A Partial-Repeatability Approach to Data Mining

Kai-Yuan Cai, Yunfei Yin, and Shichao Zhang *Department of Automatic Control, Beijing University of Aeronautics and Astronautics, Beijing , China*

***Abstract*—Unlike the data approached in traditional data mining activities, software data are featured with partial-repeatability or parepeatics, which is an invariant property that can neither be proved in mathematics nor validated to a high accuracy in physics, but still (partially) governs the behavior of the data. Parepeatics emerges as a result of the inaccurate universe. The universe comprises all possible C language programs is an example that cannot be accurately characterized since human writes defect-prone programs. In this paper we design a parepeatic mining framework for software data diming, where the mined knowledge is represented in terms of parepeatic models. A parepeatic model consists of central knowledge, a knowledge fluctuation zone and a correctness factor. Our approach can generate the required parepeatic model as a new form of knowledge representation from a given dataset and apply it to software data mining. Experimental results with real C language programs show that the proposed approach is effective.**

***Index Terms*—Knowledge representation, parepeatic model , parepeatics, Partial-repeatability, software data mining, uncertainty**

I. INTRODUCTION

Data mining is an active or vigorous area nowadays and has found extensive applications in market analysis, investment assessment, security guarantee, manufacturing process analysis, Web searching, and scientific data analysis, among others [1-4]. It attempts to reveal or discover valuable information or knowledge from vast amount of data. The valuable knowledge can be represented in terms of patterns, clusters, association rules, significant structures and so on. These forms of knowledge can be discovered using various approaches such as clustering techniques, decision trees, neural networks and case-based reasoning methods. In contrast with traditional statistical data analysis that is assumption-driven and applicable to analyzing modest amount of data, data mining is discovery-driven and applicable to analyzing vast amount of data.

This work was supported by the National Natural Science Foundation of China (60233020, 60474006, 60473067).

Kai-Yuan Cai is with the Department of Automatic Control, Beijing University of Aeronautics and Astronautics, Beijing 100083, China (corresponding author, phone/fax: +86-10-8231-7328; e-mail: kycai@buaa.edu.cn).

Shichao Zhang is an assistant president at the Guangxi Teachers University. He is also a senior research fellow at the Faculty of Information Technology, University of Technology, Sydney. Contact him at the Univ. of Technology, Sydney, Broadway NSW 2007, Australia (e-mail: Zhangsc@it.uts.edu.au).

Roughly speaking, there are two implicit assumptions underlying existing approaches for data mining. First, there are invariant valuable patterns or knowledge among the dataset under mining. Second, the intended knowledge can be represented in a conventional form such as equivalence classes (clusters), statistical models, decision trees, induction rules, and neural networks. The first assumption is concerned with technical aspects as well as non-technical aspects. From the technical viewpoint we need to consider if the intended knowledge can be mined from the given dataset or if the given dataset is minable. From the non-technical viewpoint we need to consider if the data mining process is cost-effective. The second assumption is mainly technical. For example, suppose the given dataset is $\{x_1, x_2, \ldots, x_n\}$ and we want to cluster them into a number of classes. Then existing clustering techniques [5] will generate a few disjoint classes of data, $\{C_1, C_2, \ldots, C_m\}$, from the dataset as the intended knowledge. The second assumption implies that $\{C_1, C_2, \ldots, C_m\}$ is an appropriate representation of the intended knowledge.

However, our previous work in software data analysis has revealed that second assumption mentioned above can hardly hold in software engineering as a result of partial-repeatability featured with software data unless new models of knowledge representation are introduced [6, 7]. For example, given a software program, we may calculate the number of lines of code, the number of distinct usages of operators, the number of distinct usages of operands, the number of total usages of operators, and the number of total usages of operands. Then are there any invariant laws that govern these five measures regardless of the particular features of various software programs? The laws, if any, can hardly be represented as a single absolute assertion such as a deterministic function $y = f(x)$ or a statistical model. A more appropriate representation form is the one that includes central function (knowledge), fluctuation zone, and the corresponding correctness factor [7]. Such a new model or representation form is intended to characterize the feature of partial-repeatability in vast amount of software data. By partial-repeatability it is meant that complex phenomena may demonstrate an invariant property that can neither be proved in mathematics nor validated to a high accuracy in physics, but still (partially) governs the behavior of the phenomena. (Conventional science and technology follows the top criterion of full repeatability that scientific arguments must either be proved repeatably in

mathematics or validated to a high accuracy repeatably (even in a statistical sense) in physics (experimentally)) This means, traditional data mining techniques are not appropriate to software data analysis/mining.

In this paper we design a parepeatic mining framework for software data mining for dealing with the partial-repeatability problem. The main contributions in this paper include:

The notion of partial-repeatability is further clarified as a new kind of uncertainty in comparison with randomness and fuzziness.

The parepeatic model[1] is treated as a new form of knowledge representation. This model comprises the central knowledge such as valuable patterns and association rules, a fluctuation zone, and a correctness factor. If the fluctuation zone contains only the central knowledge and the correctness factor is equal to one, then the parepeatic model reduces to an existing or conventional model of intended knowledge.

A new criterion is introduced to measure the quality of conventional clustering. This criterion takes account of not only the homogeneity within each cluster and separability between the distinct clusters, but also the number of distinct clusters.

A new approach is proposed for data mining, which generates a partial-repeatability or parepeatic model from the given dataset for the intended knowledge.

The rest of this paper is organized as follows. Section 2 clarifies the notion of partial-repeatability as a new kind of uncertainty. Section 3 presents the proposed approach for mining data with partial-repeatability. Section 4 applies the new approach to mining software data. Concluding remarks are contained in Section 5. The Appendix details our mining algorithm for generating the required parepeatic model of clustering from the given dataset.

II. Partial-Repeatability as a New Form of Uncertainty

As mentioned in Section 1, before data mining, one must decide what he or she wants to extract from the dataset under mining. The intended knowledge must be represented in an appropriate form. If no uncertainty is associated with the given dataset and what we want to extract from the dataset is a causality relation or deterministic function, then we can use classic or crisp sets to represent the relation or function. For example, suppose the given dataset is $\{\langle x_{i1}, y_{i1}, x_{i2}, y_{i2}, \ldots, x_{im}, y_{im}, z_i \rangle; i = 1, 2, \ldots, n\}$, where x_{ij} is the number of copies of books of type j sell at a book store of concern on the ith day, y_{ij} is the price of books of type j sell at the book store on the ith day, and z_i is the total income of books sell at the book store on the ith day. If we are interested in the causes of z_i being increasing or decreasing, or we are interested in whether z_i is a increasing function of x_{ij} or y_{ij}, then the answer is positive and crisp. There holds $z_i = \sum_{j=1}^{m} x_{ij} y_{ij}$. This relation is deterministic and gives all information for the intended knowledge. No uncertainty is associated with the answer or intended knowledge. It is sufficient for a crisp singleton set of deterministic function or relation to represent the intended knowledge.

However one may argue whether the crisp answers of this kind are really interesting or valuable. They look trivial. More often than not, we are interested in answers with uncertainty. For example, what is the underlying relation among $\langle x_{i1}, y_{i1}, x_{i2}, y_{i2}, \ldots, x_{im}, y_{im}, z_i \rangle$ and i ? Suppose the intended knowledge is represented in form of $f(i; x_{i1}, y_{i1}, x_{i2}, y_{i2}, \ldots, x_{im}, y_{im}, z_i) = 0$. Then how to determine or represent the intended relation f ? Obviously, uncertainty must be associated with f and the answer can not be deterministic. A natural framework to represent the underlying uncertainty is probabilistic or statistical. We may assume that f is a random function. If n or the number of days of concern becomes huge and the relation among $\langle x_{i1}, y_{i1}, x_{i2}, y_{i2}, \ldots, x_{im}, y_{im}, z_i \rangle$ and i looks over complicated, then we may exploit human experience or heuristic and assume that f is fuzzy function. No matter whatever models (deterministic, random, or fuzzy) are used to represent the intended knowledge, existing approaches for data mining assume that the intended knowledge is represented in terms of a single relation f. More specifically, the single relation can be a statistical model, a fuzzy model, a decision tree, an induction rule, a neural network and so on.

Now the problem is whether a single relation f is sufficient to characterize the intended knowledge. By adopting the single relation f, we implicitly follow the philosophy of full repeatability that have been treated as the top scientific criterion in thousands of years of history of conventional science and technology. By full repeatability it is meant that scientific arguments can either be proved repeatably in mathematics or validated to a high accuracy repeatably in physics (experimentally). The correctness of the scientific arguments should be independent of the investigators who present or prove them. It is the nature of full repeatability or high quantitative accuracy that enables existing physics or science to gain enduring respect and reputation. Existing approaches for data mining treat the intended knowledge as a conventional scientific argument that can be fully repeatable even in a statistical sense.

Unfortunately full repeatability may fall as observed in our

[1] Parepeatics is an abbreviated term for partial-repeatability, and accordingly, 'parepeatic' is the adjective form of 'parepeatics'.

previous work [6, 7]. For example, a course of fish may not be fully repeatable if cuisine is involved: no chef can make a course of fish twice at exactly identical sweetness or saltiness. The bones of a human face may be slowly evolving; the skins may be slightly faster evolving. The face pattern continues to evolve over age. One cannot assure that the face at two different time instants would be fully identical, although they might be essentially similar. Things of this kind are only partially repeatable. This can be further justified in software engineering. No human can write exactly the same program twice for a single software requirement specification. No software test process can be exactly repeated twice. The behavior of software systems and the software development process can not be fully repeatable. However different software systems produced from different development processes for a single requirement specification can work similarly and do not fail in most cases, although one is not quite sure how to measure or quantify "similarly" or "most" even in a statistical sense. Software systems and software development processes behave partially repeatably. For the example given at the beginning of this section, we can treat the underlying relation as a function of n, denoted as $f^{(n)}$. Then how can $f^{(1)}, f^{(2)}, \ldots, f^{(n)}, \ldots$ behave fully repeatably as a single relation f? In very complicated situations, can $f^{(1)}, f^{(2)}, \ldots, f^{(n)}, \ldots$ behave partially repatably and fluctuates among a number of typical relations? We argue in our previous work [6, 7] that there is something lying between full repeatability (conventional scientific arguments) and miracles (the unrepeatable). This is partial-repeatability by which it is meant that complex phenomena may demonstrate an invariant property that can neither be proved in mathematics nor validated to a high accuracy in physics, but still (partially) governs the behavior of the phenomena. Partial-repeatability is a new kind of uncertainty that is distinctly different from randomness and fuzziness. If we treat the deterministic physical laws as type I laws (where causality dominates), the statistical physical laws as type II laws (where randomness dominates), then we can treat the laws governing the phenomena of partial repeatability as type III laws. In quantitative terms, a single relation or formula describing a type III law is not valid to a high accuracy (actually, a number of relations should be given), and substitution operations of variables may lead to significant errors.

Simply, we can identify partial–repeatability as a new kind of uncertainty as follows. Suppose $U = \{u\}$ is the universe of discourse and can be accurately characterized. For example, U comprises all positive integers. Further, let A be an object of interest (e.g., integers greater than 6, integers around 9). A is a crisp set if we can absolutely assert $u \in A$ or $u \notin A$ for all $u \in U$. If in general $u \in A$ holds to some extent and $u \notin A$ to another extent simultaneously, then A is a fuzzy set. If for any $u \in A$ there must be $A = u$ or $A \neq u$, but we are sure which u leads to $A = u$ or $A \neq u$, then we say that A is a random variable. Randomness and fuzziness assume that the universe of discourse is characterized accurately. However in some circumstances it is nearly impossible to characterize the universe of discourse accurately. For example, suppose the universe of discourse comprises all C language programs. Since human writes defect-prone programs and defects are in various forms, it is impossible to describe all possible C language programs. Another example can be encountered when we need to extract invariant patterns from all possible human faces. How can all possible human faces be represented accurately if they constitute a single U? In this way uncertainty is associated with the universe of discourse. Partial-repeatability emerges as a result of uncertain universes, no matter whether the relation between the object of interest and the universe of discourse is crisp, fuzzy or random. The object of concern is accordingly referred to as a parepeatic object. An example parepeatic object is the statement that the number of distinct usages of operators is less than that of distinct usages of operands in a C language program. The statement can hardly be assessed in a fuzzy or statistical context since the universe of C language programs is not accurate.

For the book-selling example mentioned at the beginning of this section, the universe of discourse comprises all possible selling scenarios that may take place at the book-store. The numbers of copies of books of various types sell at the book store, the corresponding prices and the number of days of concern serve as the features or feature variables defined for the universe, and the given dataset $\{\langle x_{i1}, y_{i1}, x_{i2}, y_{i2}, \ldots, x_{im}, y_{im}, z_i \rangle; i = 1, 2, \ldots, n\}$ is an observation of the universe in terms of the feature variables. The interested object is just the intended knowledge that we want to extract from the observation for the universe in terms of the feature variables. If we can accurately describe all possible selling scenarios, then we can argue that the uncertainty associated with the intended knowledge acts as randomness, fuzziness or a mixture of them. Otherwise the intended knowledge should be a parepeatic object.

Therefore we can introduce a framework of parepeatic data mining as described in Figure 2.1. The universe of discourse is parepeatic. Notice that there are elements in the universe that can not be given accurately as a result of the uncertainty associated with the universe. For each given element of the universe, we define several features or feature variables, which will lead to various observations. Based on the features and the corresponding observations, we need to extract intended knowledge for the parepeatic universe. So, in such a data-mining framework, two fundamental questions must be addressed. First, how should the intended knowledge be represented? Crisp, random and fuzzy models are not enough. We need a new form of knowledge representation. This is the so-called parepeatic model that will be defined in Section 3. Second, how can the intended knowledge be extracted from the given observations? An approach will be proposed for this in Section 3 and detailed in the Appendix.

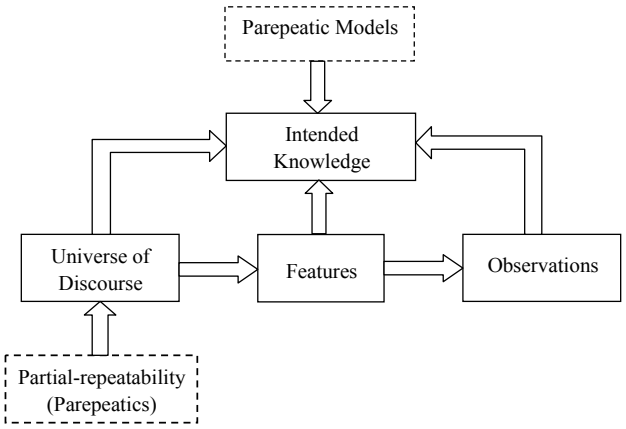


Figure 2.1 Parepeatic Data Mining

III. The Proposed Parepeatic Approach for Data Mining

A. Mathematical Definition of a Parepeatic Model

Suppose $U^{(p)}$ is the parepeatic universe of interest, and X the vector of feature variables of interest. For each element of $U^{(p)}$, X takes a vector value in $U^{(f)}$, which is referred to as the feature universe. Let f be a mapping from $2^{U^{(f)}}$ to V, where $2^{U^{(f)}}$ denotes the power set of $U^{(f)}$, and V is referred to as the knowledge universe of interest. f is the parepeatic object of interest. Suppose S is a subset of $U^{(f)}$ and defines the given set of values that X actually takes. Then we can write $f(S)$ as $f(X)$ if no confusion can otherwise arise. Note that $f(S)$ actually defines the central knowledge that we are going to extract from S for $U^{(p)}$ with the understanding that X is a simplifying representation of a generic element of $U^{(p)}$. Let $E(S)$ or $E(X)$ be a subset of V, or $E(X) \in 2^V$. We call the triplet $(f(X), E(X), c)$ a parepeatic model, where $f(X)$ is referred to as central knowledge[2], $E(X)$ the knowledge fluctuation zone, and c the corrector factor. There should hold $f(X) \in E(X) \subset V$ and $c \in [0,1]$.

In software engineering, we can use the parepeatic universe $U^{(p)}$ to represent the collection of all C language programs. Obviously, uncertainty is associated with the collection since it is nearly impossible to accurately define this collection. We have no idea how many C language programs there may be and whatever a C language program may be. This is particularly true if we consider the possibility that defects may be remaining in a C language program such that the synthetic and/or semantic requirements of C language are violated. Let $X = [x_1, x_2, \ldots, x_5]^\tau$ be the column vector of feature variables of interest, where τ denotes the transpose of a matrix and

x_1 : the number of lines of source code

x_2 : the number of distinct usages of operators

x_3 : the number of distinct usages of operands

x_4 : the number of total usages of operators

x_5 : the number of total usages of operands

X is a simplifying representation of a C language program, $X \in U^{(f)} = [0,\infty) \times [0,\infty) \times \cdots \times [0,\infty)$. Given a set of C language programs, suppose we are interested in clustering them. Then we can define $V = \{1, 2, 3, \ldots\}$. The central knowledge is $f(X)$ that defines the best number of clusters that the C language programs should be divided, the fluctuation zone $E(X)$ defines the set of all appropriate clustering, and c defines the degree of correctness of the clustering relation $f(X) \in E(X)$. Note that each cluster represents an equivalence class of C language programs in terms of X. Given a set of C language programs, there is a possibility that the given fluctuation zone $E(X)$ is inappropriate. For example, suppose the generated parepeatic model is $(3, \{2,3,5,7\}, 0.9)$. That is, the given set of C language programs suggests that all the C language programs should best be divided into 3 disjoint clusters, and it is also acceptable to divide all the C language programs into 2, 5, or 7 disjoint clusters. However these different clustering is still subject to uncertainty. One may argue that the best clustering that can be generated from the given set of C language programs should lead to $f(X) = 4$. Note $4 \notin \{2,3,5,7\}$. We only have $c = 0.9$ degree of confidence that $f(X) \in E(X)$.

B. Mining a Parepeatic Model from a Given Dataset

Suppose $\{X^{(1)}, X^{(2)}, \ldots, X^{(n)}\}$ is a set of observations for the vector of feature variables X. We want to extract intended knowledge from the given dataset. Let the intended knowledge be represented in terms of a parepeatic model $(f(X), E(X), c)$. In general, the parepeatic model can be determined as follows.

Step 1. Obtain the dataset $\{X^{(1)}, X^{(2)}, \ldots, X^{(n)}\}$ from n elements of the given parepeatic universe $U^{(p)}$.

Step 2. Pre-process the dataset $\{X^{(1)}, X^{(2)}, \ldots, X^{(n)}\}$ to remove the undersirable data or outliers. Consequently, a new

[2] In our previous work [7] we call $f(X)$ the central function. Obviously function can be a form of knowledge of interest. However knowledge of interest can be defined in other forms such as clustering, patterns, association rules, and so on.

dataset, $\{X^{(1')}, X^{(2')}, \ldots, X^{(n')}\}$ with $n' \le n$, is obtained. The new dataset can be further transformed into $\{Y^{(1')}, Y^{(2')}, \ldots, Y^{(n')}\}$ if necessary.

Step 3. Determine the knowledge fluctuation zone $E(X)$ from the dataset $\{Y^{(1')}, Y^{(2')}, \ldots, Y^{(n')}\}$ according to some evaluation criterion.

Step 4. Determine the central knowledge $f(X)$ as a representative of $E(X)$ according to some evaluation criterion.

Step 5. Determine the correctness factor c finally.

In Step 3 a sampling technique is applied to sample a number of subsets of the data from the given dataset for the purpose of determining the knowledge fluctuation zone. These sampled data can be treated as the training dataset of the fluctuation zone. A sampling technique is applied also in Step 5 for the purpose of determining the correctness factor. The resulting data can be treated as the validation dataset of the fluctuation zone. Note that the above 5 steps for parepeatic data mining is rather abstract. The Appendix details how a parepeatic model as a new form of knowledge representation is generated from a given dataset, where a new criterion is introduced to measure the quality or performance of conventional data clustering in Step (11) of the algorithm detailed in the Appendix. The new criterion takes account of not only the homogeneity within each cluster and separability between the distinct clusters, but also the number of distinct clusters.

C. Discussion

One may observe that there is some similarity between a parepeatic model presented in Section 3.1 and a confidence interval representation in conventional statistical setting. However we should note that there are several essential differences between them. First, in conventional statistical interval estimation the parameter under estimation is given a priori. In a parepeatic model the perepeatic object of interest or the central knowledge $f(X)$ must be generated from the given dataset. It is not given a priori. A given dataset may generate different central knowledge, depending on the used generation criterion. Second, $f(X)$ is widely interpreted. It can represent clustering, patterns, scenarios, association rules and so on, depending on application context of interest. $f(X)$ is not stuck to a particular parameter. Finally, as we will observe more clearly in the rest of this paper, no statistical assumptions are taken for determining a parepeatic model. However statistical assumptions are essential for conventional interval estimations.

Notice the parepeatic data mining algorithm presented in Section 3.1 and the Appendix is closely related to data clustering. This is because we assume the knowledge universe characterizes the number of disjoint clusters of the parepeatic universe. That is, the central knowledge represents the number of clusters. However the algorithm presented in this paper differs from existing clustering algorithms in data mining [8] at least in two dimensions. First, existing data clustering algorithms measure the quality of clustering in terms of the homogeneity within each cluster and separability between the distinct clusters, but the number of distinct clusters is not taken into account. This is not true for the algorithm presented in this paper. Second, existing clustering algorithms generate a partition of the universe of discourse or the central knowledge. They do not produce the knowledge fluctuation zone or correctness factor.

In general, the essential difference between existing approaches to data mining and the parepeatic data mining approach lies in the different philosophies they follow and the different knowledge representation models they adopt. The existing approaches assume that the universe of discourse is accurately given and no partial-repeatability is involved. The adopted knowledge representation model is actually given in terms of central knowledge which can be clusters, statistical models, fuzzy models, decision trees, neural networks, induction rules and so on. On the other hand, the parepeatic approach assumes that the universe of discourse involves uncertainty and can not be given accurately. The adopted knowledge representation model comprises not only central knowledge, but also a knowledge fluctuation zone and a correctness factor. A parepeatic model coincides with an existing model if the knowledge zone contains only the central knowledge and the correctness factor is equal to one. The existing knowledge representation models can be treated as a special class of parepeatic models.

Here we note that data mining is related to the so-called granular computing [9]. Data clustering results in a number of disjoint clusters, which can each be treated as an information granule. Following this philosophy, we can see that there exists potential for the parepeatic data mining approach to be interpreted from the granular computing perspective.

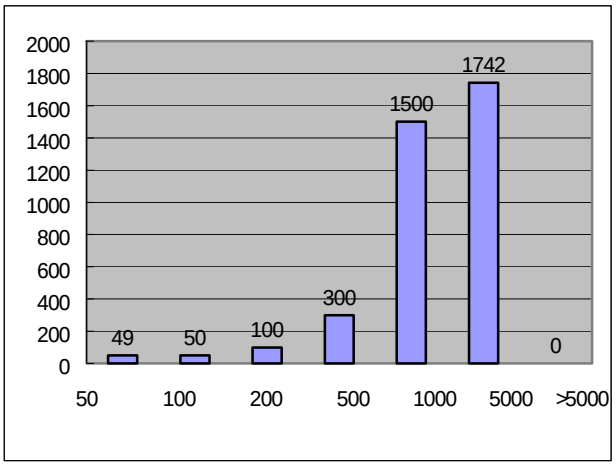
IV. Experimental Results in Software Data Mining

Software data mining, in some sense, can be traced back to Halstead's work on software science [10], although he neither adopted the terminology of data mining and nor recognized the importance of data mining. Halstead argued that there existed physics-like laws that obeyed each piece of software. He defined a number of software metrics such as those defined in Section 3 and proposed a set of the so-called software science formulae for these metrics. Empirical data were then collected from real software programs to validate the proposed formulae. We can treat these formulae as intended knowledge or intended laws and thus Halstead's work can be treated as a kind of software laws mining.

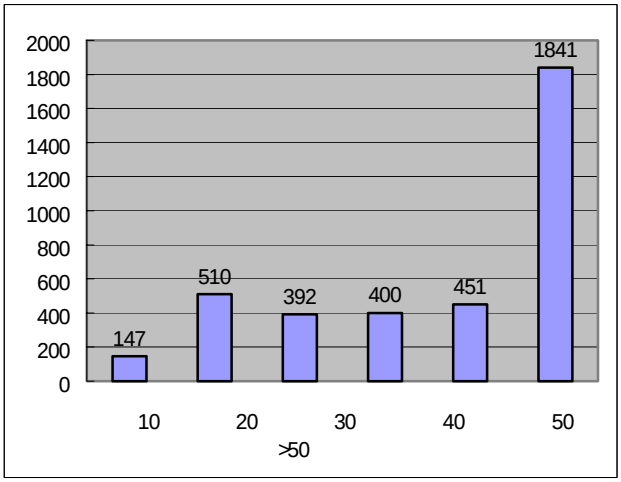
Unfortunately, as observed in our previous work [6, 7], Halstead's work still follows the philosophy of full repeatability and thus fails to stand. As argued in Section 2, the universe of software programs is parepeatic and thus software laws should be formulated in terms of parepeatic models. The

algorithm described in the Appendix can be employed to extract the required parepeatic models.

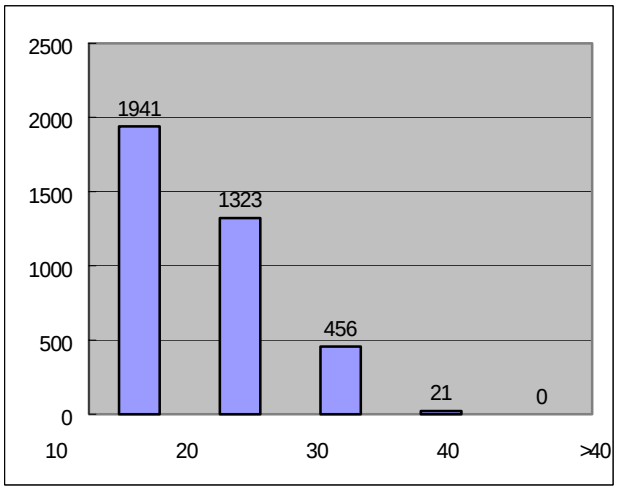
In order to test the algorithm described in the Appendix, we collected a set of 5437 C language programs. Some of the programs were downloaded from open source Web sites, and some of them came from undergraduate and graduate students' projects. For each program, we obtained a data point as that defined in Section 3. This has been performed in an automatic data collection tool. After the data pre-processing (i.e., Step (2) of the algorithm proposed in the Appendix), the dataset was reduced to a new dataset comprising 3471 data points. Figure 4.1 shows the histograms of these data points for x_1, x_2, x, x_4 and x_5. This new dataset ($\{T^{(1)}, T^{(2)}, \ldots, T^{(3471)}\}$ as specified in Step (2) in the algorithm described in Appendix) was then used to generate the required parepeatic model.



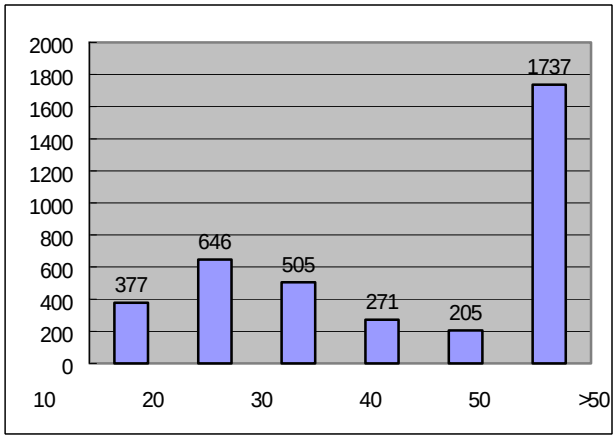
(a) Number of lines of source code



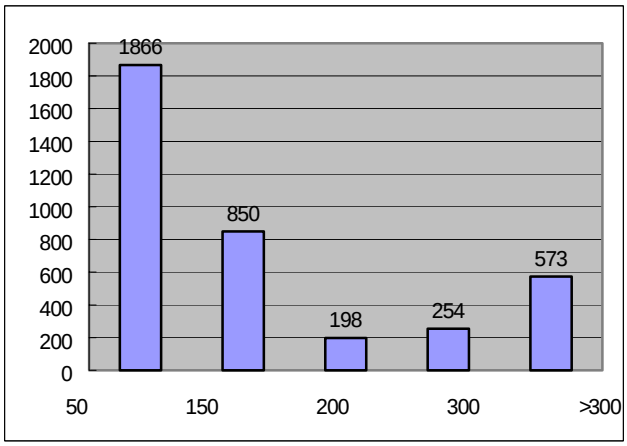
(b) Number of Distinct Usages of Operators



(c) Number of Distinct Usages of Operands



(d) Number of Total Usages of Operators



(e) Number of Total Usages of Operands

Figure 4.1 Histograms of Software Metrics for the Dataset

We carried out five experiments for the dataset $\{T^{(1)}, T^{(2)}, \ldots, T^{(3471)}\}$, that is, the algorithm described in the Appendix was applied for 5 times. In Steps (10) and (15) we applied a cut-set-based fuzzy clustering algorithm to do clustering, where the cut level was chosen as 0.975. Table 4.1 summarizes the experimental results. Different experiments

generated the training datasets of different sizes. However the validation datasets generated in these experiments had the same size ($\kappa_c \equiv 100$).

TABLE 4.1 EXPERIMENTAL RESULTS FOR PAREPEATIC SOFTWARE DATA MINING

($w_1 = 0.23, w_2 = 0.36, w_3 = 0.41, N = 25$)

Experiment No.	κ	Size of knowledge fluctuation zone	Central knowledge $f(X)$	κ_c	Correctness factor C	
					ε_0=0.001	ε_0=0.005
1	35	28	<11, 0.537088>	100	0.91	0.97
2	135	133	<11, 0.537075>	100	0.92	0.97
3	155	153	<10, 0.520748>	100	0.94	0.87
4	165	163	<10, 0.520748>	100	0.91	0.99
5	175	173	<10, 0.520748>	100	0.92	0.99

From the experimental results we can see that:

(1). As expected, greater value of ε_0 leads to greater value of C. This is understandable since Step (26) of the algorithm described in the Appendix implies that more data points are acceptable for the relation $f(X) \in E(X)$ if greater value of threshold is adopted.

(2). As the size of the training dataset grows, the central knowledge tends to be stable, although the size of the corresponding fluctuation zone tends to grow too. The corresponding correctness factor also behaves steadily. This implies that the generated parepeatic models are trustworthy and the algorithm described in the Appendix does work well.

V. CONCLUSION

Partial-repeatability or parepeatics emerges in complex phenomena if the complex phenomena demonstrate an invariant property that can neither be proved in mathematics nor validated to a high accuracy in physics, but still (partially) governs the behavior of the phenomena. This is particularly true in software engineering since software is developed by human. Software development processes and software system behavior are too complicated to be characterized accurately even in a statistical sense, but they still work and tend to serve human requirements, although they happen to fail to function. The notion of partial-repeatability was proposed in our previous work to contrast that of full repeatability that has been followed as the top criterion in traditional science and technology [6, 7]. By full repeatability it is meant that scientific arguments can either be proved repeatably in mathematics or be validated to a high accuracy repeatably (even in a statistical sense) in physics (experimentally).

In the preceding sections we have further clarified the notion of partial-repeatability as a new kind of uncertainty that is distinctly different from randomness and fuzziness. Suppose the universe of discourse is given and characterized accurately and an object is of interest. If the relation between the object and the universe is crisp, or each of the elements of the universe can be clearly identified to belong or not to belong to the object, then the relation is binary and the object is a crisp set. If the relation can be determined clearly, and each of the elements of the universe can belong to the object to some extent, and cannot belong to the object to another extent simultaneously, then the relation is fuzzy and the object is a fuzzy one. If each of the elements of the universe must either belong to or not belong to the object, but which elements belong to the object is not clearly determined, then the relation is random and the object is a random one. On the other hand, if the universe of discourse cannot be characterized accurately, then partial-repeatability emerges as a new kind of uncertainty. The corresponding object is a parepeatic one. An example universe is the one that comprises all possible C language programs.

Following the notion of partial-repeatability, we have proposed a new approach to data mining. In this approach the intended knowledge that is to be extracted from the given dataset is treated as a parepeatic object and represented in terms of parepeatic models. A parepeatic model consists of central knowledge, a knowledge fluctuation zone, and a correctness factor. Although a parepeatic model looks similar to conventional statistical confidence interval in some sense, there are essential differences between them. We have shown how to generate a parepeatic model (intended knowledge) from the given datset. The effectiveness of the proposed approach is justified by our experiments with software data mining.

The importance of the work presented in this paper is two-fold. First, partial-repeatability is clearly identified as a new kind of uncertainty that is distinctly different from randomness and fuzziness. This implies that we need to develop new mathematical framework to characterize this new kind of uncertainty and explore the underlying physical laws (or type III laws as mentioned in our previous work [7]). Second, a new framework of data mining, i.e., parepeatic data mining, is proposed. In this framework the intended knowledge is treated as a parepeatic object and parepeatic models are treated as a new form of knowledge representation. A lot of research work can be done by extending existing frameworks of data mining to the parepeatic counterpart. Pareaptic data mining reduces to conventional data mining if no uncertainty is associated with the underlying universe of discourse. This paper is only a small step towards to a new research direction and is speculative somewhat.

APPENDIX

Algorithm of Mining a Parepeatic Model:

In the context of software data mining, suppose the given parepeatic universe $U^{(p)}$ comprises all possible C language programs and the feature variables are $x_1, x_2, \ldots, x_5$ as given in Section 3.1. Given the observations

$\{X^{(1)}, X^{(2)}, \ldots, X^{(n)}\}$, with $X^{(j)} = [x_1^{(j)}, x_2^{(j)}, \ldots, x_5^{(j)}]^\tau$, where τ denotes the transpose of a matrix (vector) we want to cluster them into several classes. After data pre-processing, we have $\{Y^{(1')}, Y^{(2')}, \ldots, Y^{(n')}\}$ and want to divide these data into a number of disjoint classes, say, $\{D_1, D_2, \ldots, D_m\}$, where $D_i = \{Y^{(i_1)}, \ldots, Y^{(i_{n_i})}\}$, $D_i \cap D_j = \phi, (i \neq j)$, $n_1 + n_2 + \cdots + n_m = n'$.

In general, the amount of observations $\{X^{(1)}, X^{(2)}, \ldots, X^{(n)}\}$ is huge and these observations are not used in total to determine the knowledge fluctuation zone and the correctness factor. Rather, the sampling techniques may be applied. More specifically, we may follow the following the procedure to do parepeatic software data mining.

Step (1). Obtain the dataset $\{X^{(1)}, X^{(2)}, \ldots, X^{(n)}\}$ from n elements of the given parepeatic universe $U^{(p)}$, where $X^{(j)} = [x_1^{(j)}, x_2^{(j)}, \ldots, x_5^{(j)}]^\tau$

Step (2). Remove all the data points which satisfy the condition $x_1 x_2 \cdots x_5 = 0$, that is, the data points with no usages of any operators or operands are treated as outliers. The resulting dataset with outliers being removed is denoted as $\{T^{(1)}, T^{(2)}, \ldots, T^{(n)}\}$, with $T^{(j)} = [t_1^{(j)}, t_2^{(j)}, \ldots, t_5^{(j)}]^\tau$.

Step (3). Determine the maximal number of disjoint classes that the dataset is allowed to be divided. Denote this number as N.

Step (4). Determine the number of sampling sets which are to be obtained from $\{T^{(1)}, T^{(2)}, \ldots, T^{(n)}\}$; let the number be κ.

Step (5). Let $\alpha = 1$.

Step (6). Generate a random positive integer over the range $[s_1, s_2]$; this number is denoted as $s^{(\alpha)}$.

Step (7). Sample $s^{(\alpha)}$ data points one by one from the dataset $\{T^{(1)}, T^{(2)}, \ldots, T^{(n)}\}$ without replacement; that is, if $T^{(1)}$ is sampled, then it will not be returned back to the sampled dataset and thus the resulting $s^{(\alpha)}$ data points must be distinct; let the resulting $s^{(\alpha)}$ data points make up the dataset $\{\Gamma^{(1)}, \Gamma^{(2)}, \ldots, \Gamma^{(s^{(\alpha)})}\}$, with $T^{(j)} = [\gamma_1^{(j)}, \gamma_2^{(j)}, \ldots, \gamma_5^{(j)}]^\tau$.

Step (8). Transform the dataset $\{\Gamma^{(1)}, \Gamma^{(2)}, \ldots, \Gamma^{(s^{(\alpha)})}\}$ into $\{L^{(1)}, L^{(2)}, \ldots, L^{(s^{(\alpha)})}\}$ such that $L^{(i)} = \gamma_2^{(i)} \log_2 \gamma_2^{(i)} + \gamma_3^{(i)} \log_2 \gamma_3^{(i)}$, $i = 1, 2, \ldots, s^{(\alpha)}$; that is, $L^{(i)}$ is the Halstead length of a C language program.

Step (9). Transform the dataset $\{L^{(1)}, L^{(2)}, \ldots, L^{(s^{(\alpha)})}\}$ into $\{Y^{(1)}, Y^{(2)}, \ldots, Y^{(s^{(\alpha)})}\}$ such that $Y^{(i)} = \dfrac{L^{(i)} - \min\limits_{1 \le i \le \alpha} L^{(i)}}{\max\limits_{1 \le i \le \alpha} L^{(i)} - \min\limits_{1 \le i \le \alpha} L^{(i)}}$, $i = 1, 2, \ldots, s^{(\alpha)}$; that is, the dataset is normalized and $Y^{(i)} \in [0,1]$, $i = 1, 2, \ldots, s^{(\alpha)}$.

Step (10). Cluster the dataset $\{Y^{(1)}, Y^{(2)}, \ldots, Y^{(s^{(\alpha)})}\}$ into a number of disjoint classes, say, $(C_1^{(\alpha)}, C_2^{(\alpha)}, \ldots, C_{n_\alpha}^{(\alpha)})$ according to some clustering algorithm (e.g., fuzzy clustering algorithm); there hold $C_j^{(\alpha)} \cap C_k^{(\alpha)}$ for $j \neq k$, $C_j^{(\alpha)} = \{r_{j1}^{(\alpha)}, r_{j1}^{(\alpha)}, \ldots, r_{jd_j}^{(\alpha)}\}$, and $C_1^{(\alpha)} \cup C_2^{(\alpha)} \cup \cdots \cup C_{n_\alpha}^{(\alpha)} = \{Y^{(1)}, Y^{(2)}, \ldots, Y^{(s^{(\alpha)})}\}$.

Step (11). Evaluate the performance of the resulting clustering $(C_1^{(\alpha)}, C_2^{(\alpha)}, \ldots, C_{n_\alpha}^{(\alpha)})$ as

$$p_\alpha = \begin{cases} w_1 R^{(\alpha)} + \dfrac{w_2}{D^{(\alpha)}} + \dfrac{w_3}{N} & \text{if} \quad n_\alpha \le N \\ \infty & \text{otherwise} \end{cases}$$

where $w_1, w_2, w_3 \in [0,1]$ are weighting coefficients, with $w_1 + w_2 + w_3 = 1$, $R^{(\alpha)} = \max\limits_{1 \le j \le n_\alpha} \max\limits_{1 \le \beta, \nu \le d_j} \left| r_{j\beta}^{(\alpha)} - r_{j\nu}^{(\alpha)} \right|$, $D^{(\alpha)} = \min\limits_{1 \le i, j \le n_\alpha} \min\limits_{1 \le \beta \le d_i, 1 \le \nu \le d_j} \left| r_{i\beta}^{(\alpha)} - r_{j\nu}^{(\alpha)} \right|$; that is, $R^{(\alpha)}$ is the maximum of the diameter of a class, and $D^{(\alpha)}$ the minimum of the distances between two distinct classes. A good clustering should lead to small $R^{(\alpha)}$ and large $D^{(\alpha)}$. So, the smaller $p^{(\alpha)}$, the better the clustering.

Step (12). Obtain the result of the clustering as a data pair (n_α, p_α).

Step (13). Let $\alpha = \alpha + 1$; if $\alpha \le \kappa$, go to Step (6).

Step (14). Obtain the result of data sampling and processing as a new dataset $((n_1, p_1), (n_2, p_2), \ldots, (n_\kappa, p_\kappa))$.

Step (15). Obtain the knowledge fluctuation zone by removing the possible outliers from the dataset $((n_1, p_1), (n_2, p_2), \ldots, (n_\kappa, p_\kappa))$; $E(X) = ((n_{1'}, p_{1'}), (n_{2'}, p_{2'}), \ldots, (n_{\kappa'}, p_{\kappa'}))$, with $\kappa' \le \kappa$; that is, the dataset $((n_1, p_1), (n_2, p_2), \ldots, (n_\kappa, p_\kappa))$ is divided into two classes, the fluctuation zone and the outliers, by using some clustering algorithm (e.g., fuzzy clustering algorithm).

Step (16). Transform the dataset $((n_{1'}, p_{1'}), (n_{2'}, p_{2'}), \ldots, (n_{\kappa'}, p_{\kappa'}))$ into $((\eta_{1'}, p_{1'}), (\eta_{2'}, p_{2'}), \ldots, (\eta_{\kappa'}, p_{\kappa'}))$, with $\eta_\alpha = \dfrac{n_\alpha}{N}$.

Step (17). Let $\overline{\eta} = \frac{1}{\kappa'}\sum_{\alpha=1}^{\kappa'}\eta_\alpha$, $\overline{p} = \frac{1}{\kappa'}\sum_{\alpha=1}^{\kappa'} p_\alpha$

Step (18). Let $\varsigma = \arg\min_{1\le\alpha\le\kappa'}\sqrt{(\eta_\alpha-\overline{\eta})^2+(p_\alpha-\overline{p})^2}$; that is, $(\eta_\varsigma, p_\varsigma)$ is the one that is closest to $(\overline{\eta},\overline{p})$.

Step (19). Choose $(\eta_\varsigma, p_\varsigma)$ as the central knowledge, that is, $f(X)=(\eta_\varsigma, p_\varsigma)$.

Step (20). Determine the number of sampling sets which are to be obtained from $\{T^{(1)}, T^{(2)}, \ldots, T^{(n)}\}$; let the number be κ_c.

Step (21). Re-perform Steps (5) to (13) except that κ is replaced by κ_c.

Step (22). Obtain the output of Step (21) as a new dataset $((l_1,q_1),(l_2,q_2),\ldots,(l_{\kappa_c},q_{\kappa_c}))$ as that obtained in Step (14); this dataset is used to assess the knowledge fluctuation zone and obtain the corresponding correctness factor.

Step (23). Transform the dataset $((l_1,q_1),(l_2,q_2),\ldots,(l_{\kappa_c},q_{\kappa_c}))$ into $((\omega_1,q_1),(\omega_2,q_2),\ldots,(\omega_{\kappa_c},q_{\kappa_c}))$, with $\omega_\alpha = \frac{l_\alpha}{N}$.

Step (24). Let $\varepsilon_i = \min_{1\le j\le\kappa'}\sqrt{(\omega_i-\eta_j)^2+(q_i-p_j)^2}$, where (η_j, p_j) is specified in Step (16); ε_i measures the distance between the data point (l_i, q_i) and the knowledge fluctuation zone.

Step (25). Determine a distance threshold ε_0.

Step (26). Determine the correctness factor as $c = \frac{1}{\kappa_c}\sum_{i=1}^{\kappa_c}\delta_i$, where

$$\delta_i = \begin{cases} 1 & \text{if } \varepsilon_i \le \varepsilon_0 \\ 0 & \text{otherwise} \end{cases}$$

Step (27).

End.

Remarks

(1). The central knowledge we want to extract from the given dataset by using the above algorithm is an approximate clustering. Different intended central knowledge should lead to different data mining algorithm. In our previous work [7], the intended central knowledge is a functional relation among several feature variables.

(2). Data pre-processing takes place in Step (2). However it also takes place in Steps (7), (8), (9) and (21). That is, besides in the beginning of data mining, data pre-processing may take place throughout the rest process of data mining.

(3). Steps (3) to (15) are devoted to determining the fluctuation zone, Steps (16) to (19) to determining the central knowledge, and Steps (20) to (26) to determining the correctness factor.

(4). The sampling technique plays a major role in the parepeatic data mining algorithm given above. It is used both in determining the fluctuation zone and in determining the correctness factor. The dataset $((n_1,p_1),(n_2,p_2),\ldots,(n_\kappa,p_\kappa))$ obtained in Step (14) can be treated as the training dataset for the parepeatic model of the intended knowledge, whereas the dataset $((l_1,q_1),(l_2,q_2),\ldots,(l_{\kappa_c},q_{\kappa_c}))$ obtained in Step (22) can be treated as the validation dataset for the parepeatic model.

(5). The performance evaluation criterion p_α adopted in Step (11) is key part of the intended knowledge. It evaluates how good the clustering is. It makes trade-offs among the diameters of a class, the distances among distinct classes, and the number of distinct classes by using the weighting coefficients w_1, w_2 and w_3. The allowed maximal number of distinct classes, N , is also required for the performance evaluation.

(6). In determining the correctness factor c, the distance threshold ε_0 must be specified. Different values of ε_0 will lead to different values for c.

(7). Besides the parameters $w_1, w_2, w_3, N, \varepsilon_0$, the algorithm given above also requires the positive integer interval $[s_1, s_2]$ to be specified for the sampling processes of determining the fluctuation zone and the correctness factor.

REFERENCES

[1] U. Fayyad, P. Stolotz. "Data Mining and KDD: Promise and Challenges", *Future Generation Computer Systems*, Vol.13, 1997, pp99-115.

[2] J. Han, "Data Mining", in: J.Urban, P.Dasgupta (eds), *Encyclopedia of Distributed Computing*, Kluwer Academic Publishers, 1999.

[3] X. Wu, "Building Intelligent Learning Database Systems", *AI Magazine*, Vol.21, No.3, 2000, pp59-65.

[4] C. Zhang, S. Zhang, *Association Rule Mining: Models and Algorithms*, Springer, 2002.

[5] A. Jain, M. Murty, P. Flynn, Data Clustering: a Review", *ACM Computing Survey*, Vol.31, No.3, 1999, pp264-323.

[6] K. Y. Cai, J. H. Liao, "Software Pattern Laws and Partial Repeatability" in: G. Q. Chen, M. S. Ying, K. Y. Cai, (editors.) *Fuzzy Logic and Soft Computing*, Kluwer Academic Publishers, 1999, pp89-120.

[7] K. Y. Cai, L. Chen, "Analyzing Software Science Data with Partial Repeatability", *Journal of Systems and Software*, Vol.63, 2002, pp173-186.

[8] J. Granmeier, A. Rudolph, "Techniques of Cluster Algorithms in Data Mining", *Data Mining and Knowledge Discovery*, Vol.6, 2002, pp303-360.

[9] Y. Li, N. Zhong, "Interpretations of Association Rules by Granular Computing," *Proc. the Third IEEE International Conference on Data Mining*, 2003, pp.593-596.

[10] M. H. Halstead, *Elements of Software Science*, Elsevier, 1977.