 + \alpha_1 f_1(t-1) + \alpha_2 f_1(t-2) + \cdots + \alpha_p f_1(t-p) +$$

$$+\beta_1 f_2(t-1) + \beta_2 f_2(t-2) + \cdots + \beta_p f_2(t-p) + u(t)$$

$$f_2(t) = c_1 + \gamma_1 f_1(t-1) + \gamma_2 f_1(t-2) + \cdots + \gamma_p f_1(t-p) + e(t)$$

where p is the lag length, which can be estimated with various criteria [5], $u(t)$ and $e(t)$ are independent and identically distributed random variables. To test whether $f_1(t)$ Granger-cause $f_2(t)$ the null hypothesis to reject (H_0: f_1 does not Granger-cause f_2) is defined as:

$$H_0 : \beta_1 = \beta_2 = \cdots = \beta_p = 0$$

which can be implemented by calculating the sum of squared residuals (RSS):

$$RSS_1 = \sum_{t=1}^{T} \widehat{u}(t)^2; \quad RSS_0 = \sum_{t=1}^{T} \widehat{e}(t)^2$$

The null hypothesis is rejected if

$$S = \frac{(RSS_0 - RSS_1)/p}{RSS_1/(T-2p-1)}$$

is greater than the 5% critical value for an $F(p, T-2p-1)$ distribution.

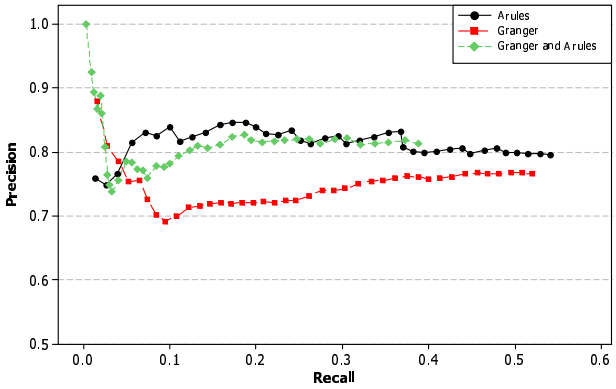

Figure 1: Precision and Recall

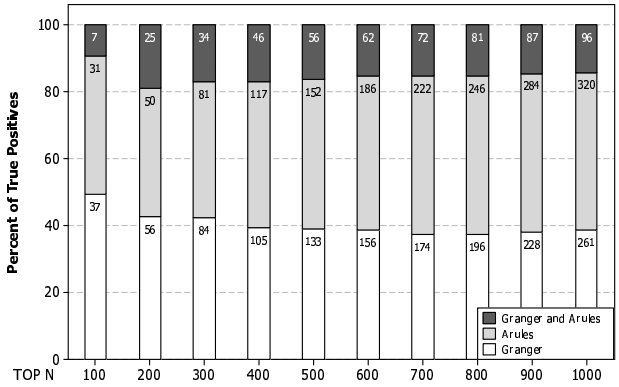

Figure 2: True positive impacts for association rules, Granger, and the combination of both

To find the pair of source files, f_i and f_j, such that $f_i \rightarrowtail f_j$ (f_j is impacted by a change in f_i), for each $f_i, f_j \in S$, with $i \neq j$, we consider the top N in the list of pair of source files ranked by S in decreasing order.

3. PRELIMINARY EMPIRICAL RESULTS

This section reports an empirical study we performed with the aim of getting preliminary insights about the proposed change impact analysis approach. The *goal* of this study is to analyze to what extent Granger causality can be used as a complement to association rules to predict source code files change impact by using data from versioning systems. The *quality focus* is the increase of precision / recall in predicting change impacts, as well as the capability of different techniques to detect complementary change impact sets. The *perspective* is of a maintainer interested to understand what files could be potentially affected by changes occurring in a given file. The *context* consists of change sets extracted from the versioning history of Samba[1]—a well-known file and print service developed to ensure interoperability between Microsoft Windows and Unixes. Specifically, we analyzed 10531 snapshots, from release 1.9.16 to release 3.0.0, and restricted our analysis to files belonging to the Samba daemon (i.e., *smbd* only). We used half of the snapshots (5265) to build the models, and the remaining ones to predict change impacts, using the actual changes occurring in these snapshots as an oracle for evaluating the performances of our prediction.

Figure 1 shows, for different values of top-n (i.e., links with highest strength, increasing from the left to the right), and for different techniques (association rules, Granger causality, and the combination of both) the average precision and recall obtained when identifying the change impact of a class. Precision is defined as the ratio between the number predicted true changes and the total number of predicted changes, while recall is defined as the ratio between the number of predicted true changes and the total number of true changes.

As it can be seen, for lower values of n Granger starts with a high precision (90%), that however quickly decreases with increasing top-n to about 70%, without positively increasing the recall. The precision for association rules starts lower ($\sim$ 75%) and even increases with increasing top-n, up to 82%, then remains almost constant. Finally, the precision for the combination of the two techniques starts very high ($\sim$ 100%, with, however, few links retrieved), then it drops to $\sim$ 75%, and finally rises again to $\sim$ 82%. Overall, the combination of both techniques seems to exhibit the best precision/recall compromise, i.e., to start with a very high precision, but then to keep it as high as association rules.

The complementarity of the two techniques is even more evident when looking at Figure 2, which shows, for different values of top-n, the number of true impacts found by Granger causality (and not found by association rules), by association rules (and not by Granger causality), and those found by both techniques at the same time. This figure shows that the intersection between the two techniques is always lower (from 7 vs 37 and 31 for top-100, up to 25 vs. 50 and 56 for top-200) than the number of links found by one technique only.

Finally, Figure 3 shows an excerpt of an impact graph highlighting impact relations identified by association rules (plain lines), by Granger causality (dotted lines) and by both (bold lines). For space reasons, the graph is limited to files related to Samba authentication (`auth_*.c`), `error.c` (responsible of handling errors), `reply.c` (providing requests to any connection to the Samba server) and `uid.c`, handling user ids. As it can be noticed, `auth_*` files are strongly coupled, and exhibit dependencies highlighted by both techniques. Changes performed to `auth_*` files immediately impact on `reply.c` and on `uid.c`, as shown by relations identified by association rules. Instead, the graph shows that changes are propagated only after a while to `error.c`: in fact the change impact relation is identified by Granger causality only. For instance, on Aug 8, 2001 both `auth_*` files and `reply.c` (revision 1.316) were changed, and the commit note (*"smbd/auth_server: Doco we want to use cli_nt_error here soon smbd/password.c ..."*) mentioned that the error management needs to be updated soon. This happened on Aug 27, 2001 (commit note: *"... added automatic mapping between dos and nt error codes ..."*), where `error.c` underwent a change (to revision 1.6).

The performed study suffers of some threats to validity. In particular, threats to *internal validity*, as this kind of impact analysis (Granger test and its combination with as-

[1]http://www.samba.org/

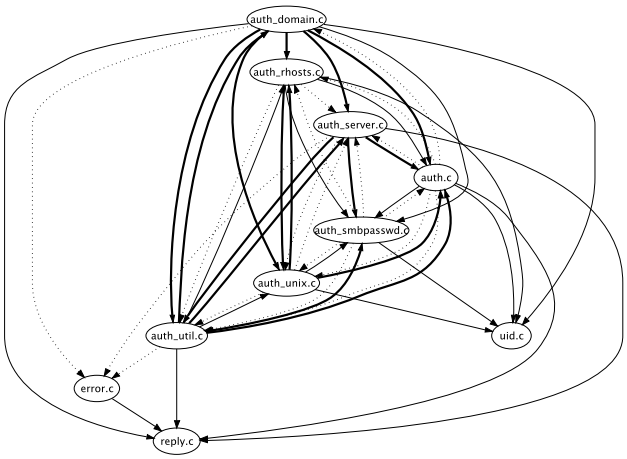

Figure 3: Excerpt of impact graph

sociation rules) is only able to identify the presence of a significant correlation between a change occurring to a file and future changes occurring to other files. Our approach is purely statistical, hence there is no guarantee to ensure source code change causality. Also, the study can suffer of threats to *external validity*, as this is only one, small preliminary study. Further, larger studies are desirable to better assess the performances and the scalability of the proposed approach.

4. EMERGING RESEARCH DIRECTIONS

Driven by bioinformatics, where the Granger causality test is largely used to derive causal relationships among different elements, such as genes and proteins, we have developed a novel change impact analysis method that uses the Granger causality test to learn impact relationships among software artifacts. The main idea of the method is to infer the mutual dependencies between software artifacts by measuring the statistical confidence that the time series representing the history of changes of a software artifact can be used to predict the changes of another artifact. The problem is therefore posed as a statistical test, evaluating the quality of forecasting of a variable given another one.

In this paper the method has been compared with the application of association rules [9, 10], and the preliminary results suggest that the Granger causality test can be a viable approach to change impact analysis, as it complements existing approaches. As a matter of fact, while association rules capture co-changes, Granger causality helps learning consequent changes. While not shown in this paper for the sake of space, we have compared the results of the proposed impact analysis method against traditional methods based on static analysis, and also in this case we found that they are able to highlight complementary, yet useful, impact relationships. This opens the way towards an eclectic impact analysis approach that combines existing methods, to overcome the limitation of the individual methods, and provides software engineers with a richer set of information useful to assess the impact of a change.

There are several directions in which the work presented here can be expanded. Of course, a first direction is the development of further empirical studies to fully understand advantages and limitation of the proposed method, to compare it with other (traditional) impact analysis techniques, and to develop heuristics for improvement.

The application of Granger causality depends entirely on the appropriate selection of variables. In this paper we have used an impulsive (0,1) time series to represent file changes in a snapshot. However, other variables should be explored, for example the variation of some metric values. Furthermore, the method could be applied for object-oriented system, considering classes and methods instead of functions.

We are interested to compare the Granger causality approach against the dynamic Bayesian network inference, which is a well-known approach used to derive causal relationships. Finally, we are investigating the use of information theoretic approaches based on mutual information.

5. REFERENCES

[1] R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 26-28, 1993*, pages 207–216. ACM Press, 1993.

[2] R. S. Arnold and S. A. Bohner. Impact analysis - towards a framework for comparison. In *Proceedings of the Conference on Software Maintenance, ICSM 1993, Montréal, Quebec, Canada, September 1993*, pages 292–301, 1993.

[3] G. Canfora and L. Cerulo. Impact analysis by mining software and change request repositories. In *11th IEEE International Symposium on Software Metrics (METRICS 2005), 19-22 September 2005, Como Italy*, page 29. IEEE Computer Society, 2005.

[4] C. W. J. Granger. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica*, 37(3):424–438, 1969.

[5] J. D. Hamilton. *Time Series Analysis*. Princeton University Press, January 1994.

[6] A. Hindle, M. W. Godfrey, and R. C. Holt. Mining recurrent activities: Fourier analysis of change events. In *31st International Conference on Software Engineering, ICSE 2009, May 16-24, 2009, Vancouver, Canada, Companion Volume*, pages 295–298, 2009.

[7] J. Law and G. Rothermel. Whole program path-based dynamic impact analysis. In *Proceedings of the 25th International Conference on Software Engineering, May 3-10, 2003, Portland, Oregon, USA*, pages 308–318. IEEE Computer Society, 2003.

[8] N. D. Mukhopadhyay and S. Chatterjee. Causality and pathway search in microarray time series experiment. *Bioinformatics*, 23(4):442–449, 2007.

[9] A. T. T. Ying, J. L. Wright, and S. Abrams. Source code that talks: an exploration of Eclipse task comments and their implication to repository mining. In *Proceedings of the 2005 International Workshop on Mining Software Repositories, MSR 2005, Saint Louis, Missouri, USA, May 17, 2005*. ACM, 2005.

[10] T. Zimmermann, P. Weisgerber, S. Diehl, and A. Zeller. Mining version histories to guide software changes. In *ICSE '04: Proceedings of the 26th International Conference on Software Engineering*, pages 563–572, 2004.