Suggesting Software Components for Reuse in Search Engines Using Discovered Knowledge Techniques

Alexandre Costa Martins
Federal University of Pernambuco (UFPE)
Reuse in Software Engineering (RiSE)
Recife, Brazil
acm3@cin.ufe.br

Vinicius Cardoso Garcia
Federal University of Pernambuco (UFPE)
Reuse in Software Engineering (RiSE)
Recife, Brazil
vcg@cin.ufe.br

Eduardo Santana de Almeida
Reuse in Software Engineering (RiSE)
Recife, Brazil
esa@rise.com.br

Silvio Romero de Lemos Meira
Federal University of Pernambuco (UFPE)
Reuse in Software Engineering (RiSE)
Recife, Brazil
srlm@cin.ufe.br

Abstract—The software reuse initiatives are better implemented when there is an efficient way to find the reusable assets. However, the search and retrieval of such information is considered a big deal in literature, once there is a gap between what the software engineer would like to retrieve and what is stored in the repository. Thus, this paper presents an efficient way to reduce this problem and aid search engines applying data mining techniques on log mechanisms to extract knowledge about the historic data. In order to evaluate the results, an initial experiment is also discussed.

Keywords—software reuse; search engines; information retrieval; data mining; association rules;

I. INTRODUCTION

Software reuse is the process of creating software systems from existing software rather than building them from scratch [14]. Hence, when it is systematically performed, it presents benefits such as improvements in time-to-market, quality and costs reduction, since it uses assets already tested, certified and documented [8]. However, to achieve success on reuse initiatives, often, it is necessary to provide an efficient way to search and retrieve the assets since, according to [23], *"To reuse a software component, you first have to find it"*.

The literature presents some search and retrieval tools, as pointed by Garcia et al. [9], to aid the reuse process. However, some research [19], [15] show that the main obstacle with these tools is to retrieve a component that corresponds to the developer's need, once there is a gap between the problem formulation, in developer's mind, and the component description in the repository [12]. Several solutions [19], [15] have been proposed to attenuate it. However, the user still needs to build queries every time, and it is hard as also highlighted by [29].

Another problem is associated with the dependencies among the components [27], since sometimes, the returned assets do not solve the entire problem, and the developer has to find additional components to complement the entire solution. Thus, the dependencies are identified only on demand, when the developer realizes it and tries to search for another component. In this way, this paper

proposes an approach that uses data mining techniques to solve it by reducing the conceptual gap on the queries built. The solution is based on the reduction of performed queries. Thus, the main goal is to optimize the component search and retrieval, through its historic use which is monitored by a log mechanism such as file or database.

This work is part of the **B.A.R.T (Basic Asset Retrieval Tool)** [10] project whose main goal is to develop a robust tool to search and retrieve software components. In this project, we had experimented new trends related to search and retrieval such as active search [17], context [24], folksonomy [26], semantic [5], facets [18] and Data Mining [16]. These efforts have presented initial findings which stimulate the research in this direction. Hence, we believe that our approach is a complement solution for the previous approaches because the suggestions have no relations with a specific technique of search, i.e., It can be used together.

II. RELATED WORK

In the literature, some work propose different tools to aid the information search and retrieval. CodeBroker [29] presents a tool that finds asset about components that are relevant for activities realized in the same moment. According to [29], ad-hoc evaluations presented that this type of strategy is effective to promote reuse. It allows the exploration from a new style of collaboration between users and machines. Based on the same concepts, Strathcona [13] uses software agents to implement an active search mechanism to search source code examples and to aid the developers in the codification process. It presents examples extracted of repositories based on six different heuristics.

The related work presented are associated with source code reuse based on search and retrieval. Our approach complement these work allowing reuse based on the historic use. We believe that the approaches are complementary and can be used together, as we did, with any search engine.

The use of association rules was experimented too in [6] to improve search engines. However, the rules were used to suggest queries for users and its dataset was composed of almost 3 millions of lines from yahoo website.

III. B.A.R.T SEARCH ENGINE

According to the idea that reuse can be performed in a systematic way [7], supported by an environment to aid in the software development process activities, we constructed *B.A.R.T (Basic Asset Retrieval Tool)*. *B.A.R.T* is a component search engine to aid the software engineer to find component, which implements the problem that needs to be solved [10].

B.A.R.T is based on the client-server architecture, where the client side communicates to the server side through a web service layer. The client side is represented by Eclipse and MS Word plug-ins besides a brand new web interface which searches reusable assets on the server side [10]. Its architecture was designed to be extensible by providing the capacity to add new features by including new components. Initially, this proposal concentrates its efforts only on an external client and in architectural modules of server side. The basic modules which comprise the entire architecture are: searcher, retriever, indexer, filter and repository manager.

The main components of *B.A.R.T* are detailed next.

- **Repository manager:** this module checkouts assets from Concurrent Version System (CVS) repositories on the Internet and maintains the entire repository scheme of the system. It is also possible to schedule updates from previous checkouts in order to capture brand new assets.
- **Filter:** after automatic checkout, this module filters those assets which do not satisfy some conditions such as unnecessary extension and lack of documentation (particularly applied to source codes).
- **Indexer:** this module indexes the reuse repository and creates the index base which is essential for the search occurrence.
- **Searcher:** this module is responsible for processing the user query and prepares it to seek the assets through the index base answering semantic constraints.
- **Retriever:** after a successful search, the user can download assets listed in the *B.A.R.T* client application. This module is responsible for transferring them to the user directory or to a specific Eclipse project.

The current version of *B.A.R.T* presents functionalities such as keyword, semantic and folksonomy search. Therefore, a robust infrastructure was implemented to manage users, extract statistics, register logs and etc.

IV. THE MAIN REQUIREMENTS

Once presented the context of our tool, we will discuss how apply the knowledge discovery techniques for software reuse. For that, some requirements were proposed. These requirements are based on studies about existing solutions in the literature [15], systematic analysis of

reuse tools that aid the software development [9], and our interaction with the industry in projects in this direction. Thus, the main specified requirements are:

1. **Suggest Associated Asset.** This tool must be developed to suggest associated assets for the search engine users. These associations are based on knowledge extraction of log files as exemplified previously.
2. **Extract Association Rules.** The associations mentioned previously are provided by association rules extraction. This approach must extract the rules using well-defined algorithms such as *Apriori* [1], *Dynamic Itemset Counting (DIC)* [3], and *FP-Growth* [11].
3. **Parser from Log Mechanisms Artifacts.** The log mechanisms accepted by this tool are log files. The knowledge extraction process uses these data to extract the rules.
4. **Provide a Visual Interface.** The tool must provide an alternative interface to aid the user to assimilate the suggestions. For this interface to become more intuitive, it must be based on graphs [21].

V. ARCHITECTURE SPECIFICATION AND THE OUTLINE OF IMPLEMENTATION

The current architecture is divided in two modules: **Association Rules Extractor (AR Extractor)** and **Association Rules Viewer (AR Viewer)**. The first one is responsible for the knowledge extraction using log files whose flow is: (i) parser the file and (ii) extract the association rules. The second module is responsible for showing the results to the user. These results are plotted as a graph and show the recommendations.

The inputs are a log mechanism result, such as (1) file or database, and a (2) **XML Descriptor**. This XML describes the log structure and aids the log parser process. Some configurations, such as limits of measures, algorithm type, are set by (3) Properties Configurations and the result is a rule collection as a (4) XML result. The result and search engine responses are received by the second module: (5) **AR Viewer**. The viewer renders the results as a graph for the user.

The modules integrate an environment whose responsibility is to provide a better way to search and retrieve assets. In Figure 1, the complete flow is showed. The first step is the query request by the user (1) using **Remote Method Invocation (RMI)** or **Web Services (WS)** connection. Next, the query is processed by the **search engine (2)** that uses the data from files localized in databases, file systems or CVS. This system usage generates a log that represents the real use of the search engine (3). Thus, the **AR Extractor** uses this file to extract association rules (4) that will aid the user. The final result is the sum of the search engine response and the rules associated with it (5). The final result is built by **AR Viewer** as a graph that presents the response of the user query and the suggestions.

Figure 1. Architecture and its interaction with the user

An important issue is that it must be integrated with any search engine that uses a log file. After this, as shown in Section IV, the algorithm can be changed and alternative implementations can be used.

A. Association Rules Extractor

The **AR Extractor** is divided in three sub-modules: *XML Descriptor Extractor*, *Algorithm Manager* and *XML Generator*, as seen in Figure 2. The *XML Descriptor* is parsed and next the lines are grouped in transactions. These transactions represent the files that each user downloaded in a specific time window. This number must be provided by search engines to demark the information collected in a time window.

Figure 2. Intern Architecture of AR Extractor Module

In the *Algorithm Manager*, the data mining phase is processed and the algorithm is selected. This choice depends on each situation as a distributed or singles databases and some characteristics must be analyzed, such as performance, CPU process, memory usage and return time. During the evaluation, some rules are accepted and the others are left behind. This step is important to improve the final rules quality.

1) *Rule Filtering*: The extracted rules can have a lot of antecessors and successors in this composition. Thus, some rules must be pruned to simplify the suggestion and reduce the noise.

- **Antecessor Quantity**: The rules that have two or less antecessors will be accept. Rules with more than

two antecessors are so complicated to the user and it causes more noises in the final visualization.

Ex.: FileA.java, FileB.java, FileC.java $\Rightarrow$ FileD.java

- **Successors Quantity**: The suggestions must be based in rules that have only one sucessor.

Ex.: FileA.java $\Rightarrow$ FileD.java, FileF.java

B. Association Rule Viewer

The **AR Viewer** is detailed in Figure 3 and it has four sub-modules. The **Communication layer** provides a way to connect the client and server side. This layer was implemented using *Web Services* and *RMI*. The **Interpreters** (Rules and Query Result) are used to transform the XML returned from the server and prepare it to graph render. This graph is provided by the last sub-module that will plot the result for the user.

Figure 3. AR Viewer Module Architecture

In Figure 4, the graph represents a final result. The rectangles, directly linked with central node, describe the result query by the search engine. The more extern rectangles show the suggestion (a). If the user requests the query "player", then the system returns some files. However, if the user selects the file "JMFPlayer.java" to download, the system suggests that the file "AudioPlayerAdapter.java" should be downloaded too (1). In the right of the Figure 4, two concepts of Visual Data Mining are used and the both were based on the tool called GraphMiner[20] that provides a visualization of the fields of each node on demand (a). In addition, some controllers are provided to improve the interactivity and visualization such as: graph rotation, edge size adjust and nodes quantity (c).

The **AR Viewer** uses other important concept that tries to help the user. This concept is related with the user interface, and uses *Visual Data Mining* which is a visual channel to become the process more intuitive for humans [21]. Another way to present the suggestions, using a non intrusive mechanism, is the simple text format. In Figure 5, there is an example using **B.A.R.T Search Engine** where the **suggestion (1)** is represented in a text line. This alternative solution cannot represent rules with two or more antecedents, because this interface cannot show the interconnection among the results.

Figure 4. Example of resulted graph

Figure 5. Example of suggestion on simple text format

VI. EVALUATION

In this section, we detail the achieved results of the experimental study performed in order to validate our tool. The experiment was focused on the quality analysis of the generated rules to assure the suggestions quality.

Previous search and retrieval experiments of the RiSE group such as [10] and [26] have definitely collaborated to this one. In addition to the important points such as structure and the required issues to be analyzed, the other points to be improved were also observed and tried to be overcome in this instantiation. Both experiments [10], [26] shared a common dataset (or test database) what, in

fact, this is positive because they can show the evaluation of the search mechanisms with more confidence. On the other hand, over than 1000 Java files were utilized without total knowledge of the dataset. By this reason, for this experiment the number of classes in the dataset was reduced to 100 with the condition that each class should be properly analyzed, classified and catalogued in standard dataset for future reuse [5], [18].

The experiment followed the guidelines defined in [28], which can be divided into the following main activities: **Definition**, where the experiment is defined in terms of problem, objective and goals. **Planning**, where the design of the experiment is determined, the instrumentation is considered and the threats to the experiment are evaluated. **Operation**, in which measurements are collected, analyzed and evaluated in the **analysis and interpretation**. Finally, the results are presented and packaged in the **presentation and package**.

Each activity is detailed next.

A. Definition

Wohlin et al. [28] follow the GQM (Goal-Question-Metric) template [2] for the goal definition. The GQM is based upon the assumption that for an organization to measure in a purposeful way, it must first specify the goals for itself and its projects, then it must trace those goals to the data that are intended to define those goals operationally, and finally provide a framework for interpreting the data with respect to the stated goals.

1) *Goal*: The goal of this experiment was to analyze the use of data mining techniques to improve search engines. In this way, it was analyzed the quality of the

generated rules for purpose of measure the effectiveness of our solution to generate suggestions. This evaluation has the view point based on a software developer that using a search engine to reuse assets. This experiment ran in the context of a in-house process in order to evolve the structure to make an industrial evaluation. For this phase, no subjects were used and the dataset was generated based on an heuristic to control the result and next evaluates its benefits.

2) *Questions*: In order to achieve this goal, several questions were defined:

- Q1.** Is it possible generate suggestions using association rule approach?
- Q2.** When the amount of records increase the quality of generated rules increases too?
- Q3.** Does the time to process the rules and transactions depends on the used algorithm?
- Q4.** Does the response time of the search engine increases when our approach is enabled?

3) *Metrics*: The quality evaluation of the generated rules is based in objective measures explained in Section V-A. The values are based using an empirical analysis about our dataset. The support value analyzes the coverage of an specific rule. Thus, the value of this metric must be close of 3% because the content of dataset is so heterogeneous. The Confidence will analyze the rule strongness, in this way, the rules must be a high value and in our case, we are using 50%.

Support. The generated rules must be generated with minimum support equals to 3%.

Confidence. The generated rules must be generated with minimum confidence equals to 50% .

Response Time. The time to return the set of results for the user.

Extraction Time. The time to extract the transactions and rules.

Correctness Percentage. The correctness is evaluated using the sets generated before the datasets generation.

B. Planning

The experiment was executed off-line (not in an industrial software development) with no subjects.

Since the focus of this evaluation is the quality of generated rules, the instruments used to validate were the data extracted during the experiment, which was composed of generated data groups based on 100 java files (as explained before). Over this base were generated groups with 2 or 3 files and 6 sceneries were designed.

For this experiment, two types of data were analyzed. The first one is a Homogeneous data, in this case, it was used a regular time between the set of downloads (10 minutes).The second one is a heterogeneous case, which was generated with a random time range (from 1 until 10 minutes).

Based on these two base types, 6 datasets were generated using the java files cited before:

- Scenery 1: Homogeneous and Small (2500 records)

- Scenery 2: Homogeneous and Medium (25000 records)
- Scenery 3: Homogeneous and Large (50000 records)
- Scenery 4: Heterogeneous and Small (2500 records)
- Scenery 5: Heterogeneous and Medium (25000 records)
- Scenery 6: Heterogeneous and Large (50000 records)

1) *Variables Selection*: Experiments usually employ a set of variables that link causes and effects [22], for this experiment, independent and dependent variables were taken into account:

- Independent variables are attributes actively manipulated when comparing different situations.
- Dependent variables are the outputs whose values are expected to change according to changes to the independent variables.

In the context of this experiment, the independent variables refer to the dataset size and type (homogeneous or heterogeneous), and which algorithm was used to extract the transaction and the rules.

The dependent variables are the time to process the transactions and rules set, percentage of and correctness for rules and transactions extracted and the response time.

This study evaluation was based on the following hypotheses:

2) *Null hypothesis*: It is the hypothesis that the experimenter wants to reject with as high significance as possible.

1) Transaction Identification:

- $H_{0a} : \mu$ The percentage of correct transaction identification is lower than 85%.
- $H_{0b} : \mu$ The percentage of execution time for the algorithm *Kmean*[25] is higher than the old algorithm by *Time Window*.
- $H_{0c} : \mu$ The percentage of correctness for the algorithm *Kmeans* is lower than the old algorithm by *Time Window*.

2) Rule Extraction:

- $H_{0d} : \mu$ The percentage of correct suggestions is lower than 90%.
- $H_{0e} : \mu$ No rules are generated with confidence higher than 50% and support higher than 3%.

3) Execution Environment:

- $H_{0f} : \mu$ The execution time with the suggestion mechanism enabled is 10% higher than without suggestions.

3) *Validity Threats*: In order to provide a set of valid results, we defined some threats to the validity of the experiment. Internal validity depended upon the number of registered downloads and searches; we assumed that the quantity and sceneries of the registreed downloads provided a good internal validity. However, we identified a conclusion validity related with a dataset as low statistical power, because our dataset was generated and it is not a real dataset.

C. Operation

The experimental study was conducted in-house, during October 1, 2008 - October 31, 2008, at Federal University of Pernambuco, Brazil.

For this experiment was used one computer with 2GB of RAM, Core2Duo Processor(1.73Ghz) and 160GB of Hard disk.

As explained in Section VI-B, six sceneries were specified and populated using as variation the algorithms for transaction identification and association rule extraction.

In this case were used 2500, 25.000 and 50.000 records of downloads for homogeneous and heterogeneous data. In order to process these data, was used the following algorithms:

- Transaction Identification: *K-means* and *Time Window*
- Association Rule Extraction: *FP-Growth* and *Apriori*

D. Analysis and Interpretation

This section describes the analysis and interpretation of the experimental study.

1) *Percentage of transaction identification correctness*: The average of the transaction identification correctness was 85,97%. Thus, this rejects the null hypothesis H_{0a} as shows Figure 6. This is the evidence that the groups identified by the algorithms of transaction identification generates a good data to extract association rules.

Figure 6. Percentage of Transaction Identification Correctness

2) *Percentage of Execution Time Comparison between Kmeans and Time Window*: Analyzing the results of the transaction identification execution time was possible to identify that the adaptation of Kmeans algorithm, proposed in this work, was in average 40% more faster than Time Window[4]. Thus, this rejects the null hypothesis H_{0b} as shows Figure 7.

3) *Percentage of correctness between Kmeans and Time Window*: The results of the transaction identifications show that did not occur variation among the results. Thus, this rejects the null hypothesis H_{0c} as shows Table I.

Figure 7. Time Window and Kmeans Time Execution Comparison (seconds)

Table I
TIME WINDOW AND KMEANS CORRECTNESS COMPARISON

TimeWindow	K Means
85,82%	86,12%

4) *Percentage of Correct Suggestions*: The generated rules presents 100% of correctness. However, it rejects the null hypothesis H_{0d} , but is necessary more tests with a real dataset.

5) *Quantity of Generated Rules*: The results of the rule extraction show that the proposed minimum support and confidence in average generates 31 rules. Thus, this rejects the null hypothesis H_{0e} as shows Figure 8.

Figure 8. Quantity of Generated Rules per Type

6) *Search Execution Time*: Analyzing the results of the execution time during the search was possible to identify that the response time between the search engine with and without suggestion did not present significant difference. For this test were used 13 queries, the average of the search with suggestion was 0.020 seconds and without suggestions was 0.019. Thus, the difference was 5% and this rejects the null hypothesis H_{0e} as shows the Figure 9.

Figure 9. Comparison of Search Execution Time

E. Lessons Learned

After concluding the experimental study, some aspects should be considered in order to repeat the experiment, since they were seen as limitations of the first execution.

Real World Data. Instead of uses a controlled dataset, it is necessary obtains a real dataset in order to validate the rules. In our case, the transaction identification phase generated a set with 89 % correctness and it generated rules with 100% of correctness. However, in a real environment with millions of log lines, the results would be more precise.

Subjects. In order to evaluate the impact of our tool for the users, the next experiment must have software engineers to evaluate the usability.

F. Improvements

The correctness of our proposed transaction identification algorithm needs improvement in its correctness. The current version is so closer of Time Window Proposal, but the time to extract in the Kmeans case was 40% lower.

VII. CURRENT STAGE

Nowadays, the first version is under test phase in an industrial context, at C.E.S.A.R.¹. The B.A.R.T Search Engine is being used in this test phase through the interfaces presented in the Section V. However, the benefits of an alternative interface based on graphs, the **AR Viewer**, will be measured using the user feedback, the amount of use of each interface type and downloads. Both interfaces are available for all user testers.

VIII. CONCLUDING REMARKS AND FUTURE WORK

Software reuse is a critical issue for companies to obtain the benefits of costs and development time reduction. However, for software reuse to be disseminated in companies, sometimes, the use of tools as search engines is a viable solution. This work proposed a way to improve the search engines avoiding unnecessary queries. For this,

¹Currently, this company is CMMi level 3 and has about 700 employees, <http://www.cesar.org.br>.

the *log files* were monitored to extract a historic pattern allowing to preview the searches. However these data from log files are so raw and huge. In order to solve it, we use data mining techniques, called association rules and clustering, to aid the knowledge extraction since the data cleaning until the result evaluation.

The extracted knowledge was presented to user using an alternative way based on graphs to become the suggestions more intuitive. However, the normal vision as plain text was kept to reduce the introduction for components suggestion.

In addition, an initial experiment was performed and it shows that there are some evidences that our tool can aid the search engines. The most important evidence is the performance of transaction identification, because the use of Kmeans to group the transactions resulted in 40% of time reduction. However, the correctness of this identification can be improved. In order to conclude the effectiveness in an industrial environment, a new experiment with a real and huge dataset must be executed.

We will integrate the **AR Viewer** on the developer environment. In this way, we will use a plug-in to Eclipse and Visual Studio to become the solution less intrusive since this environment very popular in companies.

ACKNOWLEDGMENT

The authors would like to thank the Recife Center for Advanced Studies and Systems (C.E.S.A.R.), Brazilian innovation institute that contributed with the evaluation of the mechanism.

This work is sponsored by Brazilian Agency (CNPq process number: 475743/2007-5)

REFERENCES

- [1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *VLDB '94: Proceedings of the 20th International Conference on Very Large Data Bases*, pages 487–499, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.
- [2] V. R. Basili, G. Caldiera, and H. D. Rombach. The goal question metric approach. In *Encyclopedia of Software Engineering*. Wiley, 1994.
- [3] S. Brin, R. Motwani, J. D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. In *SIGMOD '97: Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, pages 255–264, New York, NY, USA, 1997. ACM.
- [4] R. Cooley, B. Mobasher, and J. Srivastava. Data preparation for mining world wide web browsing patterns. *Knowl. Inf. Syst.*, 1(1):5–32, 1999.
- [5] F. A. Durão, T. A. Vanderlei, V. C. Garcia, E. S. Almeida, and S. R. de L. Meira. Applying a semantic layer in a source code search tool. In *SAC '08: Proceedings of the 23rd ACM symposium on Applied computing*, Fortaleza, CE, Brazil, 2007. ACM.

- [6] B. M. Fonseca, P. B. Golgher, E. S. de Moura, and N. Ziviani. Using association rules to discover search engines related queries. In *LA-WEB '03: Proceedings of the First Conference on Latin American Web Congress*, page 66, Washington, DC, USA, 2003. IEEE Computer Society.
- [7] W. B. Frakes, R. P. Díaz, and C. J. Fox. Dare: Domain analysis and reuse environment. *Annals of Software Engineering*, 5:125–141, 1998.
- [8] W. B. Frakes and S. Isoda. Success factors of systematic reuse. *IEEE Softw.*, 11(5):14–19, 1994.
- [9] V. C. Garcia, E. S. de Almeida, L. B. Lisboa, A. C. Martins, S. R. L. Meira, D. Lucredio, and R. P. de M. Fortes. Toward a code search engine based on the state-of-art and practice. In *APSEC '06: Proceedings of the XIII Asia Pacific Software Engineering Conference*, pages 61–70, Washington, DC, USA, 2006. IEEE Computer Society.
- [10] V. C. Garcia, D. Lucrédio, F. A. Durão, E. C. R. Santos, E. S. de Almeida, R. P. de Mattos Fortes, and S. R. de Lemos Meira. From specification to experimentation: A software component search engine architecture. In *CBSE '06: Proceedings of the 9th International Symposium on Component-Based Software Engineering*, pages 82–97, 2006.
- [11] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. *SIGMOD Rec.*, 29(2):1–12, 2000.
- [12] S. Henninger. An evolutionary approach to constructing effective software reuse repositories. *ACM Trans. Softw. Eng. Methodol.*, 6(2):111–140, 1997.
- [13] R. Holmes and G. C. Murphy. Using structural context to recommend source code examples. In *ICSE '05: Proceedings of the 27th international conference on Software engineering*, pages 117–125, New York, NY, USA, 2005. ACM.
- [14] C. W. Krueger. Software reuse. *ACM Comput. Surv.*, 24(2):131–183, 1992.
- [15] D. Lucredio, A. F. do Prado, and E. S. de Almeida. A survey on software components search and retrieval. In *EUROMICRO '04: Proceedings of the 30th EUROMICRO Conference (EUROMICRO'04)*, pages 152–159, Washington, DC, USA, 2004. IEEE Computer Society.
- [16] A. C. Martins, V. C. Garcia, E. S. Almeida, and S. R. L. Meira. Enhancing components search in a reuse environment using discovered knowledge techniques. In *2nd Brazilian Symposium on Software Components, Architectures, and Reuse (SBCARS)*, Porto Alegre, Brazil, 2008.
- [17] J. C. C. P. Mascena, S. R. de Lemos Meira, E. S. de Almeida, and V. C. Garcia. Towards an effective integrated reuse environment. In *GPCE '06: Proceedings of the 5th international conference on Generative programming and component engineering*, pages 95–100, New York, NY, USA, 2006. ACM.
- [18] R. C. Mendes. A search and retrieval of reusable source code using faceted classification approach. Master's thesis, Federal University of Pernambuco, Brazil, August 2008.
- [19] H. Mili, A. Mili, S. Yacoub, and E. Addy. *Reuse-based software engineering: techniques, organization, and controls*. Wiley-Interscience, New York, NY, USA, 2001.
- [20] M. C. M. Neto, M. Mendonça, and C. A. S. Santos. Graphminer: Uma ferramenta de mineração visual de dados em bases relacionais. In *Jornadas Iberoamericanas de Ingeniería del Software e Ingeniería del Conocimiento*, pages 305–316, Madrid, Spain, 2004.
- [21] O. Niggemann. *Visual Data Mining of Graph-Based Data*. PhD thesis, University of Paderborn, 2001.
- [22] P. A. Perry, D. and J. L. Votta. Scalable parallel data mining for association rules. In *SIGMOD '97: Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, pages 277–288, New York, NY, USA, 1997. ACM.
- [23] R. Prieto-Diaz and P. Freeman. Classifying software for reusability. *IEEE Softw.*, 4(1):6–16, 1987.
- [24] E. C. R. Santos, F. A. Durão, A. C. Martins, R. Mendes, B. J. M. M. Cassio A. Melo, V. C. Garcia, E. S. de Almeida, and S. R. L. Meira. Towards an effective context-aware proactive asset search and retrieval tool. In *WDBC '06: Proceedings of the 6th Workshop on Component-Based Development*, pages 105–112, December 1996.
- [25] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.
- [26] T. A. Vanderlei, a. Frederico A. Dur A. C. Martins, V. C. Garcia, E. S. Almeida, and S. R. de L. Meira. A cooperative classification mechanism for search and retrieval software components. In *SAC '07: Proceedings of the 22nd ACM symposium on Applied computing*, pages 866–871, Seoul, Korea, 2007. ACM.
- [27] M. Vieira and D. Richardson. Analyzing dependencies in large component-based systems. In *ASE '02: Proceedings of the 17th IEEE international conference on Automated software engineering*, page 241, Washington, DC, USA, 2002. IEEE Computer Society.
- [28] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in software engineering: an introduction*. Kluwer Academic Publishers, Norwell, MA, USA, 2000.
- [29] Y. Ye and G. Fischer. Supporting reuse by delivering task-relevant and personalized information. In *ICSE '02: Proceedings of the 24th International Conference on Software Engineering*, pages 513–523, New York, NY, USA, 2002. ACM.