ePOP: An Eclipse-based Extensible Research Evaluator

Paolo Arcaini[1], Angelo Gargantini[2], and Elvinia Riccobene[1]

[1] Dipartimento di Informatica, Università degli Studi di Milano, Italy
{paolo.arcaini,elvinia.riccobene}@unimi.it
[2] Dip. di Ing. dell'Informazione e Metodi Matematici, Università di Bergamo, Italy
angelo.gargantini@unibg.it

Abstract. The research evaluation process is becoming nowadays of critical importance since public and private institutions use results of this process to decide research funds allocations, promotions, and hiring positions. For this purpose, several data sources are available with different features, and different degree of openness, accuracy, completeness, and cost. It would be desirable having a tool able to integrate these data sources for comparison and measurement, possibly based on a rich and extensible set of research evaluation indexes.
In this paper, we present ePOP, an Eclipse based tool for research evaluation, developed by exploiting the Rich Client Platform. ePOP is a multiplatform, open and free software. It can access multiple data sources, and it provides different research evaluation metrics. ePOP can be easily extended by exploiting the plugin feature of the Eclipse platform in order to integrate new data sources and new research evaluation indexes.

1 Introduction

Public and private universities are lately undergoing to a thorough evaluation process of their research results. National agencies and bodies, like the Agenzia Nazionale di Valutazione del sistema Universitario e della Ricerca (ANVUR) [1] in Italy, the Research Assessment Exercise (RAE) [4] in UK, the Performance Based Research Funding (PBRF) program [3] in New Zealand are invested with increasing responsibilities in deciding funds allocations and grants to projects and individual researchers.

Because of the importance of the evaluation process, these agencies tend to collect as most data as possible and, while peer reviews are still considered the most accurate means of evaluation, they lean toward performing (part of) the evaluation in an automatic way, meanly for cost and time constraints. For this reason, there is an increasing demand of automatic tools and data repositories that should make the process outcome predictable, verifiable, accurate, repeatable, and as objective as possible.

This tendency rises several issues: how to select and rate data sources, how to aggregate the information coming from multiple sources and how to measure the quality of the research. Moreover, a parallel goal of this activity is to provide the

researchers with tools for the auto-evaluation of their activity in order to better address their research efforts and to permit a pre-evaluation of their curricula.

During these years, several data sources have become available with different degrees of openness, accuracy, completeness, and cost. For instance, it is well known that Google Scholar [2], a free service provided by Google, *provides an avenue for more transparency in tenure reviews, funding and other science policy issues, as it allows citation counts, and analyses based thereon, to be performed and duplicated by anyone* [9]. However, other services, like the Web of Science provided by Thomson ISI [6], which is only available to selected and paying academic institutions, may provide more accurate and reviewed information.

Similarly, metrics and measures used in the evaluation process are continuously devised by researchers in order to increase the accuracy and precision of the evaluation process. For instance, while the number of publications gives an immediate easily understandable measure, most researchers retain more accurate the h-index [10] since it provides a measure of how the work of the researcher is considered in the research community. More complex measures can take into consideration also the period of time during which a researcher has published his/her papers. Moreover, many data providers and research indexes are tailored to specific research fields.

For these reasons, a tool that permits to search over several data sources at the same time, to compute several research indexes, and to compare the obtained results would be useful.

In this paper, we present an Extensible Publish or Perish tool – epop. It is based on Eclipse technologies and exploits the Rich Client Platform (RCP). epop is a multi-platform, open and free software. It can access several data sources, and it provides different research evaluation indexes. epop can be easily extended by exploiting the plugin feature of the Eclipse platform in order to integrate new data sources and new research evaluation indexes.

Although epop in its current stage is the result of the author efforts, it could become a project of the Italian eclipse community where professors and students (e.g. those participating to the ETC project) contribute to add new plugins, new features, and fix the bugs.

In Sect. 2, the paper presents a list of desired features epop has to have. Sect. 3 describes epop as an Eclipse RCP platform, and introduces the suitable extension points, together with their implementation. Sect. 4 presents the overall epop architecture and discusses the testing activity. How to use epop is described in Sect. 5. Sect. 6 presents some related work. Sect. 7 concludes the papers outlining some future improvement of the platform.

2 epop Goals

The main goal of epop is to provide researchers and academic institutions with a flexible tool for research evaluation. We have identified the following desired features. epop wants to be:

free: at zero cost for users who do not have to pay for its usage and who are not restricted from giving it away to anyone else.

open: distributed under an open source license. Indeed, it is distributed under the EPL licence. The user can read the code and check how queries are performed and data processed. This is particularly important, since malicious programmers could modify the data about themselves and provide false information about others.

multi platform: running on any operative system. Indeed, it is written in Java and it is based on Eclipse. It can run on any machine OS with a JRE and the SWT graphical widgets. Currently it works on Windows, Mac, and Linux.

multi sources: accessing several research data sources. Indeed, every data source is a plugin. New data sources can be easily included by using the extension mechanism of Eclipse RCP. This fosters data completeness and accuracy. There are many bibliographic databases[3] with different degrees of completeness and precision. For instance, Google scholar is widely used, easily accessible, and rich of information, but it is not very accurate.

multi metrics: different evaluation indexes have to be available for use and comparison. Basic metrics are already included in ePOP, but researchers may introduce new ones. Indeed, in ePOP, a metric is just a plugin (extension in the terms of RCP).

easily extensible: new data sources and new evaluation metrics have to be easily added to the framework. This is possible in ePOP thanks to the plugin extension mechanism supported in Eclipse.

client-based: information have to be directly retrieved from the data sources without any intermediate server. Although web server-based applications (like QuadSearch[4] and ResEval[5]) seem more usable, since they can be accessed from any web client (mobile, tablet, and so on), and more easily maintainable, since no updates of clients are necessary, building a server that collects the data from bibliographic servers and re-exports them may be against the license of use of the bibliographic server[6]. For instance, the web server may expose data that a user has no rights to access. A client-based application, instead, gets the right from its users, and for copyright purposes it is not different from a normal web client. ePOP has been designed as a *client-based* application in this sense.

easily updatable: updating the application (or part of it) upon improvements or extensions of data sources and/or evaluation indexes should be possible in an automatic way via Internet. In the present version, ePOP lacks of this feature that is currently left as future improvement.

---

[3] See `http://en.wikipedia.org/wiki/List_of_academic_databases_and_search_engines`

[4] `http://quadsearch.csd.auth.gr/index.php?s=2`

[5] `http://project.liquidpub.org/reseval/`

[6] Google scholar term of service prohibits automatic searches like those coming from other servers

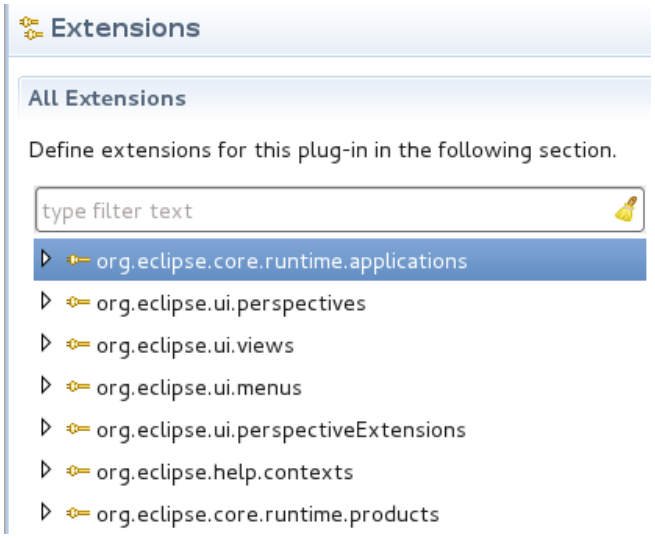

Fig. 1. Extensions of RCP extension points

3 ePOP as an eclipse RCP Application

The Eclipse platform, that is essentially a set of plugins, has been designed in a way that permits to build any client application (not necessarily an IDE) simply reusing its components. The *Rich Client Platform* (RCP) is the minimal set of plugins that is needed to build a rich client application: only two plugins are required, `org.eclipse.ui` and `org.eclipse.core.runtime`. In addition to the two core plugins, other plugins can be used to add several functionalities to the application (e.g., Help UI, Update Manager, etc.).

3.1 Reusing eclipse graphical widgets and RCP extension points

In order to build ePOP we have implemented some extensions of several classical extension points of RCP (see Figure 1). The extension of `org.eclipse.core.runtime.applications` is used to identify the entry point of the application. The extension of `org.eclipse.ui.perspective`, instead, is used to define the layout of the application. The extension of other extension points permits to define menus, views, and so on.

3.2 Definition of extension points

The extension point mechanism of Eclipse permits to build applications in which components are loosely coupled, making it easy to add/remove/replace them.

A plugin declares what are the portions of its functionality that can be extended (or customized) by defining some extension points. An extension point

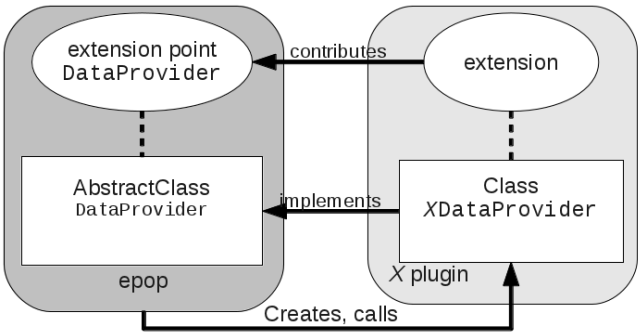

Fig. 2. Extension point mechanism

declares a contract (a combination of XML and Java interfaces) that the plugins that want to extend it must satisfy in their extensions. One can think of an extension point as a port – an entry point for other plugins to offer services. An extension is a plug that connects to the right port. An extension point defines a contract between the application and the service provider introduced as plugin. The extension implementation is the actual service which will be added to the application by using the plugin mechanism of eclipse.

There are several benefits from this architecture. New plugins can be dynamically added and removed from the application without recompiling them. Third-party tools can be easily added to the application by registering them as extensions. A plugin includes some descriptive information and the application extension point can decide how to use it. For instance, a plugin can declare to support a feature and the application can decide if it is worth loading the extension or not. The development of a plugin is strongly decoupled with the development of the application, making easy for third-parties to contribute to the framework.

Very often an extension point introduces a Java interface (or abstract class) which must be implemented by the extensions. Figure 2 shows the structure of the extension point `DataProvider` defined in `epop`. It defines the abstract class `DataProvider` declaring the methods necessary for any data provider. A plugin (`X` in Fig. 2) must define a class (`XDataProvider`) implementing the required interface in order to extend the application with a new data provider. The application will become aware of the new provider and will be able to create and call instances of that class when needed.

This mechanism permits to build applications that can be extended after their deployment, also by individuals/companies different by the developer who defined the extension point.

In `epop` we have declared two extension points:

– `org.epop.dataprovider` permits to add new data sources where to retrieve information about the researcher; actually we have extended it with exten-

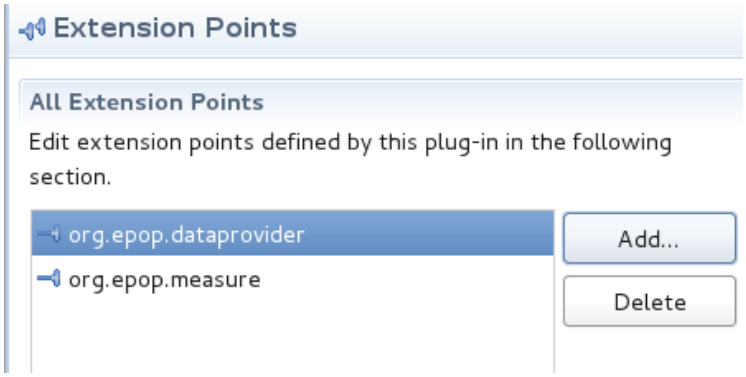

Fig. 3. epop – Extension points definition

sions for the data sources *Google Scholar*, *Microsoft Academic Search* and *Scopus*, as described in Section 3.3.

– `org.epop.measure` permits to add new metrics for the evaluation of the researcher; actually we have provided extensions for all the metrics described in Section 3.3.

Figure 3 shows the definition of the extension points.

3.3 Implementation of epop extensions

Data providers. epop can currently gather data from the following data providers:

Google Scholar It is a free search engine developed by Google. It indexes scholarly literature of different disciplines and published in different formats (both online and offline documents are indexed). It indexes journal articles, technical reports, preprints, theses, books, and other documents, including web pages that are deemed to be *scholarly*. It ranks the results using a ranking algorithm that weighs *the full text of each article, the author, the publication in which the article appears, and how often the piece has been cited in other scholarly literature*[7].

Microsoft Academic Search It is a free search engine developed by Microsoft Research. It covers different disciplines, around 38 million publications and 19 million authors. The order of results is based on the number of citations.

Scopus It is a bibliographic database containing abstracts and citations for academic journal articles. It is owned by Elsevier, an international publisher of scientific journals, and is available by subscription. Searches in Scopus incorporate searches of scientific web pages through Scirus, another Elsevier product, as well as patent databases.

[7] `http://scholar.google.com/scholar/about.html`

Metrics. epop implements the following common measures:

Number of publications It is the total number of documents retrieved for the specified author.

Overall citations count It is the sum of the citations received by each document written by the searched author.

Average number of citations per publication It is the ratio between the overall citations count and the number of publications.

h-index It is the Hirsch index, described in [10]. Basically, a scholar with an index of h has published h papers each of which has been cited by others at least h times.

g-index It was introduced by Leo Egghe in [7]. Given a set of articles ranked in decreasing order of the number of citations that they received, the g-index is the (unique) largest number such that the top g articles received (together) at least g^2 citations.

Signal to noise ratio It is the ratio between the number of publications that contributes to the computation of h-index and g-index (i.e., those that have received at least one citation), and the total number of publications. The aim of this metric is to somehow estimate the quality of the work of a researcher compared to the number of publications he/she wrote.

Moreover, the following new measures have been introduced:

Academic age Such indicator is sometimes used in combination with the previous indicators to take into consideration also the period of time during which the researcher has written the papers under consideration. Different interpretations exist for this indicator. The first interpretation considers the academic age as the time between the first and the last publication (*active academic range*). The second interpretation, instead, considers it as the time between the first publication and the current year (simple *academic age*). The first interpretation should be used for considering the career of a retired researcher, so to not invalidate the results because of the years after his/her retirement during which he/her has not published anymore. The second interpretation, instead, should be used to evaluate active researchers.

m-index It is the h-index divided by the academic age [10]. It is also called *m-quotient*.

Optimal m-index It is a pair of indexes (x, y) where x is the maximum `m-index` considering only the publications of the last y years. This is useful when one wants to maximize his/her `m-index` by limiting the pasts years for which he/she considers the publications.

hc-index The Contemporary h-index was proposed by Antonis Sidiropoulos, et al. [12]. It weights each cited article, giving less weight to older articles.

NPapers 10y The number of papers published in the last 10 years.

Citation rate It is the number of citations divided by the academic age.

Figure 4 shows the definition of the extensions to the extension point `org.epop.measure`.

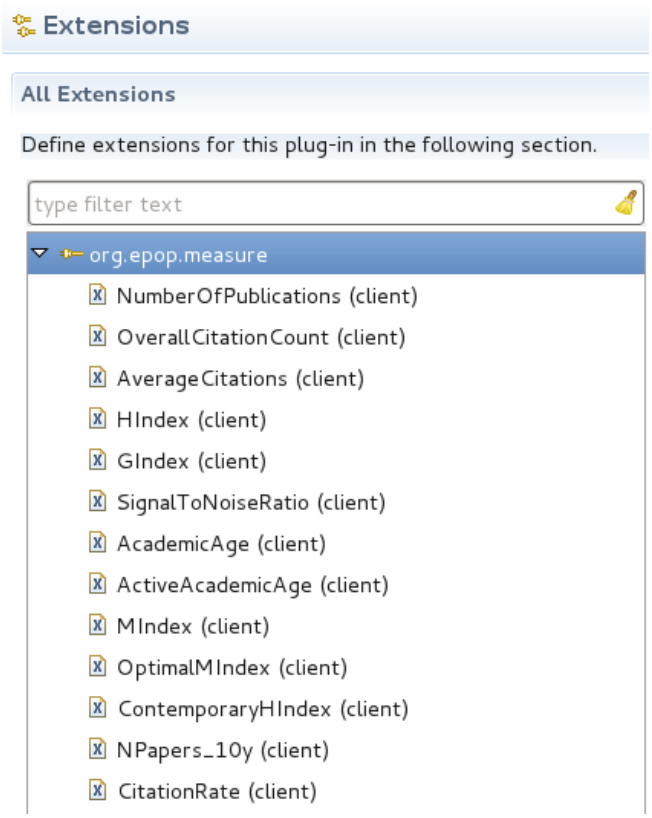

Fig. 4. Extension points definition

4 Overall architecture

Fig. 5 shows the architecture of epop. We have organized the application as a set of plugins, each defined in its own eclipse project, depicted with boxes in the Figure. The epop.core is the core plugin in which the extension points are defined. It also contains the definition of the classes used by all the others plugins (like the classes for Paper, List of papers, Measure, and so on). The core has no dependencies with other plugins, except the eclipse runtime component. The basic research indexes introduced by epop are defined in the epop.meausure plugin. The plugins that extend the data provider extension point, for the three main data sources defined in epop, are epop.googlescholar, epop.msacademic, and epop.scopus. In order to retrieve and properly parse data from the their sources, they use two auxiliary libraries, namely the apache.httpclient and jerico.htmlparser, which are organized as plugins as well. The top plugin, epop.application, defines the application with the menus, views, and dialogs. It depends, besides all the other plugins, on org.eclipse.ui.

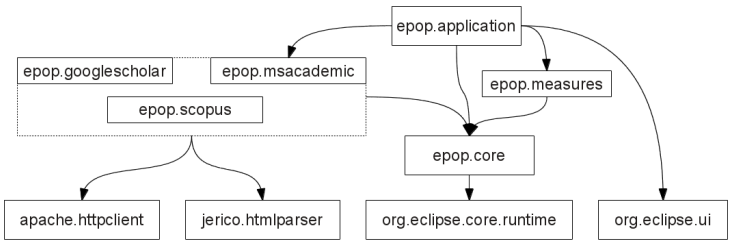

Fig. 5. epop Plugins and architecture

4.1 Testing epop

Since we are building a software that extracts sensitive information, it is important to test it deeply. We provide a set of tests that give us enough confidence that the written application is correct. We can not write tests checking the exact values of some metrics, since these values can change over time; however we can check that the returned values are in a given range.

Since the inputs used in the tests are names of researchers and the expected outputs are values related to authors' research (e.g., number of publications, h-index, ...), we do not want to add the inputs in clear text, since it could be awkward for them. So, we add the inputs in ciphertext, so that crawlers and/or users can not extract sensitive information simply browsing the code.

However, since we want to show that we do cheat about the test results, we have added the password used to encrypt the inputs directly in the code. In this way, anyone can download the code, decrypt the inputs and verify that we are not faking the results.

Code 1 shows a fragment of a JUnit test code in which we check that the number of papers written by one the authors of this paper is greater than 100.

5 Using epop

epop can be downloaded from the Download section at `http://code.google.com/a/eclipselabs.org/p/epop/`. epop does not need to be installed. You just have to download the compressed file for your target platform, unzip it wherever you want in your filesystem, and run the *epop* executable.

The starting screen of the epop application is shown in Fig. 6. The user can specify an author name in a proper text field and select the data sources he/she wants to use.

The output of the search is shown in different tabs, one for each data source, displaying the title of the paper, the year, where it has been published, and the number of citations. Moreover, a further tab shows a summary of all the results obtained from the different data sources: some papers, indeed, could be provided only by some providers, and it is possible that, for a particular paper,

```
package org.epop.dataprovider.googlescholar;

import org.junit.Test;
...

public class GoogleScholarProviderTest {

    @Test
    public void testRunQuery() throws Exception {
        getListPapers("ppGDSF/2cu7+/mZxO8+aPwoO7En2RqZhGH4jXUi0Aq0=", 100);
    }

    private void getListPapers(String authorNameCrypted, int numPapers) throws Exception {
        Query q = new Query(StringCipher.decrypt(authorNameCrypted));
        GoogleScholarProvider gs = new GoogleScholarProvider();
        List<Paper> res = gs.runQuery(q);

        // actual test: it checks that there are at least numPapers and
        // that, for each paper, all the necessary information are provided
        GetHTMLUtils.checkPapers(numPapers, res);
    }
}
```

Code 1. Test case with ciphered input

some fields provided by a provider are more accurate than those provided by another provider. An example of search result is shown in Fig. 7.

6 Related work

The tool that inspired the current work is *Publish or Perish* [8], a program that retrieves data from Google Scholar. The tool has several disadvantages: a) it uses only Google Scholar as data provider, without providing a way to add new data sources; b) it is closed source, so it is not possible to inspect the code to check that it does not fake the results; c) it is not easily portable (e.g., in order to be used in Linux it requires *Wine*, a tool that permits to execute Windows programs in Linux).

Scholarometer [5] is a plugin for Firefox and Google Chrome that permits to easily query Google Scholar. Since such browsers are available for any OS, it is more portable than *Publish or Perish*. *Scholarometer* requires to tag each query with one or more discipline names so that the developers of *Scholarometer* can collect data about the various disciplines: these collected data will be made publicly available. The process of tagging is quite annoying for the user who has to look for a proper tag among a set of predefined tags or define a new one. The data gathered from Google Scholar upon a user query are previously parsed on the servers of *Scholarometer* and then shown to the user. Since the code running on the servers is not available (just the code of the plugin is available), there is still the problem of how to guarantee that the data are not modified in malicious ways.

ResEval [11] is a web application having similar goals of ePOP. It permits to evaluate individual researchers and groups (Group Comparison Tool). Like

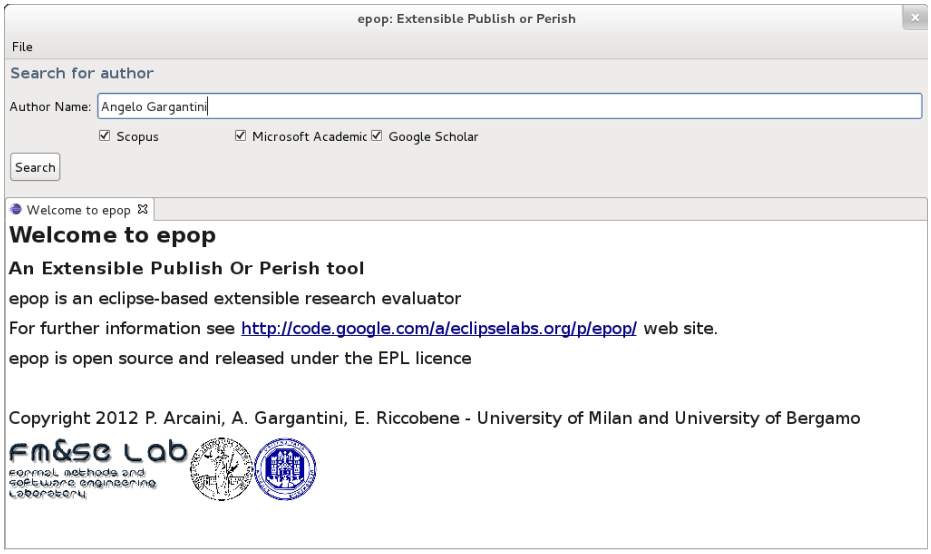

Fig. 6. epop: Welcome screen

our tool, it relies on several sources, and provides an easy way for defining and computing custom metrics. However, neither the web tool nor the code are currently accessible.

7 Summary and Plans

We here present epop, an Eclipse based tool developed to address the current request of tools able to support the research evaluation process in a flexible, complete and more accurate way w.r.t. those provided by the existing applications.

epop has been developed by exploiting the Rich Client Platform, and it has been designed as a multi-platform, open and free software. It can access multiple data sources, and it provides different research evaluation metrics for research results comparison. epop can be easily extended, either as data sources and as evaluation indexes, thanks to the the plugin feature of the Eclipse platform.

The major weakness of epop is that, in order to retrieve information from the data sources, we must parse html files that could sometimes be not well structured or change their format without notice. Moreover, some sources could provide only partial information: Google Scholar, for example, shows only a portion of the venue where the paper has been published if this is too long (just if the author does not have a google scholar account).

epop can be improved in several directions that we here shortly describe and that are left as future work.

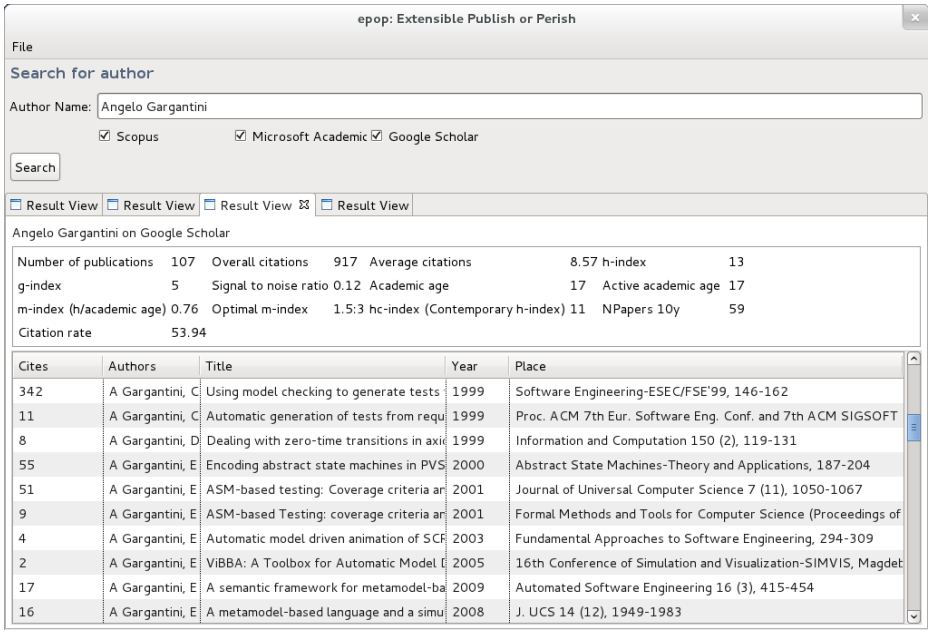

Fig. 7. epop: Results view

A first improvement is to make epop easily and automatically updatable, according to the feature described in Sect. 2.

A further plan is to add to epop the possibility to summarize the co-authors of a searched author. For each co-author c we would like to provide different indicators: (1) the number/percentage of papers written in collaboration with c; (2) the average number of citations obtained considering only the papers in which c is a co-author; (3) the h-index/g-index obtained considering only the papers in which c is a co-author.

Moreover, we would like to provide a visual representation of some indexes (e.g., an histogram showing the number of paper per year) to improve the usability of the tool.

epop could be useful also for publishing planning: an author could decide what to cite and in which journal to submit a paper depending on the impact over her/his publication measures.

References

1. Agenzia Nazionale di Valutazione del sistema Universitario e della Ricerca (ANVUR). http://www.anvur.org/.
2. Google Scholar. http://scholar.google.com/.
3. Performance Based Research Funding (PBRF). http://www.tec.govt.nz/Funding/Fund-finder/Performance-Based-Research-Fund-PBRF-/.

4. Research Assessment Exercise (RAE). http://www.rae.ac.uk/.
5. Scholarometer. http://scholarometer.indiana.edu/.
6. Web of Science. http://thomsonreuters.com/products_services/science/science_products/a-z/web_of_science/.
7. L. Egghe. Theory and practise of the g-index. *Scientometrics*, 69(1):131–152, 2006.
8. Anne-wil Harzing. Publish or Perish. http://www.harzing.com/pop.htm.
9. Anne-wil Harzing and Ron V. Wal. Google Scholar: the democratization of citation analysis? *Ethics in Science and Environmental Politics*, pages 1–27, 2007.
10. J. E. Hirsch. An index to quantify an individual's scientific research output. *Proceedings of the National Academy of Sciences*, 102(46):16569–16572, October 2005.
11. Muhammad Imran, Maurizio Marchese, Azzurra Ragone, Aliaksandr Birukou, Fabio Casati, and Juan Jose Jara Laconich. ResEval: An open and resource-oriented research impact evaluation tool. TR DISI-10-016, Department of Information Engineering and Computer Science. University of Trento, 2010.
12. Antonis Sidiropoulos, Dimitrios Katsaros, and Yannis Manolopoulos. Generalized h-index for disclosing latent facts in citation networks. *CoRR*, abs/cs/0607066, 2006.