TA-RE: An Exchange Language for Mining Software Repositories

Sunghun Kim[1], Thomas Zimmermann[2], Miryung Kim[3], Ahmed Hassan[4], Audris Mockus[5], Tudor Girba[6], Martin Pinzger[7], E. James Whitehead, Jr.[1], and Andreas Zeller[2]

[1]University of California, Santa Cruz, CA, USA
{hunkim, ejw}@cs.ucsc.edu

[2]Saarland University, Saarbrücken, Germany
{tz, zeller}@acm.org

[3]University of Washington, USA
miryung@cs.washington.edu

[4]University of Waterloo, Canada
aeehassa@plg.uwaterloo.ca

[5]Avaya labs
audris@avaya.com

[6]University of Berne, Switzerland
girba@iam.unibe.ch

[7]University of Zurich, Switzerland
pinzger@ifi.unizh.ch

ABSTRACT
Software repositories have been getting a lot of attention from researchers in recent years. In order to analyze software repositories, it is necessary to first extract raw data from the version control and problem tracking systems. This poses two challenges: (1) extraction requires a non-trivial effort, and (2) the results depend on the heuristics used during extraction. These challenges burden researchers that are new to the community and make it difficult to benchmark software repository mining since it is almost impossible to reproduce experiments done by another team. In this paper we present the TA-RE corpus. TA-RE collects extracted data from software repositories in order to build a collection of projects that will simplify extraction process. Additionally the collection can be used for benchmarking. As the first step we propose an exchange language capable of making sharing and reusing data as simple as possible.

Categories and Subject Descriptors
D.2.7 [**Software Engineering**]: Distribution, Maintenance, and Enhancement – *Restructuring, reverse engineering, and reengineering,* K.6.3 [**Management of Computing and Information Systems**]: Software Management – *Software maintenance*

General Terms
Measurement, Experimentation

Keywords
Corpus, Software Repository Mining, Prediction, Analysis

1. INTRODUCTION
Software repositories, such as version archives, problem databases, newsgroups, and mailing lists, have been getting a lot of attention from researchers in recent years. They have been used to discover previously unknown information and evaluate existing software engineering approaches and theories. Mining software repositories (MSR) is an active research area.

MSR '06, May 22-23, 2006, Shanghai, China.

This has lead to a wide range of topics including co-change analysis [1, 3, 23], origin analysis [7, 11], signature change analysis [12], defect analysis and prediction [8], investigation of code clones [10], code decay [5], estimating drivers for software change effort [9] and quality [18], identifying key features of open source development process [14], chunking of software in order to facilitate distributed development teams [17], and constructing tools to identify expert developers [15].

Even though these research topics vary, every analysis needs to first extract data from software repositories. Developing such extraction tools requires a *non-trivial effort*, particularly for researchers new to this area. Kenyon was recently developed to simplify extraction from version archives [2]. However, such tools still require knowledge about version control systems and are thus difficult to learn.

Even though common tools may facilitate research, it remains *difficult to reproduce* existing results. First, some required information that is not available in software repositories has to be inferred using heuristics and through interviews of the project participants. The latter is often essential because different projects tend have different development processes and different change and reporting practices. Typical examples are the recovery of change transactions from CVS [22] and the identification of bug fixes [16]. The algorithms used differ widely in existing research efforts. Since choosing different parameters may lead to completely different results, *benchmarking is almost impossible*. Second, when analyzing open-source projects, researchers rely on the availability of those repositories in the future. However, this assumption is very optimistic in particular since many projects are currently migrating their CVS repositories to Subversion. As a result, the original CVS repositories may be gone in a few years.

We also want to analyze closed source projects. In the rare event such a code history becomes publicly available, it is unlikely we will have direct access to its SCM repository.

Other research areas address the above problems by providing a collection of common test cases or documents. Examples are the UCI Repository [19], the Reuters corpus [13] from text classification research, and the PROMISE Repository [20]. In this paper, we propose a similar solution: a collection of extracted software repositories called the TA-RE[1] corpus.

---

[1] TA-RE is a Korean word and means "group" or "cluster".

The TA-RE corpus consists of (1) an *exchange language* and (2) *extracted data* of a set of selected software projects that will allow researchers to reproduce and benchmark their experiments. The vision of TA-RE is that every paper on mining software repositories will share its extracted data via the TA-RE repository. Other researchers can then reuse this information without spending too much time on extraction.

TA-RE *needs to be widely accepted and adopted*; otherwise it will have no impact. One key to acceptance is for the data sharing to be as easy as possible. This leads to several requirements that are discussed in Section 2. The resulting exchange language is presented in Section 3, and Section 4 discusses alternatives to TA-RE and Section 5 presents related work. Section 6 concludes the paper with an outlook and future work.

2. REQUIREMENTS

The success of the TA-RE project depends on whether the research community will adopt it. Therefore we discuss several requirements for the corpus that would increase its appeal.

2.1 Completeness of Information

(E1) The exchange language should be able to describe all information that is available from most standard SCM systems:

1. *Transactions:* the author, date, log message, and the version of each changed file. This information enables reconstruction of proper snapshots.
2. *Changes:* the files that were changed, including their new content. This information suffices for lightweight syntactic analysis like creating abstract syntax trees.
3. *Snapshots:* a consistent state of a project after each transaction. This information is needed for static and dynamic program analysis, clone detection, etc.

Not all SCM systems provide the above information. For instance, for CVS the transaction information is not stored and has to be recovered by heuristics.

(E2) Additionally the exchange language should support information that can be inferred for most SCM systems:

1. *Source code positions* of classes, methods, or functions
2. *Size and location of the change:* which lines were added, deleted or modified
3. *Nature of a change:* adaptive, corrective, or perfective changes [16], fix-inducing changes [21]
4. *Counts:* number of methods, lines, changes or fixes
5. *References to other artifacts,* such as problem databases, mailing lists, and newsgroups

(E3) All information provided by the exchange language should come with a *quality* (or trust) annotation. A transaction from a Subversion archive may be of low quality if it was migrated from a CVS repository. Such annotations should describe known data quality problems or heuristics used to calculate the relevant attribute (see E4).

(E4) For inferred information the exchange language should provide ways to identify the *algorithm* that was used. Additionally, it should be possible to use different variants of an algorithm in the same dataset (e.g., different algorithms to recognize bug fixes).

(E5) The exchange language should be extensible in anticipation for new research interests.

2.2 Applicability to Research and Industry

(A1) The corpus should support closed-source projects. Such projects might be willing to share some information without revealing their actual source code. This means that TA-RE needs to provide tools to anonymize the extracted data. To simplify this process, the source code and the description of changes should be separated in the exchange language.

2.3 Usability

(U1) The exchange language should allow any researcher to provide new data with minimal effort.

(U2) The data from the corpus should be easy to use for researchers in their projects. In particular, the exchange language should be straightforward and must not be too difficult to parse, i.e., cross-references or complicated relations should be avoided.

(U3) The corpus itself should not be restricted to any platform. It should be usable for programs that are specific to any type of machine or system.

(U4) The exchange language should be well documented.

3. TA-RE CORPUS

We describe the TA-RE corpus exchange language in this section.

3.1 Available Information

The TA-RE exchange language can represent the following classes of data:

Extraction level 1: directly extractable data from SCM systems:
- Transaction information: author, transaction time, and change log
- All file contents (deltas) with the original directory structures
- File level co-change information

Extraction level 2: data obtained by further analysis, such as source code parsing “
- Entity (class, function, and method) level information and content
- File addition and deletion Information
- Unique identifier for each transaction and content
- Entity level co-change information

Mined data: data extracted using heuristics:
- Recovered transactions (CVS [22])
- Transaction, file, and entity level bug-fix data [6]
- Fix-inducing data at file and entity levels [21]
- Accumulated bug count at file and entity levels
- Origin relationship between entities [7, 11]
- Reference links among transactions, contents, and entities.

3.2 Corpus Model

The TA-RE corpus data contains two flavors: transaction and content data. The corpus data has multiple transactions, and a transaction has multiple contents. Instead of providing all contents of each transaction, TA-RE provides only changed (added, deleted, and modified) contents in each transaction, since it is possible to recover all transaction contents from only the changed contents. The content data consist of two parts: content metadata and original file content. The content metadata has metadata for the original file content such as reference, change status, count, and entity information. We separate the content metadata and original file content for two reasons: (1) to store binary files and (2) to make original file contents optional for closed source projects.

We use sequential numbers (starting from 1) for transaction and content identifiers. The transaction identifiers are ordered chronologically, hence the transaction 1 is older than the transaction 2. We can easily determine the transaction order from transaction identifiers. The content identifier is unique for the same file name. For example, file '*/src/foo.java*' will have the same identifier over all transactions.

Since we use numeric identifiers for both transactions and contents, we use prefixes to avoid possible confusion between their identifiers. We use the prefix '*t*' for transactions and the '*c*' for contents. Content metadata is stored as a file whose name is the combination of the content prefix and a content identifier such as *'c32'*. Since content exchange language consists of metadata data and original file content, we use file extensions to distinguish them: '*.meta*' for the metadata and '*.con*' for the original file content.

Each transaction has a directory whose name is the combination of the transaction prefix and a transaction identifier. Transaction information is stored as a file, '*transaction*' in the corresponding transaction directory. All contents (*.meta and *.con) of the transaction are stored in the directory as well. For example, for transaction 1, the '*t1*' directory is created, and contains the transaction information file ('*transaction'*) and content files ('*c[content-id].meta'* and '*c[content-id].con'*) of the transaction.

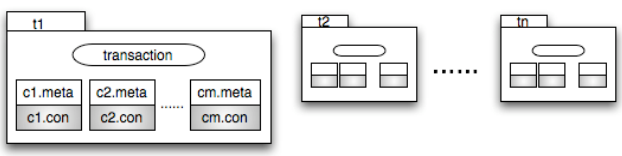

Figure 1. TA-RE Corpus Model

Figure 1 shows the TA-RE corpus model. Each transaction directory ('*t[transaction-id]'*) has three kinds of corpus files:

Transaction information ('*transaction*')**:** information of the corresponding transaction.
Content metadata ('*c[content-id].meta'*)**:** metadata of the content
Original file content ('*c[content-id].con*'): the original file content (optional)

The transaction and content metadata exchange language are formatted using XML. An example of transaction corpus exchange language is shown in Figure 2. It has the TA-RE exchange language version number, transaction id, release, author, data, indication of transaction nature, and change logs.

```
<?xml version="1.0" encoding="utf-8" ?>
<T:transaction xmlns:T="TA-RE:" id="t32">
  <T:corpus-version>0.1</T:corpus-version>
  <T:author>hunkim</T:author>
  <T:date>1995.3.1.1 xxx GMT</T:date>
  <T:nature kind="release" value="release 1.0"/>
  <T:nature kind="fix" heuristic="mockus2000"/>
  <T:nature kind="fix" heuristic="fischer2003"/>
  <T:change-log>Fixed compilation error in foo.c
  </T:change-log>
</T:transaction>
```

Figure 2. An example of transaction data

Figure 3 shows an exchange language example of a content metadata file. Only changed content data (added, deleted, modified) are present in TA-RE. The metadata have the original file name, references, counts, and entity data. The original file content can be fount at the '*c[content-id].con*' file in the same transaction directory. The detailed XML elements and DTD are described in http://tare.dforge.cse.ucsc.edu/.

```
<?xml version="1.0" encoding="utf-8" ?>
<T:content xmlns:T="TA-RE:" id="c32"
               filename="src/edu/ucsc/Kenyon.java">
  <T:corpus-version>0.1</T:corpus-version>
  <T:change-status value="modified"/>
  <T:reference kind="partof" level="transaction"
         transaction-id="t40"/>
  <T:reference kind="fixes" level="content"
        transaction-id="t29" content-id="c32"/>
  <T:reference kind="fixed-by" level="content"
        transaction-id="t45" content-id="c32"/>
  <T:reference kind="fixed-by" level="content"
        transaction-id="t99" content-id="c32"/>
  <T:count kind="accumulated-fix" value="2"/>
  <T:count kind="accumulated-fix-inducing" value="3"/>
  <T:count kind="accumulated-change" value="10"/>
  <T:entity level="class" id="class-foo" name="Foo"
      start-pos="20" end-pos="2564">
  <T:entity level="method" id=" foo" name="foo"
        return-type="void"  parameters="int I, char *var"
        start-pos="32" end-pos="95">
       <T:reference kind="fixes" level="entity">
        tansaction-id="t23" content-id="c32" entity-id="foo"/>
  </T:entity>
  <T:entity level="method" id=bar" name="bar"
         return-type="char"  parameters="int i, char c"
         start-pos="103" end-pos="195">
  </T:entity>
    ...
  </T:entity>
</T:content>
```

Figure 3. An example of a content metadata file. This content fixes the same content at transaction 29. This change includes bugs (fix-inducing changes). The bugs in this content change are fixed in transaction 45 and transaction 99. The original file content is stored in '*c32.con*' in the '*t40*' directory.

4. DISCUSSION

4.1 Why not use Traditional Extractors?

There are SCM fact extractors such as Kenyon [2] and APFEL [4]. These extractors are useful for extracting data from SCM systems without dealing with the SCM connections or protocols directly. Choosing different extractor options will yield different data from the same SCM repository. For example, the number of transactions and the number change contents of a transaction may be different when extraction tools use different CVS sliding windows times. Mined data in TA-RE such as bug-fix data or origin analysis data need to be provided by the extractor using their own heuristic options. Extracting different data from the same SCM systems makes it *difficult to reproduce* existing results.

4.2 Why not use DBMS Schemas?

Fischer et al. proposed DBMS schemas [6] to store data for software repository mining research. If the schema is complete and publicly available, the data in DBMS are beneficial for all software repository mining researchers. TA-RE provides an exchange language. It does not enforce any universal database schema because different research might need different formats. Use of an exchange format avoids this issue, as each researcher can write tools to export TA-RE to their project specific DB schema. Every researcher only has to write the import/export tools once and can reuse them for every project she downloads from TA-RE

4.3 Why not use Transaction-Aware SCM?

Transaction-aware SCM systems such as Subversion provide change based revision numbers (no need to recover transactions), log renaming events, and support metadata-setting features. Using SCM systems requires an extraction process, and it has the same limitations of using extractors (Section 4.1). TA-RE provides downloadable and ready-to-use data including all mined data such as bug-fix and big-inducing change information, which are not provided by SCM systems such as Subversion.

4.4 Closed Source Project Support

The TA-RE corpus exchange language can be used for closed source projects. First author information in transaction data files can be replaced with numeric ids to hide real author ids. The file names in the content metadata file can be omitted or replaced with obfuscated names. The original file contents stored in separate files (*c[content-id].con*) can be omitted. In addition, all entity information can be omitted.

5. RELATED WORK

The PROMISE repository provides various data sets for predictive model research in software engineering [20]. Data sets in the PROMISE repository mostly consist of features and classes or values. Using the features, researchers develop prediction models to predict classes (classification) or values (regression). PROMISE data sets are limited for general software repository mining research. Since they provide pre-defined features such LOC, count of operators, and count of blank lines, it is hard to extract new features that are not defined in the data set. The data sets are focused on developing predictive models. The non-predictive model research such as origin analysis, code clone genealogy, or co-change analysis cannot be performed using the data sets in PROMISE repository.

The UCI Repository of machine learning [19] or Reuters Corpus [13] are de facto standard benchmarking data set for text classification research. The data stets enables researchers to compare their classification results with others. TA-RE is inspired from them, but their data sets are designed for the text classification.

6. CONCLUSION AND FUTURE WORK

It is no secret that the majority of time spent during software repository mining is focused on extracting data. Additionally, the "magic" that is involved in the extracting phase makes comparison of results and benchmarking impossible. The TA-RE project addresses this issue by specifying a common exchange language that will be used to share project data. Using a common exchange language will enable reuse of data as much as possible. The next steps of this project are the following:

Finalize exchange language. This paper serves as a proposal for a common exchange language. Thus, designing a common language will heavily benefit from discussions and participation of other researchers. We hope that the discussions at the MSR workshop will give us enough feedback to finalize the exchange language.

Provide initial dataset. Once the exchange language is finalized, the participants of the TA-RE project will create an initial dataset for several selected projects.

Include other data sources. The initial exchange language will describe data only from version archives. For the next release, we plan to include additional data sources such as problem databases, mailing lists, or newsgroups.

For more information visit: http://tare.dforge.cse.ucsc.edu/
or join the discussion: http://groups.google.com/group/TaRe

7. REFERENCES

[1] J. Bevan and E. J. Whitehead, Jr., "Identification of Software Instabilities," Proc. of 2003 Working Conference on Reverse Engineering (WCRE 2003), Victoria, Canada, 2003.

[2] J. Bevan, E. J. Whitehead, Jr., S. Kim, and M. Godfrey, "Facilitating Software Evolution with Kenyon," Proc. of the 2005 European Software Engineering Conference and 2005 Foundations of Software Engineering (ESEC/FSE 2005), Lisbon, Portugal, pp. 177-186, 2005.

[3] D. Beyer and A. Noack, "Clustering Software Artifacts Based on Frequent Common Changes," Proc. of the 13th IEEE International Workshop on Program Comprehension (IWPC 2005), St. Louis, Missouri, USA, pp. 259-268, 2005.

[4] V. Dallmeier, P. Weißgerber, and T. Zimmermann, "APFEL: A Preprocessing Framework For Eclipse," 2005, http://www.st.cs.uni-sb.de/softevo/apfel/.

[5] S. G. Eick, T. L. Graves, A. F. Karr, J. S. Marron, and A. Mockus, "Does Code Decay? Assessing the Evidence from Change Management Data," *IEEE Transactions on Software Engineering*, vol. 27, pp. 1-12., 2001.

[6] M. Fischer, M. Pinzger, and H. Gall, "Populating a Release History Database from Version Control and Bug Tracking Systems," Proc. of 2003 Int'l Conference on Software Maintenance (ICSM'03), pp. 23-32, 2003.

[7] M. W. Godfrey and L. Zou, "Using Origin Analysis to Detect Merging and Splitting of Source Code Entities," *IEEE Trans. on Software Engineering*, vol. 31, pp. 166- 181, 2005.

[8] T. L. Graves, A. F. Karr, J. S. Marron, and H. Siy, "Predicting Fault Incidence Using Software Change History," *IEEE Transactions on Software Engineering*, vol. 26, pp. 653-661, 2000.

[9] T. L. Graves and A. Mockus, "Inferring Change Effort from Configuration Management Data," Proc. of In Metrics 98: Fifth International Symposium on Software Metrics, Bethesda, Maryland, pp. 267-273, 1998.

[10] M. Kim, V. Sazawal, D. Notkin, and G. Murphy, "An Empirical Study of Code Clone Genealogies," Proc. of the 2005 European Software Engineering Conference and 2005 Foundations of Software Engineering (ESEC/FSE 2005), Lisbon, Portugal, pp. 187-196, 2005.

[11] S. Kim, K. Pan, and E. J. Whitehead, Jr., "When Functions Change Their Names: Automatic Detection of Origin Relationships," Proc. of 12th Working Conference on Reverse Engineering (WCRE 2005), Pennsylvania, USA, 2005.

[12] S. Kim, E. J. Whitehead, Jr., and J. Bevan, "Analysis of Signature Change Patterns," Proc. of Int'l Workshop on Mining Software Repositories (MSR 2005), Saint Louis, Missouri, USA, pp. 64-68, 2005.

[13] D. Lewis, Y. Yang, T. Rose, and F. Li, "RCV1: A New Benchmark Collection for Text Categorization Research " *Journal of Machine Learning Research*, vol. 5, pp. 361-397, 2004.

[14] A. Mockus, R. F. Fielding, and J. Herbsleb, "A Case Study of Open Source Development: The Apache Server," Proc. of 22nd Int'l Conference on Software Engineering (ICSE 2000), Limerick, Ireland, pp. 263-272 2000.

[15] A. Mockus and J. Herbsleb, "Expertise Browser: A Quantitative Approach to Identifying Expertise," Proc. of 24rd Int'l Conference on Software Engineering (ICSE 2002), Orlando, Florida, pp. 503-512, 2002.

[16] A. Mockus and L. G. Votta, "Identifying Reasons for Software Changes Using Historic Databases," Proc. of International Conference on Software Maintenance (ICSM 2000), San Jose, California, USA, pp. 120-130, 2000.

[17] A. Mockus and D. M. Weiss, "Globalization by Chunking: a Quantitative Approach," *IEEE Software*, vol. 18, pp. 30-37, 2001.

[18] A. Mockus, P. Zhang, and P. Li, "Drivers for Customer Perceived Software Quality," Proc. of 2005 Int'l Conference on Software Engineering (ICSE 2005), Saint Louis, Missouri, USA, 2005.

[19] D. J. Newman, S. Hettich, C. L. Blake, and C. J. Merz, "UCI Repository of machine learning databases," 1988, http://www.ics.uci.edu/~mlearn/MLRepository.html.

[20] J. Sayyad Shirabad and T. J. Menzies, "The PROMISE Repository of Software Engineering Databases," 2005, http://promise.site.uottawa.ca/SERepository.

[21] J. Sliwerski, T. Zimmermann, and A. Zeller, "When Do Changes Induce Fixes?" Proc. of Int'l Workshop on Mining Software Repositories (MSR 2005), Saint Louis, Missouri, USA, pp. 24-28, 2005.

[22] T. Zimmermann and P. Weißgerber, "Preprocessing CVS Data for Fine-Grained Analysis," Proc. of Int'l Workshop on Mining Software Repositories (MSR 2004), Edinburgh, Scotland, pp. 2-6, 2004.

[23] T. Zimmermann, P. Weißgerber, S. Diehl, and A. Zeller, "Mining Version Histories to Guide Software Changes," *IEEE Trans. Software Engineering*, vol. 31, pp. 429-445, 2005.