SortingHat: Wizardry on Software Project Members

David Moreno
Santiago Dueñas
Valerio Cosentino
Bitergia
[dmoreno,sduenas,valcos]@bitergia.com

Miguel Angel Fernandez
Ahmed Zerouali
Bitergia
[mafesan,ahmed]@bitergia.com

Gregorio Robles
Jesus M. Gonzalez-Barahona
Universidad Rey Juan Carlos
[grex,jgb]@gsyc.urjc.es

ABSTRACT

Nowadays, software projects and in particular open source ones heavily rely on a plethora of tools (e.g., Git, GitHub) to support and coordinate development activities. Despite their paramount value, they foster to fragment members' contribution, since members can access them with different identities (e.g., email, username). Thus, researchers and practitioners willing to evaluate individual members contributions are often forced to develop ad-hoc scripts or perform manual work to merge identities. This comes at the risk of obtaining wrong results and hindering replication of their work. In this demo we present `SortingHat`, which helps to track unique identities of project members and their related information such as gender, country and organization enrollments. It allows to manipulate identities interactively as well as to load bulks of identities via batch files (useful for projects with large communities). `SortingHat` is a component of `GrimoireLab`, an industry strong free platform developed by `Bitergia`, which offers commercial software analytics and is part of the CHAOSS project of the Linux Foundation. A video showing `SortingHat` is available at https://youtu.be/724I1XcQV6c.

KEYWORDS

Software mining, empirical software engineering, open source software, software development, identity merging

ACM Reference Format:
David Moreno Santiago Dueñas Valerio Cosentino, Miguel Angel Fernandez Ahmed Zerouali, and Gregorio Robles Jesus M. Gonzalez-Barahona. 2019. SortingHat: Wizardry on Software Project Members. In *ICSE '19 Companion: 41th International Conference on Software Engineering Companion, May 25-May 31, 2019, Montreal, QC, Canada.* ACM, New York, NY, USA, 4 pages. https://doi.org/xxxxx

1 INTRODUCTION

In recent years, members of software projects and in particular open source ones have become more reliant on an increasing number of software development tools to support, coordinate and promote their daily activities around software. For instance, source code changes are tracked via version control systems (e.g., Git, Mercurial), while coordination activities normally take place on mailing lists or instant messanging services (e.g., Slack, IRC). In order to interact with these tools, every member is usually required to set up an identity.

An identity generally includes a combination of email address, full name or username. Examples of identities are the commit signatures (i.e., full names and email addresses) of committers and authors in Git repositories or the GitHub and Slack usernames. However, the identities used by the same member may differ across the tools used in the project. It also may happen that a given project member uses more than one identity for the same tool (typically in version control systems and mailing lists), sometimes successive in time, sometimes even contemporary. Moreover, an identity can be shared by more project members, like during pair programming (i.e., same email address for both members).

Not considering that developers use multiple identities in software repositories can distort the results of research studies and affect professionals working in different collaborative development platforms, since individual contributions may be underestimated. From an industrial perspective, many open source projects or companies need identity merging support to show statistics that demostrate they have increasing and active communities. Thus, it is paramount to have tools that allow to merge identities with the maximum precision.

In the past decade several approaches have been proposed to unify project member identities used in software development tools [4]. Robles and Barahona [7] describe heuristics to match identities on source files, versioning repositories, mailing lists and bug tracking systems, based on member full names, usernames and email addresses. Bird et al. [1] use similarity metrics to semi-automatically unify multiple email addresses to unique identities. Stephany et al. [8] and Poncin et al. [6] combine Robles' and Bird's approaches and propose heuristic-based algorithms, while Kouters et al. [5] present an algorithm based on Latent Semantic Analysis. Wiese et al. offer a comparison of the heuristics used to identify multiple addresses in mailing lists [10].

In this paper we introduce `SortingHat`, a tool that simplifies the management of project member identities. It implements and extends the approach proposed by Robles and Barahona [7], by providing more than 20 commands to manipulate identities, including support for (i) identity merging based on email addresses, usernames and full names found on

Permission to make digital or hard copies of part or all 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. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
ICSE '19 Companion, May 25-May 31, 2019, Montral, QC, Canada
© 2019 Copyright held by the owner/author(s).
ACM ISBN xxxx.
https://doi.org/xxxxx

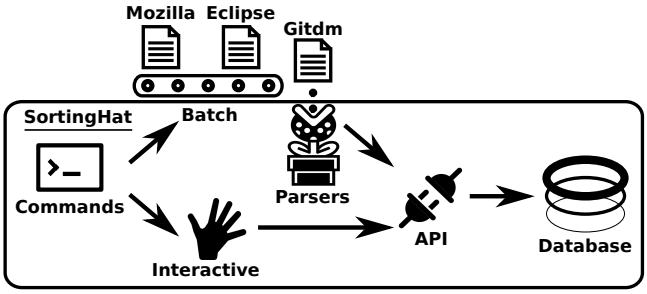

Figure 1: Overview of the approach underlying `SortingHat`.

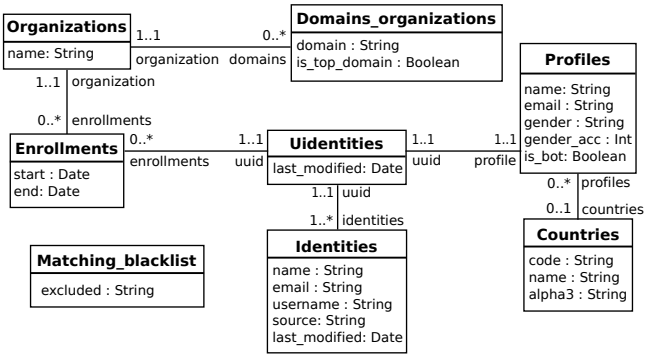

Figure 2: Overview of the SortingHat conceptual schema, where unique identities are the first-class citizens.

many tools used in software development, (ii) enrolling members to organizations for a given time span and (iii) gender assessment. `SortingHat` provides an easy-to-use shell, where commands can be interactively executed, and allows support for batch files to load large sets of identities. `SortingHat` functionalities can also be used through `Hatstall`, a web application that provides a graphical interface, thus granting no technical users to easily manage member identities. Both `SortingHat` and `Hatstall` are included in `GrimoireLab`, an industry strong free platform for software development analytics, part of the CHAOSS Linux Foundation project[1].

The remainder of the paper is organized as follows: Section 2 describes the approach underlying `SortingHat`, Section 3 shows how to install it and use it, Section 4 presents how `SortingHat`'s functionalities are exploited in GrimoireLab. Finally, Section 5 ends the paper.

2 OVERVIEW OF SORTINGHAT

`SortingHat` maintains a relational database with identities and related information extracted from different tools used in software development [3]. An identity is a tuple composed of a *name*, *email*, *username* and the name of the *source* from where it was extracted. Tuples are converted to unique identifiers (i.e., *uuid*), which provide a quick mean to compare identities among each other. By default, `SortingHat` considers all identities as unique ones. Heuristics take care to automatically merge identities based on perfect matches on (i) uuids, (ii) *name*, (iii) *email* or (iv) *username*. In case of a positive match, an identity is randomly selected as the unique one, and the other identities are linked to it.

Identities can be interactively manipulated via shell commands, which hide low-level implementation details to the user, thus decoupling the shell from the database technology in use. Then, each command is translated to one or more API calls in charge of dealing with the database specificities. Furthermore, identities can be loaded to `SortingHat` via batch files written in specific formats, thus speeding up identities imports for projects with large communities. Batch files are processed by parsers and inserted to the underlying database via API calls. Currently, the available parsers handle the following formats: `Gitdm`, `MailMap`, `Stackalytics` and the formats used for `Eclipse` and `Mozilla` committers.

The overall view of `SortingHat`'s approach is summarized in Figure 1. It is composed of three components: *Database*, *Commands* and *API*.

2.1 Database

`SortingHat` relies on open source technologies to store and manipulate identity information. It uses `MySQL` for storage and `SQLAlchemy` to bridge database relations into objects. The conceptual schema of the `SortingHat` database is shown in Figure 2. At its heart, there are the unique identities (i.e., *Uidentities*) of project members. Every unique identity can have more than one *Identities*, which are found in the software development tools of the project. They are described by the attributes *name*, *email* and *username*, plus the *source* where it was found (e.g., Git, GitHub) and the last date when its attributes were modified or when it was (un)linked to a unique identity (i.e., *last_modified*). Furthermore, a unique identity has a *Profile* and a list of *Enrollments*. A profile includes a summary of the project member such as *name*, *email*, whether it is a bot (i.e., *is_bot*), *gender*, and optionally the *Country* information (based on the ISO 3166 standard).

Enrollments express temporal relationships (i.e., *start* and *end* date attributes) between unique identities and *Organizations*. Thus, organizations and unique identities can have more than one enrollments over time, but a given enrollment exists only for an organization and a unique identity. *Organizations* are defined by a *name* and can belong to different domains (i.e., *Domains_organizations*). A domain represents affiliation relationships between organizations (i.e., *is_top_domain*). This is the case of large open source foundations like Linux and Mozilla, where several organizations contribute to.

Finally, organization names or identities with specific email addresses, usernames or full names can be easily excluded from `SortingHat` by registering their values in a *Matching blacklist*. The filter associated to the blacklist is executed every time an identity is inserted to the database or modified.

[1] https://chaoss.community/

2.2 Commands

`SortingHat` provides more than 20 commands[2] to manipulate identities data. The list below summarizes the common ones, which involve *manual* and *heuristic-based* operations.

- *Manual operations*
  - `Add`: add identities.
  - `Show`: show information about identities.
  - `Profile`: edit profile (e.g., update gender).
  - `Remove`: delete an identity.
  - `Merge`: merge unique identities.
  - `Move`: move an identity into a unique identity[3].
  - `Orgs`: list, add or delete organizations and domains.
  - `Enroll`: enroll identities into organizations.
  - `Withdraw`: unlink identities from organizations.
- *Heuristic-based operations*
  - `Unify`: merge identities using a matching algorithm.
  - `Affiliate`: enroll unique identities to organizations using email addresses and top/sub domains.
  - `Autoprofile`: auto complete profiles with emails and names found on the tools used in the project.
  - `Autogender`: auto complete gender information using the `genderize.io` API[4].

2.3 API

The shell commands are processed by the `SortingHat` API, which is based on a three-layer architecture that promotes modularization and decoupling. The first layer consists of *basic methods* that interact with the database and implement `CRUD` operations such as additions, deletions or searches (e.g., `find_organization`). The second layer contains *composed methods*, which leverage on the *basic methods*. This is the case of `move_identity`, which retrieves two identities and update their information. Finally, the top layer includes *complex methods* that have a one-to-one correspondence with the shell commands. They rely on *composed methods*.

3 SORTINGHAT IN ACTION

This section describes how to install and use `SortingHat`, highlighting its main features.

3.1 Installation

`SortingHat` has been developed in `Python` and tested mainly on `GNU/Linux` platforms. There are several ways for installing `SortingHat` on your system (from `pip`, `Docker` or source code) which are detailed in the `SortingHat` repository.

3.2 Use

Once installed, `SortingHat` can be used as a stand-alone program, a Python library, or via `HatStall`.

[2]A complete list and description of the commands is available at https://github.com/chaoss/grimoirelab-sortinghat.
[3]Note that when the `Move` operation is executed over the same identity, it allows to unmerge the identity from its unique identity.
[4]`Genderize.io` assesses the gender of a first name. However, `SortingHat` allows to modify the result with other genders not covered by the API.

Listing 1: Using SortingHat as a program.

```
# Adding identities
$ sortinghat add --name "Harry Potter"
  --email "hpotter@hogwarts.edu" --source git
  New identity 0ca..c1 added
$ sortinghat add --name "Harry Potter"
  --username "harryp" --source github
  New identity 11c..ab added
$ sortinghat add --name "H. Potter"
  --username "harryp" --source slack
  New identity 23d..r2 added
# Listing identities
$ sortinghat show 0ca..c1
  unique identity 0ca..c1
  Profile: ...
  Identities:
   0ca..c1 Harry Potter  hpotter@hogwarts.edu  - git
# Merge identities
$ sortinghat merge 0ca..c1  11c..ab
  Unique identity 0ca..c1 merged on 11c..ab
# Unify identities
$ sortinghat unify --sources github slack --matcher username
  Total unique identities processed 2
```

Listing 2: Using SortingHat as a library.

```
from sortinghat import api
def add_identity(cls, db, identity, backend):
 ...
 uuid = api.add_identity(db, backend, identity['email'],
          identity['name'], identity['username'])
 profile = {"name": .., "email": identity['email']}
 ...
 api.add_organization(db, identity['company'])
 api.add_enrollment(db, uuid, identity['company'], ...)
 return uuid
```

3.2.1 Stand-alone program. Using `SortingHat` as stand-alone program requires only some basic knowledge of GNU/Linux shell commands. Listing 1 shows how easy it is to add, list and merge identities. As can be seen, the command `add` accepts name, email, username and data source (e.g., git) of the identity to be inserted. The command `show` prints profile data and the list of identities linked to the unique identifier passed as input. Finally, the command `merge` allows to perform a manual merge on two identities using as input their unique identifiers, while the command `unify` automatically merges unique identities on a set of optional data sources using heuristics (e.g., perfect matches on usernames).

3.2.2 Python library. Including `SortingHat` functionalities to Python scripts and applications is easy, since the user only needs some basic knowledge of Python. Currently, `SortingHat` is integrated in `ELK`[5], which is an `ElasticSearch` wrapper, part of `GrimoireLab`, used to insert and process `Perceval` data [3]. Listing 2 shows how information related to identities is uploaded to `SortingHat` via `ELK`. The method *add_identity* uses the `SortingHat` API calls to add identities, organizations, enrollments.

3.2.3 HatStall. `SortingHat` functionalities are also available via `HatStall`, a Web application written in `Django`, a popular framework for Web development in Python. `HatStall` provides an intuitive graphical interface to perform operations over a `SortingHat` database. It is fully open source,

[5]https://github.com/chaoss/grimoirelab-elk

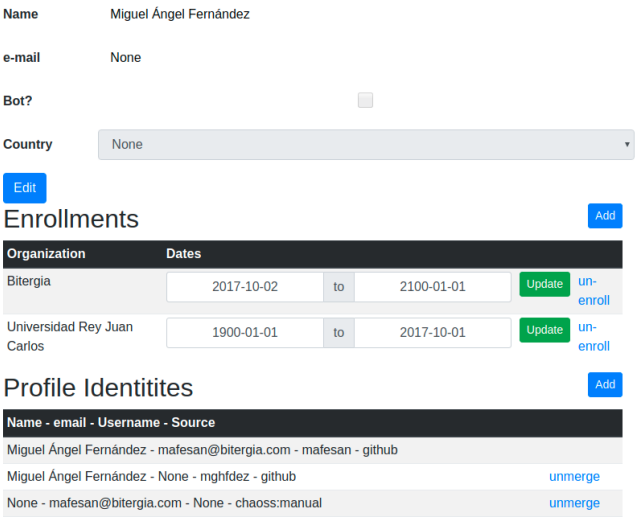

Figure 3: CHAOSS member data, including profile, enrollments and identities, visualized via *HatStall*.

available as a Docker image[6], and can be easily plugged to `GrimoireLab`. After starting `HatStall`, the user sets up the parameters (e.g., username, host) to connect to an existing `SortingHat` database, and then navigates and modifies the identities data through the application. Figure 3 shows the information of a CHAOSS project member. The page contains his profile data, enrollments and identities plus widgets to modify them.

4 EXPLOTATION

`SortingHat` is a very valuable tool for both academic and industrial purposes, since it provides support to track identities extracted from the software development tools used in a project. Thus, researchers and practictioners can rely on it to better assess the contributions of individuals and organizations in projects of any size. Furthermore, `SortingHat` can be considered a mature tool since it is used in `Bitergia` for maintaining the data of more than 30 customers. Figure 4 shows a visualization included in the dashboard of the CHAOSS project[7] which relies on `SortingHat` data. The pie chart represents the code contributions made on the CHAOSS project over the last year grouped by organizations. As can be seen Bitergia accounts for around three-quarters of the commits, followed by the universities of Missouri and Nebraska.

5 CONCLUSIONS AND FUTURE PLANS

Tracking and unifying member contributions in software projects, and in particular open source ones, is a challenging and relevant task for both researchers and practitioners.

In this demostration paper we have presented `SortingHat`, which tracks and manages identities found in the different

[6]https://hub.docker.com/r/grimoirelab/hatstall/
[7]https://chaoss.biterg.io/app/kibana

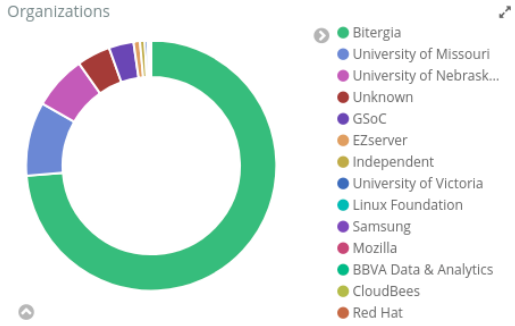

Figure 4: `SortingHat`-based visualization that shows how organizations have contributed to the CHAOSS project in the last year.

development tools used in a project, thus allowing to link identities to individual project members. `SortingHat` functionalities can be executed via Shell, embedded in Python applications or used through `HatStall`. `SortingHat` is integrated in `GrimoireLab`, and daily helps `Bitergia` in managing identities information coming from more than 30,000 development tools. Currently, we are working to expose `SortingHat` data as an API, using `GraphQL`, paying special attention to secure personal data. In the future, we would like to improve the way of resolving false positive (which are detected manually, at present) by extending the heuristics used to match identities with machine learning [5] and similarity metrics [1]. Furthermore, we plan to extend the database schema with data about roles covered in organizations and work experiences to better assess tenure diversity in software projects [9].

REFERENCES

[1] Christian Bird, Alex Gourley, Prem Devanbu, Michael Gertz, and Anand Swaminathan. 2006. Mining email social networks. In *MSR*. 137–143.
[2] A. Capiluppi, A. Serebrenik, and L. Singer. 2013. Assessing Technical Candidates on the Social Web. *IEEE Soft.* 30 (2013), 45–51.
[3] Santiago Dueñas, Valerio Cosentino, Gregorio Robles, and Jesus M Gonzalez-Barahona. 2018. Perceval: software project data at your will. In *ICSE Companion*. 1–4.
[4] Mathieu Goeminne and Tom Mens. 2013. A comparison of identity merge algorithms for software repositories. *Sci. Comput. Program.* 78, 8 (2013), 971–986.
[5] Erik Kouters, Bogdan Vasilescu, Alexander Serebrenik, and Mark GJ van den Brand. 2012. Who's who in Gnome: Using LSA to merge software repository identities. In *ICSM*. 592–595.
[6] Wouter Poncin, Alexander Serebrenik, and Mark Van Den Brand. 2011. Process mining software repositories. In *CSMR*. 5–14.
[7] Gregorio Robles and Jesus M Gonzalez-Barahona. 2005. Developer identification methods for integrated data from various sources. *ACM SIGSOFT SEN* 30, 4 (2005), 1–5.
[8] François Stephany, Tom Mens, and Tudor Gîrba. 2009. Maispion: A tool for analysing and visualising open source software developer communities. In *IWST*. 50–57.
[9] Bogdan Vasilescu, Daryl Posnett, Baishakhi Ray, Mark GJ van den Brand, Alexander Serebrenik, Premkumar Devanbu, and Vladimir Filkov. 2015. Gender and tenure diversity in GitHub teams. In *CHI*. 3789–3798.
[10] Igor Scaliante Wiese, José Teodoro da Silva, Igor Steinmacher, Christoph Treude, and Marco Aurélio Gerosa. 2016. Who is who in the mailing list? Comparing six disambiguation heuristics to identify multiple addresses of a participant. In *ICSME*. 345–355.