RESEARCH ARTICLE OPEN ACCESS

Knowledge Discovery Applied to a Database of Errors of Systems Development

Elias Delgobo Junior*, Denise Fukumi Tsunoda**, Egon Walter Wildauer***
*(Department of Science and Information Management, Federal University of Paraná, Paraná, BR)
** (Department of Science and Information Management, Federal University of Paraná, Paraná, BR)
*** (Department of Science and Information Management, Federal University of Paraná, Paraná, BR)

ABSTRACT
This paper presents the knowledge discovery process in a database related to the development of computer systems through the Apriori algorithm. This method of data mining was succesfull in discovering of patterns of relationships between kinds of non-conformities found during the software development and relationships of noncompliance with the kinds of tasks to be performed as an association between two variables "Simple" and "Average" in more than fifty percent of the cases whit tasks labeled as "Improvement". The discovered rules may assist in the decision making by development systems managers in order to reduce non-conformities related to the development of computational systems.
Keywords Data Mining, Knowledge, Errors

I. INTRODUCTION

The rapid growth of data derived from information systems is constant.These stored data are useless without, for example, the use of information extraction methods that allow the discovery of patterns, models and previously unknown relationships. Obviously, to support the decision making process, only accurate and relevant information is valuable for managers. In this case, it is valid to apply data mining methods based on statistical algorithms and machine learning techniques to discover patterns that "may be rules, affinities, correlations, trends or forecasting models" (Turban, 2010, p. 460).

This work aims to discover relationships between the kinds of errors found during the testing phase of information systems, relating kinds of errors and levels of urgency of the tasks perfomed inside a software development company.

It also seeks to demonstrate that the pressure for delivering a task can induce the insertion of system errors by programmers, using the techniques of knowledge discovery process. The following topics will cover steps of this study, the chosen database and its attributes, the mining algorithm used and the results achieved.

II. THE PROCESS OF KNOWLEDGE DISCOVERY IN DATABASE (KDD)

According to Turban "the knowledge discovery based on computer has been used since the 60s. However, the techniques which make it possible have been expanded and improved with time" (Turban, 2010, p. 405). In the 1990s with the creation of data warehouses, the volume of stored data increased and created room for the techniques of Knowledge Discovery in Databases (KDD).

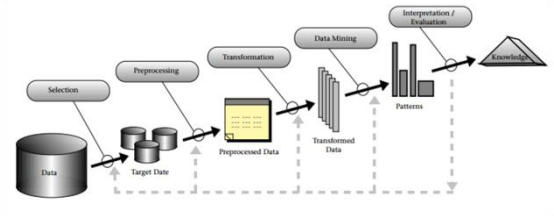


Figure 1 – KDD overview

Source: Fayyad, 1996

This concept has been used since the 1980s demonstrating that there are valuable results produced by a process ran over data to extract relevant information for decision making. ” Acording to Mainon and Rokach (2010), KDD is “an automatic, exploratory analysis and modeling of large data repositories. KDD is the organized process of identifying valid, novel, useful, and understandable patterns from large and complex data sets.”. Yet, according to the same authors, “*Data Mining* (DM) is the core of the KDD process, involving the inferring of algorithms that explore the data, develop the model and discover previously unknown patterns. The model is used for understanding phenomena from the data, analysis and prediction.”. Based on Adriaans and Zantinge (1996), the discovery process comprises:

- Selection of data: from a database, relevant data should be selected to be part of the process goals;
- Preprocessing: after being selected,data should be cleaned, in order to eliminate redundancies, inconsistencies and null values that may hinder the mining or analysis of data;

- Transformation: this step encodes data in order to facilitate the work of mining, discretizes and restructures the data, e.g. the database value AGE discriminated as {1,2, 3 ..., 27, 31, 34, ..., 91,92,97} may be grouped as {1-15, 16-30, ..., 76-90};
- Mining: "the process of extracting unknown information, while significant, from large databases to be used in business decision making." (Singh, 1998);
- Interpretation: in this phase, also known as post processing, resulting information from the mining process is analyzed and interpreted.

Fig. 1 presents a sequential overview of the KDD process, since from the aquisition of the raw data until the interpretation of these same data turned into knowledge. It´s important to notice that the phases are not necessarily required, after obtaining the data, sometimes uneeded preprocessing and/or transformation are skiped and the data mining is performed, followed by the results interpretation.

III. SOFTWARE DEVELOPMENT ERRORS DATABASE

The database chosen for the process of knowledge discovery comes from a management system used by a medium-sized software development company. Such a database holds data about non-conformities found during the compliance and quality testing of the produced software.

The company applies an adaptation of agile development methodology called Scrum, consisting basically in dividing the system development into several functional phases, called sprints. Each sprint has a cycle 2-4 weeks with planning meeting, periodic review meeting and feedback meeting. Each phase is divided into smaller parts which are called tasks. These are analyzed by specially designated teams and passed on to the developers responsible for the tasks.

When the development team completes a task, it transfers the resulting artifacts to the test teams to perform the checks in accordance with what was specified. If non-conformities are found, they are recorded in a database, and forwarded to the developers responsible for the correction.

The data used in the KDD process have been extracted from this database. Initially five attributes were extracted; however, two of these were eliminated in the selection process by presenting repeated and inconsistent values. The choosene mining algorithm changed the structure of the database becoming one of the attributes unsuitable for mining and dividing another attribute used to categorize kinds of non-conformities in five new ones, according to the non-conformities registered in the database.

The class attribute presented in table TaskType is used as an identifier for the task urgency. Such an attribute has five possible values: "Improvement" (when the task is based on an existing functionality, but at the request of the customer, analyst or tester, it is submitted for improvement), "New" (when the task is to add some new functionality), "RNC (Legacy)" (RNC is a Record of non-Conformity, when the client requests some urgent change in the system, due to an error or problem that causes loss to the customer - considered as Legacy because they are old and not fixed requests), "RNC (Recent)" (when the customer requests an urgent change in the system, due to an error or problem that causes loss to the customer – considered Recent because they are new) and "Demand" (when changes are needed to make the system to conform to laws, standards, ans so on).

Changes required by standards and laws are rare but urgent, the RNC are records of non-compliance that receive attention and classification of high degree of urgency, being labeled as "New" or "Improvement".

The database presents the kinds of non-compliances divided into five categories: "Simple" (when the error found in testing is easy to fix and/or requires less than an hour of work), "Average" (when the error found in testing is not so easy to fix and/or require more than one hour of work), "Complex" (when the error found in testing somehow stoped the activity of the user on the system), "Standardization" (when the error found in testing refers to the visual standards imposed by the company) and "Text" (when the error found in testing refers to grammar, syntax or language issue).

The five listed non-conformities are classified only at the level of existence, which means, if the error indeed exists (no matter the amount of the occurrences) is considered as an "S" and otherwise the variable will be displayed with a "?".

IV. METHODOLOGY

For the purposes of this research, the process of knowledge discovery used the methodology proposed by Fayyad (1996). Data were initially selected from three different tables: one that lists the tasks to be performed, one containing the errors found in the system and another one that lists the error levels found in the system.

Fields were chosen according to the research goals. We do not select data to identify somehow the company or employees.. Then we excluded the tasks that had no errors, since they were useless for the purposes of this data mining research.

Finally, five attributes were selected: "Task_cod" (identifies the task id, the primary key), "task_priority" (identifies a priority valued from zero to twenty, in ascending order of importance), "error_level_desc" (identifies the type of error,

"Simple", "Average", "Complex", "Standardization" and "Text"), task_level_desc (description of the task level, "Improvement", "New", ...) and "error_level" (level of error, numeric attribute).

The attributes were reviewed and it was noted that it was not appropriate to use all of them, the "error_level" and "error_level_desc" attributes, even being different, have values related in almost 100% of cases. The first one was removed.

It was noticed that the "task_priority" attribute is repeated in many rows and by a system default it was always zero, which could lead to unwanted distortions in results, depending on the method chosen. This attribute has also been removed.

The Apriori algorithm was applied because the initial goal of the process is to discover relationships between error kinds according to their presence in system testing. The algorithm aims the discovering of these relationships. According to Aurelio (1999), Apriori "is responsible for discovering the set of frequent items via multiple steps in the database. Each step starts with a seed set of items and such a seed will generate potential new seed sets called candidate items set "(AURÉLIO 1999, p. 13).
According to Pang-Ning Tan:

"The Apriori algorithm uses a level approach to generate association rules where each level corresponds to the number of items that belong to the consequent of the rule. Initially, all rules of high confidence that have only one item in the consequent are extracted. These rules are then used to generate new candidate rules. For example, if {acd} → {b} and {abd} → {c} are rules of high confidence then the candidate rule {ad} → {bc} is generated by the fusion of the consequent of both rules. Suppose the confidence to {bcd} → {a} is low. All rules containing the item in its consequent including {cd} → {ab} {bd} → {ac} {bc} → {d} and {d} → {abc} may be discarded "(TAN 2005 p.417).

With the choice of mining method it was decided to change the way the data is displayed, the choice of Apriori algorithm caused the restructuring of data, which were aligned according to the errors and then grouped by task. In this case, the field "task_cod" was considered irrelevant since it never repeats and does not add any information to the mining.

The attribute was removed and "error_level_desc" was divided into columns according to different kinds of errors previously discussed. The database was constructed in such a way that each field holds the number of errors for each kind of error for each task, which means, if a certain task has two "Average" errors, three "Simple" errors, one "Complex" error and none (zero) "Text" error, the relevant fields would show the numerical values for each task.

In this configuration it is still not possible to perform the mining because data are still in numerical format, which prevents the use of the algorithm. Therefore, fields with values greated than zero were replaced by "S" and the zero valued ones were given value "N", turning them into alphanumeric attributes.

Still, to make the mining possible, non-english, special characters and blanks were removed and a text file was built to match the input requirements of used mining tool (Weka, discussed in the next section). In a first attempt, no interesting outcome was achieved, so the values "N" were replaced by "?". Such was done because the algorithm considered the proportion of "N" higher than "S", resulting in incorrect information. After the replacement some effective results were reached. After all the processes mentioned earlier, 1255 instances of tasks with errors were found, of these, 559 are labeled as "Improvement", 201 as "New", 459 as "RNC(Legacy)", 25 as "RNC(Recent) " and 11 as "Requirement". (Table 1). Some parameters had to be ajusted so that the algorithm would return some result. The metric type was chosen for confidence with a minimum of 18%.

For the mining process it was used an open source and free tool, Weka, which provides several data mining methods. Such a software was developed and is maintained by the University of Waikato. It was chosen because its simplicity and, use easyness and because it meets the needs of this research.

Table 1 – Number of errors by type of task

Task Type	**Total Errors**	**Percentage of total errors (%)**
Improvement	559	44,54
New	201	16,01
RNC(Legacy)	459	36,57
RNC(Recent)	25	1,99
Requirement	11	0,87

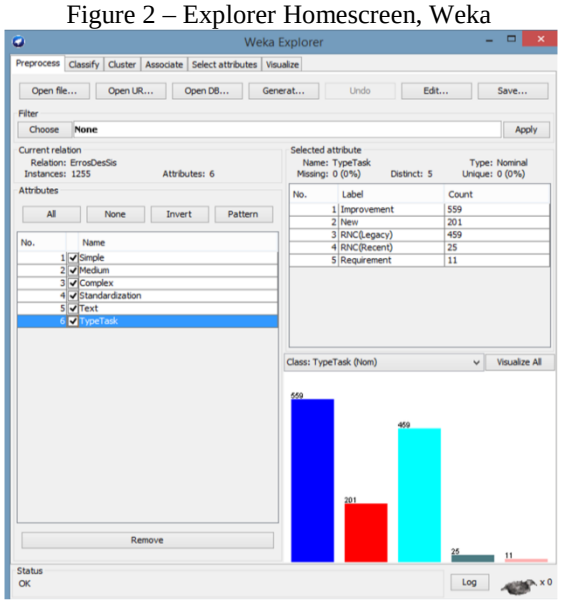
V. RESULTS AND DISCUSSION

Before mining, it was carried out an analysis of the data by using the Weka homescreen, which presents the amount of each attribute values. The amount of "Improvement" and "RNC (Legacy)" tasks outnumbers the other kinds, as shown in Table 1 and Fig. 2, but the proportion is not considered in this case because the object of study is the discovery of relationships between the kinds of errors. The research on the reasons for this numerical disproportion is left for a further study.

Fig. 3 displays the amount of each error kind related to the tasks. Since the the count of errors per task was replaced by the variable "S", the result of

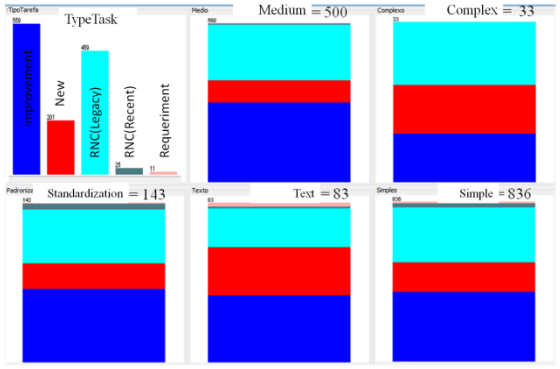
the analysis is that each frame gives the graphical proportion of the occurrence of certain error relation to the "TaskType" attribute e.g.: the "Simple" attribute shows that there are 836 tasks with at least one occurrence of this non-compliance condition, which are divided according to the class attribute ("TaskType"). Regarding the non-conformities labeled as "Simple", 43.66% of these are related to "Improvement" and 34.30% are related to "RNC (Legacy)".

Figure 2 – Explorer Homescreen, Weka



The graphic results remain proportional to "Simple", "Average" and "Standardization" when compared to the chart in Fig. 2. For non-conformities labeled as "Complex", Fig. 3 indicates a larger amount of “RNC (Legacy)” non-conformities, and a smaller one for "Improvement", which means that according to data, developers make more complex errors in more urgent tasks. The “Text” chart shows an increase of this kind of error in tasks labeled as "New".

Figure 3 - Amount of data attributes



The discovery of these trends can support the decision of a manager who wants to avoid or mitigate certain kinds of errors. It can even support the creation of standards for a company to stablish quality metrics.

After preliminary analysis, the algorithm was used to find relationships between the errors. Apriori gave even more credit the comments above and found 14 associations, the first one with a confidence level of 76.00% being 153 situations of “Simple” non-conformities related to "New" tasks.. The second relationship shows the "Simple" attribute is associated to a "RNC (Legacy)" task in 63.00% of the cases. These factors do not demonstrate a discovery, because charts allow the perception that the amount of “Simple” non-conformity is high in "RNC (Legacy)" and "Improvements" tasks.

The third relationship associates the ”Improvement” task with the "Simple" and "Average" attributes with 52.00% of reliability, meaning that “Improvement” tasks have 52.00% of chances to suffer “Simple” and “Average” errors at the same time. Such a condition cannot be verified in the chart, since it combines two attributes of non-compliance and it was unknown before this process. This is particularly relevant because such knowledge allows managers to interfere in software development process in order to avoid the occurrence of those noncompliances, making it more accurate and agile.

The whole list with fourteen association rules generated can be seen in Table 4, which presents the relationship between the attributes taken as base and related attributes, instance quantities of the base attribute and the number of instances of the associated attribute and, finally, the reliability of discovered rule.

Some rules were found again when the algorithm crossed the attributes during the rule confirmation phase, but with different reliability rates according to the base attribute. For example, rules 3, 6, 8, 11, 14 are equal but in the 3 rule the "Simple" and “Average” attributes are taken as base and "TaskType = Improvement" is the associated attribute. In rule 6 the "Simple" attribute becomes associated with "Average" and "TaskType = Improvement". In rule 8, "Average" comes to be associated with the other two attributes.

Table 2 - Depending on the rules generated Apriori algorithm

Nº	**Attribute base**	**Qnt.**	**Associate attribute**	**Qnt.**	**Reliability (%)**
1	TypeTask=New	201	Simple=S	153	76,00
2	TypeTask =RNC(Legacy)	459	Simple=S	287	63,00
3	Simple=	244	TypeTas	128	52,00

	S Medium =S		k =Improv ement		
4	TypeTas k =Improv ement	559	Medium =S	281	50,00
5	Medium =S	560	TypeTas k =Improv ement	281	50,00
6	Medium =S TypeTas k =Improv ement	281	Simple= S	128	46,00
7	Medium =S	560	Simple= S	244	44,00
8	Simple= S TypeTas k =Improv ement	365	Medium =S	128	35,00
9	Simple= S	836	TypeTas k =RNC(L egacy)	287	34,00
10	Simple= S	836	Medium =S	244	29,00
11	TypeTas k =Improv ement	559	Simple= S Medium =S	128	23,00
12	Medium =S	560	Simple= S TypeTas k =Improv ement	128	23,00
13	Simple= S	836	TypeTas k =New	153	18,00
14	Simple= S	836	Medium =S TypeTas k =Improv ement	128	15,00

VI. CONCLUSIONS

The KDD process allowed the analisys of a database presenting the results of in the testing process of system development, where it was investigated the relationship between the error kinds and the task kinds.

The study of the results disclosed the relationship between "Average" and “Simple” errors in over forty percent of these cases and, in those, more than fifty percent are related to the same kind of task.

These findings give development managers relevant input to support the solution of specific problems, improving the final product quality and reducing the software development time.

The work helped to enhance notion of the value added by the use of data mining to find meaningful relationships in large amount data in an efficient fashion and with minimal loss of information. The use of a data mining tool to analyze the correlations between non-conformities in software development proved effective during the validations and the intuitive interface and way of operation demanded little learning time, what reduced the workload to validate the data extracted from the source database.

REFERENCES

[1] M. Aurelio, M. Vellasco, C. H. Lopes, *Descoberta de Conhecimento e Mineração de Dados* (Handout Department of Electrical Engineering, PUC-Rio, RJ, 1999).

[2] U, Fayyad. G, Piatetsky-Shapiro. P. Smyth, *From Data Mining to Knowledge Discovery in Databases. American Association for Artificial Intelligence*, 1996, p. 37-54.

[3] H. S. Singh. *Interactive data warehousing*.(Makron Books, São Paulo, 1998).

[4] T. Pang-Ning, M. Steinbach, K. Vipin. *Introduction to Data Mining* (University of Minnesota, Minnesota, 2005).

[5] O. Maimon, L. Rokach, *Introduction to Knowledge Discovery and Data Mining* (Springer, London, 2010).

[6] E. Turban, J.C. Wetherbe, E. Mclean, *Tecnologia da Informação para Gestão* (Bookman Companhia Editora, São Paulo, 2010).

[7] P. Adriaans, D. Zantige. *Data mining* (Addison-Wesley, 1996).