

Sharing Jazz development data: a community perspective

Annie T.T. Ying, Kate Ehrlich, Li-Te Cheng, Harold L. Ossher,
Thomas V. Fraunhofer, Frank van Ham
IBM Watson Research Center

{aying, katee, li-te_cheng, ossher, tvf, fvanham}@us.ibm.com

ABSTRACT

Common data sets—and the communities around them—often benefit a research community. To contribute to the research community, IBM is making the Jazz development data available via a community led by academic researchers. The contribution of the Jazz development data is important, as this data is the only substantial data set collected by Jazz[®], a platform that records richer data with more types of artifacts and traceability than data collected by existing platforms. In this paper, we describe the technical challenges in preparing this data for a small group of external researchers, with the intent of eventually building a community around the data.

Categories and Subject Descriptors

D.2.7 [Software Engineering]: Distribution, Maintenance, and Enhancement – *Restructuring, reverse engineering, and reengineering.*

General Terms

Measurement, Experimentation.

Keywords

Jazz, Corpus, Software Repository Mining, Infrastructure

1. INTRODUCTION

Common data sets—and the communities around them—often provide a point for dissemination for a research community. Such data and communities can make evaluation more comparable and authoritative. Researchers can more easily and effectively collaborate with others, share their work with others, and build upon each others' contributions. Many of the public funding agencies such as the National Science Foundation have recently emphasized the importance of sharing large datasets [2]. Examples of de facto common data sets in Computer Science include UCI ML Repository¹ for machine learning and TREC data sets² for natural language processing. However, there are both technical and social challenges in making data more publicly available and accessible (e.g. [13]). Technical challenges include a) providing appropriate restrictions to prevent the data from getting into the wrong hands or being misused, b) hiding private or confidential information, c) using common formats so that researchers can access the data on a variety of hardware and software platforms and d) providing provenance that validates the

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

authenticity of the data and indicates how it was generated. It is also necessary to provide documentation to inform researchers the content and organization. Besides these technical challenges, more vexing social challenges include communicating the context, purpose and intent of the data. Social challenges include: a) the willingness of researchers to share data especially in competitive research settings, b) understanding enough about the conditions and context in which the data were created to use and analyze the data appropriately and c) having sufficient shared language between researchers to communicate accurately about the data [4].

To contribute to the research community, IBM is making the Jazz³ development data available. Jazz is an integrated end-to-end solution from IBM aimed at collaborative software development for software engineering stakeholders. To researchers, Jazz provides comprehensive types of artifacts, rich traceability information, and a great infrastructure for research in collaborative software development. Being a new platform released as Beta to on the Jazz community website in June 2007, Jazz has only been deployed on one large team: the Jazz development organization itself, consisting of over 160 IBM professional software engineers spanning multiple teams and time zones. To help encourage research on Jazz, IBM is making this data set available via a community led by academic researchers.

In this paper, we describe the technical challenges we encountered preparing this Jazz data for a small group of external researchers, with the intent of eventually building a community around the data. These challenges mirror the four challenges we mentioned in other communities centered on data sharing. We describe the steps we took to approach the challenges, and how the Jazz platform helped and impeded our work.

This paper has two major goals: The first is to provide transparency and to present how we plan to make the Jazz development data available to the research community. The second is to present our experience as an example for companies thinking about contributing their software engineering data to the academic research community.

The rest of the paper is organized as follows. Section 2 summarizes Jazz and related work. Section 3 describes the challenges in externalizing the Jazz development data. Section 4 concludes and provides some next steps.

¹ <http://archive.ics.uci.edu/ml/>

² <http://trec.nist.gov>

³ <http://jazz.net>

2. JAZZ AND RELATED WORK

Jazz is a team collaboration platform for seamlessly integrating tasks across the software lifecycle. There are three major themes in the Jazz platform: enabling team collaboration, empowering governance, and ease of administration. First, Jazz enables teams to collaborate, by raising team awareness and automation; providing collaboration features; and supporting a broad range of business and technology users by providing them both an IDE-integrated and a web-based interface. Second, Jazz empowers governance by providing process awareness and automation for different processes, such as agile and RUP-based [11] processes; flexible, rules-based process definition; and project dashboards. Finally, Jazz provides support for administration by providing a flexible administration model; connector framework, with integration to other change management systems; and open middleware interface, enabling Jazz to use open source middleware (Apache Tomcat⁴ server and Derby⁵ database) and commercial middleware (IBM DB2⁶ and Oracle Database⁷).

In the rest of this section, we show how the data collected by the Jazz platform differs from that by similar systems, from the perspective of a researcher analyzing software engineering data.

2.1 SCM Systems

We compare Jazz's software configuration management (SCM) system with other existing SCM systems in the following aspects:

Database vs. file formats: SCM systems such as CVS⁸ and Subversion⁹ typically store data in file formats. In contrast, the Jazz repository back-end is a relational database. A database back-end provides many benefits for researchers analyzing the data. First, the structured data stored in database makes retrieval of basic information easy. Information is already stored according a schema of the database in Jazz, rather than in textual files that require parsing in many other SCM systems. Second, a database provides many desirable features for data retrieval purposes, such as scalability, availability, and the ability to parallel process.

More traceability and types of artifacts in Jazz: Jazz collects more contextual data, and traceability information to other artifacts, such as Work Items (discussed in Section 2.2), author information, team information, and build information. For example, a researcher can find out who delivers the code (sharing and creating a version of the code in Jazz) to the repository as well as detailed information about this person, such as what team and role he/she is in, whereas CVS does not provide such information.

Cleaner data in Jazz: Many capabilities provided by Jazz make the data collected by Jazz much cleaner than data recorded by other SCM systems. For example, CVS does not automatically store a semantic change that typically spread across multiple files as a single unit, called transactions, or Change Sets in Jazz. Researchers analyzing the CVS data often have to recover these

transactions using heuristics. SCM systems such as Subversion and Jazz's SCM system store changes to files belonging to the same transaction.

Jazz lacks documentation and support: Documentation on Jazz's database targeted to help researchers mining data collected by Jazz is nearly non-existent. This is not surprising since Jazz is designed for software engineering stakeholders and Jazz is still in its Beta release. More mature SCM systems such as CVS benefit from more documentation and a range of existing tools to support analysis (e.g., CVSSuck¹⁰).

2.2 Change Tracking Data

Work Items (reports on bugs, enhancements, or tasks in Jazz) and reports in Bugzilla¹¹, a popular open-source change tracking system, share many similarities. The differences between Work Items and Bugzilla-like reports are mostly a result of the Jazz's integration with other features for collaborative software engineering.

Traceability in Jazz, e.g., between Work Items and Change Sets: Rich traceability in Jazz enables Jazz to collect valuable information that is not usually possible with change tracking systems. In particular, because Jazz integrates the change tracking system and SCM, Jazz can provide linkages between a Work Item and code changes that pertain to the Work Item.

Work Items and Bugzilla reports have similar fields: These fields include creator, owner, severity, status, some notion of target completion time, some notion of what component the task belongs to, and timestamp. Work Items have additional fields that indicate the team that the Work Item belongs to, whereas Bugzilla does not have the notion of teams.

Work Items and Bugzilla reports have similar life-cycle: They both go through a "New" status, typically through a triaged status, and typically to a resolved status. The resolved status could mean that the work is complete, invalid, a duplicate, or the work will not be done for now. Work Items has an additional "Start working" state, which can benefit researchers who want to determine the amount of time to complete a task more accurately.

2.3 Communication Data

Communication data is typically hard to obtain for inferring social network relationships. Ehrlich et al relied on a survey to infer social networks in their analysis of social-technical congruence gaps [8]. The disadvantage of this approach is that the survey is retrospective and is not tied to particular tasks. Cataldo et al used the content of the Internet Relay Chat systems to determine when particular communication between a pair of people was related to a particular modification request [5]. Algorithms for mining other communication sources such as email or chat exist, but privacy considerations often prevent access to such data. Even with access, researchers often still need to relate this communication data to work artifacts. Both Bugzilla and the Jazz Work Items keep track of the task-based communication that researchers can use for building social network. Jazz also provides data on team structure, a valuable

⁴ <http://tomcat.apache.org/>

⁵ <http://db.apache.org/derby/>

⁶ <http://www-306.ibm.com/software/data/db2/>

⁷ <http://www.oracle.com/database/index.html>

⁸ <http://www.nongnu.org/cvs/>

⁹ <http://subversion.tigris.org/>

¹⁰ <http://cvs.m17n.org/~akr/cvssuck/>

¹¹ <http://www.bugzilla.org/>

data source for social-network related analyses that other platforms such as Bugzilla do not provide.

2.4 Infrastructure and Data Sets

Before analyzing the data from various repositories, researchers often need to extract the data to a more usable form. Several existing infrastructures aim to ease this task. We compare Jazz with these infrastructures on the following aspects:

Support for storing analysis results: The Jazz platform supports extensions to the repository for storing user-defined items. Because the Jazz development data we plan to distribute is compatible with the Jazz tools, researchers can take advantage of this Jazz repository's extensibility to store their analysis results. Kenyon is designed to incorporate analysis results through its object-relational mapping system [3]. Other infrastructures such as Bloof developed by Drahelim and Pekacki are not designed to incorporate analysis results [7].

Support for different data sources: Many mining infrastructures support multiple types of repositories. For example, Kenyon is designed to support multiple data source [3]. Others have more limited data sources, such as RHDB on CVS [8]. Jazz is not designed as a mining infrastructure that supports different data sources, but its connector framework can integrate with other source control and change management systems such as IBM ClearCase¹², IBM ClearQuest¹³, and Subversion.

Database vs. other formats: Jazz uses database as a back-end, as many infrastructures such as RHDB and Minero [1]. These infrastructures can take advantage of the capabilities provided by the database. One disadvantage is with the fixed schemas, which may not be suitable for all analyses. Other proposals attempt to solve this problem. For example, TA-RE is an exchange language developed by a community of researchers in the mining software repositories area [11], and is meant to be used as a format for transformation between different data sources. Kenyon supports a database backend and does not place significant requirements on the capabilities of its underlying database.

Raw data vs. extracted data: Many existing mining infrastructures provide some support on some basic extraction of data or commonly data cleaning operations. For example, the PROMISE¹⁴ repository contains datasets of extracted features, and thus is limited to predictive model research, as it is hard to extract other features or information from these datasets. RHDB associates bug-tracking data with version control data and stores the results for later use. OSSmole is intended for project-level analysis. Kenyon focuses on program-level analysis [6]. Bloof allows users to define custom evolution metrics. Jazz does not explicitly provide these data cleaning and preprocessing operations, but the raw data collected by Jazz already contain some of the information that existing mining infrastructure tries to extract, such as change set information.

Live analyses vs. mirroring the data: The database backend in Jazz allows on-line analysis on data collected by the Jazz platform and avoids the need to mirror the data in the repository.

Exchange languages, such as TA-RE, require data to be exported from the data source, hindering the ability to do live mining. Other infrastructures, such as RHDB and Minero, are designed to mirror the data, and hence require extra steps to keep the source and the mirror synchronized for on-line analysis. On the other hand, replicating the data provides reliability for the external data that researchers have no control on (e.g., the data source can be discontinued.)

Communities: The MSR Mining Challenge¹⁵ brings together researchers and practitioners who are interested in applying, comparing, and challenging their mining tools and approaches on software repositories for the Eclipse open source project. The annual social network conference (INSNA) runs a competition called Vizards where a single dataset is made available and all the presenters present their approach to visualizing it.

3. TECHNICAL CHALLENGES

This section presents the technical challenges in preparing the Jazz data. These challenges mirror the ones we mentioned in other communities centered on data sharing: providing restrictions on the data (Section 3.1), hiding private and confidential information (Section 3.2), using common formats (Section 3.3), and providing provenance (Section 3.4).

3.1 Restrictions on the data

We initially offer the Jazz development data to a small group of external researchers who received Jazz grants. Jazz grants are established by IBM, with the purpose of promoting research on the Jazz platform. Offering the data only to a small initial group makes the distribution and restrictions on the data easier. There are two major questions: how we are planning to distribute this large database, and how we construct the terms of use.

Regarding the distribution mechanism to this small group, we are first distributing the data through a physical media at this Infrastructure for Research in Collaborative Software Engineering workshop. Distributing the data—which is 21GB—through a physical media, together with the term of use, is a relatively easy and inexpensive option to implement restrictions on the data, compared to other options such as a server with user accounts. In addition, given the huge size of the data, the physical media makes the data transfer faster and easier.

The term of use agreement is a one-year Non Disclosure Agreement. The agreement includes prohibition on unauthorized copying and some notion of fair use. Renewal of the agreement may be considered. We further discuss the notion of fair use in Section 3.2.

3.2 Private Information -- Anonymization

Anonymization of the data makes publishing privacy-violating results tougher. We replace user information—names, user ids, and email addresses—with generated strings, *User1, ..., UserN*, to replace user names of the first person to the *N*th person. We deliberately did not eliminate all the information about individuals affiliated with Jazz, for two reasons. First, complete anonymization of the data is nearly impossible. Names of people often appear in Work Item discussions. A message such as "Hi

¹² <http://www.ibm.com/software/awdtools/clearcase>

¹³ <http://www.ibm.com/software/awdtools/clearquest>

¹⁴ <http://promisedata.org/>

¹⁵ <http://msr.uwaterloo.ca/msr2008/challenge/>

Paul, Please fix this bug." in the Work Item discussion would reveal the first name of the developer. It would be extremely tedious to filter out all such information manually, or a hard computational problem to automatically filter out such information. Second, the Work Items from Jazz development are public on the Jazz community website. Thus, researchers can figure out the mapping of the ID's to real names. However, this attempt would violate the term of use, and with publishing their results with the names of Jazz affiliated people.

How does Jazz infrastructure help and impede the pruning of confidential and private information? We had to do two major steps for this anonymization, one supported by the Jazz API, which was relatively easy, and one was not. In the first task, we used the Jazz API to change the user information through the Jazz IContributor interface. The second task, the more tricky one, is to clean up the snapshots of the contributor information in the data warehouse, a facility that backups the data in the Jazz repository periodically and automatically. In this task, we had to manipulate the database using SQL update statements (more specially, on the COMMON_SNAPSHOTS.CONTRIBUTOR table). The difficulty lies in that the schema of the database is not well documented and that the Jazz database is intended to be manipulated through the API and the Jazz user interface, but through direct manipulation of the database.

3.3 Common formats

We distribute the anonymized data in the standard Jazz repository export format. This achieves two aspects related to common formats. First, the data recipients can use the Jazz built-in repository import tool to populate the Jazz repository and the data to the database backend. Standard database, such as DB2, Oracle, and Derby (although not recommended for this huge amount of data) are among the databases supported by the import command. Second, once the data has been populated into the Jazz repository, the data recipients can view and analyze the data in two ways that are supported by Jazz: examine the repository using the Jazz clients (Eclipse-based and Web-based) and access the data programmatically from the Jazz API. Although the user can also directly query the database, this access method is not a use case that is explicitly supported by the Jazz platform.

3.4 Provenance

Provenance relates to confidence in the validity of the data, that is, the data represents what the researcher thinks it represents. Because the Jazz team has been very transparent in its work and made a lot of work public on the Jazz community web site, external researchers can be confident that the data really do represent Jazz development. This assumes that the researchers have confidence that our data preparation did not alter the data in any radical way. This paper is intended to provide some transparency to the initial work on data preparation.

4. CONCLUSION

In this paper, we describe the technical challenges we encountered preparing the Jazz development data for a small group of external researchers. Even though Jazz is designed for software engineering stakeholders, we find that Jazz provides great utilities for data providers to externalize data and for data consumers who analyze the data. Next steps include building a

community around this data. Fortunately, we already have the beginnings: a group of people has already formed a nascent community through shared research interests and who all have access to Jazz for their own exploration and development. Several tools would assist this community building activities, including tools and shared space to exchange information (e.g., supporting FAQs, Q&As, and discussions, sharing data, papers, and tools) and providing profiles to individuals in the community. We plan to use the Jazz platform to center some of these activities. For example, use Jazz Work Items and discuss threads; leverage the Jazz Web UI dashboard to create a page with points to resources stored in the Jazz repository; use the Jazz source control to create separate streams, components, and projects; and use the profile facility in Jazz for defining user profiles.

5. ACKNOWLEDGMENTS

We thank the Jazz team for the Jazz data, approving the data distribution and constructing the legal agreement. We also thank Daniela Damian, André van der Hoek, Anita Sarma, Adrian Schröter, Ben Waber, and Andreas Zeller for the discussion and help on this work. IBM, Jazz, and Rational are trademarks of IBM Corporation, in the United States, other countries or both.

6. REFERENCES

- [1] Alonso, O., Gertz, M., Devanbu, P., Database Techniques for the Analysis and Exploration of Software Repositories. In Int'l Workshop on Mining Software Repositories, 2004.
- [2] Atkins, D. E., Droegemeier, K. K., Feldman, S. I., Garcia-, Molina, H., Klein, M. L., & Messina, P. Revolutionizing science and engineering through cyberinfrastructure: Report of the national science foundation blue-ribbon advisory panel on cyberinfrastructure, 2003.
- [3] Bevan, J., Whitehead, Jr., J., Kim, S., and Godfrey, M. Facilitating Software Evolution with Kenyon. Proc. of ESEC/FSE'05.
- [4] Birnholtz, J. P. and Bietz, M. J. Data at work: Supporting sharing in science and engineering. Proc. of Group 2003.
- [5] Cataldo, M., Wagstrom, P., Herbsleb, J. and Carley, K. Identification of Coordination Requirements: Implications for the Design of Collaboration and Awareness Tools. Proc. of CSCW'06.
- [6] Conklin, M., Howison, J., and Crowston, K. 2005. Collaboration using OSSmole: a repository of FLOSS data and analyses. In Int'l Workshop on Mining Software Repositories, 2005.
- [7] Draheim, D. and Pekacki, L. Process-Centric Analytical Processing of Version Control Data, Proc. of Int'l Workshop on Principles of Software Evolution, 2003.
- [8] Ehrlich, K., Helander, M., Valetto, G., Davies, S., and Williams, C. An Analysis of Congruence Gaps and Their Effect on Distributed Software Development. In Int'l Workshop on Socio-Technical Congruence Workshop, 2008.
- [9] Fischer, M., Pinzger, M., and Gall, H. Populating a Release History Database from Version Control and Bug Tracking Systems, Proc. of International Conference on Software Maintenance, 2003.
- [10] German, D. Mining CVS repositories, the softChange experience. In Int'l Workshop on Mining Software Repositories, 2004.
- [11] Kruchten, P. The Rational Unified Process: an introduction, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, 1999
- [12] Kim, S., Zimmermann, T., Kim, M., Hassan, A., Mockus, A., Girba, T., Pinzger, M., Whitehead, Jr., E. J. and Zeller, A. TA-RE: An Exchange Language for Mining Software Repositories. In Int'l Workshop on Mining Software Repositories, 2006.
- [13] Louis, K. S., Jones, L. M., & Campbell, E. G. 2002. Sharing in science. American Scientist, 90(4). 304-30