Code4Thought Project: Employing the ISO/IEC-9126 standard for Software Engineering - Product Quality Assessment

Panos Antonellis[1], Dimitris Antoniou[1], Yiannis Kanellopoulos[2], Christos Makris[1], Christos Tjortjis[3], Vangelis Theodoridis[1], Nikos Tsirakis[1]

1. University Of Patras, Department of Computer Engineering and Informatics, Greece
2. The University Of Manchester, School of Computer Science, U.K.
3. University of Ioannina and University of Western Macedonia, Greece

adonel, antonid, makri, theodori, tsirakis@ceid.upatras.gr,
Yiannis.Kanellopoulos@postgrad.manchester.ac.uk, christos.tjortjis@manchester.ac.uk

Abstract

The aim of the Code4Thought project was to deliver a tool supported methodology that would facilitate the evaluation of a software product's quality according to ISO/IEC-9126 software engineering quality standard. It was a joint collaboration between Dynacomp S.A. and the Laboratory for Graphics, Multimedia and GIS of the Department of Computer Engineering and Informatics of the University of Patras. The Code4thought project focused its research on extending the ISO/IEC-9126 standard by employing additional metrics and developing new methods for facilitating system evaluators to define their own set of evaluation attributes. Furthermore, to develop innovative and platform-free methods for the extraction of elements and metrics from source code data. Finally, to design and implement new data mining algorithms tailored for the analysis of software engineering data.

1. Introduction

Software is playing a crucial role in modern societies. Not only people rely on it for their daily operations or business, but for their lives as well. For this reason a correct and consistent behaviour of a software system is a fundamental part of users' expectations.

Therefore the demand for software quality is increasing and is setting it as a differentiator, which can determine the success or failure of a software product. Moreover delivering high quality products is becoming not just a competitive advantage but a necessary factor for companies to be successful.

The goal of the Code4Thought project was to deliver a tool supported methodology that would facilitate the evaluation of a software product's quality according to ISO/IEC-9126 software engineering quality standard.

It was a joint collaboration between Dynacomp S.A. and the Laboratory for Graphics, Multimedia and GIS of the Department of Computer Engineering and Informatics of the University of Patras. It was a 2 years R&D project co-funded by G.S.R.T. (General Secretariat of Research and Technology) and Dynacomp S.A. The project ended in May 2008 and its budget was 422,000 Euro. The members consisting the project's team were Dr Christos Tjortjis as the Project Coordinator, from the Manchester School of Computer Science and Dynacomp S.A. Then Dr Christos Makris University of Patras Coordinator, from CEID (School of Engineering Department of Computer Engineering and Informatics). The Project Manager was Yiannis Kanellopoulos, PhD student from Dynacomp S.A. and Manchester School of Computer Science. Then there were Evangelos Theodoridis, Antoniou Dimitris, Tsirakis Nikos, and Antonellis Panagiotis, all PhD candidates from CEID.

The SQO-OSS project was also related to Code4Thought **Error! Reference source not found.**. Both projects were focusing on software quality and on employing data mining techniques for the analysis of the derived software measurement data.

2. Research Focus

The research on the Code4Thought project was related to software quality and data mining. For this reason, the research focus in this project was the on extending at first, the ISO/IEC-9126 standard by employing additional metrics and developing new methods for facilitating system evaluators to define their own set of evaluation attributes [1], [4]. Furthermore innovative and platform free methods for the extraction of elements and metrics from source code data were developed. Finally new data mining algorithms or combination of data mining techniques tailored for the analysis of software engineering data were designed and implemented [2], [3].

3. Project Achievements

The Code4Thought project developed at first a generalised quality model for the assessment of both structural and OO software systems. The characteristics of this model were:

The main characteristics of this methodology are:

- The necessary metrics were elements extracted solely from source code.
- It employed the ISO/IEC-9126 standard as a frame of reference for communication concerning software product quality.
- It proposed a three-step approach and an associated model, in order to link system level quality characteristics to code-level metrics.
- It applied the Analytic Hierarchy Process (AHP) in every level of the model's hierarchy in order to reflect the importance of metrics and system properties on evaluating quality characteristics according to ISO/IEC-9126.

Furthermore, k-Attractors, a clustering algorithm tailored for the analysis of software measurement data was developed for the purposes of this project [3]. The characteristics of this algorithm are:

- It defines the desired number of clusters (i.e. the number of k), without user intervention.
- It locates the initial attractors of cluster centers with great precision.
- It measures similarity based on a composite metric that combines the Hamming distance and the inner product of transactions and clusters' attractors.
- It can be used for large data sets.

4. Project Deliverable

The final deliverable of the project was a data mining tool that was responsible for the extraction, analysis and visualisation of data and results concerning the evaluation of a software system. This tool consisted of the following modules:

- Data Extraction and Preparation.
- Data Analysis.
- Results Visualization.

The objective of the data extraction and preparation module was two-fold:

- At first to collect appropriate elements that describe the software architecture and its characteristics. These elements included native source code attributes and metrics.
- Then to analyze the collected elements, choose a refinement subset of them and store them in a relational database system for further analysis.

Native attributes included definition files, classes, structure blocks etc. Metrics, on the other hand, provided additional system information and described more effectively the system's characteristics and behaviour.

All the metrics were associated with a native source code attribute, e.g. the lack of cohesion is associated with a class member method. All of the above collected attributes and metrics were stored into appropriate structured XML files. XML was chosen because of its interoperability and its wide acceptance as a de facto standard for data representation and exchange. Storing the metrics in XML files enables further processing and analysis with a variety of tools.

For simplicity, a refinement subset of the most important collected elements was chosen for analysis. This subset should be small enough in order to be easily analyzed and large enough to contain all the necessary system information. Based on this requirement, only the metrics and their associated native attributes were stored and further analyzed.

The elements chosen needed to be extracted from the XML files and stored permanently in a relational database. For this reason tools that mapped XML elements and nodes into any relational database, keeping the extraction method transparent from the underlying database were used.

Figure 1 depicts the general architecture of the data extraction and preparation module.

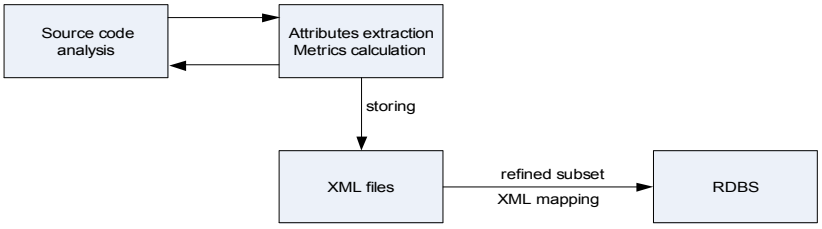

Figure 1: Data Preparation and Extraction Module

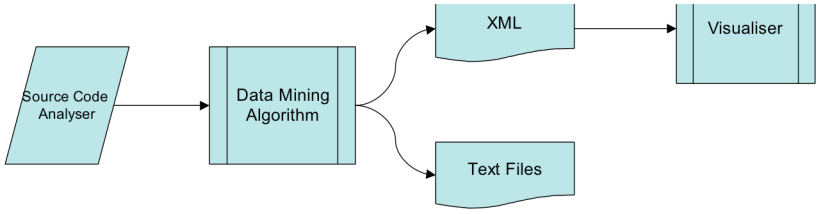

Figure 2: Data Analysis Module

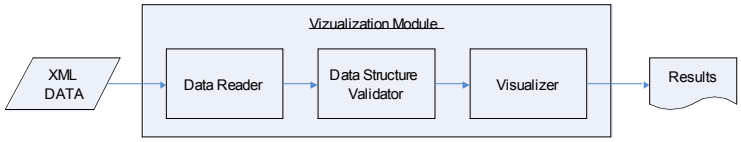

Figure 3: Data Visualisation Module

As depicted in Figure 2, the data analysis module is the core of the Code 4 Thought methodology. At first, the data mining algorithm, accepts data from the source code analyzer, by performing queries on the database, where the data reside. The outcome of the analysis is stored in XML files, in order to be visualized by the corresponding module. The primary objective of the data analysis is to obtain a general but illuminating view of a software system that may lead maintenance engineers to useful conclusions about its structure and maintainability.

The visualization module now consists of the following parts:

- An XML reader for the clustering results.
- Text files for the results derived from association rules and classification algorithms.

Figure 3 shows the parts of the visualization procedure for the XML files, from the moment data are being imported until the final results are available to the user.

5. Research Evaluation

The work during this project was evaluated on both open source and proprietary systems of industrial scale. Results were reviewed by domain experts who provided their comments and assessments. The criteria for this evaluation included accuracy of the tool and the ability to capture knowledge that is valid, novel and potentially useful to maintenance engineers.

In the first case study [2] a framework that combined clustering and association rules mining was evaluated. An industrial-scale proprietary system concerning books publication developed in C#, was used as a test-bed. By examining the patterns found from clustering, several observations were derived. At first the existence of "god classes" was discovered as there are classes that their number of methods is much above the average which is 5 methods per class. Patterns that reflected the niches of the Books Publishing System were also discovered. For example a cluster of the Member Methods consists of methods that were responsible for the format and rounding of numbers. Some also from the derived association rules like Some of them like supervisorNameTxt->addressTxt, contract_id->_published_book_id, Assign Parameters->Get Parameters were characterised by system's developers interesting.

In the second case study clustering was applied in JBoss AS, an open source application server [5]. The aim of this study was to facilitate maintenance engineers to comprehend the structure of a software system and assess its maintainability. For this reason source code elements and metrics were collected. As derived from the clustering (based on its structural characteristics), JBoss is an OO system that is built based on the JMX (Java Management Extension) infrastructure and supports also Aspect Oriented Programming middleware. The assumption here was that packages which play the most significant role in formulating clusters would be the main packages of JBoss. The inspection of clusters showed that this assumption was valid. For example a cluster was formed by classes that belong to org.jboss.console which provides a simple web interface for managing the MBean server and org.jboss.ejb3 which is the most basic package for an EJB implementation. A very interesting cluster which comprises of packages that implement AOP (Aspect Oriented Programming) – based services, was also discovered. Another observation was that most of the classes of JBoss have very low complexity values and they do not expose too much afferent or efferent couplings. They also have no children and they do not use inheritance very much. Interestingly, classes related to the implementation of the AOP services support, have very low complexity (WMC) values. It was also observed that most of the classes with high WMC and CBO values have member methods that their return types are not primitives.

The second case employed as test-bed Apache Geronimo, another open source application server [1]. It combined clustering with Analytic Hierarchy Process (AHP) for the evaluation of a software product's maintainability according to the ISO/IEC-9126 software product quality standard. The intuition was to integrate measurement data extracted from source code's elements with the expertise of a system's engineers by providing them the ability to define a number of attributes suitable for such evaluation. In this case study a cluster that contained classes that exhibited very low maintainability values was discovered. These classes were provided to

domain experts (that is, software engineers familiar with the Geronimo Apache source code) in order to inspect them and give their feedback concerning additional attention for improving their maintainability. From the classes of this cluster, they indicated that CdrOutputStream and CdrInputStream needed further attention. These classes were used fairly widely within the application server, for, among others, serializing non-primitive data structures, hence the high complexity values. They were of interest to the domain experts, since they are at Geronimo's core and widely used, so for maintainability and runtime performance they were important classes.

The third case study was similar to the previous one, in the sense that combined clustering and AHP for the evaluation of maintainability according the ISO/IEC-9126 standard [4]. However its main difference was that clustering was applied on the raw source code metrics and not on the derived ISO/IEC-9126 maintainability values. The aim here was to demonstrate how to summarise information concerning the evaluation of maintainability in the various levels of a software system, from raw source code metrics describing a system's artefacts, to the system itself as a whole. By using the proposed methodology maintenance engineers were able to quickly obtain high-level, but constructive information about the system's maintainability. The method compared well with the most naïve technique for interpreting source code metrics, being the identification of outliers for individual metrics. Secondly, it compared well with existing methods for translating source code measurements into high-level quality characteristics. The introduction of clustering into the process ensured that aggregation was done on relatively homogenous groups of units. Thirdly, the use of AHP, rather than other weighting schemes turned out to scale up nicely to the high number of metrics and system properties addressed in this case study.

Finally, k-Attractors, a clustering algorithm presented in [3] was tested in two proprietary systems. The results of this experiment showed that in its main phase k-Attractors is about 600% faster than k-Means. This is attributed to its initialization phase which adds a significant overhead to the overall algorithm's performance. It was also demonstrated that regarding software measurement data, k-Attractors appears to form better, in terms of quality, and more concrete clusters. So as long as the initialisation phase contributes significantly to the algorithm's accuracy this overhead is considered justifiable.

6. Conclusions

The evaluation results shown at first, that descriptive data mining techniques have the ability to support program comprehension and maintainability evaluation according to ISO-IEC/9126. It was shown that clustering can be complementary to association rules mining in the sense that their combination can form a single and coherent framework [2]. Then it was demonstrated that k-Attractors is significantly faster than k-Means, a prominent clustering algorithm. Additionally, regarding software measurement data, k-Attractors appeared to form better, in terms of quality, and more concrete clusters [3]. Also it was demonstrated that clustering software metrics can create groups of artefacts with similar measurements and spot potentially important maintainability issues [1], [2], [5]. Finally, it was shown how to translate source code measurements into high-level quality characteristics (like maintainability) [1], [4].

All cases have shown promising results concerning the combination of data mining techniques with a model based on the ISO/IEC-9126. Furthermore, concerning the project's planned activities there were difficulties getting data from proprietary systems. In some cases non-disclosure agreements had to be signed and in other cases there was no access granted. That is the reason that most of the experiments were conducted using open source systems.

References

[1.] Antonellis et al. "A Data Mining Methodology for Evaluating Maintainability according to ISO/IEC-9126 Software Engineering-Product Quality Standard", in the proceedings of IEEE 11th Conference on Software Maintenance and Reengineering (CSMR 2007) special session on System Quality and Maintainability (SQM 2007)

[2.] Kanellopoulos Y., Makris C. and Tjortjis C., "An Improved Methodology on Information Distillation by Mining Program Source Code", Data & Knowledge Engineering, (Elsevier) May 2007, Volume 61, Issue 2, pp. 359 - 383.

[3.] Kanellopoulos Y., Antonellis P., Tjortjis C, Makris C., "k-Attractors: A Clustering Algorithm for Software Measurement Data Analysis", In the proceedings of IEEE 19th International Conference on Tools with Artificial Intelligence, (ICTAI 2007)

[4.] Kanellopoulos Y., Heitlager I., Tjortjis C., Visser J., "Interpretation of source code clusters in terms of ISO/IEC-9126 Quality Aspects", in the proceedings of IEEE 12th Conference of Software Maintenance and Reengineering (CSMR 2008).

[5.] Kanellopoulos Y., Dimopoulos T., Tjortjis C. and Makris C., "Mining Source Code Elements for Comprehending Object-Oriented Systems and Evaluating Their Maintainability" ACM SIGKDD Explorations Vol. 8 Issue 1, Special Issue on Successful Real-World Data Mining Applications, pp 33-40 June 2006

[6.] Code4Thought: www.code4thought.org

[7.] SQO-OSS: www.sqo-oss.org