INFORMATION SYSTEMS IN MANAGEMENT

Information Systems in Management (2017) Vol. 6 (3) 169−180

MINING SOFTWARE REPOSITORY FOR IMPROVEMENT OF IT PROJECT MANAGEMENT PROCESS

JAKUB GUZIUR [a)], ANETA PONISZEWSKA-MARAŃDA [b)]

[a)] *Institute of Informatics, Silesian University of Technology*
[b)] *Institute of Information Technology, Lodz University of Technology*

With the growing number of IT projects and their complexity their deeper analysis became necessary. For this purpose, the historical data on previously completed projects began to collect – it has enabled projects undergo a detailed analysis, which allowed the efficient implementation of projects and allowed in advance to detect the potential problems. Thus a new area of research, Mining Software Repository (MSR), was born. Research in the field of MSR brings very promising results. The increasing importance of research for gaining the tools to support the process of collecting and analyzing the data related to software development is observed. The data from version control systems, bug tracking systems and issue tracking systems is the most popular. This paper presents the existing MSR tools carries out its analysis and describes the weaknesses of each individual solutions.

Keywords: software mining, software analytics, MSR tools

1. Introduction

With the growing number of IT projects and their complexity their deeper analysis became necessary. Number of data collected by project management systems, bug tracking systems, issue tracking systems and similar, has become too big to analyze it them by team managers, project managers or architects. It became necessary to create tools that supported the decisions taken by the managers and

detect dependencies that are not visible at first glance. With the need to meet these needs a new field of research: Mining Software Repository (MSR) was established.

Mining Software Repository is a field of software engineering which focuses on retrieving software development related data and extract knowledge about the software from this data. Software development related data is widely understood. It can be a history of issue from issue tracking system or feature requirement document. All this data is collecting and the knowledge is extracting from it using statistical or data mining techniques.

The first stage of any MSR research is to select the project data, from which metrics about the IT project will be extracted. After selecting the data, it is necessary to download it. It can be done: manually, using a written script or automatically using MSR tool, if it supports the selected project data. The final step is to calculate the metrics that will evaluate the project and, if necessary, make the necessary amendments in the places indicated by the metrics. The calculation of metrics, like downloading project data can be done: using a written script or tool, if it supports selected metrics. Metrics can be presented in the form of a numerical value, a percentage or using one or multi-dimensional graph, depending on how the metric is complex.

MSR research is the source of many interesting information about IT projects [13], e.g. gives an answer to the question which problems the modern code review can fix [19], allow to determine the impact of code review coverage for software quality [18] and to understand how students develop the software [17]. Obtaining such information allows to detect the weaknesses and bottlenecks in the project, which elimination will help to improve the software quality and the management process of IT project.

This paper presents the MSR tools existing on a market, the information which kind of data these tools need and the role of these tools in improvement of IT project management process. Section 2 presents what kind of data is interest for MSR area researches and from which sources we can deliver it. Section 3 describes what information can be extracted from the project data and how to use this information to improve the management process of IT project. Section 4 deals with MSR tools, divided into four categories: on-demand data sets, tools to collect and analyze one type of source data, tools to collect and analyze multi type of source data, tools to support MSR tools development. Section 5 presents the architecture for MSR tools, which main goal is unified internal workflow of MSR tools to help the developers to join the developing MSR tools by unified internal structure of all MSR tools.

2. What data is interesting for MSR researches?

The projects data is the most important for MSR researches. Through the related data we mean all data generated during the IT project. The data the most frequently used by the MSR tools includes:

a) Source code (C#, Java, Python) – easily accessible type of project data, written in plain text. It can be used without having to use an external API or translators. Each IT project of software development includes this type of project data.
b) Version Control Systems (Git, SVN) – software for managing multiple versions of files and helps people work together with the same files [14]. In most cases, this software is used for managing the versions of source code, but can be used to hold any type of files, e.g. documentation or requirements.
c) Bug tracking systems (Bugzilla, Redmine) – software for storing and managing bugs found in the software. It allows to report the bugs, gives them priorities, changes their status and closed them after fixed by developer.
d) Issue tracking systems (JIRA, Asana) – computer software package that manages and maintains the lists of issues [21]. In IT projects by issue understand all the tasks performed during the IT project: implementation of functionality, reviewing the code, reviewing the documentation, preparation of test scenarios or preparation of equipment.

The above-described data sources are not the only ones that can be used. Data sources can include design documentation, timesheets or even salary of the employees. In fact, the researcher decides which data can show interesting properties.

The next section presents what information can be extracted from the project data and how to use this information to improve the management process of IT project.

3. MSR role in improvement of management process of IT projects

For each type of project data, the managers can extract a lot of information which can help them to improve the management process of IT projects. This section presents what information can be extracted from the project data and how managers can use it to improve the management process of IT project.

A lot of interesting information about the language in which the code is written and the person who wrote it can be extracted from the source code. Finding

the least used language statements can allow the creator of the language to verify that the language statement is needed in the language or it can be removed. One of example of such type research is paper where authors check how new features based on Java generics are introduced, championed or ignored [16]. Interesting information that can be extracted from the source code would be also the average number of lines of code per file, depending on its priority. It can be a correlation between the priority of task and the number of lines of code needed to implement it. If yes, it allows better to estimate the time needed for the task according to its priority. The average number of lines of code per file can be also compared with the number of bugs existing in the file. It may be possible that with a certain or less number of lines of code bugs occur rare, while exceeding this number of lines can cause a sharp increase of occurring the bugs.

Another data source from which we can be extract the information that will allow to improve the management process of IT project is version control systems. From the these systems, we can get for example the average number of changed lines of code needed for fixing the bug depending on its priority. This information allows project managers to predict the time needed to fix all existing bugs and if necessary warning about project delay. Information about number of written lines of code per developer correlated with the information about number of bugs assigned to each developer can give the answer for the question “If more line of codes means better code?” The average number of changed lines of code per single commit can give the information how big a single commit should be to decrease the chance of bug occur. The ten largest single commits can show potential malicious behavior. Maybe someone pushed false code. Impact analysis of change requests on source code based on interaction and commit histories [15] is very interesting information which can be extracted from version control systems.

From the issue tracking systems we can extract the average duration of issue depending on its priority. This average duration can be used to predict the timeframe for the next project. The number of open/in progress/closed issues/bugs in the selected time period gives project manager the information about possible delay especially if he can compare this with similar time period for the historical projects. Similarly, the growth of reported bugs in the given period of application development compared to the similar period for previous version can gives project manager the information that finishes the project on time is in danger.

The last data source discussed in this section is bug tracking systems. Information extracted from this data source can be interesting especially for testers. The number of bugs for each feature can allow the tester to better choose the features that may be more prone to bugs and focus on it during testing phase. The total number of bugs in the project by priority shows the maturity of the software. The further phase of software development, the software should contain less critical bugs. If not, then this is a warning for testers that perhaps software

should not be release. Mean time to fix the bug, depending on its priority, can be used by project manager to calculate time necessary to fixed all bugs. If this time is greater than the time remaining to release the software then it may be a warning for the project manager that he should increase the number of testers, delay release or decide which errors will be fixed after the release.

Data sources described in this section can provide the managers a lot of useful information that can significantly improve the management process of IT projects. The next section presents the tools which provide such type of information based on project data provided by the user.

4. Mining Software Repository tools

There are many applications supporting research in the field of Mining Software Repository. They differ in the type of supported data and offered features. This section presents different types of tools, their advantages and disadvantages, as well as examples of tools available on the market.

For researchers who do not have their own project data, and would like to test their hypothesis on real data, the good choice is *data sets* shared by other researchers. These data sets contain data from open source projects that can be downloaded to disk and then used to calculate metrics and confirm or reject the hypotheses. Using data sets allows bypassing "API restrictions" and starting right from the data analysis. Examples of data sets are: OpenHub [2], GHTorrent [6, 7, 8] and GitHub Archive [9].

GHTorrent is a project of scalable offline mirror of data offered through the Github REST API. It releases the data as downloadable archives. For each event from *Github public event time line* GHTorrent retrieves its content and stores: raw JSON response to a MongoDB database, while extracted structure in MySQL database. GHTorrent is a huge data set (around 4TB of compressed JSON data stores in MongoDB and more than 1.5 billion rows of extracted metadata in MySQL) contains data about all projects since 2012. It is an open source project which uses its own model for data storage [6, 7, 8].

Open Hub (formerly Ohloh.net) is a public directory of free and open source software. It offers search services for discovering and analytics methods to compare open source code and projects. OpenHub index over 21,000,000,000 lines of open source code. It provides the reports about composition and activity of project code by connecting to project source code repositories and analyze code's history. Currently it contains information about around 345 000 projects [2].

GitHub Archive is a web project which goal is recording GitHub timeline, archive and share it for further analysis. GitHub events are aggregated as hourly archives. GitHub Archive provides its resources in two ways: via HTTP request as an archive contains JSON encoded data or via public dataset on Google BigQuery

with access by arbitrary SQL-like queries. Using Google BigQuery is restricted by 1 TB of data processed per month free of charge [9].

To summarize, the data sets are a great solution for researchers who do not have their own data, and would like to start the MSR research. The summary of the advantages and disadvantages of data sets is as follows:

a) advantages:

- huge data sets: 345000 projects (OpenHub), 4TB of data (GHTorrent),
- on-demand data sets without API restrictions,
- no need to write a tool to retrieve the data – focus on data analysis,
- open source license: full (GitHub Archive, GHTorrent) or partly (OpenHub),

b) disadvantages:

- one type of project data: CVS (GitHub, e.g. GHTorrent, GitHubArchive; SVN, Git, Mercurial, e.g. OpenHub),
- predetermined data structure and the type of database: GitHub Archive, GHTorrent, OpenHub,
- metrics (need to develop by own (GitHub Archive), only offered by tool (OpenHub), both own and offered by too (GHTorrent).

If the researcher already has project data and would like to carry out MSR research on it, he can take advantage of *tools for collecting and analyzing project data*. These tools are the most complex tools that exist on the market. They allow collect different type of project data from multiple projects and using it calculates very complex metrics. Examples of tools for collecting and analyzing project data are: BuCo Reporter [12], Candoia [11], OSSMeter [5, 10], Bitergia [1] and SmartShark [11].

BuCo Reporter is an easy to use, extensible standalone application written in Java that analyses relevant project data such as commits, committers, source code and bugs, and provides useful reports about the project evolution history [13]. Data from project commits are using to calculate metrics for version control systems such as Historical Commit Distribution, Average Lines per Commit, Average Commits over a time period and many more. From bug reports BuCo calculate metrics such as Average Bug Correction Time and Rate of Unresolved Bugs. Generally BuCo Reporter enables software practitioners to easily combine information from both Version Control and Bug Tracking systems [12].

OSSMeter is a platform to support decision makers in the process of discovering and monitoring the health, quality and activity of open-source project. To achieve this it computes the quality indicators by performing analysis of information coming from different sources such as project metadata, source code repositories or bug tracking systems. OSSMeter is available under open source license

and support analyzing the existing OSS projects from platforms such as SourceForge, Google Code, GitHub, Mozilla and Apache [5, 10].

Candoia is a platform to support development of MSR programs. It is for MSR tools as Android for mobile applications. Researcher can build MSR programs, without worrying about access to project data such as bugs or source code. He can use also Candoia application store which allow download programs created by other researchers and share with others their own applications. Candoia offers ready to use abstractions for popular MSR artifacts e.g. VCS or bug. Researcher can perform all tasks via web interface [11].

Bitergia is open source software to analyze project data. The application through the web interface provides an overview about whole project ecosystem (e.g. code review and management, documentation processes). The biggest advantages of Bitergia are provided reports: KPIs, benchmarking between projects, quantitative studies, maturity level analysis, good\bad practices of cooperation during software development. Bitergia supports almost 30 types of project data [1].

SmartShark is a general-purpose platform for supporting MSR research. It supports a variety of data. Architecture of SmartShark allows the researcher to reduce the tasks to two tasks: select a project from which the data will be analyzed, write analysis program. This allows the researcher to focus on the analysis. ETL process is responsible for preparing the data for the analysis. The result of its action is saved to MongoDB. Analysis program must base on Apache Spark. Programs are executed on the Hadoop cluster, where they can access the MongoDB. The researcher communicates with the platform via this web based interface [11].

To summarize, the tools for collecting and analyzing project data are very complex. They allow researchers to extract the knowledge from project data and focus only on analyses of data, not on infrastructure problems. The summary of the advantages and disadvantages of this type of tools is as follows:

a) advantages:
   - support various types of project data: BuCo Reporter supports SVN, Jira, Bugzilla; OSSMeter supports SVN, Git, Bugzilla, BitBucket, Jira, Redmine, SourceForge; Bitergia supports almost 30 types of project data; Candoia supports GitHub, Jira, Bugzilla,
   - open source license: BuCo Reporter, OSSMeter, Candoia, Bitergia,
   - ability to define the data format: yes for OSSMeter, BuCoReporter and no for Candoia, Bitergia,

b) disadvantages:
   - metrics only offered by tool but the huge number of them: BuCo Reporter, OSSMeter, Candoia, Bitergia,
   - ability to use data from multiple projects: no for Candoia, BuCo Reporter and yes for OSSMeter, Bitergia.

Sometimes researchers do not want or cannot use the tools available on the market. Then one way to solve such problem is to create own tool. Tools supporting MSR tools development can help in creation of own tools. These tools contain special data structures for the most popular project data types as well as methods to access and write the project data which significantly shorts the time required to create the own tool. Examples of tools to support MSR tools development are Boa [3, 4] and TA-RE [20].

Boa is a domain specific language and infrastructure which goal is to make ease the testing MSR hypothesis [3, 4]. Boa reduces programming efforts providing several domain-specific types for mining software repositories. Each type provides several attributes that can be thought of as read-only fields and improves the scalability of programs by running it on Boa infrastructure which use Map Reduce and manages the details of downloading the projects data. It offers the web-based interface for submitting programs, compiling them, running in the cluster and providing an output from these programs [3, 4].

TA-RE corpus consists of an exchange language capable of making the sharing and reusing data as simple as possible and extracts data from software repositories that will allow the researchers to reproduce and benchmark their experiments [20]. Main goal of this solution is to encourage the researchers to share their extracted data via TA-RE repository. TA-RE anonymizes the extracted data by separate source code and the description of changes in the exchange language [20].

To summarize, the tools supporting MSR tools development may help the researchers to develop their own tools and allow the researchers focusing on the business logic without worry about technical details. The summary of the advantages and disadvantages of this type of tools is as follows:

a) advantages
- making sharing and reusing data simple (Boa, TA-RE),
- easy to learn – high level of abstraction (Boa, TA-RE),
- scalable (BOA),

b) disadvantages
- need to write MSR tool by own (Boa, TA-RE),
- support only for source code (BOA).

The next section describes the unified architecture for MSR tools which we use for development of MSR tools.

5. Unified architecture for MSR tools

The main reason, which was behind the creation of unified architecture for MSR tools, is to standardize the operation of MSR tools. Currently, the existing tools often offer the same functionality, but their architectures and workflows are

very different. This causes the difficulties in learning new tools, as well as the need to explore every new MSR tool. Unified architecture lets the researchers easily learn new MSR tools and focus as more on functionality than on understanding tool architecture, as MVC pattern makes easy learn new tool based on MVC, if someone previously using another tool based on MVC. The architecture does not impose a programming language and way of implementation of components. It describes only the flow of data in the system and the role of system components (Fig. 2).

Our architecture consists of four main modules and databases in which data and calculated metrics are stored. Researcher in the sub-module *Mapper* from *Core* module defines how to save the data from data source to the database table. Figure 1 shows an example of sub-module *Mapper* definition, where the researcher has defined that the field *created* from *Jira* issue will be saved to the table *bug* as a column *created*.

```
"Jira.issue.renderedFields.priority.name": "bug.priority_name",
"Jira.issue.renderedFields.created": "bug.created",
"Jira.issue.renderedFields.resolution": "bug.resolution",
```

Figure 1. Definitions from *mapper* sub-module

Architecture involves the use of Object-relational mapping (ORM) technique to communicate with the database. For this reason, it is necessary to define the data models that will be used by ORM. The definitions of data models are made in the module *Data Models*. The data models used to store project data are located in the sub-module *Data sources models*, while the data models for calculated metrics are located in the sub-module *Metrics Models*. It provides the transparency and logical separation of two different types of models.

Sub-module *Validator* is used to check the compatibility between the definitions defined in module *Data models* and definitions defined in sub-module *Mapper* from *Core* module. In case of detection of non-compliance definitions, the sub-module *Validator* will suspend the application until the correction of non-compliance.

To download the project data from the sources the sub-modules are included in module *Data Crawlers*. It includes the sub-modules, each responsible for obtaining data from a single source (e.g. Jira, GitLab or Redmine). Sub-modules use the data contained in the sub-module *Configuration* from the module *Core* to authenticate with the API and to determine which data is to be collected. Data collected by the sub-modules of *Data Crawlers* module is sent to the sub-module *Translator* from the module *Core*. Sub-module *Translator* converts the data according to the definitions in sub-module *Mapper* and then sends it to the sub-module *ORM* that saves the converted data into the database as defined in sub-module *Mapper*.

The sub-module *ORM* obtains from sub-module *Configuration* the data needed to connect to the database.

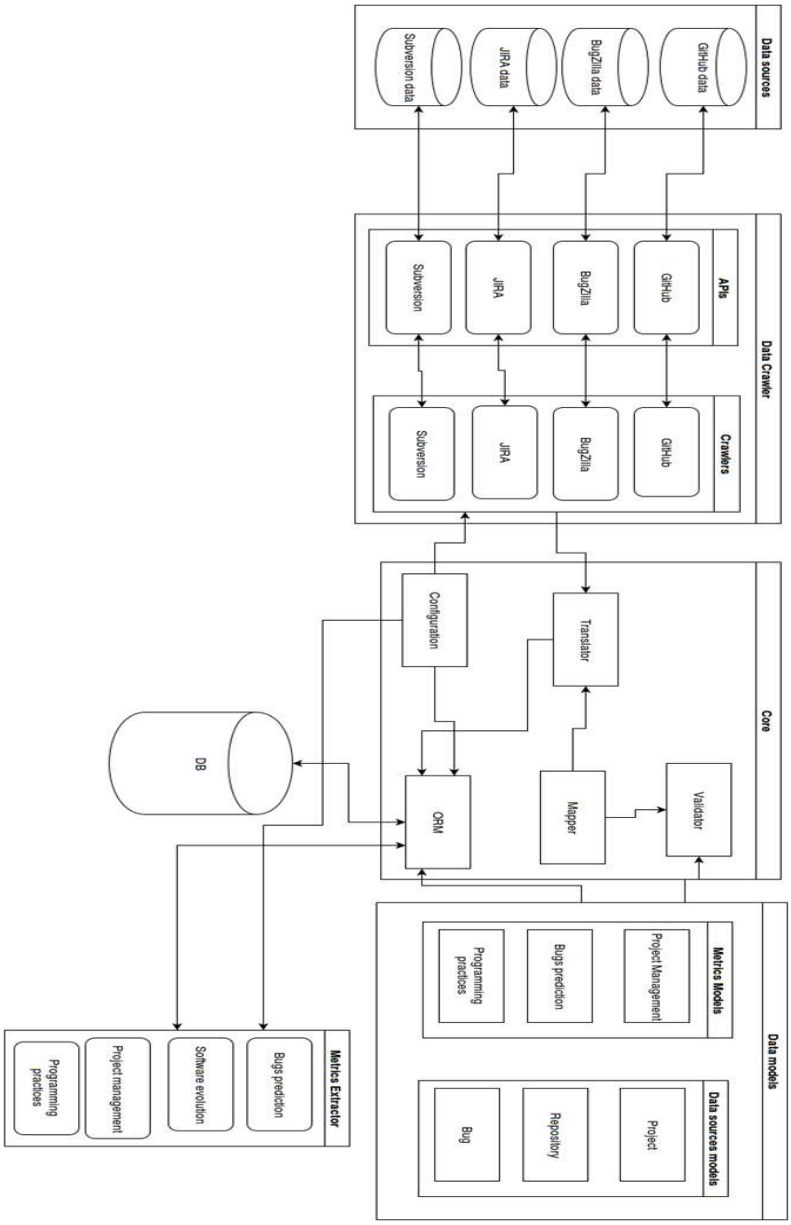

Figure 2. Definitions from *mapper* sub-module

The sub-modules located in module *Metrics Extractor* provide calculation of metrics. They use sub-module *ORM* from the module *Core* to load the data and then calculate the metrics. Each type of metrics is logically one sub-module which ensures the transparency and makes easier to find the interesting metrics.

The discussed architecture has many advantages that allow to create complex MSR tools: ability to choose which data should be saved, ability to define the format of recorded data, specialized modules and sub-modules with a clearly defined scope of activities.

6. Conclusion

Today IT projects are very complex and have many non-obvious connections between the individual elements. Managers find increasingly difficult to make project decisions based only on their knowledge and experience. MSR tools can assist in the effective management of IT projects. These tools allow the analyzing thousands of data to detect non-obvious dependences and predict the occurrence of events in the project, such as delays in the project or the number of reported bugs in a week.

There are many very different tools that can be used depending on the needs. Tools presented in the paper can be used to start the MSR researches. The researcher can use existing data sets to test his predictions on real data. If the researcher uses only one type of data, e.g. source code, he can take the advantage of MSR tools that work with one type of data. If he wants to use multiple sources of data he can use MSR tools which support multiple data types. There are also available on the market the frameworks to accelerate the development of MSR tools.

The unified architecture can be an attempt to standardize the development of MSR tools. The use of a common architecture for all creators of tools will allow their users to easily learn the new tools and researchers who would like to develop the existing tools used for the development team.

REFERENCES

[1] Bitergia, http://bitergia.com/, 2016.

[2] Black Duck Software Inc., Open HUB, https://www.openhub.net/, 2016.

[3] R. Dyer, H. A. Nguyen, H. Rajan, and T. Nguyen, *Boa: Ultra-Large-Scale Software Repository and Source-Code Mining*, ACM Transactions on Software Engineering and Methodology, 2015.

[4] R. Dyer, H. A. Nguyen, H. Rajan, and T. N. Nguyen, *Boa: A language and infrastructure for analyzing ultra-large-scale software repositories*, In Proceedings of the IEEE/ACM 35th International Conference on Software Engineering (ICSE), 2013.

[5] D. Di Ruscio, D. S. Kolovos, I. Korkontzelos, N. Matragkas, and J. Vinju, *Ossmeter: A software measurement platform for automatically analyzing open source software projects*, In ESEC/FSE 2015 Tool Demonstrations Track, 2015.

[6] Gousios, Georgios, and Diomidis Spinellis, *GHTorrent: GitHub's data from a firehose*, 9th IEEE Conference on. Mining software repositories (MSR), 2012.

[7] Gousios, Georgios, et al. *Lean GHTorrent: GitHub data on demand*, Proceedings of the 11th working conference on mining software repositories, ACM, 2014.

[8] G. Gousios, *The GHTorrent dataset and tool suite*, MSR '13, pages 233–236, 2013.

[9] I. Grigorik, GitHub Archive, https://www.githubarchive.org/, 2016.

[10] Tiwari, Nitin M., et al., *Candoia: A Platform and an Ecosystem for Building and Deploying Versatile Mining Software Repositories Tools*, 2015.

[11] Trautsch, Fabian, et al., *Adressing problems with external validity of repository mining studies through a smart data platform*, Proceedings of the 13th International Workshop on Mining Software Repositories. ACM, 2016.

[12] Ligu, Elvis, Theodoros Chaikalis, and Alexander Chatzigeorgiou, *BuCo Reporter: Mining Software and Bug Repositories*, BCI (Local), 2013.

[13] Hemmati, Hadi, et al., *The msr cookbook: Mining a decade of research*, Mining Software Repositories (MSR), 2013 10th IEEE Working Conference on. IEEE, 2013.

[14] Rahman, Mohammad Masudur, and Chanchal K. Roy, *An insight into the pull requests of github*, Proceedings of the 11th Working Conference on Mining Software Repositories. ACM, 2014.

[15] Zanjani, Motahareh Bahrami, George Swartzendruber, and Huzefa Kagdi., *Impact analysis of change requests on source code based on interaction and commit histories*, Proceedings of 11th Conference on Mining Software Repositories. ACM, 2014.

[16] Parnin, Chris, Christian Bird, and Emerson Murphy-Hill., *Java generics adoption: how new features are introduced, championed, or ignored*, Proceedings of 8th Working Conference on Mining Software Repositories. ACM, 2011.

[17] Liu, Ying, et al., *Using CVS historical information to understand how students develop software*, 1st international workshop on mining software repositories. 2004.

[18] McIntosh, Shane, et al., *The impact of code review coverage and code review participation on software quality: A case study of the qt, vtk, and itk projects*, Proceedings of the 11th Working Conference on Mining Software Repositories. ACM, 2014.

[19] Beller, Moritz, et al., *Modern code reviews in open-source projects: which problems do they fix?*, Proc. of 11th conference on mining software repositories. ACM, 2014.

[20] Kim, Sunghun, et al, *TA-RE: An exchange language for mining software repositories*, Proceedings of the 2006 international workshop on Mining software repositories. ACM, 2006.

[21] Bertram, Dane, *The social nature of issue tracking in software engineering*, Diss. University of Calgary, 2009.