Serdica J. Computing **8** (2014), No 2, 183–198

Serdica
Journal of Computing

Bulgarian Academy of Sciences
Institute of Mathematics and Informatics

DATA MINING FOR SOFTWARE DEVELOPMENT LIFE CYCLE QUALITY MANAGEMENT

Galia Novakova Nedeltcheva

Abstract. Computer software plays an important role in business, government, society and sciences. To solve real-world problems, it is very important to measure the quality and reliability in the software development life cycle (SDLC). Software Engineering (SE) is the computing field concerned with designing, developing, implementing, maintaining and modifying software.

The present paper gives an overview of the Data Mining (DM) techniques that can be applied to various types of SE data in order to solve the challenges posed by SE tasks such as programming, bug detection, debugging and maintenance. A specific DM software is discussed, namely one of the analytical tools for analyzing data and summarizing the relationships that have been identified. The paper concludes that the proposed techniques of DM within the domain of SE could be well applied in fields such as Customer Relationship Management (CRM), eCommerce and eGovernment.

1. Introduction.

Over the recent years Data Mining has been establishing itself as one of the major disciplines in computer science with growing industrial impact. DM is a hot topic of computer science research in recent

ACM Computing Classification System (1998): H.2.8.
Key words: Data mining, software engineering, software development, data mining, quality management.

years and has an extensive application in various fields. DM technology is an application-oriented technology. It is not only a simple search, query and transfer on the particular database, but also analyzes and integrates that data to guide the solution of practical problems and find the relations between events, and even to predict future activities through using the existing data.

Data Mining techniques are extensively used by private organizations and research communities to uncover hidden trends and knowledge from historical data. Data Mining concepts are successfully implemented in several areas such as "Banking", "Credit Card Business", "Insurance", "Super Store Sales data analysis", "Stock Market", "Gaming", "Network and Security", "Financial Market", "Telecommunication", "Oil and Gas exploration", "Weather Forecasting", etc. In recent years, government organizations have also recognized the potential use of Data Mining on eGovernance data to find hidden trends and knowledge from historical data.

For example, a large amount of data is generated and disseminated by different government departments at various levels of administration. It is important to integrate the different departments in terms of data sharing so that all departments can work under a single controlling authority without repetition of work. It is important to create a centralized nationwide data warehouse which has horizontal as well as vertical interconnections having limited accessibility at lower-level authorities and fully accessible at the higher levels. Use of efficient DM techniques may surely enhance government decision-making capabilities. So the DM technologies discussed in this paper can have good applications in areas as eGovernance.

Undoubtedly, research in DM will continue and even increase over coming decades, involving mining complex objects of arbitrary type, fast, transparent and structured data preprocessing, and increased usability. All aim at understanding consumer behavior, forecasting product demand, managing and building the brand, tracking performance of customers or products in the market and driving incremental revenue from transforming data into information, and information into knowledge. By using pattern recognition technologies and statistical and mathematical techniques to sift through warehouse information, DM helps analysts recognize significant facts, relationships, trends, patterns, exceptions and anomalies that might otherwise go unnoticed.

Modern data mining combines familiar and novel statistical methods to identify reproducible patterns in big data. The objective outcome is prediction. If a given model predicts new data better than the alternatives, then it has made a contribution. Rather than build a model that relates one or two experimental results to a response, data mining involves searching for patterns. Such searches commonly scan thousands of features, looking for the few that are predictive of the response. The search might be entirely automated or allow expert insight. Data mining is needed when dealing with wide data tables, those having more variables than cases, and it does not require exotic hardware or software. Recently, however, one of the most innovative technological challenges for DM is the application of cloud computing.

When we put the complexities of big data aside and strip it down to its fundamentals, the true definition of this historical shift in business analytics is about connecting many pieces of data to identify patterns which help us make better decisions. Good data analysis begins by looking at the data. That can be hard to do when the dataset has hundreds of columns, but it is not impossible. The objectives are to gain familiarity with the data, spot unusual patterns, recognize collinear variables and form conjectures. Hypothesis generation, not just testing, is an important aspect of DM. Most datasets contain *missing values*, and this class discusses simple methods for handling these anomalies – which are far too common – during data mining. To find interesting views of data it is also good to borrow ideas from *multivariate analysis* (*e.g.*, principal components and cluster analysis).

The purpose of this study is to explore how DM techniques can be applied to improve SE. So the objectives of the present study are:

- To review the concept of SE and DM;
- To determine some of the problems in SE;
- To enhance the SDLC quality assessment process;
- To identify some of the most efficient DM techniques and to propose available DM software that can be applied to solve SE problems.

The paper is structured in four sections: In the Introduction, a short overview on the important role of the DM methods is presented. A discussion on the proposed DM techniques for SE, in particular for SDLC quality management, is offered in Section 2. Some of the available DM software and its disvantages are given in Section 3. Finally, some conclusions are drawn and future research directions are outlined in Section 4.

2. Data Mining for software development quality management. Data Mining represents a shift from verification-driven data analysis approaches to discovery-driven data analysis approaches. In the former approach, a decision maker must hypothesize the existence of information of interest, collect this information and test the posed hypothesis against the information collected.

Discovery-driven approaches sift through large amounts of data and automatically (or semi-automatically) discover important information hidden in the data.

Some of the popular DM methods are as follows:

- Decision trees and rules
- Nonlinear regression and classification methods
- Example-based methods
- Probabilistic graphical dependency models
- Relational learning models

DM problems can be resolved by employing supervised and unsupervised algorithms. In the first case, a learning phase is necessary to build a predictive model from historical labeled data records, which is used later to make predictions about new, unlabeled data. Unsupervised algorithms are used in knowledge discovery modeling. This is a descriptive task whose objective is to detect patterns in actual data without need of previous learning.

In predictive modeling there is a special attribute called the "label" that one intends to predict. By encoding the relation between the label and the other attributes, the model can make predictions about new, unlabeled data. The two most common supervised modeling methods are *classification and regression.* If the label is discrete, the task is called classification; if the label is continuous, the task is called regression.

The goal in Knowledge Discovery (KD) modeling is to discover rules and segments of the data that behave similarly (clusters). These are unsupervised tasks. Unsupervised modeling is a descriptive task, not a predictive task. *Associations and clustering* are two unsupervised modeling tasks. The techniques listed in Table 1 are classified in these categories.

Table 1. Supervised and unsupervised data mining techniques

Supervised	**Unsupervised**
Decision tree	Deviation detection
Neural induction	Clustering
Regression	Association rules
Time series	Sequential pattern discovery

DM algorithms can be complemented with data visualization techniques taking advantage of the human brain's amazing pattern recognition capability. Usually the data is not organized in a way that will facilitate automated or semi-automated knowledge induction. In the DM process, visualization tools help to explore data before modeling and verify the results of other DM techniques. Visualization tools are particularly useful for detecting patterns found in only small areas of the overall data and they are very useful for noticing phenomena that hold for a relatively small subset of the data. There are many ways of visualizing data. Three-dimensional scatterplot and *splat* graphs can be used for multidimensional data. Color, opacity and other features can be used to represent variables.

SE text data includes bug reports, e-mails, code comments, and documentation for API methods. Common types of text mining algorithms include text clustering, classification and matching.

One of the main problems for data mining is that the number of possible relationships is very large, thus prohibiting the search for the correct ones by simply validating each of them. Hence, we need intelligent search strategies, as taken from the area of machine learning. Another important problem is that information in data objects is often corrupted or missing. Hence, statistical techniques should be applied to estimate the reliability of the discovered relationships.

The economies of all developed countries are dependent on software, because software controls systems that affect our daily needs. Thus, SE has become more and more important and it concerns computer-based system development; this includes system specification, architectural design, integration and deployment.

With the increasing importance of SE, the difficulty of maintaining, creating and developing software has also risen. Challenges in SE include SDLC quality assessment, requirement gathering, systems integration and evolution, maintainability, pattern discovery, fault detection, reliability and complexity of software development [15, 5].

DM is a process that employs various analytic tools to extract patterns and information from large datasets. Today, large numbers of datasets are collected in various fields, including eGovernment for instance, and stored.

Humans are much better at storing data than extracting knowledge from it, especially accurate and valuable information needed to create good software. There are seven steps in the process of extracting knowledge: data integration, data cleaning, data selection, data transformation, data mining, pattern evaluation and knowledge presentation (see Fig. 1).

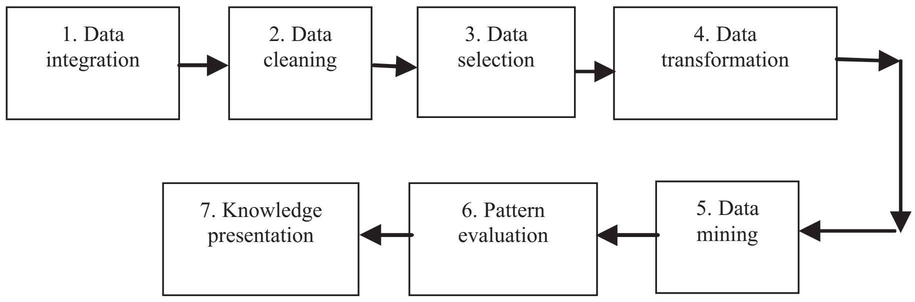

Fig. 1. Steps in the process of extracting knowledge from large datasets

DM techniques that can be applied in improving SE include generalization, characterization, classification, clustering, associative tree, decision tree or rule induction, frequent pattern mining, etc. [4]. SE has become increasingly important these days and research on its problems has proposed various methods for improving SE. The study of SE problems has followed several approaches. The early research of Thayer et al. [14] was concerned with SE project management. They introduced planning, organizing, staffing, and controlling problems as major challenges in this area. Ramamoorthy et al. [13] stated that as more complex software applications are required, programmers will fall further behind the demand. This causes the development of poor quality software and higher maintenance costs. The problems stated by Thayer are more closely tied to the processes of SE project management, while those introduced by Ramamoorthy are more closely tied to the limitations of human beings. Later work by Clarke [3] identified challenges in SE associated with the complexity of the software development process. He stated that the complexity of software development causes the software to become harder to maintain. This leads to other problems such as software integrity and difficulty in detecting application bugs or flaws. A bug is a flaw in a computer program that can ultimately cause glitches, program failure or software destruction.

Data perturbation techniques for preserving privacy in DM were proposed by Islam and Brankovic [9]. Aouf proposed the clustering technique to identify patterns in the underlying data [1]. Later work by Ma and Chan [11] suggested iterative mining for mining overlapping patterns in noisy data. The three data mining approaches discussed above are all clustering techniques. The data perturbation technique described by Islam and Brankovic [9] involves adding noisy data into some part of the dataset in order to preserve privacy, while Ma and Chan [11] were concerned with the elimination of noisy data to enable extracting valuable information. So different types of data require different DM techniques.

In the business environment, complex DM projects may require the coordinate efforts of various experts, stakeholders, or departments throughout an entire organization. In the data mining literature, various "general frameworks" have been proposed to serve as blueprints for how to organize the process of gathering data, analyzing data, disseminating results, implementing results, and monitoring improvements. One such model, CRISP (Cross-Industry Standard Process for DM) was proposed in the mid-1990s by a European consortium of companies to serve as a non-proprietary standard process model for data mining. This general approach postulates the following (perhaps not particularly controversial) general sequence of steps for data mining projects:

$$\begin{tikzcd}
\text{Business Understanding} \arrow[r, leftrightarrow] \arrow[d] & \text{Data Understanding} \\
\text{Data Preparation} \arrow[r, leftrightarrow] & \text{Modeling} \arrow[d] \\
& \text{Evaluation} \arrow[d] \\
& \text{Deployment}
\end{tikzcd}$$

Another approach – the Six Sigma methodology—is a well-structured, data-driven methodology for eliminating defects, waste or quality control problems of all kinds in manufacturing, service delivery, management, and other business activities. This model has recently become very popular (due to its successful implementations) in various US industries, and it appears to gain favor worldwide. It postulated a sequence of so called DMAIC steps –

$$\text{Define} \rightarrow \text{Measure} \rightarrow \text{Analyze} \rightarrow \text{Improve} \rightarrow \text{Control}$$

– that grew up from the manufacturing, quality improvement and process control traditions, and is particularly well suited to production environments (including "production of services", i.e., service industries).

Another framework of this kind (actually somewhat similar to Six Sigma) is the approach proposed by SAS Institute called SEMMA –

$$\text{Sample} \rightarrow \text{Explore} \rightarrow \text{Modify} \rightarrow \text{Model} \rightarrow \text{Assess}$$

– which is focusing more on the technical activities typically involved in a data mining project.

All these models are concerned with the problem of how to integrate data mining methodology into an organization, how to "convert data into information", how to involve important stakeholders, and how to disseminate the information in a form that can easily be converted by stakeholders into resources for strategic decision making.

2.1 Analysis. Different DM algorithms produce patterns that reflect different levels of information, and which algorithm to choose depends on the specific SE task's mining requirements.

SE data can be divided into three categories as follows [16]:

- Sequences, such as execution traces collected at run-time, and static traces extracted from source code;
- Graphs, such as dynamic call graphs extracted from source code;
- Text, such as code comments, documentation and bug reports.

Software quality attributes provide the means for measuring the fitness and aptness of a software product. The desired attributes for a good software system are *Reliability, Efficiency, Security, Maintainability, Supportability, Performance, Usability, etc.* During the complete process of software development, at each phase, effort is made to achieve these attributes.

When problems such as bugs or flaws arise in systems or software, it is difficult for developers or programmers to determine the cause(s). DM is a valuable tool for solving such SE problems. DM techniques can be applied to solve problems in all three categories of SE data. The following sub-topics review and describe DM techniques that can solve problems in each type of SE data.

A. Sequence Data Mining. Examples of SE sequence data include method-call data that is dynamically collected during program execution, or statically extracted from source code. The challenge inherent in SE sequence data is the difficulty of extracting sequential patterns and information from the source code [16] or program during program execution [2] for software bug or flaw detection. DM techniques that can be applied to solve these problems are *frequent item set mining* (FIM) and *frequent sequence mining* (FSM). FIM and FSM are types of frequent pattern mining that use alternative support counting techniques [16].

Figure 2 shows the process of frequent pattern mining on program source code. A program source code is composed of elements written in a particular programming language. To mine sequential data from source code, we need to

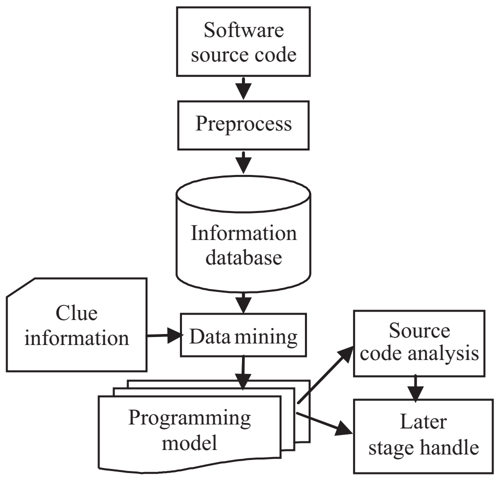

Fig. 2. The procedure of frequent pattern mining on source code [17]

divide it into different units, such as tags, blocks, function, and classes, for further mining. The information extracted from different units of source code can be used to remove the wrong source code [17]. In this case, FIM can be used to characterize the importance of program elements. FIM is used to mine the frequency of program elements, and to extract the procedural rules of fault and bug detection. FIM is only suitable for mining static traces or paths extracted from source code as it does not reflect the sequential order of information in the mined pattern [16, 12]. FIM uses a bottom-up approach and follows the principle wherein an item set X of variables can only be frequent if all its subsets are also frequent [12]. Thus, it is not suitable for application to run-time data [17].

B. Graph Data Mining. Most SE data can be represented as graphs. Dynamic-call graphs generated during execution of a program and static call graphs generated from program source code are instances of SE data represented in graphs [16]. A graph is an expressive representation for SE data and is useful for modeling complicated structures and their relationships [7]. Mining graph data has great potential to help in software development quality management. Thus, graph DM is often an active research area in SE.

As graph data is usually large and complex, it is hard for the human eye to uncover patterns in them and classify the patterns to ease program bug analysis [3]. Application of graph classification can solve this problem, since the technique automates the identification of subgraph patterns from graph data and constructs a graph classification model for automating the classification process.

Software is unlikely to ever be failure-free. The more failures and bugs encountered in the software, the more unreliable the software is. By detecting bugs and failures and pinpointing their origin software programmers or developers can fix them and thus increase the software reliability. There are two types of software bugs: crashing and non-crashing [10]. Crashing bugs are those that cause program execution to halt under non-ordinary conditions. Examples of crashing bugs include inputting characters into a field of integer data type, or referring to a null pointer. Non-crashing bugs do not terminate the program [10] but may result in errors in execution or output. An example of a non-crashing bug would be logical errors such as when the final result generated by the program is not the result that it should be. Non-crashing bugs are more difficult to detect as they have no crashing point——that is, the program continues running past the point of error [10].

In order to disclose traces of non-crashing bugs, a graph classification technique is used. Software behavior graphs from program execution comprise the data to mine. However, the number of frequent subgraphs mined from behavior graphs is often large, which may result in low performance of graph mining and classification [10] due to the difficulties and the time consuming nature of mining huge numbers of frequent subgraphs. Closed frequent subgraphs are a legitimate substitute for frequent graphs for classification purposes, as the number of closed frequent subgraphs generated is lower. The mined closed frequent subgraphs are used for classifier construction [7]. A classifier is built at each checkpoint for every function in a program to detect the location of the bug [10].

Bug detection is achieved by monitoring for classification accuracy boost. The classification accuracy of each classifier will remain low before the bug is triggered [16], because the classifier lacks information about the behavior of the bug. When the bug is triggered, the classification accuracy of the classifier at that checkpoint increases, and therefore the location of that classifier is probably the location of the bug [10]. This eases the process of debugging for non-crashing bugs, since in this way programmers can deal with the bugs directly without going through the difficulty and time to search for the bugs on their own.

Figure 3 shows the process of bug report (BR) classification using text data mining's Natural Language Description Technique. The initial step is to obtain a set of labeled BR data that contains textual descriptions of bugs correctly labeled to indicate whether they should be classified security BR or non-security BR. Labeling of the BR data set is required for creating and evaluating a Natural Language Predictive Model.

An example of an architecture for mining software engineering data is

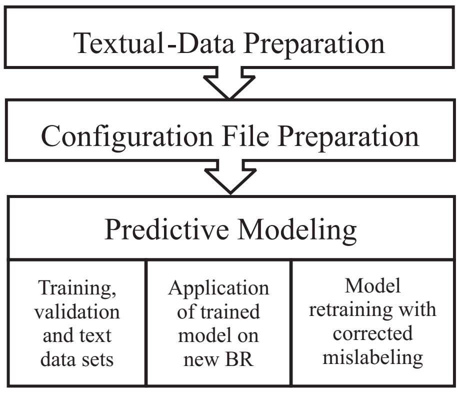
Textual-Data Preparation
Configuration File Preparation
Predictive Modeling
Training, validation and text data sets
Application of trained model on new BR
Model retraining with corrected mislabeling

Fig. 3. Bug reports classification using text data mining [6]

given in Fig. 4 below.

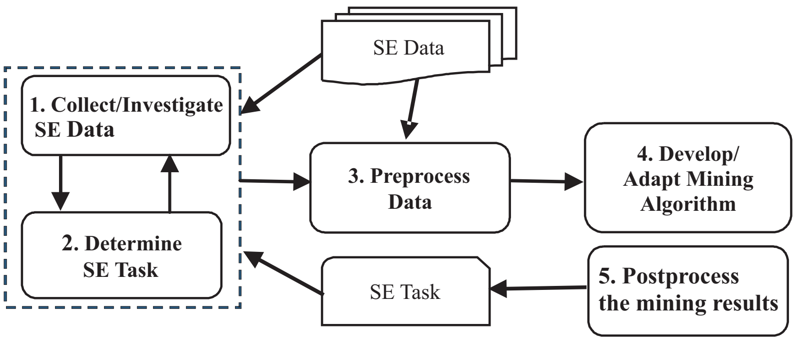
SE Data
1. Collect/Investigate SE Data
2. Determine SE Task
3. Preprocess Data
4. Develop/ Adapt Mining Algorithm
SE Task
5. Postprocess the mining results

Fig. 4. Architecture for mining software engineering data

3. Data mining tools in software engineering. Several researchers and organizations have conducted reviews of DM tools and surveys of data miners. They identify some of the strengths and weaknesses of the software packages and also provide an overview of the behaviors, preferences and views of data miners.

Some examples of software packages for DM are the following:

- 2CEE Cost Estimation
- Data.Mining.Fox (`www.easydatamining.com`)

- AdvancedMiner Professional (`www.statconsulting.eu`)
- Coheris SPAD (`www.coheris.fr/en/page/produits/Spad.html`)
- PEPITo software (`www.pepite.be`)
- SAS Enterprise miner (`www.sas.com/en_us/software/analytics/enterprise-miner.html`)
- *STATISTICA* Data Miner

The general underlying philosophy of StatSoft's *STATISTICA* Data Miner is to provide a flexible data mining workbench that can be integrated into any organization, industry, or organizational culture, regardless of the general data mining process-model that the organization chooses to adopt. For example, *STATISTICA* Data Miner can include the complete set of (specific) necessary tools for ongoing company-wide Six Sigma quality control efforts, and users can take advantage of its (still optional) DMAIC-centric user interface for industrial data mining tools. It can equally well be integrated into ongoing marketing research, CRM (Customer Relationship Management) projects, etc., that follow either the CRISP or SEMMA approach—it fits both of them perfectly well without favoring either one. Also, *STATISTICA* Data Miner offers all the advantages of a general data mining oriented development kit that includes easy-to-use tools for incorporating not only such components as custom database gateway solutions, prompted interactive queries, or proprietary algorithms into the projects, but also systems of access privileges, workgroup management, and other collaborative work tools that allow for designing large scale, enterprise-wide systems (e.g., following the CRISP, SEMMA, or a combination of both models) that involve the entire organization.

Some of the disadvantages of DM software packages. DM brings many benefits to business, society, government as well as individuals. However privacy, security and misuse of information are the big problem if it is not addressed correctly.

Privacy Issues—Concerns about personal privacy have been increasing enormously recently, especially when the Internet is booming with social networks, e-commerce, forums, blogs, etc. Because of privacy issues, people are afraid that their personal information is collected and used in unethical ways, potentially causing them a lot of trouble. Businesses collect information about their customers in many ways for understanding their purchasing behavior trends. However, businesses don't last forever, some day they may be acquired by others or gone. At this time the personal information they own is probably sold to another or leaks.

Security issues—Security is a big issue. Businesses own information about their employees and customers including social security number, birthday, payroll, etc. However, how properly this information is taken is still in question. There have been many cases when hackers accessed and stole customers' data from big corporations such as Ford Motor Credit Company, Sony, etc. with so much personal and financial information available, credit card stolen, identity theft become a big problem.

Misuse of information/inaccurate information—Information collected through DM intended for marketing or ethical purposes can be misused. This information is exploited by unethical people or businesses to take benefit of vulnerable people or discriminate against groups of people.

In addition, a DM technique is not perfectly accurate therefore if inaccurate information is used for decision making it will cause serious consequence.

The challenges of DM can be outlined as follows:

- Software development quality assessment
- Scalability
- Dimensionality
- Complex and heterogeneous data
- Data quality
- Data ownership and distribution
- Privacy preservation
- Streaming data

So while DM techniques have been applied across broad domains, they have rarely been applied in the field of software development quality assessment, a subfield of SE [4].

4. Conclusions and future enhancement. Future work needs to investigate more DM algorithms that can help to improve the process of SDLC quality assessment and are easy to use.

Some predictive DM problems are of the non-linear (NL) type. For very complex prediction (or forecasting) problems, NL algorithms or a blend of both linear and NL will be best. This means that blends of different algorithms or techniques which combine strengths will be more useful. Effort should be geared towards building supermodels that combine two or more of these techniques. Much work is going on in evaluating the strengths of the techniques: neural networks (NN), support vector regression (SVR), regression trees, kernel regression, kernel SVR, etc. Some of the breakthroughs are kernel support vector machines, kernel principal component analysis and least square support vector machines.

Another area of DM that has been and will continue to be a fertile ground for researchers is the area of data acquisition and storage. The ability to correctly acquire, clean and store data sets for subsequent mining is not an easy task. A lot of work is going on in this area to improve on what is available today.

There are many commercial software packages produced to solve some of the problems but most are uniquely made to solve a particular type of problem. It would be desirable to have mining tools that can switch to multiple techniques and support multiple outcomes. Current DM tools operate on structured data, but most of the data in the field is unstructured. Since large amounts of data are acquired, for example in the World Wide Web, there should be tools that would manage and mine data from this source, a tool that can handle dynamic, sparse, incomplete or uncertain data. The dream looks very high but given the time and energy invested in this field and the results which are produced, the development of such software is not far away.

As well, DM technique is used in CRM. Nowadays it is one of the hot topics of research in the industry, because CRM have attracted both practitioners and academics. Research on the application of DM in CRM will increase significantly in the future based upon past publication rates and the increasing interest in the area. The majority of the existing articles relate to customer retention.

E-commerce is also the most prospective domain for DM. It is ideal because many of the ingredients required for successful DM are easily available: data records are plentiful, electronic collection provides reliable data, insight can easily be turned into action, and return on investment can be measured. The integration of e-commerce and DM significantly improves the results and guide the users in generating knowledge and making correct business decisions. This integration effectively solves several major problems associated with horizontal DM tools including the enormous effort required in pre-processing of the data before it can be used for mining, and making the results of mining actionable.

Besides, the exploration of the DM techniques discussed in the paper could be for empowering eGovernment applications and services for the citizen's benefit. Existing eGovernment applications entail a limited adaptation. In particular, the need to transform eGovernment services to e-inclusion applications is a motivation for utilization of data mining techniques for processing the governmental data so as to extract and associate information fragments with real citizen needs and thus enable the encapsulation of the latter in future governmental decisions. DM techniques have a potential adoption in many government sectors such as healthcare, agriculture, education, social security funds, pollution control, electronic voting, rainfall prediction, customer complain, road traffic violation, crime control, crime forecasting, taxing, etc.

Through the proposed DM techniques in the paper effective and efficient detection of anomalies in massive data sets can be achieved, e.g., outliers, duplicates, inconsistencies, and dubious data. There is a potential to apply the DM approach also for ontology validation in the data collection.

REFERENCES

[1] Aouf M., L. Lyanage, S. Hansen. Critical review of data mining techniques for gene expression analysis. In: Proceedings of the International Conference on Information and Automation for Sustainability (ICIAFS), 2008, 367–371.

[2] Bhasker B. An algorithm for mining large sequence in databases. *Communications of the IBIMA*, **6** (2008), 149–153.

[3] Clarke J. et al. Reformulating software engineer as a search problem. *IEEE Proceeding Software*, **150** (2003), No. 3, 161–175.

[4] DePree R. W. Pattern recognition in software engineering. *IEEE Computer*, **16** (1983), No 5, 48–53.

[5] Fern X. L., C. Komireddy, V. Gregoreanu, M. Burnett. Mining problem-solving strategies from HCL data. ACM Trans. *Computer-Human Interaction,* **17** **(**2010), No 1, Article 3, 1–22.

[6] Gegick M., P. Rotella, T. Xie. Identifying security bug reports via text mining: an industrial case study. In: Proceedings of the 7th IEEE Working Conf. Mining Software Repositories (MSR), Cape Town, 2010, 11–20.

[7] Han J., M. Kamber. Data mining concepts and techniques. 2nd Edition, The Morgon Kaufman Series in Data Management Systems, 2005.

[8] Hihn J., K. Lum. 2CEE, A twenty first century effort estimation methodology. In: Proceedings of the ISPA/ SCEA Joint International Conference, 2009, Lane Dept. CSEE West Virginia University.

[9] Islam M. Z., L. Brankovic. Detective: a decision tree based categorical value clustering and perturbation technique for preserving privacy in data mining. In : Proceedings of the Third IEEE Conference on Industrial Informatics (INDIN), 2005, 701–708.

[10] Liu C. et al. Mining behavior graphs for "backtrace" of noncrashing bugs. In: Proceedings of the 2005 SIAM Int. Conf. on Data Mining (SDM'05), 2005, 286–287.

[11] Ma P. C. H., K. C. C. Chan. An iterative data mining approach for mining overlapping coexpression patterns in noisy gene expression data. *IEEE Trans. Nano Bioscience*, **8** (2009), No 3, 252–258.

[12] Parsons T., J. Murphy, P. O Sullivan. Applying frequent sequence mining to identify design flaws in enterprise software systems. In: Proceedings of the 5th Int. Conf. MLDM, 2007, 261–275.

[13] Ramamoorthy C. V., A. Prakash, W. T. Tsai, Y. Usuda. Software engineering: problems and perspectives. *Computer archive*, **17** (1984), No 10, IEEE Computer Society, 191–209.

[14] Thayer R. H., A. Pyster, R. C. Wood. Validating solutions to major problems in software engineering project management. *Computer archive,* **15** (1982), No 8, IEEE Computer Society, 65–77, 1982.

[15] Wahidah H., Pey Ven Low, Lee Koon Ng, Zhen Li Ong. Application of Data Mining Techniques for Improving Software Engineering. In: Proceedings of the 5th International Conference on Information Technology ICIT, Amman, Jordan, 2011.

[16] Xie T., S. Thummalapenta, D. Lo, C. Liu. Data mining for software engineering. *Computer archive.* **42** (2009), No 8, IEEE Computer Society, 55–62.

[17] Zong C. M., Z. L. Li. Applying data mining techniques in software development. In: Proceedings of the 2nd IEEE Int. Conf., Chengdu, China, 16–18 Apr. 2010, 535–538.

Galia Novakova Nedeltcheva
Faculty of Mathematics and Informatics
Department of Computing Systems
Sofia University
5, J. Bourchier Blvd
1164 Sofia, Bulgaria
e-mail: `g.novak@fmi.uni-sofia.bg`

Received December 4, 2014
Final Accepted February 23, 2015