On Software Engineering Repositories and Their Open Problems

Daniel Rodriguez
University of Alcala
Alcala de Henares, Spain
daniel.rodriguez@uah.es

Israel Herraiz
Technical University of Madrid
Madrid, Spain
israel.herraiz@upm.es

Rachel Harrison
Oxford Brookes University
Oxford, United Kingdom
rachel.harrison@brookes.ac.uk

***Abstract*—In the last decade, a large number of software repositories have been created for different purposes. In this paper we present a survey of the publicly available repositories and classify the most common ones as well as discussing the problems faced by researchers when applying machine learning or statistical techniques to them.**

***Keywords*-quality; software engineering repositories; preprocessing software engineering data; data quality**

I. Introduction

Many research studies in the field of Empirical Software Engineering are based on a few case studies or small samples. For instance, Mockus *et al.* [1] found that free / open source software contained fewer defects than proprietary systems, basing their conclusions on two cases of open source projects (Apache and Mozilla). Another example is the distribution of bugs in software, which was recently found to be a Weibull distribution [2] based on some releases of a single case study (the Eclipse IDE). But this result was refuted by a report with contradictory results, stating that the distribution of bugs can also be described using other statistical models [3].

Empirical research that is based on small datasets will have to refute contradictory results because of lack of generalisation. However, gathering a large amount of software and data for empirical studies can be a cumbersome task, prone to the introduction of unintentional errors, and potentially causing more problems than they solve.

The popularization and rise of the free / open source software development phenomenon has made available vast amounts of data which are useful for research purposes. Thus, we can find several opportunities in the research community to obtain data for large samples of software projects, and in an integrated and structured manner, so these repositories can be easily queried to extract information. Even some *closed* repositories with more specialized information have appeared.

These repositories can be applied in any area of the Empirical Software Engineering field. We highlight the case of Search Based Software Engineering (SBSE), because it is particularly suitable for large amounts of data [1] [4].

[1]http://crestweb.cs.ucl.ac.uk/resources/sbse_repository/

SBSE deals with research into search and metaheuristic techniques in software engineering and has become an important area of research. Many SBSE problems are composed of one or more fitness functions that evaluate a search space, which can be generated while searching for the solution or from repositories forming a combinatorial problem from dataset attributes.

Another source of data is simulation, for example, using System Dynamics. Shepperd and Kaoda [5] use simulated data to compare effort estimation methods and it can share some of the problems highlighted later on. We however do not review the technique and specific issues of this approach, and focus on the case of reusable research datasets.

In this position paper we want to provide a preliminary review and classification/characterization of currently available repositories as well as to highlight the most common problems that users and researchers face when dealing with such repositories.

II. Software Engineering Repositories

Software projects leave a trail in different kinds of repositories, and this trail can be used to reconstruct the history of the project, and to study the software development and maintenance processes. We classify this trail in the following categories:

- Source code
  This is the most obvious product of the a software project. Source code can be studied to measure its properties, such as size or complexity.
- Source Code Management Systems (SCM)
  SCM repositories make it possible to store all the changes that the different source code files undergo during the project. Also, SCM systems allow for work to be done in parallel by different developers over the same source code tree. Every change recorded in the system is accompanied with meta-information (author, date, reason for the change, etc) that can be used for research purposes.
- Issue tracking systems
  Bugs, defects and user requests are managed in issue tracking systems, where users and developers can fill tickets with a description of a defect found, or a desired new functionality. All the changes to the ticket are

recorded in the system, and most of the systems also record the comments and communications among all the users and developers implied in the task.

- Messages between developers and users
  In the case of free / open source software, the projects are open to the world, and the messages are archived in the form of mailing lists, which can also be mined for research purposes. There are also some other open message systems, such as IRC or forums. Other projects which are developed in public can also store messages, but it is unusual to have that information for research purposes.
- Meta-data about the projects
  As well as the low level information of the software processes, we can also find meta-data about the software projects which can be useful for research. This meta-data may include intended-audience, programming language, domain of application, license (in the case of open source), etc.
- Usage data
  In the case of the user side, the trail that projects leave is virtually invisible. There are statistics about software downloads on the Internet, but that is not the only way users get their software. Some of the research datasets we describe in this paper include information about usage data, which is recorded thanks to the collaboration of users.

III. Research datasets

As stated previously there is a large number of repositories that have been created in the last decade that allow researchers to study different aspects within the software engineering field with statistical or data mining techniques.

In this paper we analyze the following repositories:

- FLOSSMole [6]
  `http://flossmole.org/`
- FLOSSMetrics [7]:
  `http://flossmetrics.org/`
- PROMISE (PRedictOr Models In Software Engineering) [8]:
  `http://promisedata.org/`
- Qualitas Corpus (QC) [9]:
  `http://qualitascorpus.com/`
- Sourcerer Project [10]:
  `http://sourcerer.ics.uci.edu/`
- Ultimate Debian Database (UDD) [11]
  `http://udd.debian.org/`
- Bug Prediction Dataset (BPD) [12], [13]:
  `http://bug.inf.usi.ch/`
- The International Software Benchmarking Standards Group (ISBSG) [14]
  `http://www.isbsg.org/`
- Eclipse Bug Data (EBD) [11], [15]:
  `http://www.st.cs.uni-saarland.de/softevo/bug-data/eclipse/`
- Software-artifact Infrastructure Repository (SIR) [16]
  `http://sir.unl.edu`
- ohloh [17]
  `http://www.ohloh.net/`
- SourceForge Research Data Archive (SRDA) [18]
  `http://zerlot.cse.nd.edu/`
- Helix Data Set [19]:
  `http://www.ict.swin.edu.au/research/projects/helix/`

IV. Classification of the datasets

We can classify the repositories into several orthogonal dimensions:

- Type of information stored in the repositories:
  - Meta-information about the project itself and the people that participated.
  - Low-level information
    - Mailing Lists (ML)
    - Bugs Tracking Systems (BTS) or Project Tracker System (PTS)
    - Software Configuration Management Systems (SCM)
  - Processed information. For example Project management information about the effort estimation and cost of the project.
- Whether the repository is public or not
- Single project vs. multiprojects. Whether the repository contains information of a single project with multiples versions or multiples projects and/or versions.
- Type of content, open source or industrial projects
- Format in which the information is stored. The repositories analysed here provide the information using different formats or technologies for accessing the information:
  - Text. It can be just plain text, CSV (Comma Separated Values) files, Attribute-Relation File Format (ARFF) [20] or its variants
  - Through databases. Downloading dumps of the database.
  - Remote access such as APIs of Web services or REST.

V. Quality issues / Open problems

We can also classify the problems related to the extraction of information or from the actual data stored in the repositories.

A. *Problems generated from extracting the information*

Robles et al. [21] describe the processes and tools to extract information needed to analyse software repositories.

Table I
SUMMARY OF THE REPOSITORIES

	Meta-info	*Low-level Info*	*Public?*	*Single vs. Multi*	*OSS*	*Format/Access*
FLOSSMole	Yes	No	Yes	Multi	Yes	DB dumps, text, DB access
FLOSSMetrics	No	Yes	Yes	Multi	Yes	DB dumps, web service, web
PROMISE	Some datasets	Some datasets	Yes	Multi	Most datasets not from OSS	Mostly ARFF
QC	Yes	Yes	Yes	Multi	Yes	CSV, source code, JAR
Sourcerer	No	Yes	Yes	Multi	Yes	DB dumps?
UDD	Yes	Yes	Yes	Single (Debian)	Yes	DB dump
BPD	No	Yes	Yes	Yes (5 Java Systems)	Yes	CSV
ISBSG	Project management data	No	No	Multi	No	Spreadsheet
EBD	No	yes	Yes	Single (Eclipse)	Yes	ARFF, CSV
SIR	No	Actual code for testing	Needs registration (commercial licence)	Multi	Yes	C/Java /C#
ohloh	Yes	Yes	Yes	Multi	Yes	Web service (limited)
SRDA	Yes (from SF.net)	Yes	Needs registration, only research	Multi	Yes	BD dumps

Although the process is quite similar to the general process of data mining described by Fayyad et al. [22], it has it own characteristics and difficulties. There is large variability in the formats and tools needed, standards, etc. that make the data gathering process a very labour intensive one. Another example is the mining of textual data to deal with bugs for classification, clustering, etc. This is a difficult task even with human intervention because change requests and incident reports are often mixed together in the BTS or PTS.

B. Replicability

Replicability is one of the main reasons to adopt open repositories [23]. Kitchenham [24] also discusses the risk of replicating experiments without using the original sources. It is a well known fact that in the data mining process, one of the hardest tasks is to preprocess the data. However, trusting the preprocessed data from others is a poisened chalice. For example, Shepperd has reported differences between using an original dataset or a preprocessed one downloaded from the PROMISE repository [8]. Among the repositories discussed, EBD not only contains the data but also the necessary scripts to replicate the study.

C. Data quality problems related to machine learning

From the statistical and data mining point of view, we face many of the generic problems we discuss in this section. Therefore, in addition to specific tool issues (e.g., [25]) we need to be aware of many of the statistical and data mining problems we may face when dealing with software engineering repositories.

- Outliers.
  Although this statistical problem is well known in the literature, it is not always properly reported for example in many estimation studies as stated by Turhan [26].
- Missing values and inconsistencies.
  Some of the repositories such as the ISBSG, are composed of a large number of attributes, however, many of those attributes are missing values that need to be discarded in order to apply machine learning algorithms. There are also inconsistencies in the way information is stored [27]. In this particular dataset, cleaning inconsistencies (e.g., languages classified as 3GL or 4GL, Cobol 2 or Cobol II) can be risky.
- Redundant and irrelevant attributes and instances.
  It is also well known that the existence of irrelevant and redundant features in the datasets has a negative impact in most data mining algorithms, which assume a certain level of balance between the class attributes. Feature Selection has been applied and studied by the software engineering community, not so much instance selection which needs further research (a few exceptions for effort estimation include [28], [29]). It is known, however, that feature selection algorithms do not perform well with imbalanced datasets, resulting in a selection of metrics that are not adequate for the learning algorithms. This problem can happen in most effort estimation or defect prediction datasets. For example, the ISBSG that has over 60 attributes most of them are irrelevant or the 8000 repeated rows in JM1 from NASA's defect prediction datasets in PROMISE. Also the defect prediction datasets such the EB data are highly unbalanced. Some further research into robust algorithms such as Subgroup Discovery techniques is also needed [30] or weighting of attributes and instances.
- Overlapping or class separability.
  When dealing with classification, we may also face the problem of overlapping between classes in which a

region of the data space contains samples from different values for the class. We have found that many samples from the NASA dataset contained in the PROMISE repository are contradictory or inconsistent, many instances have the same values for all attributes with the exception of the class, making the induction of good predictive models difficult.

- Data shifting.
  The data shift problem happens when the test data distribution differs from the training distribution. Turhan [26] discusses the *dataset shift* problem in software engineering (effort estimation and defect prediction). It is customary in data mining, to preform the evaluation using cross-validation, i.e., divide the dataset into k-folds for training and testing and report the averages of the k folds. This problem can easy happen when we are dealing with small datasets [31]. Also when we are dealing with small datasets, it can happen that the number of instances that remain in the training dataset is skewed. Many software effort estimation datasets are very small (around 20 effort estimation datasets contained in PROMISE repository contain just over a dozen samples, e.g., the Kemerer or Telecom datasets)
- Imbalance.
  This happens when samples of some classes vastly outnumber the cases of other classes. Under this situation, when the imbalanced data is not considered, many learning algorithms generate distorted models for which (i) the impact of some factors can be hidden and (ii) the prediction accuracy can be misleading. Although this is a well-known problem in the data mining community, this problem has not been addressed in detail by the software engineering community. This is typically addressed by preprocessing the datasets with sampling techniques or considering cost in the data mining algorithms (making the algorithms more robust). This problem happens in many of the defect prediction datasets (e.g. the PROMISE repository has around 60 defect prediction datasets). The previous problems, redundant and irrelevant attributes, overlapping, data shifting and small datasets are made worse when datasets are imbalanced [32].
- Metrics.
  In relation to the measurements, either from the social network data, mailing lists or code, there can be differences depending on the tool used in those repositories that contain source code such FLOSMetrics, EBD, or BPD. For example, Lincke et al. [33] report on large differences in metrics collected from the code depending on the tool used.
- Evaluation metrics and the evaluation of models.
  For example, Shepperd and MacDonell [34] report on the the use and abuse of using MMRE (Mean Magnitude of Relative Error) when dealing with effort estimation. Despite the fact that MMRE has been known to be biased and favours underestimation, perhaps because it is easy to apply, it has been used to wrongly validate and compare different estimation methods or models. Furthermore, as such metrics can be used as fitness functions in metaheuristic algorithms [35], the solutions obtained may be suboptimal.

VI. Conclusions

In this position paper we have discussed the current data repositories that are available for Software Engineering research. We classified them and discussed some common problems faced when extracting information from them. We have also discussed data related problems when applying machine learning techniques. Although some of the problems such as outliers or noise have been extensively studied in software engineering, others need futher research, in particular, imbalance and data shifting from the machine learning point of view and replicability in general, providing not only the data but also the tools to replicate the empirical work.

References

[1] A. Mockus, R. T. Fielding, and J. D. Herbsleb, "Two case studies of Open Source software development: Apache and Mozilla," *ACM Transactions on Software Engineering and Methodology*, vol. 11, no. 3, pp. 309–346, 2002.

[2] H. Zhang, "On the distribution of software faults," *IEEE Transactions on Software Engineering*, vol. 34, no. 2, pp. 301–302, march-april 2008.

[3] G. Concas, M. Marchesi, A. Murgia, R. Tonelli, and I. Turnu, "On the distribution of bugs in the eclipse system," *IEEE Transactions on Software Engineering*, vol. 37, no. 6, pp. 872–877, Nov-Dec 2011.

[4] M. Harman and B. F. Jones, "Search-based software engineering," *Information and Software Technology*, vol. 43, no. 14, pp. 833–839, 2001.

[5] M. Shepperd and G. Kadoda, "Comparing software prediction techniques using simulation," *IEEE Transactions on Software Engineering*, vol. 27, no. 11, pp. 1014–1022, nov 2001.

[6] J. Howison, M. Conklin, and K. Crowston, "FLOSSmole: A collaborative repository for FLOSS research data and analyses," *International Journal of Information Technology and Web Engineering*, vol. 1, no. 3, 2006.

[7] I. Herraiz, D. Izquierdo-Cortazar, F. Rivas-Hernandez, J. M. Gonzalez-Barahona, G. Robles, S. D. nas Dominguez, C. Garcia-Campos, J. F. Gato, and L. Tovar, "FLOSSMetrics: Free / libre / open source software metrics," in *Proceedings of the 13th European Conference on Software Maintenance and Reengineering (CSMR)*. IEEE Computer Society, 2009.

[8] G. Boetticher, T. Menzies, and T. Ostrand. (2007) Promise repository of empirical software engineering data. West Virginia University, Department of Computer Science. [Online]. Available: http://promisedata.org/ repository

[9] E. Tempero, C. Anslow, J. Dietrich, T. Han, J. Li, M. Lumpe, H. Melton, and J. Noble, "Qualitas corpus: A curated collection of java code for empirical studies," in *2010 Asia Pacific Software Engineering Conference (APSEC2010)*, Dec 2010.

[10] E. Linstead, S. Bajracharya, T. Ngo, P. Rigor, C. Lopes, and P. Baldi, "Sourcerer: mining and searching internet-scale software repositories," *Data Mining and Knowledge Discovery*, vol. 18, pp. 300–336, 2009, 10.1007/s10618-008-0118-x.

[11] T. Zimmermann, R. Premraj, and A. Zeller, "Predicting defects for eclipse," in *Proceedings of the Third International Workshop on Predictor Models in Software Engineering (PROMISE'07)*, ser. PROMISE '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 9–.

[12] M. D'Ambros, M. Lanza, and R. Robbes, "An extensive comparison of bug prediction approaches," in *Proceedings of the 7th IEEE Working Conference on Mining Software Repositories (MSR07)*. IEEE CS Press, 2010, pp. 31 – 41.

[13] M. D'Ambros, M. Lanza, and R. Robbes, "Evaluating defect prediction approaches: a benchmark and an extensive comparison," *Empirical Software Engineering*, pp. 1–47, 10.1007/s10664-011-9173-9.

[14] C. Lokan, T. Wright, P. Hill, and M. Stringer, "Organizational benchmarking using the isbsg data repository," *IEEE Software*, vol. 18, no. 5, pp. 26 –32, sep/oct 2001.

[15] N. Nagappan, A. Zeller, T. Zimmermann, K. Herzig, and B. Murphy, "Change bursts as defect predictors," in *Proceedings of the 21st IEEE International Symposium on Software Reliability Engineering (ISSRE 2012)*, November 2010.

[16] H. Do, S. Elbaum, and G. Rothermel, "Supporting controlled experimentation with testing techniques: An infrastructure and its potential impact," *Empirical Software Engineering*, vol. 10, pp. 405–435, October 2005. [Online]. Available: http://dl.acm.org/citation.cfm?id=1089922.1089928

[17] Ohloh, "Ohloh." [Online]. Available: http://www.ohloh.net/

[18] M. Van Antwerp and G. Madey, "Advances in the sourceforge research data archive (srda)," in *Fourth International Conference on Open Source Systems, IFIP 2.13 (WoPDaSD 2008)*, Milan, Italy, September 2008.

[19] R. Vasa, "Growth and change dynamics in open source software systems," Ph.D. dissertation, Faculty of Information and Communication Technologies Swinburne University of Technology Melbourne, Australia, 2010.

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

[21] G. Robles, J. M. Gonzalez-Barahona, D. Izquierdo-Cortazar, and I. Herraiz, "Tools for the study of the usual data sources found in libre software projects," *International Journal of Open Source Software and Processes*, vol. 1, no. 1, pp. 24–45, Jan-March 2009.

[22] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "The kdd process for extracting useful knowledge from volumes of data," *Communications of the ACM*, vol. 39, pp. 27–34, November 1996.

[23] J. González-Barahona and G. Robles, "On the reproducibility of empirical software engineering studies based on data retrieved from development repositories," *Empirical Software Engineering*, vol. 17, pp. 75–89, 2012, 10.1007/s10664-011-9181-9. [Online]. Available: http://dx.doi.org/10.1007/s10664-011-9181-9

[24] B. Kitchenham, "The role of replications in empirical software engineering: a word of warning," *Empirical Software Engineering*, vol. 13, no. 2, pp. 219–221, 2008.

[25] R. P. Buse and T. Zimmermann, "Information needs for software development analytics," in *Proceedings of the 34th International Conference on Software Engineering*, June 2012.

[26] B. Turhan, "On the dataset shift problem in software engineering prediction models," *Empirical Software Engineering*, vol. 17, pp. 62–74, 2012, 10.1007/s10664-011-9182-8.

[27] D. Rodríguez, M. Sicilia, E. García, and R. Harrison, "Empirical findings on team size and productivity in software development," *Journal of Systems and Software*, vol. 85, no. 3, pp. 562–570, 2012.

[28] C. Kirsopp and M. Shepperd, "Case and feature subset selection in case-based software project effort prediction," in *Proceedings of 22nd International Conference on Knowledge-Based Systems and Applied Artificial Intelligence (SGAI'02)*, 2002.

[29] Z. Chen, T. Menzies, D. Port, and D. Boehm, "Finding the right data for software cost modeling," *IEEE Software*, vol. 22, no. 6, pp. 38–46, Nov-Dec 2005.

[30] D. Rodríguez, R. Ruiz, J. Riquelme, and J. Aguilar-Ruiz, "Searching for rules to detect defective modules: A subgroup discovery approach," *Information Sciences*, vol. 191, no. 0, pp. 14–30, 2012.

[31] S. Raudys and A. Jain, "Small sample size effects in statistical pattern recognition: recommendations for practitioners," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 3, pp. 252–264, mar 1991.

[32] A. Fernández, S. García, and F. Herrera, "Addressing the classification with imbalanced data: Open problems and new challenges on class distribution," in *6th International Conference on Hybrid Artificial Intelligence Systems (HAIS)*, 2011, pp. 1–10.

[33] R. Lincke, J. Lundberg, and W. Löwe, "Comparing software metrics tools," in *Proceedings of the 2008 International Symposium on Software Testing and Analysis (ISSTA'08)*. ACM, 2008, pp. 131–142.

[34] M. Shepperd and S. MacDonell, “Evaluating prediction systems in software project estimation,” *Information and Software Technology*, pp. –, 2012.

[35] M. Harman and J. Clark, “Metrics are fitness functions too,” in *Proceedings. 10th International Symposium on Software Metrics 2004*, sept. 2004, pp. 58–69.