BMR: Benchmarking Metrics Recommender for Personnel issues in Software Development Projects

Ángel García-Crespo, Ricardo Colomo-Palacios,
Juan Miguel Gómez-Berbís, Myriam Mencke
Computer Science Department, Universidad Carlos III de Madrid,
Av. Universidad 30, Leganés, 28911, Madrid, Spain
[angel.garcia, ricardo.colomo, juanmiguel.gomez, myriam.mencke]@uc3m.es
www.uc3m.es

(leave space here)

Abstract

This paper presents an architecture which applies document similarity measures to the documentation produced during the phases of software development in order to generate recommendations of process and people metrics for similar projects. The application makes a judgment of similarity of the Service Provision Offer (SPO) document of a new proposed project to a collection of Project History Documents (PHD), stored in a repository of unstructured texts. The process is carried out in three stages: firstly, clustering of the Offer document with the set of PHDs which are most similar to it; this provides the initial indication of whether similar previous projects exist, and signifies similarity. Secondly, determination of which PHD in the set is most comparable with the Offer document, based on various parameters: project effort, project duration (time), project resources (members/size of team), costs, and sector(s) involved, indicating comparability of projects. The comparable parameters are extracted using the GATE Natural Language Processing architecture. Lastly, a recommendation of metrics for the new project is made, which is based on the transferability of the metrics of the most similar and comparable PHD extracted, here referred to as recommendation.

Keywords: Ontologies, Software Metrics, Semantics, GATE, Natural Language Processing.

1. Introduction

The importance of software in today's industry is without doubt. Given the critical role of software, the requirement for project plans adjusted for time, effort, cost and quality has become a fundamental element for organizations producing software. Demonstrating the advancement of the field, since the end of the 1970s until the present, initiatives have been developed which aim to accurately plan projects in relation to their actual realization. In this environment, outsourced software services are drawn up in response to offer requests from the perspective of the invisible development process[1], that is, managers make their decisions based on their personal perceptions rather than on contrasted data. Various authors have proposed the use of metrics to improve software development's visibility, for example [2, 3]. Many years ago, Basili [4] wrote "All the data collected on the project should be stored in a computerized data base. Data analysis routines can be written to collect derived data from the raw data in the data base". It is precisely this statement which is the motivation of the current work, adapted to present-day – to recollect metrics and parameters of past projects with the objective of planning future projects with better precision, based on the Offer documentation.

The system is a tool to enable organizations embarking on new software development projects to utilize automatic benchmarking, as it compares the Offer document of a new project with sets of similar PHDs,

and consequently recommends metrics based on the PHD which is most comparable to the Offer document from a set of similar PHDs. Benchmarking is implemented, because only the metrics from the most comparable PHD document are recommended, whose appropriateness in the project has already been proven in the project previously completed.

The paper consists of the following sections. Section 1 introduces the setting of the research for the software development process, in particular with regard to software engineering metrics. This is followed by an introduction to the theory of information extraction, and an overview of the Natural Language Processing (NLP) techniques used for the practical implementation of such tasks, namely the GATE (General Architecture for Text Engineering) architecture and document clustering methods. Section 2 describes the architecture of the system and the components it is comprised of. Section 3 presents a use case which illustrates the uses of the system, and Section 4 discusses conclusions and future research work.

1.1. *Software Metrics*

In short, according to Boehm [5], software metrics help us to make better decisions. The first book dedicated to describe Software Metrics dates from 1976 [6], but the history of active software metrics dates back to the mid-1960's when the Lines of Code metric was used as the basis for measuring programming productivity and effort [7] . Thus, as has just been mentioned, the first book dates from 1976, but the first initial efforts to use metrics, in this case, Lines of Code, dates from 1971. The focus of this effort was to oversee the quality of software produced. Another study [8] dealt with module defect density (number of defects per KLOC) in terms of the module size measured in KLOC.

Fenton and Pfleeger [9] classify software metrics into three main categories: product, process and resources metrics. According to this taxonomy, personnel metrics are under resources metrics category. Without a doubt, the effective combination of the three categories produces hybrid metrics rich in information, in relation to individual and group productivity. The current work is focused on these types of metrics as well as the central category of personnel metrics.

More precisely, a metric is a quantifiable measurement of software product, process, or project that is directly observed, calculated, or predicted [10]. Considering this definition, software metrics may be obtained by means of observation, they may be calculated, or predicted. Additionally, within the metrics universe the research work is focused on the establishment of these types of metrics, in particular, metrics related to personnel factors, such as skills, experience, work load, and productivity.

1.2. *Personnel in Software Metrics*

The decision to concentrate the research on personnel metrics was not taken trivially. According to [11], Personnel attributes and Human Resource activities provide by far the largest source of opportunity for improving software development productivity. Previous work by [5] states that "After product size, people factors have the strongest influence in determining the amount of effort required to develop a software product". Failure rates in software projects are high and the qualified software engineers able to deal with software development processes, and their shortcomings and caveats [12] represent a scarce resource. Software development teams are composed of professionals with a heterogeneous training, background and expertise [13], that management must be able to evaluate and provide with a professional view, with the ultimate goal of improving the competences of the workforce and their results [6].

Taking into account, on the one hand, the importance of personnel in development projects, and on the other hand, the benefits of reliable metrics for the most appropriate estimation and constant improvement of the software process, the current work proposes an architecture capable of extracting personnel related metrics using Natural Language Processing techniques.

These metrics will be extracted from repositories of metrics generated through the application of Natural Language Processing to repositories of documents. Numerous authors have carried out work in the field of the use of software repositories applied to software metrics, in relation to their design [14], or their application to specific problems in the field of software engineering [15]. Concerning the software industry, since the 1970s initiatives for repositories of metrics have emerged, for example, DACS Productivity Dataset (http://www.thedacs.com/databases/sled/prod.shtml), The architecture research facility (ARF) dataset, (http://www.thedacs.com/about/services/pdf/Data-Brochure.pdf), the NASA/SEL Dataset

(http://www.dacs.com/databases/sled/sel.shtml), or the repository of the International Software Benchmarking Standards Group (http://www.isbsg.org/). The proposed research work does not aim to be based on data unconnected with an organization, rather, it is focused on an organization's own data. Thus, taking account of documents generated in previous projects, the objective is to automatically construct a set of metrics relative to project personnel, taking advantages of the functionalities provided by Information Extraction with Natural Language Processing

1.3. *Information Extraction with Natural Language Processing*

The use of NLP to derive parameters related to project size, effort, time, and resources is an example of the application of computational techniques which originate in the Information Extraction (IE) field. Information Extraction refers to the processing of free (unstructured) text documents in order to annotate them with a meaningful, predefined structure relevant for a specific task, and readable by a particular system. Other definitions have been proposed by [16], who refers to information extraction as the identification of instances of a particular class of events or relationships in a natural language text, and the extraction of the associated features of these entities. The problem at hand is usually restricted to a defined domain, in other words, it is domain dependent. In the case of this work, it is evident that the application of the techniques is limited to the software engineering domain. The information derived from a text may be divided into particular categories of linguistic content, such as named entities; references to people, locations, names of corporations (proper nouns) and numerical and temporal expressions, attributes associated with the entities, for example, a person's job title, real world facts, and events.

Systems which represent the information captured in software engineering documentation for knowledge reuse in itself is not new, however, the current architecture uses automatic extraction of features specifically for the recommendation of metrics. A system which specifically applied the extraction of linguistic information for the task of validation of software documentation is the SIFT (Specific Information from Text) system [17], which was executed on online software reference manuals and help systems semi-formatted with XML. The system extracts sentences and their semantics defined by a linguistic formalism, the generative lexicon [18]. The system was used to evaluate an online help system for the Adept series of structured editors [19]. In particular, sentences which defined specific information in the description of the repository API were extracted, those referring to the return codes for routines for accessing document and document fragments stored in an external repository. Thus, an extremely useful functionality of sentence extraction using NLP techniques is exhibited in this situation: a developer noted that one of the routines contained an incorrect return code, and by using SIFT, 38 sentences about error return codes out of 46 descriptions of routines were extracted automatically and could be verified for accuracy.

Álvarez-Macías et al. [20] evaluated the performance of the application of two data mining algorithms to the values of the attributes which comprise a companies' management process, such as staff hiring, staff dismissal and staff adaptation, to construct rules which measure the influence of these variables on outcome parameters like effort assignment, personnel, and delivery time. The algorithms tested were based on Evolutionary Algorithms, GAR, an unsupervised method which builds association rules between the variables in projects, and ELLIPSES, a supervised classification method which constructs mathematical regions for project parameters and determines which rules are most appropriate for each region.

The current paper focuses on the novel application of IE methods by specifically extracting information relevant to project planning and organizing, and thus the associated metrics. The extraction of the objects in the current work which refer to size, time, effort and resources are an example of an IE task known as noun chunk extraction, where the items extracted are noun chunks predefined by JAPE grammar rules, which will be explained further below. To perform this task, the GATE (General Architecture for Text Engineering) platform has been incorporated into the platform, and the capabilities of the language of GATE, JAPE, have been exploited. JAPE can be used to recognize the regular expressions contained in the text annotations made by GATE. A brief overview of the functionalities of GATE will be given, and its accompanying rule recognition language JAPE.

1.3.1. *GATE*

GATE is a NLP architecture specifically designed to perform the tasks referred to above, for example, Named Entity Recognition and Coreference resolution , determining attributes associated with entities, which indicate equivalence between entities. However, these are just two examples of the functionalities of GATE. In fact, it consists of three principal components which enable the execution of a host of adaptable language engineering tools, whose successful functioning has been demonstrated in a number of IR tasks throughout literature.

Despite of GATE is a well know tool that enables NLP, it is still present in many recent research projects E.g. [21,22,23,24].

The main elements of GATE are comprised of an architecture for language processing, a Java framework which forms the backbone of such a system, and a graphical development environment which allows manipulation of the framework for language engineers to build their own personalized language engineering tools and processing resources. GATE initially comes with a set of built-in processing resources, referred to in the platform as ANNIE (A Nearly New Information Extraction System). These are linguistic tools which have specific language processing functions [25], namely a tokeniser, gazetteer, sentence splitter, POS (Part of Speech) tagger, named entity transducer and an orthographic name-matcher.

The Gazetteer component of ANNIE is particularly useful in this architecture, as it consists of a set of predefined lists of nouns. Each item in the list has been pre-assigned an attribute, for example, organization, currency_unit, or manufacturer. The attributes are input to JAPE grammars (discussed below). This functionality enables the identification of the resources used in projects, for example, “IBM Requisite PRO”, or “CSW”.

When documents are processed by GATE, they are input to what is referred to as a GATE document pipeline, and the language processing tasks are performed sequentially. The language used to modify the capabilities of the processing resources is called JAPE. A description of JAPE is given below.

1.3.2. *JAPE*

Fundamentally, JAPE provides a tool for language engineers to define the characteristics of the sentences or phrases which they wish to extract in the particular application in question. It is a language for matching GATE annotations to regular expressions, thus it is essentially using pattern matching to construct more annotations using finite state autonoma. The patterns are defined as rules, a JAPE Grammar, which constitute a finite state machine. The rules are invoked on each text in sequence when it is input to the GATE document pipeline, as previously described.

In the current architecture, the first step is the clustering of the Project History Documents (PHDs) using a document clustering technique, an overview of which will be provided below. Once the PHDs have been grouped, and the input Offer document is grouped with the most similar set, the PHD which is most similar to the to the Offer document is determined by automatic analysis of the sentences which refer to project effort, time, and resources. It is the application of JAPE rules which allow such a comparison between relevant content of the two documents. Therefore, in order to extract all of the relevant phrases, the JAPE rules search for all possible sequences of annotations which match the rules. For the current research, the aim is to construct rules for phrases which indicate comparability of projects, such as project size, costs, sector, effort, time, and resources. Options exist for assigning priorities to the application of rules, that is, for example, if several phrases match the rule, only the one which matches the longest set of annotations from the input is accepted. In natural language, such a rule may be written as "If the sequence of tokens 'staff', 'hours' is preceded by a numerical value annotation, then create a new Time annotation for the three tokens" This sample rule will match the phrase "200 staff hours" as a name of the time involved in the project and annotate it accordingly, even if this phrase is not included in Gazetteer lookup lists. Equivalently, the same rule could be written for the sequence of tokens ‘man’ ‘hours’ preceded by a numerical value. This enables extraction of all variables which refer to time in hours. Or, for example, a similar rule could be used to identify that the phrases “6 team members”, “team of 6” and “team size of 6” refer to a team size variable. The rules which are written here in natural language are converted to a formal JAPE grammar.

1.4. *Document Clustering Techniques*

As mentioned above, the first component of the architecture groups the PHDs using a document clustering method. A support vector machine (SVM) has been used to cluster the documents, however, any of a number of clustering methods could be applied to perform document similarity measures.

The first stage of knowledge acquisition and reduction of complexity concerning a group of objects is to partition or divide the objects into groups based on their attributes or characteristics [26].

Document clustering is a form of unsupervised machine learning, which given a set of input documents, extracts features from the documents and groups the documents into clusters based on the presence or absence of the features. Document clustering has been defined by [27] as "Cluster analysis is the art of finding groups in data". Defined formally, D denotes a domain of documents and C = {c1, c2, c3, ...c|c|} a set of categories. The pair (di, cj) represents (document, category). A Boolean value b ∈ {T, F} is assigned for each pair (di, cj) ∈ D × C, where the value T indicates that the document di will be attributed to class cj, and the value F implies that the document will not be assigned to the class [28]. This definition has been defined in the context of text classification, where the set of categories is defined a priori by the automated classifier user. The essential difference introduced by text clustering techniques is that classes are not previously defined, instead the clustering algorithm constructs the classes based on feature frequencies and/or weights assigned to features. Examples of machine learning approaches for text clustering include bisecting K-means, Support Vector Machines, Latent Semantic Indexing, Naïve Bayes, K-medians. Additionally, these techniques may be divided into two groupings: the K-means method and agglomerative hierarchical methods. This division may also be viewed as the division between partitioning algorithms such as k-means or k-medoid, and hierarchical algorithms such as Single-Link or Average-Link [27].

1.4.1. *Vector Space Model*

The majority of document clustering techniques which have been proposed in the literature apply the Vector Space Model [29]. The vector space model is an algebraic model used for information filtering, information retrieval, indexing and relevancy rankings. It represents natural language documents (or any objects, in general) in a formal manner through the use of vectors (of identifiers, such as, for example, index terms) in a multidimensional linear space. Each document is represented by a vector in the term space. The set of terms is a predefined collection of terms, for example the set of all unique words occurring in the document corpus. Relevancy rankings of documents in a keyword search can be calculated, using the assumptions of document similarities theory, by comparing the deviation of angles between each document vector and the original query vector, where the query is represented as same kind of vector as the documents.

1.4.2. *Neural Networks*

A neural network (NN) model is an artificial intelligence framework which is closely related to SVMs, as both models involve machine learning. As with SVMs, the NN is trained to learn from examples. The techniques are similar in the sense that they both consist of a black box which can 'learn'; the feature values are the input to the box, and the output, the class the text falls into.

1.4.3. *Latent Semantic Indexing*

In the SVM model, frequency vectors are normalized for text length and may be allocated importance weights. Zipf's law is the factor that underlies normalization and the assignment of weights to features in the SVM calculations, as it is a mathematical model which assumes that the frequencies of common linguistic features in texts are high, and that frequencies decrease proportionally. However, even when weights are assigned to features, the construction of vectors is based on the assumption that the features are independently distributed. The existence of semantic relations in text such as synonymous and polysemous words breaches this assumption. Latent Semantic Indexing (LSI) is a model of text categorization which attempts to overcome the presence of ambiguous lexical relations in texts. Sebastiani [30] describes LSI as a method of dimensionality reduction by term extraction which exploits the inter-relationships between synonymous, near-synonymous and polysemous lexical relations. It is viewed as a dimension reduction technique because it is a similar term extraction model to SVMs, but the vectors have a lower-dimensional space, as their dimensions are generated from the

patterns of co-occurrence in the dimensions of the original vectors. The terms extracted represent the 'latent' semantic relations in the texts.

1.4.4. *Support Vector Machines SVMs*

SVMs were alluded as a particular model of machine learning. In this technique, which was proposed by Vapnik [31], the model for classification is generated from the training process with the training data. Owing to its usefulness, it has been widely adopted in various fields of classification problems in recent years, including medical diagnoses [32], tourism projections [33], sound processing [34] or recommender systems [35].

The SVM algorithm exploits the use of vectors which model the distributions of features in texts. Each vector is a point in a n-dimensional space (n is the number of features) , which can hold either a Boolean value signifying whether or not the feature exists in the document, or the frequency of occurrence of the feature [28]. The objective of SVM modeling is to define the optimal line (hyperplane) which divides groups of vectors into separate categories. In its simplest form, SVMs can be used to differentiate two categories. The support vectors are the vectors in closest proximity to the line. The task is to determine which of these vectors best describe the division between the two categories. Diederich and Kindermann [36] refer to the distance of the hyperplane which separates the two categories as the maximum interclass distance, the margin.

SVMs can also be used when points are categorized by a non-linear region, which requires a non-linear model. Frequency vectors are generally normalized to account for text length, and the raw feature frequencies or log-transformed feature frequencies may be assigned importance weights. The primary advantage of SVMs for clustering is that they can measure thousands of features, if necessary all of the n-grams in the text.

2. BMR: Benchmarking Metrics Recommender

The current section describes the architecture of the system. The component which provides the initial interaction of the customer with the system is the web-based user interface, which has functionalities for uploading two classes of documents: the SPO and PHD. HDs may be uploaded at company level. For example, the manager of a software development company can upload the entire set of PHDs of the company, and continue to upload them systematically as new products are developed over time, or he can upload a number of PHDs which he considers to be related to an Offer document he is about to upload. All of the documents uploaded are later stored in two separate repositories: a rich PHD repository and an SPO repository. Metrics are extracted from each PHD during the Natural Language Processing phase, and stored in a Metrics Repository. In order to clearly illustrate the architecture, the repositories have been described as three distinct components. However, the three repositories together in fact comprise a single repository, and may be conceptualized as one repository with three different parts.

Each document which is input to the system, regardless of whether it is a SPO or a PHD, is first subject to text processing. This is the first step of the algorithm. The specific components of the architecture are described in Fig. 1, which illustrates the architecture. Mentioned architecture consists of the following components:

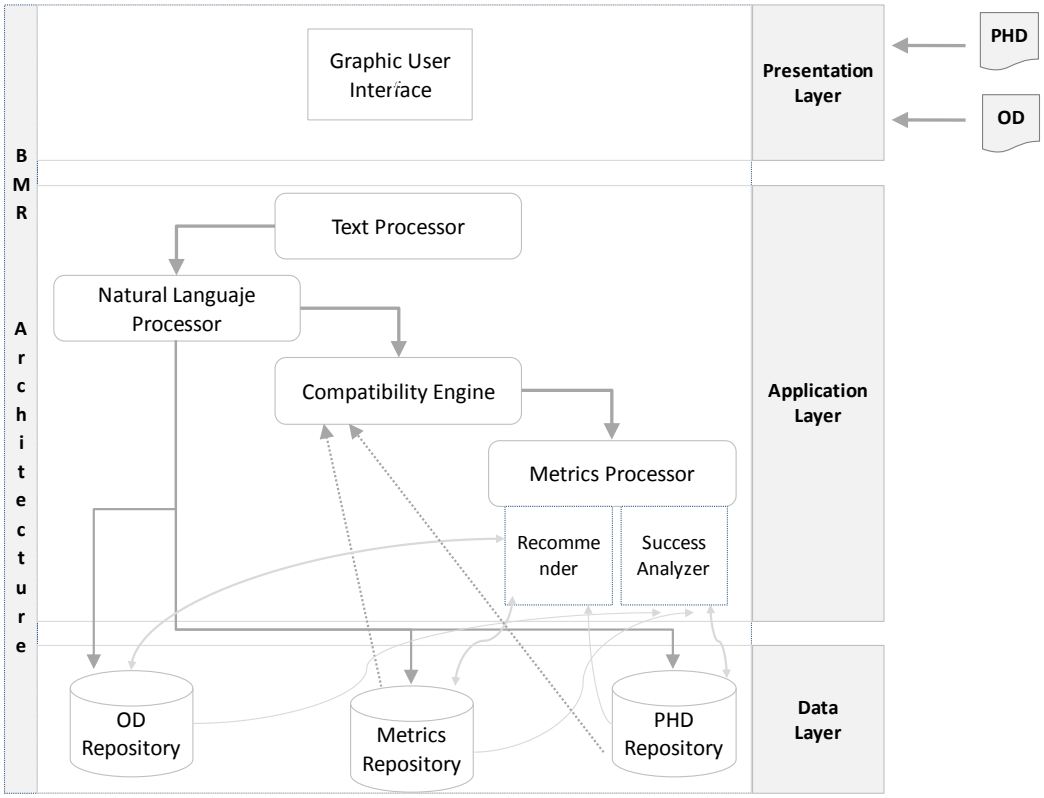

Fig. 1. BMR Architecture.

Text processor

This component converts each input document to plain text (including tables or graphics which contain text), and extracts the sections of each document which contain information relevant for measuring document similarity. These sections are then concatenated in order to re-construct the document into its final format, prior to its input to the GATE NLP pipeline. In existing NLP software architectures, often GATE is used to perform all text processing required. However, in the current architecture, in order to extract relevant parameters, specific sections of the PHDs, SPOs, and Metrics documents which contain parameters related to project comparability are required. Thus, it was decided to carry out pre-processing of the documents in order to parse only the sections needed for comparison. This also contributed to computational efficiency, by reducing processing time. Three parts of the document content are relevant for extraction:

a) Description of the Project
b) Software Production factors such as effort, time, resources, costs, and industry sector. These comprise the comparability variables.
c) Text relating to project metrics – those which comprise part of the content of the PHDs. The specific metrics extracted are later transferred to a separate repository containing project metrics relevant to each particular document, during the Natural Language Processing phase.

Natural Language Processor

Each text which is uploaded becomes part of a GATE document pipeline. All of the NLP tasks in GATE which are required in any particular application can be executed on each of the documents in the pipeline in sequence. In the architecture, GATE libraries are used to perform the following NLP tasks:

a) Syntactic annotation of noun phrases, using GATE's NP_Chunker
b) Application of JAPE rules to extract all phrases related to project comparability. The only comparability factors extracted which do not have a numerical value associated with them are the variables which describe the industry sector. These are annotated noun phrases such as "fish stock management application", or "bookmarking web application". Gazetteer lists

have been added to GATE with phrases which define each sector. The lists have been grouped according to EU industry specifications (http://ec.europa.eu/enterprise/sectors_en.htm). If the industry is not mentioned or has not been annotated correctly by a JAPE rule, it is omitted.

The GATE NLP component results in the storage of each PHD and each SPO in their respective repositories, with an associated list of comparability variables. For example, PHD DOCID1 will have a list of variables with corresponding values, for example, Variable Q, Value 100, Variable R, Value 200, Variable S, Value 3…Variable N, Value n. Each SPO document will have an ID and a similar associated list of variables.

The metrics from each PHD uploaded are also extracted and stored in a Metrics repository. In the current research work, this process is referred to as Metrics Extraction. The repository will contain a list of metrics associated with each PHD, for example, PHD DocID1 will have Metric 1, Value a, Metric 2, Value b…Metric n, value x.

Comparability Engine

The annotated documents are the input to a comparability engine. In the comparability engine, the PHDs are clustered using a text clustering algorithm. Each time a PHD is added to the repository, the document is clustered with the group of PHDs to which it is most similar, through the application of a Support Vector Machine algorithm based on lexical content. The comparability engine has three main functions:

a) Clustering of each PHD document using a Support Vector Machine
b) Clustering the input Offer Document with the most similar set of PHDs
c) Gauging comparability of the Offer document with each of the PHDs in this set, based on comparability variables.

The output of the execution of the Comparability Engine results in the following structure. The metrics associated with each PHD have already been stored in a metrics repository, as they were extracted from each PHD during the text processing phase. Therefore, the output of the comparability engine is the PHD which is most similar to the input Offer document (based on the values of the comparability variables), and its associated metrics. The most appropriate metrics from the most similar PHD are then recommended to the user by the metrics recommender. The metrics engine will now be described.

Metrics Processor

The metrics processor consists of two components: a Metrics Recommender, and a Success Analyzer.

The Metrics Recommender proposes the metrics for the input Offer document to the user, based on the variables described above. It is at this point where the novelty of the system is exhibited: not only does the system recommend suitable metrics, but the user can modify his SPO based on the metrics recommended, and a subsequent evaluation of the success of the metrics suggested for the Offer is performed. This is carried out by the Success Analyzer.

The Success Analyzer can be described as follows. As part of the metrics recommender phase, the user modifies his Offer document based on the new metrics recommended. He then carries out the software development process according to the revised Offer document, with new values for each variable. This results in the production of a PHD. The values of the comparability variables for the following documents are thus available: SPO (Version 1), SPO (Version 2), and PHD (based on Version 2). Thus, it is possible to apply an algorithm to perform the following comparisons: the similarity of SPO V1 and SPO V2, the similarity of SPO V1 and the PHD, and the similarity of SPO V2 and the PHD. The metrics in the PHD are then assigned weights according to their actual importance in SPO V2. The metrics and their corresponding weights are then transferred back to the metrics repository.

A simple mathematical algorithm is applied to determine the distance between the comparability variables, and thus, the similarity of the variables in the documents. For example, it is clearly possible to evaluate that two projects with respective team sizes of 6 and 10 are more similar than two projects of team size 6 and 150. During the execution of the algorithm, it may result from the processing of the values in the PHD (generated from Version 2 of the Offer document) that in fact the value associated with the metric "New Recruitment Rates" in the PHD is high, indicating the importance of this metric. This leads to the consequent assignment of weights accordingly.

In future research, it is intended to evaluate the effect of a number of weighting schemes on the performance of the architecture (E.g. [37, 38]).

The objective of the system is achieved, as the most appropriate metrics are recommended based on principally numerical comparability variables, and subsequently assigned importance weights as a measure of their suitability. The weights are continually adjusted based on the variables, thus as the volume and range of PHDs and SPOs in the repositories increase, the appropriateness of the metrics correspondingly becomes more refined.

3. Use case scenario

The use of the BMR architecture presents two distinct use case scenarios, which can be differentiated by the type of document under processing. The first use case which illustrates the function of the tool is the processing of the PHD. The processing of the PHD document enables the user to generate a repository of metrics and establish the variables associated with each document. It also allows the user to process a PHD which may be part of a batch of PHDs produced in the company, or the result of a project currently being carried out in the company, with the objective of uploading several documents in order to generate accurate data, and keep the repository up to date. The second use case scenario enables the user to upload an Offer document in order to receive metrics recommendations for a prospective project, once the first completed version of the Offer is prepared.

In order to set the scene for the use case, it is assumed that the PHD repository is already populated, and that the repository of the metrics extracted from the PHDs analyzed has been created, including the weights of the metrics as a function of their suitability for being used in projects. At this point, the company QMECC (fictional name) has drafted an offer document entitled OD_1. The offer is a document outlining the work plan for the parameterization of an ERP (Enterprise Resource Planning) system in the environment of an editorial company, in particular, focusing on billing and stock management. In order to carry out the customized tasks, it is established that a group of 7 consultants is required (2 senior, 5 junior), lead by a project leader over a 3 month time period, with a total effort of 1 month per staff member, in the case of the project leader, 2 months per staff member for each of the senior consultants, and 3 months per staff member for each of the junior employees.

Using the BMR Graphic User Interface, the document is uploaded and temporarily stored on the server, with its pending destination being the text processing component. The text processor converts the file to plain text and extracts the sections of the document which contain the details of the comparability variables relevant for NLP processing. Thus, the output is a plain text file, which contains the relevant sections of the SPO. This intermediate product is sent to the NLP component, which, in the case of the PHD documents, extracts metrics, and in the case of the both the PHD and the SPOs, determines the comparability variables and their values. Once this process in completed (described in the Architecture section), it is established which cluster of PHDs is most similar to the SPO. Subsequently, the SPO is compared to all of the documents in this cluster, in order to determine to which PHD it is most comparable. Once the most comparable PHD is established, the Metrics Processor makes a recommendation of metrics, which it is able to extract from the Metrics Repository. The repository contains the metrics associated with all of the PHDs in the cluster to which OD_1 is most similar.

With the objective of ensuring traceability of the recommendations, not only is the Offer document stored, but the metrics which have been recommended are also stored. At this moment, using the metrics information provided, the user has the possibility to incorporate the metrics recommended into his project planning, and subsequently upload a new version of the SPO. The variables in the new version of the SPO may have changed, based on the metrics previously recommended.

Suppose that the metrics relating to staff productivity which were suggested have implicated a mayor increase in the number of hours required by the project leader in client supervision tasks (Metric MT1), corresponding to 1.5 months of staff hours. This circumstance implies a significant additional cost for the company. Based on this analysis, the user can incorporate SPO document OD_2 to the repository.

If the offer presented is accepted by the client, QMCC has the possibility to reload the system with the PHD of the project which it has undertaken. Thus, at this point, the system contains OD_1 and OD_2, the PHD and their associated variables. OD_2 is uploaded and processed by the components in the usual way the system processes documents, but with the final objective of enriching the PHD repository. It is evident that the metrics processing phase represents an extremely novel

function. The metrics processor, aided by the output of the NLP component, locates and stores the metrics extracted from the PHD in the Metrics Repository. The subsequent task consists of an examination of the success of the metrics specified in OD_1 and OD_2, and a comparison with the actual metrics in the content of the PHD, which were specified upon termination of the project.

In the current SPO of QMECC, it can be assumed that the metric recommended for MT1 was 20 days per staff member, while in the PHD, the value transpired to be 19 days per staff member. Additionally, regarding the effort variables, they changed from 1 month per staff member in OD_1 to 1.5 months in OD_2, in the case of the project leader. However, the PHD indicated that the actual number of months for this worker was 1.4. Supposing that this was the only metric indicated, the Success Analyzer would have the objective to evaluate the MT1 metric according to how valuable it was considered, and assign a weight accordingly.

4. Conclusions and future work

During the last decades, the specification of software metrics has arisen as one of the possible solutions to the software crisis. Initiatives have been produced to extract, structure and apply software metrics in organizations based on internal data and external projects. In this paper, on the one hand, we have presented a novel initiative based on the success of software metrics, and on the other hand, on the use of an organization's own information. With this approximation, the initiative is based on the data and metrics produced in the organization itself, whose current situation is reacted to by the recommendation of the most appropriate metrics.

At the point of the development of the framework, one of the first decisions which was required to be taken was the establishment of a set of metrics which were considered applicable. In this way, and due to the importance attached to personnel factors, it was decided to focus on these factors as those which would be recommended, selecting size, time, and cost metrics, among others. Particularly, those parameters which are a significant indication of the comparability of projects and are crucial decision factors for a corporation.

Numerous possibilities for future research work arose during the current research. With regard to the clustering of the PHD documents, it is intended to apply a supervised learning technique with the categories of the PHD documents established a priori, in order to measure the effect on the recommendation of metrics. Precision and recall measures may then be applied in order to evaluate the performance of the categorization. It may also be possible to build on [20] work and construct rules for the values of the variables extracted using a genetic algorithm, and use the rules generated to recommend metrics, by assigning the most appropriate metrics for the rules in each classification region.

Concerning the extraction of project parameters, additional variables which indicate project similarity could be extracted to determine their effect on the recommendation of metrics, by the inclusion of more complex JAPE rules. Additional metrics could be stored in the metrics repository in order for the system to be able to recommend metrics for a larger range of project types. It is also intended to test the application of different algorithms for assigning weights to metrics. A further objective of future work is to collect and evaluate user feedback about their experiences with using the system.

Acknowledgements

This work is supported by the Spanish Ministry of Industry, Tourism, and Commerce under the project SONAR (TSI-340000-2007-212), GODO2 (TSI-020100-2008-564) and SONAR2 (TSI-020100-2008-665) and the MID-CBR project of the Spanish Committee of Education & Science (TIN2006-15140-C03-02).

References

1. P. Hsia, Making Software Development Visible, *IEEE Software*, **13** (3) (1996) 23–26.
2. M. Butcher, H. Munro and T. Kratschmer, Improving Software Testing via ODC: Three Case Studies. *IBM Systems Journal*, **41** (1) (2002) 31–44.
3. L.H. Putnam and W. Myers, *Five Core Metrics: The Intelligence Behind Successful Software Management*. (Dorset House Publishing, NY, 2003).
4. V.R. Basili, Data collection, validation, and analysis. In *Tutorial on Models and Metrics for Software Management and Engineering*, ed. V. R. Basili (IEEE Computer Society Press, CA, 1980), pp. 310-313.
5. B. Boehm, *Software Engineering Economics*. (Prentice-Hall, Englewood Cliffs, NJ, 1981).
6. T. Gilb, *Software Metrics*. (Chartwell-Bratt, Cambridge, Mass, 1976).
7. N.E. Fenton and M. Neil, Software Metrics: Successes, Failures and New Directions, *The Journal of Systems and Software*, **47**(2/3) (1999) 149-157.
8. F. Akiyama, An Example of Software System Debugging, *Information Processing*, **71** (1971) 353-379.

9. N.E. Fenton and S. Pfleeger. *Software Metrics: a Rigorous and Practical Approach* (Thomson Computer Press, 1997).
10. R.T. Futrell, D.F. Shafer and L.I. Safer, *Quality Software Project Management*. (Prentice Hall, Englewood, NJ, 2002).
11. B. Boehm, E. Horowitz, R. Madachy, D. Reifer, B.K. Clark, B. Steece, A.W. Brown, S. Chulani, S. and C. Abts, *Software Cost Estimation with COCOMO II*. (Prentice Hall, Upper Saddle River, NJ, 2000).
12. R. S. Pressman, *Software Engineering: A Practitioners Approach* (McGraw Hill, New York, NY, 2005).
13. S. McConnell, *Professional Software Development*. (Addison-Wesley, Reading, MA, 2003).
14. W. A. Harrison, Flexible method for maintaining software metrics data: a universal metrics repository, *The Journal of Systems and Software* **72** (2) (2004) 225–234.
15. Y. Zhang and D. Sheth, Mining Software Repositories for Model-Driven Development, *IEEE Software*, **23** (1) (2006) 82-90.
16. R. Grishman, Information Extraction: Techniques and Challenges, In *Proceedings of the International Summer School on Information Extraction: A Multidisciplinary Approach to an Emerging Information Technology*, ed. M.T. Pazienza, *Lecture Notes in Artificial Intelligence* 1299 (1997) 10-27.
17. P. Lutsky, Information Extraction for Validation of Software Documentation, In *Proceedings of the 13th international conference on Industrial and engineering applications of artificial intelligence and expert systems: Intelligent problem solving: methodologies and approaches*, ed. R. Loganantharaj, G. Palm, M. Ali, *Lecture Notes in Computer Science* 1821 (2000) 29-60.
18. J. Pustejovsky, *The Generative Lexicon*. (MIT Press, Cambridge, MA, 1995).
19. Arbortext, Inc. Adept Online Help, Version 8.2., 1999.
20. J.L. Álvarez-Macías, J. Mata-Vázquez and J.C. Riquelme-Santos, Data Mining for the Management of Software Development Process, *International Journal of Software Engineering and Knowledge Engineering*, **14** (6) (2004) 665-695.
21. R. Gacitua, P. Sawyer and P. Rayson, A flexible framework to experiment with ontology learning techniques, *Knowledge-Based Systems*, **21** (3) (2008) 192-199.
22. Y. Li, K. Bontcheva and H Cunningham, Adapting SVM for data sparseness and imbalance: a case study in information extraction, *Natural Language Engineering*, **15** (2009) 241-271.
23. H. Harkema, J.N. Dowling, T. Thornblade and W.W. Chapman, ConText: An algorithm for determining negation, experiencer, and temporal status from clinical reports, *Journal of Biomedical Informatics*, In Press 2009.
24. A. Roberts, R. Gaizauskas, M. Hepple, G. Demetriou, Y. Guo, I. Roberts and A. Setzer, Building a semantically annotated corpus of clinical texts, *Journal of Biomedical Informatics*, In Press 2009.
25. H. Cunningham, GATE, a General Architecture for Text Engineering, *Computers and the Humanities*, **36** (2) (2002) 223-254.
26. R.K. Brouwer, Clustering feature vectors with mixed numerical and categorical attributes, *International Journal of Computational Intelligence Systems* **1** (4) (2008) 285-298.
27. L. Kaufman and P. Rouseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis* (John Wiley and Sons, New York, NY, 1990).
28. K. H. Lee, *Text Categorization with a Small Number of Labeled Training Examples*. Ph.D. thesis, School of Information Technologies, University of Sydney 2003.
29. G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Journal of Information Processing and management*, **24** (5) (1988) 513-523.
30. F. Sebastiani, Machine Learning in Automated Text Categorization, *ACM Computing Surveys* **34** (1) (2002) 1-47.
31. V. N. Vapnik, *The nature of statistical learning theory* (Springer, New York, NY, 1995).
32. E.D. Übelli, Multiclass support vector machines for diagnosis of erythemato-squamous diseases, *Expert Systems with Applications* **35** (4) (2008) 1733–1740.
33. X. Xin, R. Law, T. Wu, Support Vector Machines with Manifold Learning and Probabilistic Space Projection for Tourist Expenditure Analysis, *International Journal of Computational Intelligence Systems* **2** (1) (2009), 17-26.
34. X.Y. Wang, P. Niu and W. Qi, A new adaptive digital audio watermarking based on support vector machine. *Journal of Network and Computer Applications* **31** (2008) 735-749.
35. A. García-Crespo, J.M. Gómez-Berbís, Colomo-Palacios and F. García-Sánchez, Using Support Vector Machines for feature-oriented profile-based recommendations, *International Journal of Advanced Intelligence Paradigms* **1** (4) (2009) 418-431.
36. J. Diederich and J. Kindermann, Authorship Attribution with Support Vector Machines. *Applied Intelligence* 19 (1/2) (2003) 109–123.
37. R. Jin, C. Falusos, A.G. Hauptmann, Meta-scoring: automatically evaluating term weighting schemes in IR without precision-recall, In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, eds. D.H. Kraft, W.B. Croft, D.J. Harper, J. Zobel (New Orleans, Louisiana, United States, 2001) pp. 83-89.
38. P. Pantel and D. Lin, Discovering word senses from text, In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, eds. O. R. Zaïane, R. Goebel, D. Hand, D. Keim, R. Ng (Edmonton, Alberta, Canada, 2002), pp. 613-619.