VisMinerService – A REST Web Service for Source Mining

Luis Paulo da Silva Carvalho[1,2], Renato Novais[1,3], Manoel Gomes de Mendonça Neto[2]

[1]Federal Institute of Bahia (IFBA)

[2]Federal University of Bahia (UFBA)

[3]Centro de Projeto Fraunhofer para Engenharia de Software e Sistemas at UFBA

{luiscarvalho, renato}@ifba.edu.br, manoel.mendonca@ufba.br

Abstract. *Enabling applications to perform source-mining and software metrics visualization can be a time-consuming task. With this in mind, we are positive that Web Services are fitting candidates to automate the developing of such applications. Therefore, this paper presents VisMinerService, a REST Web Service implementation built upon a static software component, VisMiner. It is described how the service can be used in order to mine metrics and other source-code related information. It is also discussed its integration with three different types of client applications (mobile, desktop/swing and web application) in order to provide a proof of concept of how VisMinerService can be used to create platform-independent software to visualize the mined information.*

1. Introduction

Manual collecting of metrics is an unreliable activity, since "too many errors or missing data badly affect the analysis process" [Sillitti et al. 2003]. Thus, providing ways to automate the execution of source-mining is an important activity. Plus to that, once the source-code is mined, it is necessary to portray it in a way that improves the understanding of the information, so to later aid software engineers to carry development tasks (e.g. maintenance, refactoring, and reverse engineering).

In order to better assist source-mining and visualization tasks, we consider that is important to fulfill the following objectives: (a) smoothing the effort to retrieve source code and metrics information in a way that one could easily build applications to make use of it (e.g. to either analyze or visualize the data); (b) automating the outsource of the gathered information in a well defined way, so that it becomes easier to parse and visualize the mined information; (c) grating the outsourcing in a distributed way (e.g. as with the use of Web Services), one could easily manage to create client software to reuse the routines to mine source-codes and metrics remotely.

Web Services have become a standard for sharing data between software applications over the internet. They enable the creation of technology-neutral, open, decentralized, reliable software solutions [Pautasso 2014]. Web Services provide distributed functionalities, which are independent of hardware platform, operating system and programming language. An example of Web Services implementation, which has increasingly gained acceptance due to its simpler style to use, is REST Web

Services [Al-Zoubi and Wainer 2009]. REST[1] is a Web Services' architectural style that constrains the interface of the HTTP protocol to a set of standard operations (GET, POST, PUT, and DELETE).

REST is a fitting candidate to implement service-oriented applications with flexibility and lower overhead [Hamad et al 2010]. In the same way, it can be used on the context of source-mining, metrics calculation, and visualization. However, either few works have explored REST to supply access to metrics using visualization or, if any, the potentials of using REST services in such field of application have not yet been experimented to its fullest.

VisMiner is a core library that comprises a set of functionalities dedicated to the extraction of information from source code files [Mendes et al. 2015]. It enables third-party applications to mine metrics from on-line Version Control Systems (VCS) as, for example, GitHub. So far, VisMiner has comprised the following set of metrics: Number of Lines of Code (LOC), Ciclomatic Complexity (CC), Number of Packages (NOP), and Number of Methods (NOM).

This work describes an approach by which the VisMiner Library is used as a component to create a REST Web Service, called VisMinerService. The goal is to externalize Visminer's functionalities across the web to enable the development of distributed service-oriented applications and toolsets to mine and visualize software data. We have also conducted a preliminary test in the development of multiple clients. The goal is to show the advantages of a service-oriented architecture for source-mining, metrics calculation, and visualization.

The rest of the paper is organized as follows: Section 2 discusses related work. Section 3 presents the VisMinerService. Section 4 exemplifies the use of VisMinerService. Conclusions and future works are discussed in Section 5.

2. Related work

The use of Web Service-based architectures to mine and visualize metrics obtained from software repositories is not new. Sillitti et al (2003) investigated the application of XML and SOAP (Simple Object Access Protocol) to expose functionalities in order to aid the extraction of metrics across HTTP-enabled networks. SOAP also makes possible the creation of Web Services. However, as previously stated, REST has later become a more adopted standard.

Sakamoto et al (2013) proposed the MetricWebAPI, which wraps mechanisms within a service-oriented framework to mine metrics. It also externalizes data to client applications via XML documents. However, it is not provided information about how such client applications can reuse the functionalities of MetricWebAPI. Protocols and document formats to base the interactions between the service and client applications are not covered.

Several other projects are intended to mine source-codes, but they either mention little (or no) remarks on how to be reused by client applications or there is no evidence that they were design for reuse at all. In this category we could mention: Metrics

1 Short form of REpresentational State Transfer

Grimoire [METRICSGRIMOIRE, 2015], GHTorrent project [GHTORRENT, 2015], SourceMiner [SOURCEMINER, 2015].

3. VisMinerService

VisMinerService is a REST Web Service built on the top of VisMiner core library. Its main purpose is to offer functionalities to remote distributed client applications, enabling the sharing, processing and visualization of software's mined data. Figure 1 depicts the targeted usage scenario for VisMinerService.

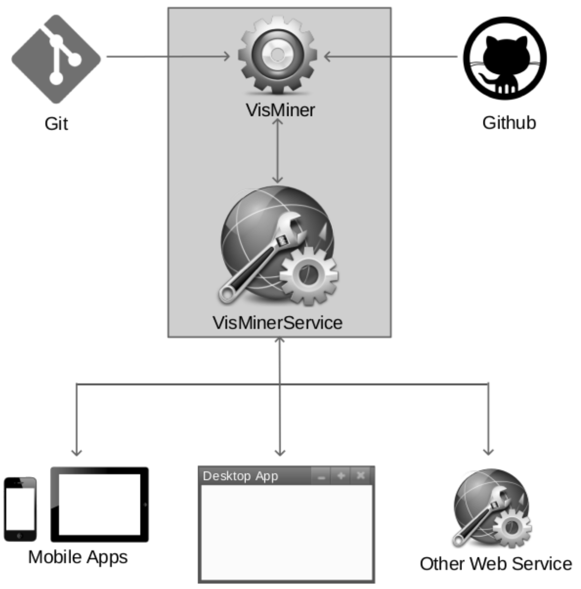


Figure 1. VisMinerService's use scenario

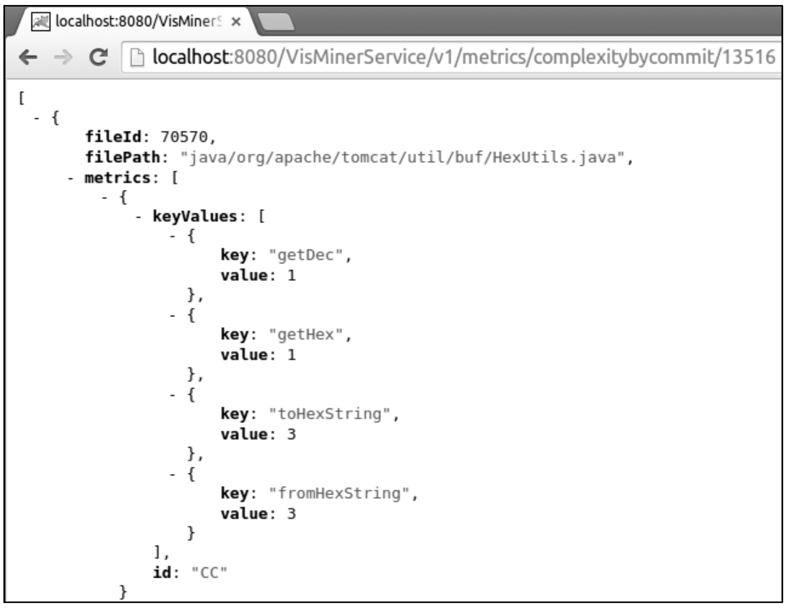
We have embedded VisMinerService with a set of URLs, each one triggering specific retrievals from the mined information. Table 1 lists examples of VisMinerService's externalized functionalities. The domain was omitted from the URLs, because it may vary depending on how the service's server is configured.

Table 1. VisMinerService's current URLs set

URL	Purpose
http://...projects/byid	It retrieves information about a project identified by its ID
http://...committers/byproject	It returns all committers from a given project
http://...commit/byproject	It retrieves information about all commits of a given project
http://...commits/bycommitter	It retrieves commits of a specific committer
http://...commits/numberofbycommitter	It returns the number of commits of a specific committer
http://...commits/bysha	It retrieves a commit identified by its SHA[2]
http://...file/bycommit	It fetches files affected by a specific commit
http://...metrics/byfile	It retrieves all metrics of an identified file
http://...metrics/bycommit	It returns metrics pertaining to a file modified by a commit

2 Secure Hash Algorithm is used by GitHub to identify commits

In the example shown in Figure 2, the “Complexity By Commit” URL filters data out after being parameterized by a commit's identification key. The commit was persisted in VisMiner's database at the moment that project's data (from TOMCAT project) was crawled from its GitHub repository. The retrieved JSON document represents a collection of key-value pairs. In the specific case of the Ciclomatic Complexity (CC) metric, key-value pairs are used to associate each file's methods to their respective complexities (e.g. the method “fromHexString” has summed up a CC's value of 3). Same as manually done on a web browser, computational agents (i.e. client software) can access the VisMinerService’s URLs programmatically. By obtaining and parsing the JSON document, one can uses the information about the metric (and the associated file) for any specific desired purpose.



localhost:8080/VisMinerService/v1/metrics/complexitybycommit/13516

```
[
  - {
        fileId: 70570,
        filePath: "java/org/apache/tomcat/util/buf/HexUtils.java",
      - metrics: [
          - {
              - keyValues: [
                  - {
                        key: "getDec",
                        value: 1
                    },
                  - {
                        key: "getHex",
                        value: 1
                    },
                  - {
                        key: "toHexString",
                        value: 3
                    },
                  - {
                        key: "fromHexString",
                        value: 3
                    }
                ],
                id: "CC"
            }
```

Figure 2. JSON representation of Ciclomatic Complexity (CC)

Client applications (e.g. Mobile Apps, Desktop Applications, and other Web Services) can reach VisMinerService via a set of URLs (contained in Table 1) in order to gain access to its remote functionalities. In this case, any software capable of navigating over the HTTP can be a client. Figure 2, for instance, shows a web browser (Google Chrome Web Browser) receiving and displaying a JSON [JSON, 2015] document from the *http://...metrics/complexitybycommit* URL. This URL is an entry-point to the service's functionality that retrieves information about metrics related to files affected by a particular commit.

Section 4 presents three examples of service-oriented applications. The purpose is to illustrate VisMinerService as a permissive tool that enables the development of remote client software. The use of the aforementioned URLs (Table 1) is also exemplified.

4. VisMinerService's multiple-client scenarios of use

In this section we show how different client applications can use VisMinerService. The purpose is: (i) to reinforce that VisMinerService is platform-independent; and (ii) to show that client software can perform a multitude of processing on the data obtained from VisMinerService.

4.1. VisMinerDroid

VisMinerDroid is an ANDROID application that makes use of VisMinerService to present data visualizations to mobile users. This example comprises two screen fragments (Figure 3):

- Committers list (left side of Figure 3) → it represents a list containing all committers from a project retrieved by the mobile application after navigating to the *http://...committers/byproject* URL;
- Commits per Committer (right side of Figure 3) → it plots a pie chart to display the percentage of commits sent by committers selected from the list. The mobile application browses the *http://...commits/numberofbycommitter* URL to obtain the total number of commits authored by each committer. The resulting dataset is rendered out to a chart. ANDROIDPLOT API [AndroidPlot, 2015] was used to draw the charts.

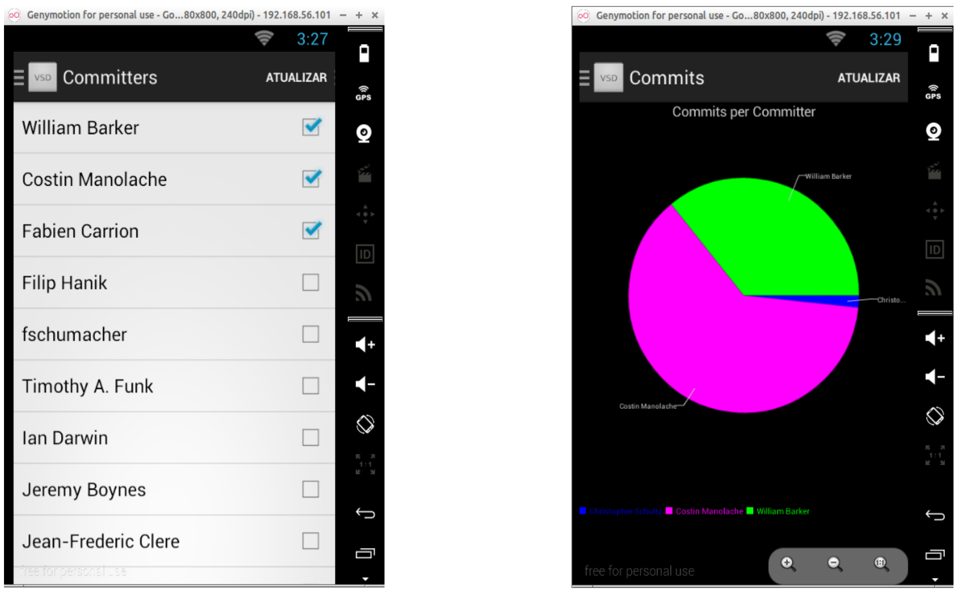


Figure 3. VisMinerDroid

VisMinerDroid is an example of how project-related information can be displayed by android applications. The advantage is to follow the global tendency of software becoming resident in mobile devices, without requiring that such applications retains all the data it might need in local databases. The information can be retrieved remotely from VisMinerService as soon as it becomes necessary.

4.2. VisMinerSwing

VisMinerSwing is a JAVAX Swing Application intended to show software-related data to desktop users. In combination with PREFUSE [PREFUSE, 2015] it shows an interactive tree, which enables the navigation through the data received from VisMinerService. Figure 4 shows a partial view of VisMinerSwing.

VisMinerSwing navigates to the *http://...committers/byproject* URL of VisMinerService to retrieve all committers from the project. After the user chooses one committer, VisMinerService is contacted again via the *http://...commits/bycommitter* URL to specifically select the commits belonging to the committer. The commits are then distributed across the tree through a stratification of temporal subsections (commits per year, month, day, time of the day) as shown on the left side of Figure 4. The right side of Figure 4 shows the files modified by a particular commit and the corresponding accumulated values of the metrics.

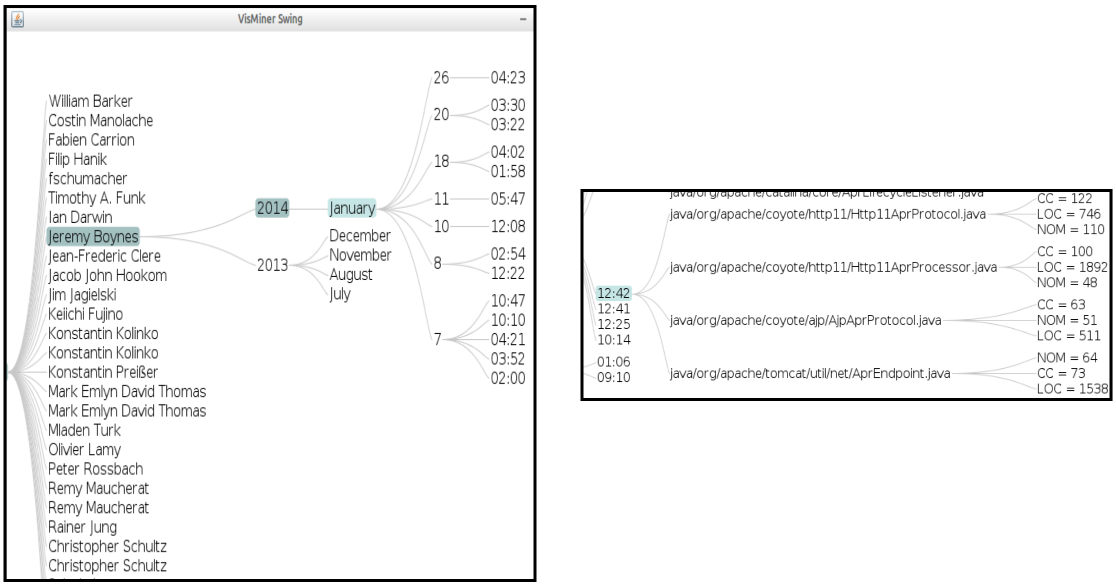


Figure 4. VisMinerSwing's commits-related information

VisMinerSwing and VisMinerDroid share the *http://...committers/byproject* URL. This is an example of how different types of application can use VisMinerService to exhibit the same information, i.e. desktop and mobile users may get access to the same information through a varied combination of visualizations. Therefore, it is possible to evidence that VisMinerService contributes to the fulfillment of sharing the use of project-related information in a platform-independent manner.

4.3. VisMinerWebViewer

The third example of VisMinerService client is a web application that enables the visualization of commits-related data via web browsers. Figure 5 presents VisMinerWebViewer. VisMinerWebViewer calls the *http://...commits/bycommitter* URL to capture commits from a chosen committer. In the example (left side of Figure 5), a timeline visualization from VIS.JS [VISJS, 2015] distributes the commits over a period of time. Users can navigate on the timeline and select a commit. After selecting a commit, another visualization based on GOOGLE CHARTS API [GCHARTS, 2015] is shown. Right side of Figure 5 contains a gauge-like visualization that exhibits metrics obtained from files affected by the commit. The set of files is extracted from

VisMinerService's response to navigations to the *http://...metrics/bycommit* URL.

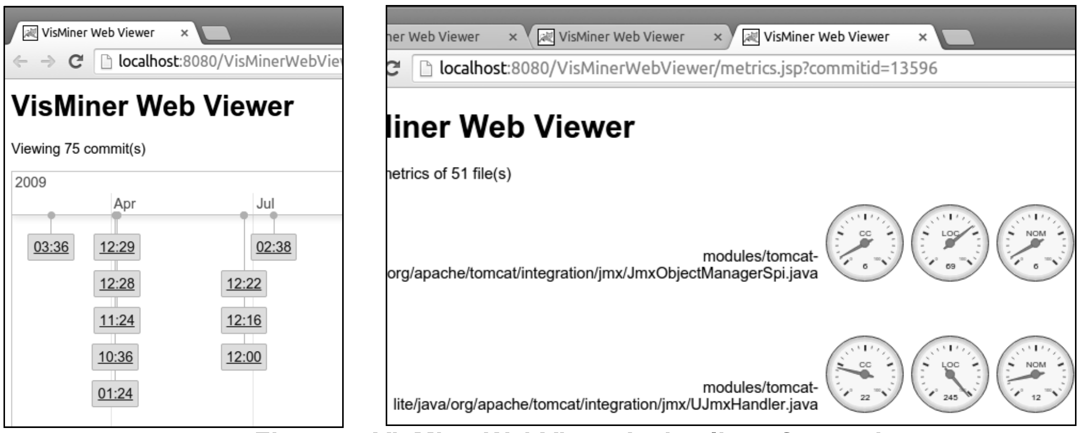

VisMiner Web Viewer
localhost:8080/VisMinerWebVie
VisMiner Web Viewer
Viewing 75 commit(s)
2009
Apr
Jul
03:36
12:29
12:28
11:24
10:36
01:24
02:38
12:22
12:16
12:00
localhost:8080/VisMinerWebViewer/metrics.jsp?commitid=13596
iner Web Viewer
etrics of 51 file(s)
modules/tomcat-
org/apache/tomcat/integration/jmx/JmxObjectManagerSpi.java
modules/tomcat-
lite/java/org/apache/tomcat/integration/jmx/UJmxHandler.java
CC
LOC
NOM

Figure 5. VisMinerWebViewer's timeline of commits

5. Conclusion and future work

In this paper, we have presented VisMinerService, a REST Web Service that intends to decentralize the extraction and visualization of source-code information. The service is capable of outputting information in a reusable and platform-independent way. We were able to prove the concept behind VisMinerService by creating different types of software, which were fully capable of showing the information gathered from the service. Such approach is highly recommended, provided that there is global tendency of information becoming pervasive [Resmini and Rosati 2011]. In this sense, source-code data (e.g. software metrics) might have to find their way through a plethora of platforms and visualization solutions while requiring either little or no adaptation at all. A service-oriented architecture is a powerful candidate to enable this.

In spite of having found strong evidences with regard to the aforesaid advantages, a more elaborated study must be conducted, because it is necessary to precisely measure the gain VisMinerService grants to the development of applications that share and visualize information mined from repositories. The experiment described in [Kaneshima et al. 2013] might be applicable in this case.

Furthering on the development of VisMinerService, we must expand the options of mined information (e.g. by adding metrics). As a consequence, more JSON documents and URLs must be created to allow new outputs. Efforts must also be made in order to automate the mining of software projects from other Version Control Systems. Ideally, VisMiner should not be limited to Git/Github's only. We must also keep the service up to date with the latest Visminer's additions: (i) inclusion of new metrics (e.g. WMC or Weighted Method Count); (ii) processing of varied targeted languages (C++ and JAVA projects) and (iii) new visualizations to externalize the new types of data that comes with such new additions.

It is also necessary to furnish end clients with visualization mechanisms; which can be achieved by increasing VisMinerService's URLs with functionalities to automate the adoption of visualization APIs. The purpose is to reduce the effort required to create graphical interfaces to show the mined data.

The integration of VisMinerService and the client applications exemplified in this work is also shown in the captured video: https://youtu.be/MOsNp45fS7c.

References

Al-Zoubi, K., & Wainer, G. (2009, June). “Using REST web-services architecture for distributed simulation”. In Proceedings of the 2009 ACM/IEEE/SCS 23rd Workshop on Principles of Advanced and Distributed Simulation (pp. 114-121). IEEE Computer Society.

GCHARTS (2015), https://developers.google.com/chart/. Acessed 2015, May.

GHTORRENT (2015), http://ghtorrent.org/. Acessed 2015, May.

Hamad, H., Saad, M. and Abed, R. (2009) “Performance Evaluation of RESTful Web Services for Mobile Devices”, In: International Arab Journal of e-Technology, Vol. 1, No. 3, January 2010.

JSON (2015), http://www.json.org/. Acessed 2015, May.

Kaneshima, Eliana, Maria Cristina Neves de Oliveira, and Rosana Teresinha Vaccare Braga. "Avaliação da Integração de Aplicações Empresariais usando Web Services: um Estudo Empírico". In X Workshop de Manutenção de Software Moderna, 2013, Salvador - BA. Anais do WMSWM 2013, 2013. v. 11. p. 1-8.

Mendes, T. S.; Almeida, D.; Alves, N.S.R; Spínola, R.O. ; Novais, R.L. ; Mendonça, M. “VisMinerTD – An Open Source Tool to Support the Monitoring of the Technical Debt Evolution using Software Visualization”. In: 17th International Conference on Enterprise Information Systems (ICEIS), 2015, Barcelona.

MetricsGrimoire (2015). http://metricsgrimoire.github.io/. Acessed 2015, May.

Pautasso, Cesare. "RESTful web services: principles, patterns, emerging technologies." Web Services Foundations. Springer New York, 2014. 31-51.

PREFUSE (2015), http://prefuse.org/. Acessed 2015, May.

Resmini, Andrea, and Rosati, Luca. “Pervasive Information Architecture: Design Cross-Channel User Experiences”. In IEEE Transactions on Professional Communication, Vol. 54, No. 4. Pp 408-409. IEEE, 2011.

Sakamoto, Yasutaka, Shinsuke Matsumoto, Sachio Saiki, and Masahide Nakamura. "Visualizing Software Metrics with Service-Oriented Mining Software Repository for Reviewing Personal Process." In Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2013 14th ACIS International Conference on, pp. 549-554. IEEE, 2013.

Sillitti, A.; Janes, A.; Succi, G.; Vernazza, T. “Collecting, integrating and analyzing software metrics and personal software process data”. Euromicro Conference, 2003. Proceedings. 29th, vol., no., pp.336,342, 1-6 Sept. 2003.

SOURCEMINER (2015), http://www.sourceminer.org/. Acessed 2015, May.

VISJS (2015), http://visjs.org. Acessed 2015, May.