Using Fuzzy Prototypes for Software Engineering Measurement and Prediction

JOSÉ A. OLIVAS, MARCELA GENERO, MARIO PIATTINI, FRANCISCO P. ROMERO
Department of Computer Science
University of Castilla-La Mancha
Ronda de Calatrava, 5, 13071, Ciudad Real.
SPAIN

Abstract: -The main objective of this work is to present an application of an extension of the original Knowledge Discovery in Databases (KDD) process called Fuzzy Prototypical Knowledge Discovery (FPKD) together with a FPKD based prediction model. This technique is applied to Software Engineering measurement. In order to get quality object-oriented information systems (OOIS), it is necessary to assess their quality focusing on diagrams which are available early in the development life-cycle, such as class diagrams. It is in this context where object-oriented measures are necessary to help designers evaluate internal quality characteristics of class diagrams, such as structural complexity, and based on these evaluations, predict external quality characteristics, such as maintainability which is (and will continue to be) one of the most critical OOIS quality characteristic. Hence, by means of the FPKD process we will build a prediction model for class diagram maintainability based on class diagram structural complexity metrics. Using the FPKD process we will search for fuzzy prototypes for characterising class diagrams maintainability, and later we will use these prototypes for predicting class diagram maintainability in a real case. The data used for prediction was obtained through a controlled experiment.

Key-Words: - Fuzzy Prototypes, Data Mining, Knowledge Discovery, Object-oriented Software Measurement, Software Quality, Complexity Metrics.

1 Introduction

There is a strong need for integrating Knowledge Discovery techniques and Software Engineering measurement. As Morasca and Ruhe [5] remark, this integration should be done in two directions: Software Engineering measurement will acquire a set of new and promising data analysis techniques and Knowledge Discovery will find a new application area. This paper shows a practical experience where we demonstrate the contribution of Knowledge Discovery in Software Engineering measurement.

The Knowledge Discovery can be used in Software Engineering for extracting knowledge from software empirical studies and save into a repository to be used in future projects. The knowledge obtained form empirical studies constitute a valuable asset that companies posses, based on which companies may assess their current products and if needed take improvement actions.

In Software Engineering it can not be expected to use the same measurement analysis techniques that are used in "exact" sciences, nor obtain the same degree of precision and accuracy. This occur due to the nature of software So that, it is necessary to find out other data analysis approaches. Is in this context where Knowledge Discovery can play a relevant role in extracting useful knowledge from Software Engineering empirical data.

For making sense of empirical data, we cannot focus on the analysis stage only but we have to consider the complete Knowledge Discovery in Databases Process (KDD) [2]. Fayyad et al. defined the KDD process as "the non-trivial process to identify valid, new, potentially useful and comprehensible patterns in data". The original KDD process was extended originating the FPKD process, which has the goal of searching fuzzy prototypes [10] from data. These prototypes form the foundation of a prediction model that can be used in different application domains. The FPKD process has been used to tackle several real problems, such as forest fire prediction, financial analysis or medical diagnosis, with very good results [6], [8]. This approach is more representative than standard approaches, because the use of an isolated algorithm or method over- simplifies the complexity of the problem. Statistical methods or decision trees (ID3, C4.5, CART) are only classification processes, and it is very important to include a clustering model for finding some kinds of patterns in the initial set of data. The use of fuzzy schemas allows us to achieve better and more understandable results, concerning patterns and prediction results.

Seeing the encouraging results obtained of the application of the FPKD process for building prediction models applied to different domains, we decided to use the FPKD process for building a model for predicting the maintainability of class

diagrams made using the standard modelling language, UML [7].

In one hand, we focus on UML class diagrams, because they constitutes the backbone of the OOIS, and they are available early in the OOIS development life-cycle. An in the other hand, we focus on maintainability because is one of the most critical quality characteristic [4]. Maintenance was (and will continue to be) the major resource waster in the whole software life cycle.

As a key artifact produced at the early phases of OOIS development life-cycle, the maintainability of a class diagram has a great impact on the quality of the OOIS that it is finally implemented, so that focusing on class diagram maintainability will be a good step towards getting better quality OOIS.

Is in this context where object-oriented measures are necessary to help OOIS designers to assess internal quality characteristics of class diagrams, such as structural complexity, and based on them predict external quality characteristics, such as maintainability. Hence, by means the FPKD process we will built a prediction model for UML class diagram maintainability based on class diagrams structural complexity metrics [3]. By this process, we will search fuzzy prototypes for characterising class diagrams maintainability, and later we will use these prototypes for predicting a real case class diagram maintainability. The data used for prediction was obtained trough a controlled experiment.

The early availability of those predictions could really help software engineers to take better decisions, soon in the OOIS development and allow them to do a better resource allocation based on these predictions.

This paper is organised thus: In section 2 we describe the steps of the FPKD process. In section 3 we apply the FPKD process for searching fuzzy prototypes that characterize UML class diagram maintainability; we describe a controlled experiment carried out for extracting empirical data to be used in the FPKD process. In section 3 we also show an example of prediction of the class diagram maintainability applied to a new real case. The paper ends with a conclusion and outlook to future work in section 4.

2. The FPKD process and the prediction model

At the moment, our ability to analyse and to understand great sets of data is far below our capacity to store them. A new generation of techniques and computational tools becomes necessary for the extraction of useful knowledge, because a fast growth of the volume of data generally occurs . These techniques and tools are the subject of a new field of investigation denominated KDD [2].

Traditionally the way to turn data into knowledge is via a manual analysis and a later interpretation. Normally this process is slow, expensive and highly subjective. In fact it becomes impassable in many domains like, for example, volumes of data that grow exponentially. When the scale of exploration, data manipulation and inference grows above the human capacity ,we look for the technology of computers to automate the process.

The task of finding patterns in data sets is known by different names, according to the different scientific communities, for example: extraction of knowledge, discovery of information, archaeology of data, processing of patterns of data, etc. The investigators in data bases, statistical experts and recently the enterprise and business communities mainly use the term Data Mining.

In this work, we use the term KDD to represent all the process of discovery of useful knowledge from data, with Data Mining as one step in this process (application of specific algorithms to extract models of the data), although other steps like preparation, selection and data cleaning, incorporation from appropriate previous expert knowledge and interpretation of the results are under consideration. Therefore, the KDD takes and contributes theories, algorithms and methods of fields like the data bases, machine learning, pattern recognition, statistic, artificial intelligence and approximate reasoning and Knowledge Acquisition in Expert Systems.

The KDD process (interactive and iterative) described by Fayyad et al. is shown in figure 1.

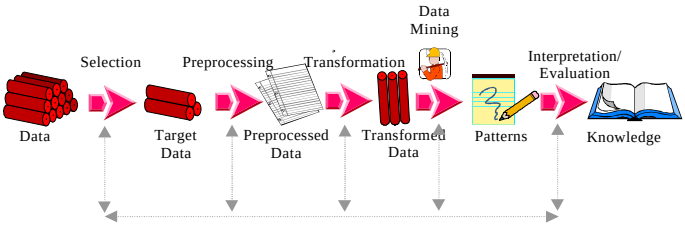

Figure 1. KDD process

The term pattern (in this work it will be denominated prototype of data) talks about a subgroup of data, along with a description and a model applicable to the same . The prototypes of data discovered must be valid for new data with some degree of certainty. These patterns must be new, at least for the system and preferably for the user, and potentially useful. Finally, these patterns must be comprehensible, if not immediately, after postprocessing. This definition implies that they must

be defined measures of the goodness of the prototypes of data; in many cases it is possible to define measures of certainty (capability of classification of new data) or utility (quality of the predictions on the basis of theses prototypes of data).

Taking the prototype theory of psychology as a reference, a single representation of ERD Maintainability could be seen as prototypical. However, in a previous approximation of the knowledge acquisition process we were able to observe that this representation excessively simplifies the behavioural guidelines of the experts. When a technician is confronted with a real situation he handles a range of prototypes determined by a series of factors and must decide which type of ERD maintainability is to be expected. Therefore, the prototype "ERD maintainability" is not unique.

Zadeh [10] mentioned the classical prototype theories from the point of view of psychology, criticizing precisely what we have just pointed out: that these theories do not fit the function that a prototype should have. Zadeh's approach to what must be taken as a prototype is less intuitive than the conceptions of psychological theories but is more rational and closer to the meaning of a prototypical concept displayed in a more detailed examination. In our case, we have observed that Zadeh's idea suggests a concept that encompasses a set of prototypes, which represent the high, medium, or low compatibility of the samples with the concept A. "The prototype is not a single object or even a group of objects in A. Rather, it is a fuzzy schema for generating a set of objects which is roughly coextensive with A" [10].

Based on these suggestions, modifications of the original KDD process are proposed, as represents fig 2. Which they involve incorporation of a new knowledge in different points and decisions of the users or experts. The aim must be to generate conceptual prototypes (Zadeh's approach: fuzzy schemas) that allow us to evaluate new situations from these patterns, and to establish predictions if these prototypes represent ordered series. The stages of the modified KDD called the FPKD are the following (see the top part of figure 2):

- *Selection*: Applying the knowledge of the dominion and excellent knowledge a priori, considering the objectives of the global process of FPKD, target data is created that will include selected sets of data or subgroups of excellent variables or examples.
- *Pre-processing*: Data cleaning, noise elimination, handling of empty fields, lost data, unknown values or by defect. Standard techniques of data bases are applied.
- *Transformation*: Reduction of the number of variables. Location of useful forms to express the data depending on the later use and on the objectives of the system. The expert knowledge and techniques of transformation and information in data bases are used.
- *Data Mining*: Selection of the algorithms of Data Mining. Decisions about the model that is derived from the algorithm of Data Mining (classification, summary of data, prediction). Search for interest patterns, as far as concerns classification, decision trees, regression, dependency, heuristics, uncertainty, etc.

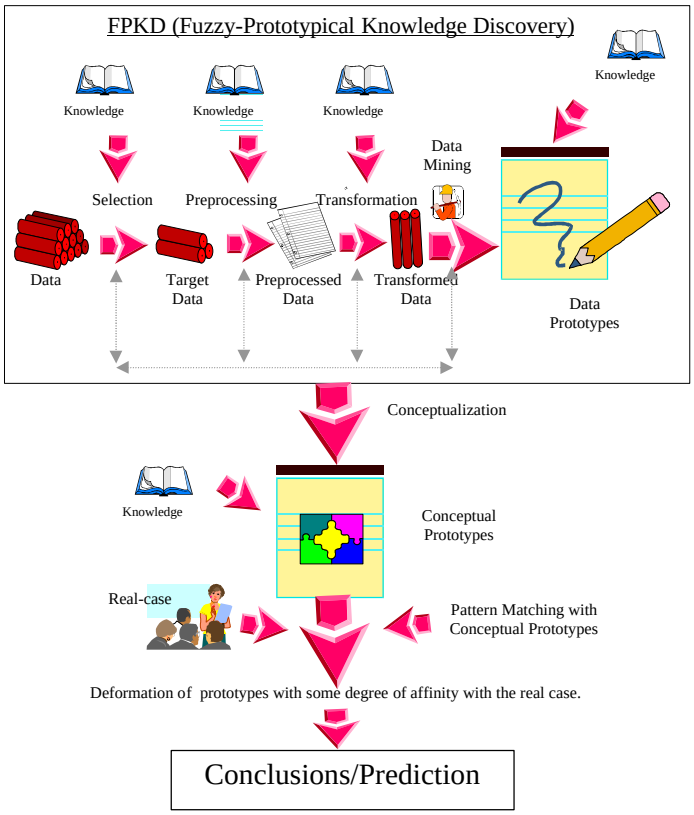

Figure 2. FPKD process and the prediction steps

Once the prototypes are found, for using them as the basis of the prediction model the formal representation must be done by the following stpes: (see the bottom part of figure 2):

- Calculate the centre of each prototype using the fuzzy prototyping data collection. Represent the prototypes in triangular fuzzy numbers.
- Combine the prototyping factors in order to obtain their affinity with each of the prototypes.
- Determine the current situation with the modification of the most similar prototype, with a linear combination using a degree of affinity with the prototypes as weight values.

3. A Practical Experience

In this section we explain a practical experience using the FPKD process and the prediction model described in the previous section.

Our idea is to use UML class diagram structural complexity metrics (see table 1) for predicting UML class diagram maintainability. So that, we will apply the FPKD to find the fuzzy prototypes for "class diagram maintainability", and later deform them and predict a new real case. The data used to built those prototypes was collected by a controlled experiment, which we describe in the next sub-section.

Metric name	Metric definition
NUMBER OF CLASSES (NC)	The total number of classes.
NUMBER OF ATTRIBUTES (NA)	The total number of attributes.
NUMBER OF METHODS (NM)	The total number of methods
NUMBER OF ASSOCIATIONS (NAssoc)	The total number of associations
NUMBER OF AGGREGATION (NAgg)	The total number of aggregation relationships within a class diagram (each whole-part pair in an aggregation relationship)
NUMBER OF DEPENDENCIES (NDep)	The total number of dependency relationships
NUMBER OF GENERALISATIONS (NGen)	Is defined as the total number of generalisation relationships within a class diagram (each parent-child pair in a generalisation relationship)
NUMBER OF GENERALISATION HIERARCHIES (NGenH)	The total number of generalisation hierarchies in a class diagram
NUMBER OF AGGREGATION HIERARCHIES (NAGGH)	The total number of aggregation hierarchies in a class diagram
MAXIMUM DIT (MaxDIT)	It is the maximum between the DIT value obtained for each class of the class diagram. The DIT value for a class within a generalisation hierarchy is the longest path from the class to the root of the hierarchy.
MAXIMUM HAGG (MaxHAgg)	It is the maximum between the HAgg value obtained for each class of the class diagram. The HAgg value for a class within an aggregation hierarchy is the longest path from the class to the leaves.

Table 1. Metrics for UML class diagram structural complexity [3]

3.1 A controlled experiment to search fuzzy prototypes for class diagram maintainability

Taking into account some suggestions provided in [1], [9] about how to do empirical studies in Software Engineering, we carried out a controlled experiment with the goal of collecting empirical data to be used for predicting class diagrams maintainability from metric values obtained at the early phases of OOIS life-cycle.

The experimental subjects used in this study were students enrolled in the third year of Computer Science at the University of Castilla La-Mancha. Even though the subjects are students, we consider they have enough experience in the design and development of OO software to do the kind of tasks required in the experiment. Moreover, subjects were given an intensive training session related to UML class diagram design and also about metrics applied to class diagrams at a high level design, before the experiment took place.

The independent variable is UML class diagram structural complexity, measured by the proposed metrics. The dependent variable is class diagram mainatainbility, measured by the time the subjects spend doing the experiment. This time is influenced by the time the subjects take to understand the diagram, which also have a great impact on the time spend in maintenance tasks, so we called it "maintenance time"

The subjects were given eight UML class diagrams of the same universe of discourse, related to Bank Information Systems. At first, they had to take each diagram write down the initial time, calculate 11 metrics (see section 3.2), and finally the final time. The difference between the initial and the final time is which we consider the maintenance time.

Once the experiment was carried out, and the measurement empirical data was collected, we needed to predict UML class diagram maintainability. For doing this, we have used the FPKD process and the concept of fuzzy deformable prototypes for establishing the prediction model.

3.2 Applying the FPKD process and the prediction model

We will explain each of the steps we have followed in the FPKD process, an we will also show how to predict class diagram maintainability, based on metrics values.

- Selection of the target data. We have taken as a starting set a relational database that contains 168 records (with 12 fields, 11 represent metrics values, 1 represents the maintenance time) obtained from the calculation of the metric values (for each class diagram) and the time spent by each subject doing the experiment, called maintenance time.
- Preprocessing. The Data-Cleaning was not necessary because we did not find any errors.
- Transformation. This step was performed doing different tasks:

Summarising subject responses. We built a unique table with 8 records (one record for each class diagram) and 12 fields (11 metrics and a field for the maintenance time). The metric values were calculated measuring each diagram, and the values for the maintenance time were obtained aggregating maintenance time using the mean of time.

Clustering by Repertory Grids. In order to detect the relationships between the class diagrams, for obtaining those which are easy, medium or difficult to maintain (based on the maintenance time), we have carried out a hierarchical clustering process by Repertory Grids. The set of elements is constituted by the 8 class diagrams and the clustering data are the medium maintenance time to accomplish an analysis of clusters on elements, we have built a proximity matrix that represents the different similarities of the elements, a matrix of 8 x 8 elements (the diagrams) that above the diagonal represents the distances between the different diagrams. Converting these values to percentages, a new table is created and the application of Repertory Grids Analysis Algorithm returns a graphic as a final result (see figure 3).

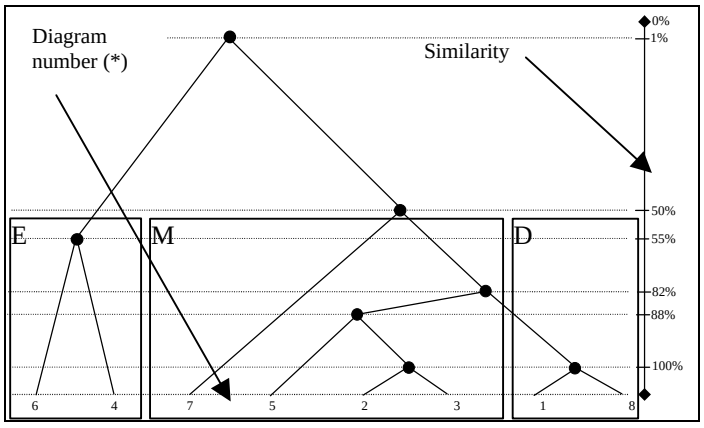

Figure 3. Clustering results (E: Easy to maintain, M: Medium to maintain, D: Difficult to maintain)

- Data Mining. The selected algorithm for data mining process was summarise functions (calculating factors such as medium, minimum and maximum time spent for maintaining each diagram, and finding for each one the average values). Table 2 shows the parametric definition of the prototypes. These parameters will be modified taking into account the degree of affinity of a new class diagram with the prototypes. With the new modified prototype we will be able to predict the maintainability of a new class diagram.

	Maintenance Time
Difficult	
Average	4 minutes 50 seconds
Maximum	16 minutes 5 seconds
Minimum	4 minutes
Medium	
Average	3 minutes
Maximum	10 minutes
Minimum	4 minutes 45 seconds
Easy	
Average	2 minutes
Maximum	5 minutes 24 seconds
Minimum	0 minutes 24 seconds

Table 2. Prototypes "Easy, Medium and Difficult to understand"

- Formal Representation of conceptual prototypes. The prototypes have been represented as fuzzy numbers, which are going to allow us to obtain a degree of membership in the concept. For the sake of simplicity in the model, they have been represented by triangular fuzzy numbers. Therefore, in order to construct the prototypes (triangular fuzzy numbers) we only need to know their centrepoints ("centre of the prototype"), which are obtained by normalising and aggregating the metric values corresponding to the class diagrams of each of the prototypes (see figure 4).

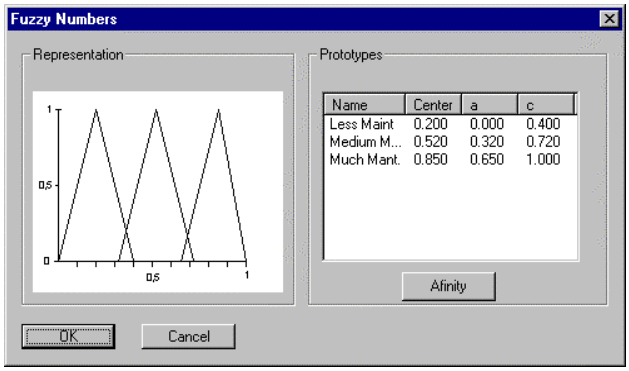

Figure 4. Representation of the prototypes

- Prediction of UML class diagram maintainability. Using Fuzzy Deformable Prototypes we can deform the most similar prototype to a new class diagram, and define the factors for a new situation, using a linear combination with the degrees of membership as coefficients. We will show an example of how to deform the fuzzy prototypes previously found. Given the following metric values corresponding to a new class diagram:

NC	NA	NM	NAssoc	NAgg	NDep	NGen	NAggH	NGenH	MaxDIT	Max HAgg
21	30	70	10	6	2	16	3	2	3	3

And their normalised values:

NC	NA	NM	NAssoc	NAgg	Ndep	NGen	NAggH	NGenH	MaxDIT	Max HAgg
0.5	0.49	0.6	0.8	0.67	1.0	0.7	0.6	0.4	1	1

The final average is 0.72. The affinity with the prototypes is shown in figure 5.

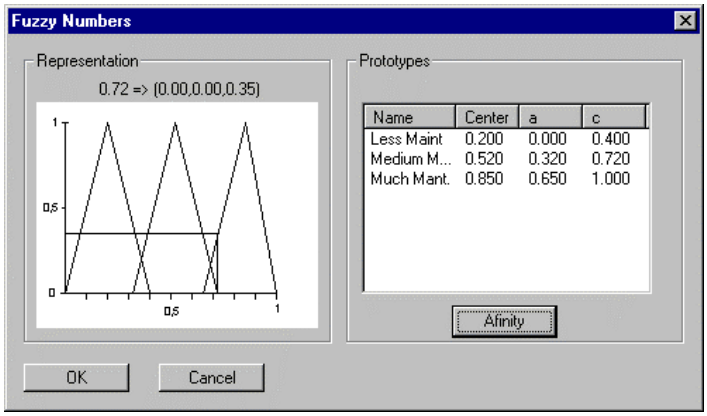

Figure 5. Affinity of the real case with the prototypes

The most similar prototype for this new class diagram is "Difficult to maintain", with a degree of membership of 0.35. Then, the prediction is shown in table 3:

	Maintenance Time
Average	2 minutes 45 seconds
Maximum	5 minutes 46 seconds
Minimum	1 minute 24 seconds

Table 3. Predicted value of the maintenance time

The predicted maintenance time could be useful to designers in order to have some insights about their designs early in the development life-cycle.

4. Conclusions and future work

We demonstrated in this paper that Software Engineering measurement and Knowledge Discovery can be integrated to provide fruitful results.

We applied and extension of the original KDD, the FPKD process for searching fuzzy prototypes for characterising class diagram maintainability. Based on these prototypes we have established a prediction model for UML class diagram maintainability by deforming the original ones. The input of the prediction model are the UML structural complexity metrics [3].

The early availability of a prediction model for class diagram maintainability, at the initial phases of the OOIS life-cycle, could really help OOIS designers to take better decisions in their design tasks, which is the most important goal that must pursue any measurement proposal if it pretends to be useful.

As future work we plan to apply the FPKD process to data obtained from real OOIS projects in order to corroborate not only the utility of the FPKD process and the fuzzy deformable prototypes but also the capability of UML class diagram structural complexity metrics [3] as early quality indicators.

Acknowledgements

This research is part of the DOLMEN project supported by CICYT (TIC 2000-1673-C06-06) and the CIPRESES project supported by CICYT (TIC 2000-1362-C02-02).

References:

[1] Briand L., Bunse C. and Daly J. (1999). A Controlled Experiment for evaluating Quality Guidelines on the Maintainability of Object-Oriented Designs. Technical Report IESE 002.99/E, Fraunhofer Institute for Experimental Software Engineering, Kaiserslautern, Germany.

[2] Fayyad, U., Piatetsky-Shapiro, G. and Smyth, P. (1996).The KDD Process for Extracting Useful Knowledge from Volumes of Data. Communications of the ACM, 39(11), 27 – 34.

[3] Genero, M., Piattini, M. and Calero, C. Early Measures For UML class diagrams. (2000). L´Objet. 6(4), Hermes Science Publications, 489-515.

[4] ISO/IEC 9126-1 (1999). Information Technology-Software product quality – Part 1: Quality Model.

[5] Morasca S. and Ruhe G. Knowledge Discovery from Empirical Software-Engineering Data. (2000). International Journal of Software Engineering and Knowledge Engineering. 9 (5). 495-498.

[6] Olivas J. and Romero F. (2000). FPKD. Fuzzy Prototypical Knowledge Discovery. Application to Forest Fire Prediction. Proceedings of the SEKE'2000, Knowledge Systems Institute, Chicago, Ill. USA, 47 – 54.

[7] Object Management Group (1999). UML Revision Task Force, OMG Unified Modeling Language Specification, v. 1.3. document ad/99-06-08., 1999.

[8] Olivas, J. A. (2000). Contribution to the Experimental Study of the Prediction based on Fuzzy Deformable Categories, PhD Thesis, University of Castilla-La Mancha, Spain.

[9] Wohlin C., Runeson P., Höst M., Ohlson M., Regnell B. and Wesslén A. (2000). Experimentation in Software Engineering: An Introduction. Kluwer Academic Publishers.

[10] Zadeh, L. (1982). A note on prototype set theory and fuzzy sets. Cognition 12, 291- 297.