Science of Computer Programming 153 (2018) 30–33

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

Original Software Publication

GITANA: A software project inspector

Valerio Cosentino [a,*], Javier Luis Cánovas Izquierdo [a], Jordi Cabot [a,b]

[a] UOC, Barcelona, Spain
[b] ICREA, Barcelona, Spain

ARTICLE INFO

Article history:
Received 29 May 2017
Received in revised form 29 November 2017
Accepted 3 December 2017
Available online 14 December 2017

Keywords:
Software project analysis
Software analysis
Mining software

ABSTRACT

The development of software projects entails significant implementation and collaboration activities, typically supported by tools such as issue trackers, code review tools and Version Control Systems. However, these tools only provide a partial view of the project and often lack of advanced querying mechanisms, thus hampering the analysis of the status of the project and endangering the decision making process on the best way to drive the development process. We present GITANA, a software project inspector able to import the activity of the different support tools into a single relational database, thus providing a central point to perform all kinds of cross-cutting analysis on the software project data.
Tool website: https://github.com/SOM-Research/gitana.

Software metadata

(executable) Software metadata description	
Current software version	1.0.1
Permanent link to executables of this version	https://github.com/ScienceofComputerProgramming/SCICO-D-17-00124
Legal Software License	MIT
Computing platform/Operating System	Python-compatible platform
Installation requirements & dependencies	The following python packages are required (as noted in `setup.py` file): networkx, mysql-connector-python-rf, gitpython, python-bugzilla, pygithub, selenium, py-stackexchange, beautifulsoup, slacker, pygal.
If available, link to user manual - if formally published include a reference to the publication in the reference list	http://gitanadocs.getforge.io
Support email for questions	valcos@bitergia.com

Code metadata

Code metadata description	
Current code version	1.0.1
Permanent link to code/repository used of this code version	https://github.com/ScienceofComputerProgramming/SCICO-D-17-00124
Legal Code License	MIT
Code versioning system used	Git
Software code languages, tools, and services used	Python
Compilation requirements, operating environments & dependencies	Python-compatible system, MySQL database
If available Link to developer documentation/manual	http://gitanadocs.getforge.io
Support email for questions	valcos@bitergia.com

* Corresponding author.
E-mail address: valcos@bitergia.com (V. Cosentino).

https://doi.org/10.1016/j.scico.2017.12.002
0167-6423/

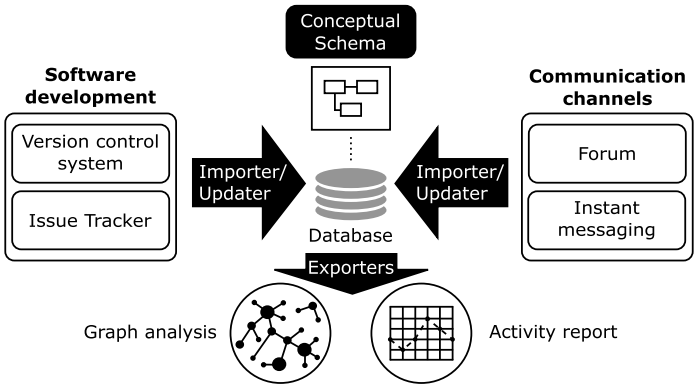

Fig. 1. Our approach.

1. Introduction

Software development processes are supported by a plethora of tools. Each tool helping to manage the complexity of a specific aspect of the project: source code is tracked by using Version Control Systems (VCSs) such as Git, development tasks are reported in issue trackers, and collaboration and coordination activities are normally developed in forums, mailing lists or chats. While all these tools store relevant information, they are actually silos that provide partial views of the project and lack of advanced query and data integration mechanisms. Thus, project managers often struggle to understand how projects perform, a must to make knowledgeable decisions about the development process.

In this paper we present GITANA, a project inspector that analyzes the support tools used in software projects and imports the information in a relational database, thus providing a central point to perform all kinds of cross-cutting analysis on project data. The current version of the tool provides support to inspect Git repositories, Bugzilla/GitHub issue trackers, Eclipse forums and Slack instant messages. To ensure efficiency, GITANA comes with an incremental propagation mechanism that refreshes the database content with the latest modifications available on the data sources. The approach also incorporates exporters to enable further data analysis with third-party tools.

2. Problems and background

Nowadays, software development activities are independently tracked and recorded in different tools. These tools have limited support for querying their internal data and typically rely on scripts (e.g., git commands) or APIs (e.g., the Bugzilla API) to do so. As a result any advanced analysis of the data quickly becomes a hard and complex task. Especially for perform any data integration or combination operation involving more than one tool, as it is typically the case. As consequence, comparison of data from different projects is also challenging.

This situation limits the scope of existing research studies and tools devoted to understand (and improve) software development [1]. For instance, GHTorrent [2] is a dataset only devoted to analyze GitHub repositories, the work presented by Kahani et al. [3] target the analysis of Eclipse forums and Wang et al. [4] study the context of StackOverflow. Examples of more general tools are GrimoireLab[1] and Kibble.[2] They are similar to Gitana in their goals though they target a different user profile. In short, Gitana is the only one providing SQL-based access to the data which is a plus for users not familiar with Python libraries for data manipulation (e.g., Panda) and NoSQL storage engines (e.g., ElasticSearch) as required in those other tools. Therefore, we believe Gitana is especially useful for users (like, for instance, project managers) that want an easy way to explore the project data (and compare data across projects) by easily writing and running adhoc queries.

3. Software framework and architecture

We propose a unified access to the information made available by all support tools. Fig. 1 depicts our approach. From a global conceptual schema for project-related concepts, we derived a relational database, which is then populated from a variety of partial sources through an incremental update mechanism. The database can be manually explored using standard SQL and/or used as the input for different kinds of deeper analysis processes such as complex network analysis and OLAP multidimensional analysis. GITANA architecture is divided into four main components (black-filled elements in Fig. 1).

The conceptual schema, which models the project activity. This schema is an extension of [5]. Full details are available here.[3]

[1] http://grimoirelab.github.io/.
[2] https://kibble.apache.org/.
[3] http://gitanadocs.getforge.io/conceptual.