Enhancing Components Search in a Reuse Environment Using Discovered Knowledge Techniques

Alexandre C. Martins[1,2], **Vinicius C. Garcia**[1,2], **Eduardo S. Almeida**[2], **Silvio R. L. Meira**[1,2]

[1]Federal University of Pernambuco (UFPE) and
Recife – PE – Brazil

[2]Recife Center for Advanced Studies and Systems (CESAR)
Recife – PE – Brazil

{acm3,vcg}@cin.ufpe.br, esa@rise.com.br, srlm@cin.ufpe.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 aids search engines applying data mining techniques on log mechanism to extract knowledge about the historic data. In order to evaluate the results, a preliminary experimentation is also discussed.*

1. Introduction

Software reuse is the process of creating software systems from existing software rather than building them from scratch [Krueger 1992]. 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 [Frakes and Isoda 1994]. 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 [Prieto-Diaz and Freeman 1987], *"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. [Garcia et al. 2006a], to aid the reuse process. However, some research [Mili et al. 2001, Lucredio et al. 2004] 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 [Henninger 1997]. Several solutions [Mili et al. 2001, Lucredio et al. 2004] have been proposed to attenuate it. However, the user still needs to build queries every time, and it is hard as also highlighted by [Ye and Fischer 2002].

Another problem is associated with the dependencies among the components [Vieira and Richardson 2002], since sometimes, the returned assets do not solve the entire problem, and the developer has to find other 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,

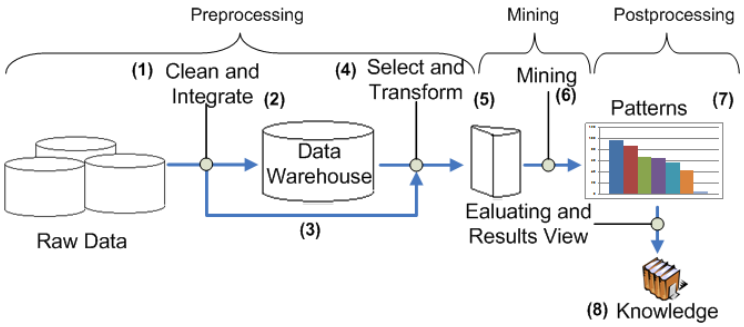

Figure 1. KDD steps

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)** [Garcia et al. 2006b] 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 [Mascena et al. 2006], context [Santos et al. 1996], folksonomy [Vanderlei et al. 2007] and semantic [Durão et al. 2007]. 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.

The remainder of this paper is organized as follows. Section 2 presents the background on data mining and its application on this work. Section 3 and 4 discusses the main requirements of B.A.R.T Search Engine and the proposed approach. Section 5 presents the architecture and implementation. Section 6 shows a preliminary experimentation and Section 7 the current stage. Section 8 discusses the related work, and, finally, Section 9 presents the concluding remarks.

2. Applying Data Mining Over Log Files

The *Knowledge Discovery in Databases* (KDD) is a process that aims at discovering useful information from large collection of data. The discovered knowledge can be rules describing data properties, frequently occurring patterns, clustering of objects in the database, etc. Basically, KDD is divided in three phases: preprocessing, mining and post-processing as shown in Figure 1.

In this Section 2, these phases will be explained and contextualized in our project. In our context, the KDD is applied over log files in order to guide the data preparation until the data evaluation. Figure 2 represents a log file that contains information such as date, hour, user and asset downloaded. For each operation, a line is recorded corresponding to the user actions.

2.1. Preprocessing

As shown in Figure 1, the first phase prepares the raw data to be mined. This phase is extremely important because the inconsistencies can change the **knowledge extracted (8)**. In this way, this phase is divided in five steps. The first one is to **clean (1)** the raw

Figure 2. A log line example

data to remove the ”trash”. The ”trash” lines are the ones that do not match the format in Figure 2, for instance, lines 2 and 5 of Figure 3. This is necessary because during the inserts transaction, some data can be damaged for instance by system crash. Next, it is necessary to **integrate (2)** all files to optimize the knowledge extract process. In this moment, the results can be used to compose a **data warehouse (3)**, such as a historic database copy. Sometimes, the date is divided in many parts to improve their compression or organization. The next steps are to **select (4)** and **transform (5)** the information in groups of transactions that represent each user during a time or activity. The process of association rules extraction needs to have these groups to find repeated events or patterns [Tan et al. 2005]. Section 2.1.1 presents more detail about that.

Figure 3. A raw log file example

2.1.1. Identifying the Transactions from Log Mechanisms

The association rules extraction is based on the pattern detection in groups of records called *itemsets* [Agrawal and Srikant 1994]. However, when log files are used, these groups are not perfectly separated. The classic situation of association rules is the Market Basket Analysis [Tan et al. 2005] that is associated with the products' organization in a supermarket. In this case, the transactions are defined by the consumer's ticket. Thus, the products purchased are perfectly described and separated for each consumers. Although in a log file the records are not sorted, it is necessary to separate its lines in transaction sets where each transaction will contain one or more log lines. Figure 3 and 4 show respectively a log example generated by the **B.A.R.T Search Engine** and the identified transactions.

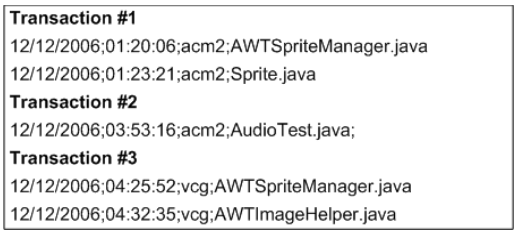

Figure 4. A log file example cleaned and divided in transactions

The transaction identification is based on time window [Cooley et al. 1999]. However, we do not use a fixed time window. Our approach uses the cluster algorithm k-Means [Tan et al. 2005]. The main task is to separate the log lines by user and after applying the k-means to group these lines. Figure 5 shows an example of downloads performed by some user in the timeline. In this case, these downloads are executed in **three periods (1)**, which correspond to when the transactions happened. However, the cluster algorithm uses centroids to make the comparisons and each centroid is transformed in a cluster. In our problem, the centroid amount is based on the time window of *10 minutes* with no download **(2)**. This number was based on an empirical analysis of our base. During the iterations cluster algorithm, the centroid positions are recalculated based on the Squared Sum Error (SSE) [Tan et al. 2005].

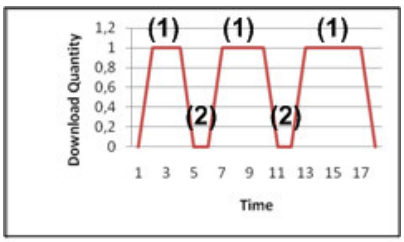

Figure 5. The mapping of the hypothetic user downloads quantity

The result, as shown in Figure 6, is a set of **clusters (1)** and each **centroid (2)** that represents the related downloads and it will be the input to the association rule algorithms. Figure 6 shows the result represented by clusters.

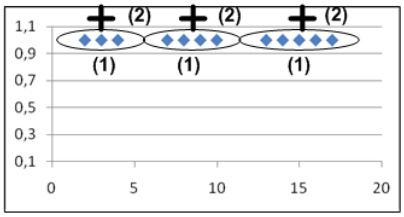

Figure 6. Result of Cluster Analysis to Transaction Identification

2.2. Mining

In the Mining phase, Figure 1, step 6, some algorithms can be used to extract the knowledge such as clustering, classification, association rules, among others [Tan et al. 2005]. However, in our case, we are interested in extracting the relations among the stored components. Thus, we used association rules algorithms [Agrawal and Srikant 1994] to extract a set of implications. Formally, let **I=**$\{i_1, i_2, \cdots, i_n\}$ be a set of literals, called items. An association rule is an implication of the form $(\bigwedge_x \in X^x) \Rightarrow (\bigwedge_y \in Y^y)$, which we write more compactly as $X \Rightarrow Y$, where $X \subset I, Y \subset I$, and $X \cap Y = \emptyset$. These rules represent the patterns (Figure 1, step 7) and each rule means that the occurrence of ITEM X suggests the ITEM Y occurrence. In our approach, the rules will indicate the dependency among the components, for instance, `"FileUtil.java ⇒ File.java"`. Thus, it is possible to suggest the `"File.java"` every time `"FileUtil.java"` is returned by some query.

2.2.1. Association Rules Algorithms

In the literature, several algorithms were proposed to extract association rules. The **Apriori** [Agrawal and Srikant 1994] is the base for several of these algorithms. However, each algorithm is focused on a specific quality characteristic. For instance, the **Partitioning** [Savasere et al. 1995] algorithm reduces the database scans. On the other hand, if the performance is the focus, the parallel algorithms such as **IDD** (Intelligent Data Distribution) [Han et al. 1997] can be used.

The analysis of association rules algorithm shows that a perfect algorithm does not exist. However, the algorithm used can be updated for each situation. Thus, our solution allows the extension for other association rules implementations.

2.3. Post-processing

The mined rules must be evaluated because the quantity can be too large ($\approx 10^{79}$) [Tan et al. 2005]. For this, there are several measures to evaluate the quality of each generated rule. These measures can be classified in *subjective* or *objective*. The subjective measures are related with external questions, for instance, business decisions. The objective measures are related with technical questions and event occurrences such as access, searches, record stored, and etc. In this work, we use two objective measures: *confidence* and *support* [Sheikh et al. 2004]. This choice was based on the information type that the search engines can provide using log files.

a. Confidence (X⇒Y)
The confidence evaluates the relation level among the rule components (X and Y). Thus, it is possible to see that: if X occurs then Y will occur too or if this happened just as a coincidence.

$$conf = \frac{X \cup Y}{X}$$

b. Support (X⇒Y)
The support measure represents the rule occurrence rate in all database and it is represented by the union of X and Y occurrence divided by the total of occurrences in database (n). Thus, if a rule occurs few turns then it is not a pattern.

$$sup = \frac{X \cup Y}{n}$$

2.4. 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 ⇒ FileD.java`
- Successors Quantity: The suggestions must be based in rules that have only one sucessor.
  Ex.: `FileA.java ⇒ FileD.java, FileF.java`

3. B.A.R.T Search Engine

According to the idea that reuse can be performed in a systematic way [Frakes et al. 1998], 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 [Garcia et al. 2006b].

The *B.A.R.T* is based on 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 [Garcia et al. 2006b]. 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.

4. The Main Requirements

Once presented the background on data mining and the KDD process, we will discuss how these ideas are being used in our tool. The requirements proposed for this work are based on studies about existing solutions in the literature [Lucredio et al. 2004], systematic analysis of reuse tools that aid the software development [Garcia et al. 2006a], and our interaction with the industry in projects in this direction. Thus, the main specified requirements are:

1. **Suggest Associated Asset**. This approach 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* [Agrawal and Srikant 1994], *Dynamic Itemset Counting (DIC)* [Brin et al. 1997], and *FP-Growth* [Han et al. 2000].
3. **Parser from Log Mechanisms Artifacts**. The log mechanisms accepted by this approach are log files. The knowledge extraction process uses these data to extract the rules.
4. **Provide a Visual Interface**. The approach 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 [Niggemann 2001].

5. Architecture Specification and Out line of Implementation

The 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. Figure **??** shows the interaction among the architectural modules.

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 response are received by the second module: (5) **AR Viewer**. The viewer renders the results as a graph to the user.

The modules integrate an environment whose responsibility is to provide a better way to locate assets. In the Figure 7, the complete flow is showed. The first step is the query request by 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.

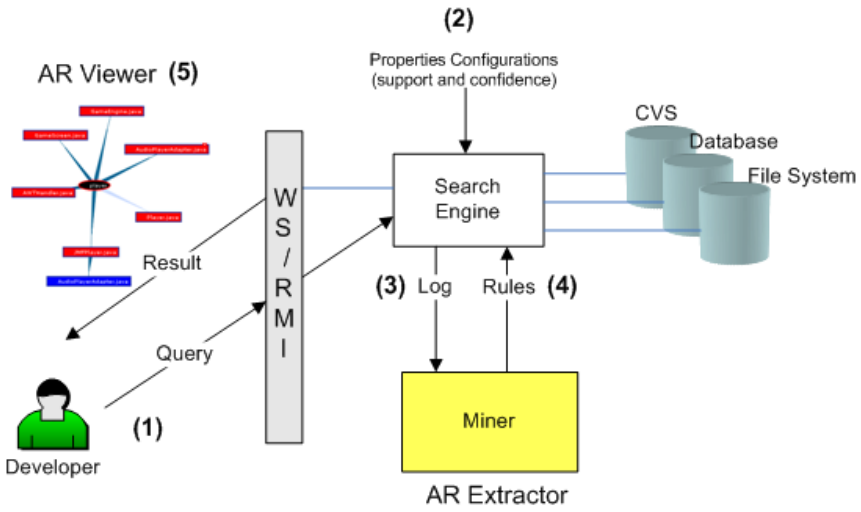

Figure 7. Architecture and its interaction with the user

An important issue is that it must be integrated with any search engine that uses a log file. After this, as shown in Section 4, the algorithm can be changed, thus, alternative implementations can be used.

5.1. Association Rules Extractor

The **AR Extractor** is divided in three sub modules: *XML Descriptor Extractor*, *Algorithm Manager* and *XML Generator*, as seen in Figure 8. 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 informatio collected in a time window.

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; some characteristics must be analyzed, such as performance, CPU process, memory usage and return time. The first version uses the *FP-Growth* [Han et al. 2000], which is a modern implementation for an association rules extraction. This solution uses a new tree called *FP-Tree* that provides a reduction on the database scans, shrinks the number of candidates

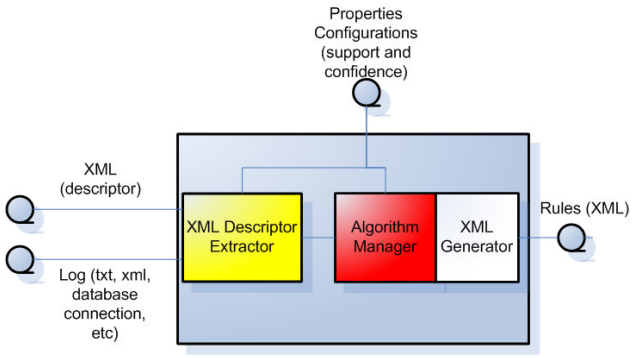

Figure 8. Intern Architecture of AR Extractor Module

and facilitates support counting of candidates [Han et al. 2000]. During the evaluation, some rules are accepted and the others are left behind. This step is important to improve the final rules quality. In Figure 9, two rules are presented, the **Rule #111** was accepted as a suggestion because the values support and confidence represents that it is a pattern on the historic data. However, the **Rule #112** was not accepted because the **0.1%** of support shows that the occurrence of "`IAudioPlayer.java ⇒ GameEngine.java`" is a coincidence. Thus, only the best rules are codified by XML Generator.

Rule #111: ***(accepted)***
JMFPlayer.java ® AudioPlayerAdapter.java
- Confidence: 90 % ↑
- Support: 5 % ↑

Rule #112: ***(rejected)***
IAudioPlayer.java ® GameEngine.java
- Confidence: 70 % ≈
- Support: 0.1% ↓

Figure 9. Evaluation of the Rules Mined

5.2. Association Rule Viewer

The **AR Viewer** is detailed in Figure 10 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 to the user.

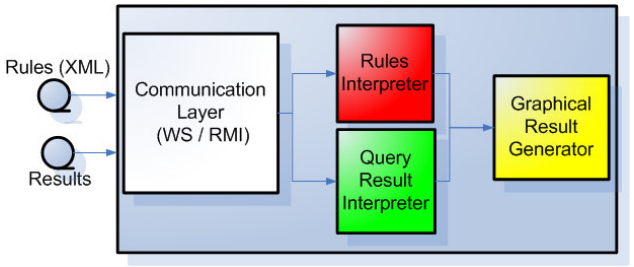

Figure 10. Architecture of AR Viewer Module

In Figure 11, 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. If the user requests the query ”player”, then the system returns

the 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)**. This suggestion is generated from the **Rule #111** showed in the Figure 9.

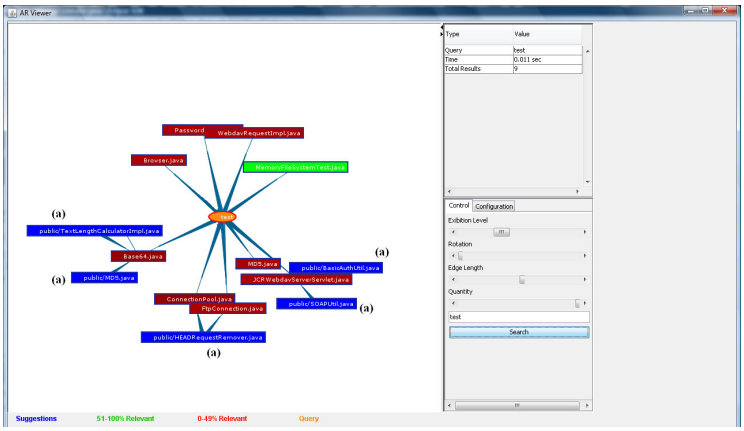

Figure 11. Example of resulted graph

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 [Niggemann 2001]. Another way to present the suggestions, using a non intrusive mechanism, is the simple text format. In Figure 12, 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 between the results.

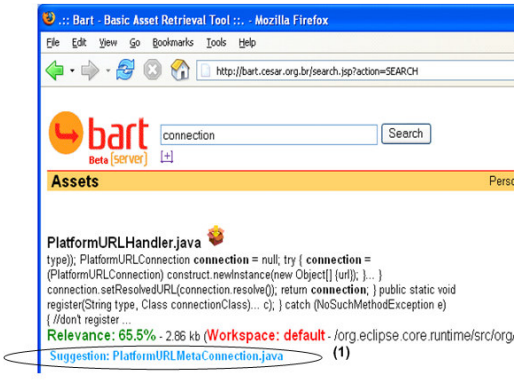

Figure 12. Example of suggestion on simple text format

6. Experimentation

In this section, we present and discuss a preliminary experiment for the data mining algorithms and the improvements in the search engine. The log mechanism selected was the log file model and it was based on our test repository. This repository is composed by *100* classes and it is used for information retrieval tests in our tools. The log file used has *5.000* lines and represents the executed downloads over the source code.

In order to formalize the experiment some hypotheses were raised. The null hypotheses (H_0), i.e. the hypotheses intended to be rejected will indicate if the our approach does not have contributed for quality of suggestions. Therefore, two types of hypotheses were defined in order to organize the experiment. The first hypotheses group is related with data mining aspects. The second one is related with the gain of the new visual interface.

Data Mining hypotheses

- H_{0a} : μ The percetagem of correct suggestions is lower than 70%.
- H_{0b} : μ No rules are generated with confidence higher than 70% and support higher than 4%.

Visual Interface hypotheses

- H_{0c} : μ The amount of rules that are not supported by the simple text interface is lower than 50%.
- H_{0d} : μ The visual interface decreases the search engine response time in 30%.

The first step to execute the experiment was to clean the files and separate the lines by users to identify the transactions. In this phase **3.308 transactions** were identified from five users. Table 1 shows the quantity of transactions identified by user.

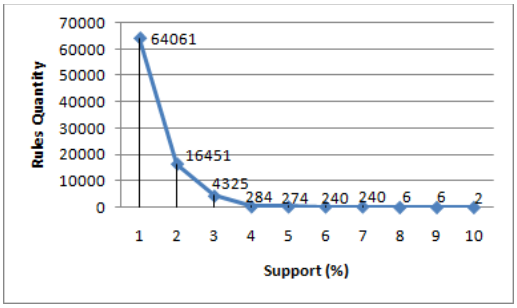

Figure 13. Comparison between Rules Quantity and Support

Table 1. Transactions Identified by User

$User$	$TransactionsIdentified$
#1	685
#2	617
#3	709
#4	576
#5	721

In the second step, we test the mining algorithms to extract the rules. Figure 13 shows the comparison between support and rule quantity using 70% of confidence. The top five rules are showed in Table 2. A **low support**, as explained in Section 2.3, suggests more rules, however, the quality decreases due to their redundancy. However, a high support restricts the rules quantity and reduces the aid rate.

The generated rules using 4% of Support and 70% of confidence were evaluated, because the support with 3% generates a lot of rules that's affect the quality. Table 3

Table 2. Top 5 extracted rules

Rule	*Antecedent*	*Sucessor*	*Support(%)*	*Confidence(%)*
1	$JeevesServlet \wedge HttpUtils$	WebdavRequestImpl	4	100
2	$WebdavRequestImpl \wedge HttpServiceRequest$	JeevesServlet	4	100
3	$WebdavRequestImpl \wedge JCRURLConnection$	JeevesServlet	4	100
4	$SimpleListener \wedge FramerImpl$	JdbcConnectionPoolConnection	4	99.55
5	$DatabasePersistenceManager \wedge JdbcConnectionPoolConnection$	FramerImpl	4	99.55

Table 3. Percentage of correct suggestions

$Total$	$Correct$	$Wrong$
195	88,08%	11,92%

shows the rules quality analysis. Thus, the null hypothesis H_{0a} was reject, because *195* rules were generated.

The preliminary evaluation shows that the final results can be improved, because works like [Fonseca et al. 2003] obtained 93.45% of correct suggestions to queries recommendation. However, the results show that the historic data aids the results in 88% of cases and it rejects the null hypothesis H_{0b}. However, we need to experiment the other transaction identification process and algorithms in a larger test base.

The Visual Interface hypotheses (H_{0c} and H_{0d}) were reject because **86,62%** of the generated rules had more than one antecedents. This rule type cannot be showed in a simple text interface with an intuitive way, because it needs the representation of relations among the assets. Hence, once that the rules are generated **in batch**, the suggestion process does not interview in the final response time.

7. 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 5. However, the gain 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 to all user testers.

8. Related Work

In the literature, some works propose different tools to aid the information search and retrieval. CodeBroker [Ye and Fischer 2002] presents a tool that finds information about components that are relevant for activities realized in the same moment. According to [Ye and Fischer 2002], Ad-hoc evaluations presented that this type of strategy is effective to promoter reuse. It allows the exploration from a new style of collaboration between men and machines. Based on the same concepts, Strathcona [Holmes and Murphy 2005] 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 in six different heuristics.

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

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

did, with any search engine. Other related work use the association rules in Web Mining [Cooley et al. 1997] for sites recommendations. However, these solutions are not used to suggest assets, only web pages.

9. Concluding Remarks and Future Works

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 proposes a way to improve the search engines avoiding unnecessary queries. For this, the *log files* were monitored to extract a historic pattern. This extraction allows previewing the searches. However this data from log files are so raw and huge. In order to solve it, we use data mining technique, 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 to components suggestion. Some tests with software developer teams were executed in order to measure the improvements that this new interface can provide.

The future version 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 is being most used in companies.

10. Acknowledgments

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

[Agrawal and Srikant 1994] Agrawal, R. and Srikant, R. (1994). 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. Morgan Kaufmann Publishers Inc. 2.1.1, 2.2, 2.2.1, 2

[Brin et al. 1997] Brin, S., Motwani, R., Ullman, J. D., and Tsur, S. (1997). 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. ACM. 2

[Cooley et al. 1999] Cooley, R., Mobasher, B., and Srivastava, J. (1999). Data preparation for mining world wide web browsing patterns. *Knowl. Inf. Syst.*, 1(1):5–32. 2.1.1

[Cooley et al. 1997] Cooley, R., Srivastava, J., and Mobasher, B. (1997). Web mining: Information and pattern discovery on the world wide web. In *ICTAI '97: Proceedings of the 9th International Conference on Tools with Artificial Intelligence*, page 558, Washington, DC, USA. IEEE Computer Society. 8

[Durão et al. 2007] Durão, F. A., Vanderlei, T. A., Garcia, V. C., Almeida, E. S., and de L. Meira, S. R. (2007). 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. ACM. 1

[Fonseca et al. 2003] Fonseca, B. M., Golgher, P. B., de Moura, E. S., and Ziviani, N. (2003). 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. IEEE Computer Society. 6

[Frakes et al. 1998] Frakes, W. B., Díaz, R. P., and Fox, C. J. (1998). Dare: Domain analysis and reuse environment. *Annals of Software Engineering*, 5:125–141. 3

[Frakes and Isoda 1994] Frakes, W. B. and Isoda, S. (1994). Success factors of systematic reuse. *IEEE Softw.*, 11(5):14–19. 1

[Garcia et al. 2006a] Garcia, V. C., de Almeida, E. S., Lisboa, L. B., Martins, A. C., Meira, S. R. L., Lucredio, D., and de M. Fortes, R. P. (2006a). 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. IEEE Computer Society. 1, 4

[Garcia et al. 2006b] Garcia, V. C., Lucrédio, D., Durão, F. A., Santos, E. C. R., de Almeida, E. S., de Mattos Fortes, R. P., and de Lemos Meira, S. R. (2006b). 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. 1, 3

[Han et al. 1997] Han, E.-H., Karypis, G., and Kumar, V. (1997). 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. ACM. 2.2.1

[Han et al. 2000] Han, J., Pei, J., and Yin, Y. (2000). Mining frequent patterns without candidate generation. *SIGMOD Rec.*, 29(2):1–12. 2, 5.1

[Henninger 1997] Henninger, S. (1997). An evolutionary approach to constructing effective software reuse repositories. *ACM Trans. Softw. Eng. Methodol.*, 6(2):111–140. 1

[Holmes and Murphy 2005] Holmes, R. and Murphy, G. C. (2005). 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. ACM. 8

[Krueger 1992] Krueger, C. W. (1992). Software reuse. *ACM Comput. Surv.*, 24(2):131–183. 1

[Lucredio et al. 2004] Lucredio, D., do Prado, A. F., and de Almeida, E. S. (2004). 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. IEEE Computer Society. 1, 4

[Mascena et al. 2006] Mascena, J. C. C. P., de Lemos Meira, S. R., de Almeida, E. S., and Garcia, V. C. (2006). 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. ACM. 1

[Michail 2000] Michail, A. (2000). Data mining library reuse patterns using generalized association rules. In *ICSE '00: Proceedings of the 22nd international conference on Software engineering*, pages 167–176, New York, NY, USA. ACM.

[Mili et al. 2001] Mili, H., Mili, A., Yacoub, S., and Addy, E. (2001). *Reuse-based software engineering: techniques, organization, and controls*. Wiley-Interscience, New York, NY, USA. 1

[Niggemann 2001] Niggemann, O. (2001). *Visual Data Mining of Graph-Based Data*. PhD thesis, University of Paderborn. 4, 5.2

[Prieto-Diaz and Freeman 1987] Prieto-Diaz, R. and Freeman, P. (1987). Classifying software for reusability. *IEEE Softw.*, 4(1):6–16. 1

[Santos et al. 1996] Santos, E. C. R., Durão, F. A., Martins, A. C., Mendes, R., Cassio A. Melo, B. J. M. M., Garcia, V. C., de Almeida, E. S., and Meira, S. R. L. (1996). 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. 1

[Savasere et al. 1995] Savasere, A., Omiecinski, E., and Navathe, S. B. (1995). An efficient algorithm for mining association rules in large databases. In *VLDB '95: Proceedings of the 21th International Conference on Very Large Data Bases*, pages 432–444, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc. 2.2.1

[Sheikh et al. 2004] Sheikh, L. M., Tanveer, B., and Hamdani, S. M. A. (2004). Interesting measures for mining association rules. In *INMIC '04: Proceedings of the 8th IEEE International Multi-topic conference*, pages 641–644. 2.3

[Tan et al. 2005] Tan, P.-N., Steinbach, M., and Kumar, V. (2005). *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA. 2.1, 2.1.1, 2.1.1, 2.2, 2.3

[Vanderlei et al. 2007] Vanderlei, T. A., Frederico A. Dur a., Martins, A. C., Garcia, V. C., Almeida, E. S., and de L. Meira, S. R. (2007). 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. ACM. 1

[Vieira and Richardson 2002] Vieira, M. and Richardson, D. (2002). 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. IEEE Computer Society. 1

[Ye and Fischer 2002] Ye, Y. and Fischer, G. (2002). 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. ACM. 1, 8