Fundamenta Informaticae 71 (2006) 63–82
IOS Press

An Empirical Study of Using Rule Induction and Rough Sets to Software Cost Estimation[*]

Jerzy Stefanowski[†]
Institute of Computing Sciences
Poznań University of Technology
ul. Piotrowo 3a, 60-965 Poznań, Poland
Jerzy.Stefanowski@cs.put.poznan.pl

Abstract. This paper concerns problems of applying the approach based on rough sets and rule induction to a software engineering data analysis. More precisely, we focus our interest on a software cost estimation problem, which includes predicting the effort required to develop a software system basing on values of cost factors. The case study of analysing the COCOMO data set, containing descriptions of representative historical projects, allows us to discuss how this approach could be used to: identify the most discriminatory cost factors, extract meaningful rule representation of classification knowledge from data, construct accurate rule based classifiers.

Keywords: Software Engineering, Rough Sets, Machine Learning, Preference Modeling, Software Cost Prediction, COCOMO model.

1. Motivations

In software engineering various approaches have been proposed to improve software development and management of projects. Many of these approaches are rather qualitative ones, e.g. based on human expert knowledge, experience or developed practices. However, in particular for large scale projects, besides qualitative practices there has also been observed an increasing interest in investigating *quantitative approaches*. These approaches are based on the analysis of either historical data from similar projects or measurement data obtained from carefully designed experiments. Some researchers introduced the

[*]Research on this paper has been supported by KBN grant no. 3T11C 050 26.
[†]Address for correspondence: Wyższa Szkoła Bankowa w Poznaniu, al. Niepodległości 2, 61-874 Poznań, Poland

paradigm of *measurement based improvement-oriented software development*, which provides an experimental view of the software activities with a focus on learning and the quality improvement and implies the need for *software measurement* and quantitative approaches, see e.g. [1, 6]. The analysis of software measurement data can be mainly used for the following aims [6, 26]:

- recognizing and estimating the important factors (e.g., personnel capability, storage constraints) on various aspects of software quality or other issues of interest (e.g., productivity improvement, effort or cost estimation) for better understanding current practices, determining their limitations, controlling the software development process, etc.;
- constructing predictive models for the software process or products (e.g. effort, costs, schedule) based on their characteristics;
- evaluating products and processes by comparing them to the projects with similar characteristics and building recommendations for future projects.

In this paper we focus our interest on a *software cost estimation* problem, which includes predicting the effort required to develop a software system. It is one of the most important research problem in the software engineering data analysis [17] and it is also ”critical” for practical applications to both software companies and customers – accurate estimates should be used for preparing projects, contract negotiations, scheduling, monitoring, etc. Usually it involves the determination of one of the following estimates: *effort* (in person-months), project duration or money costs. An effort is the most important among them. Many estimation models have already been proposed over the last years, for reviews see, e.g., [17, 26]. Shortly speaking, some of them are called ”non-algorithmic” ones, e.g. based on expert judgment, analogy costing. Others, called *algorithmic methods*, provide a cost estimate as a mathematical function of variables, which are considered to be the *cost factors*. The COCOMO model, introduced by Boehm [4], is the most known and practically used approach among these methods. The classical, statistical techniques, e.g. based on stepwise regression, are also popular. However, several researchers indicate that the cost estimation still remains a complex research problem and algorithmic approaches are often unable to adequately model the complex relationships existing in many software development projects and their results are frequently unsatisfactory; For more extensive discussions the reader can consult, e.g. [17, 14]. Therefore, researchers have attempted different approaches. Recently, more advanced techniques coming from machine learning or artificial intelligence are often considered, e.g. decision or regression tress, neural networks, CBR, Bayesian Networks or mixed approached - for reviews c.f. [18].

In this context, we pose a question about the role of *rough sets theory* and related rule based approaches [21, 31]. In general, rough sets theory is claimed to be a well suited approach for handling vague and inconsistent information [16]. However, up to now there were not so many works on its use to software measurement data. The research on this topic was started by Gunter Ruhe. He has considered several software cost and software maintenance problems as a *supervised classification* task [26, 27] – the subject of his study was described by a set of *conditioned attributes* (explanatory or independent factors / predictors) and *decision attributes* (*discrete dependent* variables expressing the given classification of objects being considered software projects). According to him the rough sets could be useful for the following issues in the analysis of software measurement data:

- discovering the most important relationships between decision and condition attributes,

- representation of these relationships in a form of *if–then* rules,
- evaluation of attribute importance for objects classification and reduction of superfluous condition attributes,
- classification of new objects on the basis of rules.

In his papers Ruhe showed how to achieve these aims for predicting software costs based on cost factors coming from the COCOMO model and for maintaining software by using the ”standard” rough set approach (with pre-discretization techniques, rough approximations and reducts [26]) and rule induction algorithm LEM2 [11] available in RoughDAS software [28]. Other research were undertaken by Ramanna and Peters for predicting decisions concerning the number of changes that the given software program required [22, 25]. However, their research were more focused on studying other research problems than cost estimation or were more oriented to evaluating different approaches, as e.g. rough neural networks. To sum up, both research directions provided promising results, although the software applications seemed to be more difficult than other domains - we will come back to it in section 2.

This paper is focused on the methodological discussion and follows the direction of predicting software cost task, which is re-formulated as a supervised classification problem. Our aims are to discuss how rough sets and rule induction algorithms could be used to solve this task using the case study of analysing data set, which is coming from the original COCOMO database [4].

In comparison to the previous related research (which were based on rather standard and simpler rough sets approach [26]), here we want to verify the effectiveness of more advanced techniques for selecting the important attributes and data reduction. This will be combined with using the new rule induction algorithm, called MODLEM (developed by the author [30, 32]). One of the characteristic feature of this algorithm is that it does not require any preliminary discretization of numerical attributes performed before the induction phase and provides an efficient compact rule representation.

Furthermore, while analysing the characteristics of cost factors, we can say that their domains may be associated with additional *semantic information about preference orders* of the attribute values. Taking into account *preference-ordered data* requires the concept of the *semantic correlation* of criteria within the context of the *multiple criteria classification problems*. In these problems criteria are semantically correlated with preference–ordered decision classes [7]. Thus, analysing such preference orders in data requires new, other methods than the standard rough approach. In this paper we will use the extension of rough set theory based on the dominance relation, which is called *Dominance-based Rough Set Approach* - introduced by Greco, Matarazzo and Slowinski, see e.g. [7, 8]. We will also consider dominance-based decision rules having a special syntax, which will be induced by a specific algorithm, called DOMLEM [9]. According to our best knowledge it will be the first study of using the dominance based rough sets approach to software engineering problems.

The final aim of this paper is to perform a comparative study of using ”basic” rough sets and ”dominance based” rough set approaches against other typical machine learning algorithms and to evaluate which approach could provide the most accurate cost predictions.

The paper is organized as follows. In section 2 we shortly discuss peculiarities of software measurement data and difficulties in building accurate prediction models. It is followed by presenting main problems of cost estimation and characteristics of the data set. Then, in section 4 we briefly give main aspects of the used methodology. In section 5, experimental results are presented. After analyzing these results we conclude with final remarks.

2. Software Engineering Data Analysis

Before presenting the methodology and experimental results we think that it is worth to point out some critical aspects for performing the software engineering data analysis in an appropriate way. In general, the analysis of real data coming from historical projects or even carefully designed experiments seems to be quite difficult data analysis problem, even comparing to previous applications of rough sets, e.g., in medicine, financial data analysis or technical diagnostics (where the author of this paper has an experience of many years). Summarizing some literature discussions (see, e.g. [6, 26]) the following sources of difficulties should be taken into account:

- reliable data sets containing measurements of the software projects are quite difficult to obtain; Moreover, commercial companies do not provide access to their own data or collect them in an inappropriate way for further analysis,
- available data sets are often of small size; It mainly refers to a limited number of examples comparing to a number of attributes characterizing them,
- the data analyst should rather avoid making too many assumptions about the relationships between attributes or the probability density distributions; As it is discussed in [6], it is very difficult to make such valid assumptions - therefore the capabilities of classical statistical methods may be limited.
- data contain a large number of different influence factors, which are modeled by attributes of different types,
- quite often the attributes are dependent each other, so the considered analysis tools should take into account the issue of interdependencies among the attributes,
- the data analysis approach should be robust to outliers, incompleteness or uncertainty present in data.

Furthermore, one should remember that the software development is a human based technology, the process is very dynamic and the requirements of the final product may often change.

Taking into account the above points we should agree with opinions saying that ”considering the objective difficulties and the subjective problems occurring within an organization, the question is which kind of analysis is appropriate. (...) For an organization of low maturity in its development process qualitative analysis seems to be more adequate. On the other side, if you have comprehensive data sets which are based on stable environments within organization, you can ask more elaborative quantitative questions” [26]. Therefore, for cost estimation problem considered in this study we have decided to exploit the original COCOMO data base coming from previous Boehm research. Although it is quite an old historical data set and certainly having rather a limited size, it includes larger projects described by well established cost factors and it was used as a benchmark set for evaluating several different methods.

3. Software Cost Estimation and COCOMO Data Set

Most of cost estimation methods focus on the aspect of human efforts in software development and attempt to provide its estimate in terms of person-months. The literature review shows that many quantitative software estimation methods have been already proposed. For a comprehensive review the reader can look in [17]. In this study we focus our attention on, the so called, algorithmic methods. These are mathematical models that produce a cost estimate as a function of a number of variables (attributes) characterizing software project development, which are considered to be *cost factors*. The open research questions are mainly two following ones: which cost factors should be selected and what kind of a function form should chosen [17]. The size of the software, measured in thousands of delivered source instructions, is a quite important factor. However, the existing methods differ in selecting other factors. As to the software engineering literature the quite comprehensive set of cost factors has been introduced by Boehm and his collaborators in the COCOMO II model [5]. The COCOMO is an abbreviation from the name **Co**nstructive **C**ost **Mo**del. In fact, this is a family of various models which could be applied depending on the complexity of the software and available information. All of them use the power function having the general form:

$$\textit{Effort} = a \times S^b$$

where S is the code-size and a, b are usually simple functions of other cost factors. These models are very popular and widely used in practice. Examples of other analytical models are Putnam's SLIM, Price-S. However, as it is discussed in the literature, see e.g. [17, 26], in practice it may be difficult to calibrate / adjust parameters of these models, their results may be unsatisfactory for some practical problems. Moreover, they do not provide a *direct explanation* which factors mainly influence the efforts of the considered project. Therefore, other methods are still developed. This is also a motivation for us to discuss the use of rough sets and rule induction approaches, where we stress aspects of identifying the most promising factors (attributes) and extracting meaningful patterns from data.

The data set for this experimental study is the original COCOMO data base as provided by Boehm. It contains 63 completed projects described by 22 condition and two decision attributes. The decision attributes are: *TKDSI* - total delivered source instructions (i.e. size of the project measured in thousands of lines), *Effort* - (actual man-month required for the project). Both are continuous attributes defined on real valued scales.

The condition attributes are cost factors coming from the COCOMO models specification. They are divided into four types:

Product factors, e.g., required software reliability, size of used data base; product complexity; language level;

Computer factors, e.g., execution time constraint, main storage constraint, virtual machine volatility (platform under which works analysed system), computer turnaround time constraint, type of computer;

Personnel factors, e.g., analyst capability, application experience, programming capability, platform experience, programing language and tool experience, personnel continuity;

Project factors, e.g., use of modern programming practices, use of software tools, required development schedule, requirements volatility.

Moreover, there are other factors not included into the original COCOMO specification, e.g.: software development mode, type of application and adaptation adjustment factor (it is a special adjustment factor for a size of instructions that were adopted from the previous software into a newly developed one).

In the available data base many of these attributes are defined on numerical real valued scales while a few (e.g., mode, type of application, computer) are described by nominal or ordinal values. Although at the first glance the data base seems to contain real numbers, we should be cautious. The investigation of the semantics behind the majority of cost factors and finding their precise documentation showed that many of them are originally defined basing on qualitative judgments, which have been transformed into some scores / weights; consult e.g. [5, 17]. For instance, the cost factor *RELY - required software reliability* - is rated by 5 ordinal terms: very low, low, nominal, high, very high, which are valued by the corresponding numbers 0.75; 0.88; 1.0; 1.15 and 1.4, respectively. In particular, personnel factors as analyst, application or programmer capability are also originally rated as very low, low, normal, high and very high. A similar interpretation exits for some other cost factors. To sum up, many of condition attributes contain only few different real values. It, somehow, supports our point of view that we can analyse this data base by using approaches, which employ a kind of data quantization and provide a qualitative generalization inside the induced knowledge representation.

4. Methodology

In this section, we shortly discuss which basic elements of rough sets, its dominance based generalization and rule induction approaches are used in this study. A reader can find a more exhaustive presentation of rough sets in such positions as [16, 21, 23], its generalizations in [7, 8] and rule induction in [11, 31].

4.1. Basic Rough Approximations of the Object Classification

Rough sets theory deals with uncertainty and vagueness in available information [21]. A *data table* (also called an information system) is a formal representation of the analysed data set and is defined as a pair $S = (U, A)$, where U is a finite set of *objects* and A is a finite set of *attributes*, $a_i(x)$ is a value of an attribute $a_i \in A$ for an object $x \in U$. In this study we use a *decision table*. It is a pair $(U, C \cup \{d\})$, where $d \notin C$ is a distinguished *decision attribute* and C is a subset of *condition attributes*. The decision attribute d determines the partition of U into k disjoint *decision* classes $X_1, X_2, \ldots, X_k$, i.e. classification $\mathcal{X}$, where k is a number of different values of attribute d.

The rough sets theory is founded on the assumption that having some information about considered objects one can establish *relations* between these objects. The basic relation is an *indiscernibility relation*. Given a subset of attributes $B \subseteq C$, an indiscernibility relation $IND(B)$ is defined as:

$$IND(B) = \{(x, y) \in U \times U : \ \forall c_i \in B \ \ c_i(x) = c_i(y)\}$$

Other generalized relations, e.g., tolerance or similarity, are discussed, e.g., in [16, 8, 32]. As some *elementary classes* of the indiscernibility relation may be *inconsistent* (i.e., may contain objects described by the same values of condition attributes and assigned to different decision classes), the precise definition of a subset of objects $X \subset U$ in terms of these elementary sets is not always possible. To handle it the concept of a *rough set* has been introduced, i.e. each imprecise set X is characterized by its *lower and upper approximation*, denoted $\underline{B}X$ and $\overline{B}X$ respectively. The lower approximation contains objects of U, which are certainly assigned to X, while the upper approximation contains these objects which possibly belong to it, following their descriptions by a set of attributes B. Rough approximations of a set are described by their cardinalities and chracterized by a coefficient of the *approximation accuracy*. The above concepts are generalized into an approximation of objects classification $\mathcal{X}$ by the set of attributes $B \subseteq C$, which is numerically characterized by the coefficient called a *quality of the approximation of classification* $\mathcal{X}$ - defined as:

$$\gamma_B(\mathcal{X}) = \sum_{i=1}^{k} card(\underline{B}X_i)/card(U)$$

Using these concepts one can perform the other rough set operations [16]. In this study we focus on determining importance of attributes for object classification and reducing data by means of reducts.

An importance of attributes is evaluated by their influence on the quality of the classification approximation – measured during stepwise adding or removing the single attribute to the current attribute subset, see e.g. [28]. Yet another option to select attributes is using reducts or a core of attributes. A *reduct* is a minimal subset of attributes ensuring the same quality of classification as the entire set of attributes. The information system may contain more reducts than one. A *core* is an intersection of all reducts in the information system. In general, a concept of the reduct seems to be attractive as it may allow to reduce the data size. In other words, we can keep only those attributes that preserve approximation of classification and reject such attributes whose removal cannot worsen the classification. However, this kind of reduction refers to a situation when the number of reducts is not too high; In other cases we need more sophisticated techniques, see e.g. [2, 16].

4.2. Rule Induction

Representing relationships between values of condition attributes C and a decision class can be expressed in a form of *decision rules*. They are logical expressions of the following form:

IF (*conditions*) THEN (*decision class*),

where conditions are formed as a conjunction of elementary tests on values of condition attributes.

A number of various algorithms have been developed to induce such rules, mainly in machine learning or knowledge discovery literature. For a review see, e.g., [11, 19, 32, 15]. Let us shortly comment the peculiarity of the rough sets context. A decision table can be seen as a set of *learning examples* which enable induction of decision rules. If the decision table is consistent, rules are induced from decision classes. Otherwise decision rules could be generated from examples belonging to rough approximations of decision classes. This special way of treating inconsistencies in the input data is the main point where the concept of the rough sets theory is used in the rule induction phase. The step of induction itself may follow the inductive principle which is a common aspect with many machine learning algorithms, see e.g. minimal cover algorithms [11, 31], although there exist specific methods developed in the rough set community - mainly *Boolean reasoning* [29]. As a consequence of using the approximations of decision classes, induced decision rules are categorized into *certain (exact)* and *approximate* ones, depending on the used lower and upper approximations (or boundaries), respectively.

Rules can be evaluated by several measures, see, e.g., reviews [32, 33]. In this study we will use: *strength* of a rule - a (relative) number of positive examples covered by the rule; *consistency* of a rule represented by its *discrimination level* (also called a rule confidence) - a number of positive examples covered by the rule divided by a number of all examples covered by the rule. Furthermore, one can prefer the more compact rule representation, i.e. a smaller number of more general rules.

4.3. MODLEM Rule Induction Algorithm

In this study we have chosen an algorithm called MODLEM, introduced by Stefanowski in [30]; see also its more precise description in [32] or [13]. This choice has resulted from its ability to work with rough approximations and the fact that we can analyse numerical attributes without their preliminary discretization. Furthermore, the previous experience with using this algorithm indicated that it can produce quite efficient rule based classifiers [13, 32].

Here, we skip the formal presentation of this algorithm and we only discuss its main idea. It is based on the scheme of a *sequential covering* and it heuristically generates a *minimal set* of decision rules for every decision concept (decision class or its rough approximation in a case of inconsistent examples). Such a set of rules attempts to cover all (or the most significant) *positive examples* of the given concept and not to cover any *negative examples* (or as little as possible of them). The main procedure for rule induction starts from creating a first rule by choosing sequentially the 'best' elementary conditions according to chosen criteria (i.e., the first candidate for the condition part is one elementary condition; If it does not fulfill the requirement to be accepted as a rule, then the next - currently best evaluated - elementary condition is added to the candidate for a condition part, etc.; This specialization is performed until the rule could be accepted). When the rule is stored, all learning positive examples that match this rule are removed from consideration. The process is iteratively repeated while some significant positive examples of the decision concept remain still uncovered. Then, the procedure is sequentially repeated for each set of examples from a succeeding decision concept. In the basic version of the MODLEM algorithm elementary conditions are evaluated by using one of two measures either *class entropy* or *Laplace accuracy* [30, 32]. It is also possible to consider a lexicographic order of two criteria measuring the rule positive cover and then its conditional probability (originally considered by Grzymala in his LEM2 algorithm or its last, quite interesting modification called MLEM2).

The extra property of the MODLEM algorithm is handling directly numerical attributes during rule induction while elementary conditions of rules are created, without any preliminary discretization phase. In MODLEM, elementary conditions are represented as either $(a < v_a)$ or $(a \geq v_a)$, where a denotes an attribute and v_a is its value. If the same attribute is chosen twice while building a single rule, one may also obtain the condition $(a = [v_1, v_2))$ that results from an intersection of two conditions $(a < v_2)$ and $(a \geq v_1)$ such that $v_1 < v_2$. For nominal attributes, conditions are represented as $(a = v_a)$.

Finally, the unordered set of induced rules is applied to classify examples using the classification strategy introduced by Grzymala in LERS system [12], which takes into account strength of all rules completely matched and also allows partially matches if no rule fits the description of the tested example.

4.4. Dominance based Rough Set Approach

The motivations for this approach comes from the domain of Multiple Criteria Decision Aid (MCDA), where objects (decision variants) are described by *multiple criteria* representing conflicting points of view. Criteria are attributes with *preference-ordered domains*. For example, if decisions about some projects are based on such characteristics as programmer or analyst capability, etc., they could be treated as criteria because a decision maker usually considers higher experience as better than lower. Regular attributes, such as e.g. programming language and computer type, are different from criteria because their domains are not preference-ordered. In this interpretation decision classes are also preference ordered, e.g. taking into account customer point of view the class of less expensive projects is preferred to more expensive ones. In this paper we formulate our problem as a *multiple-criteria sorting* problem - it concerns an assignment of objects evaluated by a set of criteria into pre-defined preference-ordered decision classes. Any reasonable regularities to be discovered have to take into account a *semantic correlation* between criteria and decision classes [7, 8]. This is expressed by the *dominance principle*, which requires that objects having better evaluations (or at least the same evaluations) cannot be assigned to a worse class. This kind of semantic information is often present in data related to cost or quality; however, it is neglected by standard methods. For this reason, Greco, Matarazzo and Slowinski have proposed an extension of the rough sets theory, called Dominance-based Rough Set Approach (DRSA), that is able to deal with this inconsistency typical to MCDA problems [7, 8].

This innovation is mainly based on substitution of the indiscernibility relation by a dominance relation. A preference scale of each criterion a induces a *complete preorder* (i.e. a strongly complete and transitive binary relation) $\succeq$ in the domain of criterion $a \in A$ - for objects $x, y \in U$, $a(x) \succeq a(y)$ means that object x is "*at least as good*" as y with respect to criterion a. Let us assume that object can be described both by a subset of criteria and a subset of regular attributes (the classic indiscernibility relation still holds for them). The binary dominance relation R_P is defined as follows: for each $x, y \in U$ xR_Py if $a(x) \succeq a(y)$ for each criterion $a \in P$ and $a(x) = a(y)$ for each attribute $a \in P$.

The decision classes are preference ordered, i.e. for all r, s such that $s > r$, objects from X_s are preferred to objects from X_r. In multiple criteria classification (sorting) problems it is typical to consider *upward union* and *downward union* of classes instead of single decision classes. The upward and downward unions are defined, respectively, as:

$$X_g^{\geq} = \bigcup_{h \geq g} X_h, \quad X_g^{\leq} = \bigcup_{f \leq g} X_f, \; g = 1, \ldots k$$

The statement $x \in X_g^{\geq}$ means "x belongs to at least class X_g", while $x \in X_g^{\leq}$ means "x belongs to at most class X_g. This requires a change of approximating items from elementary sets to dominating and dominated sets. Given object x, a *dominating set* is composed of all objects evaluated not worse than x on all considered criteria, while *a dominated set* is composed of all objects evaluated not better than x on all considered criteria. Then decision classes, the sets to be approximated are, *upward* and *downward unions* of decision classes. For instance the, lower approximation of union $X_g^{\geq}$ by the set of P contains all P dominating sets included in this union.

The syntax of dominance based rules is different from "classical" ones. In general, two kinds of these rules are distinguished: $D_{\succeq}$–decision rules with the following syntax: *if* $(a_1(x) \succeq r_{a_1}) \wedge \ldots \wedge (a_m(x) \succeq r_{a_m}) \wedge (a_{m+1}(x) = r_{a_{m+1}}) \wedge \ldots \wedge (a_p(x) = r_{a_p})$ *then* $x \in X_g^{\geq}$, $D_{\preceq}$–decision rules with the following syntax: *if* $(a_1(x) \preceq r_{a_1}) \wedge \ldots \wedge (a_m(x) \preceq r_{a_m}) \wedge (a_{m+1}(x) = r_{a_{m+1}}) \wedge \ldots \wedge (a_p(x) = r_{a_p})$ *then* $x \in X_g^{\leq}$, where $P^{\geq} = \{a_1, \ldots, a_m\} \subseteq A$ is a subset of criteria and $P^{=} = \{a_{m+1}, \ldots, a_p\}$ is a subset of regular attributes, $(r_1, \ldots, r_p) \in V_{a1} \times V_{a2} \times V_{ap}$ and $g = 1, \ldots k$.

The $D_{\succeq}$–rule says: "if an evaluation of object x on criterion a_i is at least as good as a threshold value r_i $(i = 1, \ldots, m)$ and on attribute a_j object x is indiscernible with value r_j $(i = m + 1, \ldots, p)$ then object x belongs to a least class Cl_g". Similarly the $D_{\preceq}$–decision rule means that an object is evaluated as "at most good as a value" and belongs to at most a given class.

Several rule induction algorithms specific for this DRSA context have been proposed - for review see [32]. In this paper we will use an algorithm, called DOMLEM introduced in [9]. It is focused on inducing a minimal set of rules that cover all examples in the input data and can be seen as a multiple-criteria counterpart of the MODLEM algorithm.

5. Experimental Results

5.1. Problem Statement

In this study the prediction of software cost is re-formulated as a supervised classification problem. Our aims are to discuss how rough sets and rule induction algorithms could be used to:

- identify the most discriminatory cost factors (condition attributes),
- extract meaningful representation of classification knowledge from data in a form of rules,
- construct accurate rule based classifiers.

Moreover, we want to take into account an extra interpretation of preference ordered domains of attribute, which leads us to applying the dominance based rough set approach to this problem. Discussing differences between ”classical” and ”dominance” based approaches is an additional issue in this study. Finally, we want to perform a comparative study of classification performance of our approach against other methods of learning classifiers.

The analysed data set is coming from the original COCOMO database of 63 projects described by 22 attributes [4]. For the aims of the current analysis we decided to skip three attributes: *software development mode*, *type of application* and *product language*. All of them were defined on nominal scales, while the rest were defined on numerical or ordinal ones. This is the main motivation to skip them as we additionally want to analyse the problem of software cost estimation within the multiple-criteria context by using the dominance based rough set approach. Although regular attributes could be also handled, we decided to stay with the original DRSA concept focusing on the preference ordered criteria only. Furthermore, looking into the previous or related research with this data set we noticed that some of these attributes were quite often removed from analysis, c.f. [1, 14, 26]. The list of finally chosen attributes with their codes is given in the appendix. In the further analysis the condition attributes were not pre-discretized. It is not an obstacle for our approaches - see remarks given in section 3.

An important question is how to choose the dependent / output attribute. In the data set specification we have two proposals of decision attributes, i.e.: *TKDSI* - total delivered source instructions (i.e. size of the project measured in thousands of lines) and *Effort* - (actual man-month required for the project). Originally, they are continuous attributes defined on real valued scales. In some cost estimation models *TKDSI* is used as the main condition attribute. Here, we checked its relationship with *Effort* and discovered that they are both highly correlated (the Pearson correlation coefficient was greater than 0.9). So, putting it inside the set condition attributes could ”dominate” the searched relationships with other cost factors. In previous research some authors attempted to aggregate their values. For instance, in [1] the new decision attribute was defined as a ratio *TKDSI/Effort* and discretized into five sub-intervals by an equal frequency method. We have studied this proposal for our data set but it did not provide better classification results than the decision attribute finally chosen by us - so we skip it. Another proposal was considered in the previous rough sets study by Ruhe [26]. He discretized both attributes into two intervals corresponding to expert's evaluation $\{$*low,high*$\}$ and, then, constructed their logical combination. Unfortunately, the details behind this operation were not described, so we could not compare our results to this proposal. Yet another option is to consider attributes independently and to look for some functional transformations, e.g. *log(Effort)*, however this refers to a continuous version of this task in a form of regression.

In our study, we decided to consider both attributes *Effort, TKDSI* independently. We put more attention to the attribute *Effort* because of the reasons discussed in section 3. For the supervised classification task we needed to discretize these attributes. After few attempts with methods of equal sub-intervals, equal frequency sub-intervals and k-means cluster analysis we decided to employ the second method (with the small change of two objects between 1 and 2 interval). The domains of both attributes were dived into three sub-intervals, assuming the following interpretation of decision classes corresponding to them, i.e. *small, medium, high* - so we had an ordered list of classes. These discretizations were as follows:

- *Effort* - $(0, 10); [10, 37.5); [> 37.5);$
- *TKDSI* - $(0, 50); [50, 170); [> 170).$

The obtained decision classes were approximately balanced considering their cardinalities - see e.g. table RST. Let us shortly comment that we also tried 5 sub-intervals version. However, the available data set contained a limited number of projects. So, the cardinalities of decision classes were too small and further classification results were unsatisfactory. This is why we further present the three class version of discretized decision attributes.

5.2. The Basic Rough Sets and Rule Induction Approach

First we used the basic (equivalence classes) rough sets based approach. Calculating approximations of the classification according to the *Effort* attribute showed that all classes were precisely defined. The quality of approximation was equal to 1.0. The analysed data set was consistent and selected cost factors completely discriminated projects' descriptions from different classes. On the other hand, the number of equivalence classes was equal to 62, which could be interepreted that these project descriptions were rather unique for numeric data without any discretization. Details of approximations are given in table 1.

Table 1. Rough approximations of the project classification - *Effort*

Class code	Class name	Cardinality of lower approx.	Cardinality of upper approx.	Accuracy of approximation
1	low	18	18	1.0
2	medium	24	24	1.0
3	high	21	21	1.0

Further on, we used the algorithm MODLEM2 to induce decision rules. The entropy measure was chosen to evaluate elementary conditions while adding them to condition parts of rules. We obtained 16 rules having the characteristics presented in table 2. One can notice that we obtained quite a balanced number of rules for each decision class, which used 3.5 elementary conditions on an average and were supported by usually 4 learning examples. Exemplary rules are presented below (numbers in brackets refer to projects covered by a rule, i.e., the first number to the absolute strength of the rule calculated as a number of examples, and the second number to the relative strength in the decision class):
rule 1. *if* (TYPE $\geq$ 4) *then* Class 1 [4; 22.2%]
rule 2. *if* (CONT $\geq$ 2)) $\wedge$ (DATA $<$ 1.02) $\wedge$ (ACAP $\geq$ 0.93) *then* Class 1 [7; 38.9%]

Table 2. Rule evaluation measures of the rule sets induced by MODLEM for decision attribute *Effort*

Total no. of rules	No. of rules in classes	Average length	Average strength
16	6 / 5 / 5	3.42	4.13

rule 3. *if* (DATA < 1.02) $\wedge$ (MODP ≥ 1.05)$\wedge$ (CPLX < 0.93) $\wedge$ (AAF ≥ 0.72) *then* Class 1 [3; 16.7%]
...
rule 7. *if* (CPLX ≥ 0.93) $\wedge$ (AAF ≥ 0.86) $\wedge$ (SCED < 1.06) $\wedge$ (TOOL $\in [0.93,1.05]$) *then* Class 2 [12; 50%]
rule 8. *if* (RVOL < 1.29) $\wedge$ (CONT $\in [1,3]$) $\wedge$ (TYPE $\in [2,3]$)) $\wedge$ (TOOL ≥ 1.05) *then* Class 2 [4; 16.7%]
...
rule 12. *if* (DATA ≥ 1.07)) $\wedge$ (RELAY ≥ 1.27) *then* Class 3 [3; 14.3%]
rule 15. *if* (STOR < 1.1)) $\wedge$ (AAF < 0.8) *then* Class 3 [6; 28.6%]

As one can notice some of these rules generalize descriptions of many examples inside decision classes, e.g. rule 2 is supported by 39% while rule 7 even by 50% examples from the appropriate classes. Furthermore, the syntax of these rules is quite easy for human inspection. For instance, rule 2 could be interpreted as *if personnel continuity is not worse than normal and size of the data base is below 1.02 units and analyst capability is not smaller than 0.93 units, then Project effort is low*. We also checked the possibility of using Laplace accuracy as an evaluation measure inside MODLEM - it gave slightly more rules having a similar classification accuracy.

In the next step we evaluated the classification accuracy of the MODLEM rules (entropy measure + classification strategy described in [12]) by performing the ”leaving-one-out” verification. Results are presented in table 3.

Table 3. Classification accuracy [in %] for the rule sets induced by MODLEM

Decision attribute	Total classification accuracy	Accuracy in classes		
		1	2	3
Effort	63.49	50	62.5	76.2
TKDSI	61.08	55.6	51.6	78.6

Finally, we looked for the most important cost factors for the classification of objects according to the attribute *Effort*. Using *Rose* software[1] we discovered that the core of attributes was empty. Number of all reducts was very high (the process was stopped after obtained few hundreds ones). However, it was possible to apply the heuristic procedure of looking for the shortest reduct by adding attributes having the highest influence on the quality of object classification. In this way we obtained 19 smallest reducts having 5 elements (alternative ways were possible as at the first step several attributes could be chosen with the same increase of the quality). We decided to use a simple heuristic manual inspection of this

[1]Rose is an implementation of rough sets approach and several rule induction algorithms.

process results by counting how many times the given attribute was chosen during the three first adding to the attribute subset. The following attributes were identified (the list is ordered starting from the most frequent ones): AAF, AEXP, TIME, RELY, CPLX, TYPE, ACAP, TOOL, PCAP, MODP.

As the number of reducts was very high and the core was empty we also looked for an alternative approach to identify the most discriminatory attributes. The induced rule set had an acceptable classification performance (detailed analysis is performed in section 5.4). Thus, we decided to use another heuristic, which was a manual inspection of the syntax of the strongest rules (i.e. supported by a number of examples not smaller than average strength in the given class) and counting how many times the given attribute was used in their elementary conditions. Proceeding in this way we identified the following attributes: DATA, CPLX, AAF, STOR, TYPE, ACAP, CONT, MODP, SCEP.

The similar analysis was performed for the second decision attribute *TKDSI*. The data set was again consistent. The core of attributes was empty and the number of reducts very high. Applying above heuristic procedure of inspecting the adding attributes we could identify the following attributes: DATA, AAF, RVOL, TYPE, CONT. The use of the MODLEM algorithm led to the set of 18 decision rules. Some examples of these rules are given below. Their classification performance is given in table 3.

rule 1. *if* (TIME < 1.03) $\wedge$ (DATA < 1.02) $\wedge$ (AEXP < 0.87) *then* Class 1 [8; 40%]
rule 2. *if* (RELAY < 0.91)) $\wedge$ (DATA < 1.02) $\wedge$ (PCAP < 0.9) *then* Class 1 [8; 40%]
rule 3. *if* (TIME < 1.03) $\wedge$ (DATA < 1.02)$\wedge$ (RELY < 0.82) *then* Class 1 [3; 15%]
rule 4. *if* (AAF < 0.62) *then* Class 1 [4; 20%]
. . .
rule 7. *if* (AEXP ≥ 0.96) $\wedge$ (CPLX ≥ 1.04) $\wedge$ (AAF ≥ 0.77) $\wedge$ (TYPE ≥ 3) $\wedge$ (TURN < 0.91) *then* Class 2 [2; 11.8%]
rule 9. *if* (AEXP ≥ 0.96) $\wedge$ (TYPE ≤ 2) $\wedge$ (AAF $\in$ [0.48,0.99)) *then* Class 2 [7; 41.2%]
. . .
rule 15. *if* (DATA ≥ 1.07)) $\wedge$ (CPLX < 1.23) *then* Class 3 [10; 38%]
rule 16. *if* (TIME ≥ 1.03)) $\wedge$ (CPLX < 1.04) *then* Class 3 [8; 30.8%]

5.3. The Dominance based Rough Sets

In this step of the analysis we took into account the semantic interpretation of the preference order of attribute domains and applied the dominance based rough set approach implemented in the *Foremka* software[2]. This approach required determining *monotonic preference orders* for each criteria: either *decreasing* or *increasing* one. In other terms, values of the given criterion were interpreted either as cost (the smaller, the better) or gain (the higher, the better) type of expert's preference. Furthermore, decision classes were considered as preference ordered - smaller project Effort was more preferred than higher one! It led us to analyse unions of decision classes instead of single unordered classes.

Starting from the *Effort* decision attribute we discovered the *inconsistency* of project descriptions with the dominance principle. More precisely it referred to 4 objects. Projects numbered 12 and 60 were inconsistent considering decision unions *most class 1* and *at least class 2*. Then, projects numbered 42 and 45 were inconsistent due to decision unions *most class 2* and *at least class 3*. As the result of theses inconsistencies the quality of classification was equal to 0.94. The other characteristics of union approximations are given in table 4.

[2]4mka - Foremka - is an implementation of the basic version of DRSA done during the student project in laboratory of Intelligent Decision Support Systems, ICS, Poznań University of Technology. It is available from http://www-idss.cs.put.poznan.pl

Table 4. Dominance based rough approximations of unions - *Effort*

Union class code	Union class name	Cardinality of lower approx.	Cardinality of upper approx.	Accuracy of approximation
$X_1^{\leq}$	at most low	17	19	0.89
$X_2^{\leq}$	at most medium	41	43	0.95
$X_2^{\geq}$	at least medium	44	46	0.96
$X_3^{\geq}$	at least high	20	22	0.91

Then, we identified the most important cost factors for the classification of objects according to the decision attribute *Effort*. The core of attribute was non-empty and contained the following attributes: DATA, STOR, PCAP, TOOL, SCED and RELY. The number of all reducts was equal to 32.

In the next step of the analysis, we used DOMLEM algorithm and induced the set 35 certain dominance based decision rules. Their characteristics is given in table 5. The examples of these rule are given below (the syntax is consistent with this one presented in section 4.4).

Table 5. Rule evaluation measures of the rule sets induced by DOMLEM in DRSA

Decision attribute	Total no. of rules	No. of rules in unions	Average length	Average strength
Effort	35	10 / 7 / 8 /10	2.32	5.11
TKDSI	31	7 / 9 / 5 /10	2.39	4.94

rule 1. *if* (TYPE $\succeq$ 4) *then* Union $X_1^{\leq}$ [4; 22.2%]
rule 2. *if* (AAF $\preceq$ 0.6) $\wedge$ (CPLX $\succeq$ 1.15) *then* Union $X_1^{\leq}$ [2; 11.7%]
rule 3. *if* (AEXP $\preceq$ 0.82) $\wedge$ (RELAY $\preceq$ 1.02) $\wedge$ (LEXP $\preceq$ 0.95) *then* Union $X_1^{\leq}$ [3; 17.7%]
. . .
rule 12. *if* (DATA $\preceq$ 0.94) $\wedge$ (CPLX $\succeq$ 1.0) *then* Union $X_2^{\leq}$ [24; 58.5%]
rule 13. *if* (TYPE $\succeq$ 3) $\wedge$ (RVOL $\succeq$ 1.19) *then* Union $X_2^{\leq}$ [13; 31.7%]
rule 14. *if* (CPLX $\succeq$ 1.5) $\wedge$ (DATA $\preceq$ 1.0) *then* Union $X_2^{\leq}$ [25; 60.9%]
. . .
rule 18. (DATA $\succeq$ 1.08) $\wedge$ (TYPE $\preceq$ 2) *then* Union $X_3^{\geq}$ [10; 50%]

rule 19. (CPLX $\succeq$ 0.85) $\wedge$ (RELY $\succeq$ 0.94) *then* Union $X_3^{\geq}$ [3; 15%]
. . .
rule 26. (DATA $\succeq$ 1.04) *then* Union $X_2^{\geq}$ [20; 45%]

The syntax of these rules is different than previous ones. For instance, rule no 2 has the following interpretation: *if adjustment adaptation factor is at most 0.6 units and product complexity is at least 1.5 units, then effort is at most low*. Finally, we evaluated the classification performance of the dominance-based rules in a "leaving-one-out" verification. The important issue was to choose the classification strategy of using dominance based rules. Unlike in the "classical" case, here we could met the situation of multiple matching of the classified object to rules indicating different unions of decision classes. We used a strategy discussed in [3], which computes the most likely class basing on the learning examples supporting conflicting rules. The total classification accuracy was equal to 54%, while the accuracy in particular classes was the following: class 1 - 50%, class 2 - 62.5% and class 3 - 76.2%.

The analysis for the *TKDSI* decision attribute was performed in a similar way. The data set was also inconsistent. Inconsistent objects were the following 1, 2, 12, 60. The quality of classification was equal to 0.97. Accuracies of union approximations varied from 0.93 to 1.0 (union at most low - so class 1 - was precisely described). The core of attributes was the following: RELY, DATA, TURN, PCAP, SCED. The number of all reducts was equal to 70. The induced set of rules is described in table 5. We skip presenting examples of these rules. The classification accuracy was higher (57.14%), see table 6.

5.4. The Comparative Study of Various Classifiers

Table 6. Total classification accuracy [in %] for all compared classifiers

Learning approach	Decision attribute	
	Effort	*TKDSI*
MODLEM	63.49	58.73
4emka	54	57.14
C4.5	61.9	63.49
IBL	52.38	50.79
ANN (BP)	60.31	57.14
ANN (RBF)	46.26	46.26
Logit	63.49	58.73
bagging	63.49	60.3

In previous sections we presented the classification performance of the rule based classifiers induced either by MODLEM or DOMLEM algorithms. The question is whether the obtained classification accuracy is acceptable or not. To examine it, we performed a comparative study with other learning systems on the same data set. As our approach discovers knowledge represented in a symbolic way, we considered other methods that induce a similar kind of knowledge representation, i.e. *C4.5 system inducing decision trees* [24]. As the original data were mainly defined on numeric scales we also considered several approaches suitable for this kind of attributes, i.e. instance based learning approach using the 1–nearest neighbor principle (denoted as *IBL*), two types of neural networks (the first one was standard feed forward network trained by the Back Propagation algorithm and the other was the radial based network) and logistic regression (abbreviated as Logit). Moreover, to see how works the multiple classifier specialized for increasing classification accuracy we have chosen the bagging approach with using C4.5 to train component classifiers. Faster algorithms as tree learner, instance based learning were evaluated in the same way as our approaches, i.e. by means of ”leaving-one-out”. In a case of more time consuming approaches we decided to use the 10-fold cross validation. The majority of these implementations were available inside the WEKA software package[3] The obtained results are presented in table 6.

[3]WEKA is a data mining software toolkit developed at Waikato University, New Zeland

6. Discussion of the Results and Final Remarks

Let us summarize main conclusions resulting from the performed experiments:

1. Using the basic rough sets approach it was possible to precisely describe all decision classes for both classifications *Effort* and *TKDSI*. Although the data sets were consistent, the equivalence classes were mainly singletons, what indicated limited generalization abilities.

2. A unique identification of the most discriminatory attributes is not easy to done because of the empty core and too many reducts. However, it is possible to apply heuristic procedures and obtain the following subset of attributes (for the decision attribute *Effort*): DATA, AAF, CPLX, TYPE, RELY, ACAP, TIME. In a case of the decision attribute *TKDSI* the selected subset is: DATA, AAF, RVOL, TYPE, CONT. Studying the related research, some authors already indicate these and other attributes. For instance Ruhe in [26] selected the following attributes: TYPE, AAF, ACAP, VEXP, MODP, PCAP. Briand et al. identified the attributes ACAP, PCAP and STOR.

3. The use of the MODLEM algorithm resulted in inducing a compact set of around 16 rules, which contain on a average 3.4 elementary conditions and supported on a average by 4 learning examples. It is a quite satisfactory result knowing the number of equivalence classes in the rough sets analysis. Moreover, the syntax of these rules is general because of applying operators $<$ and $\geq$ in elementary conditions.

4. An evaluation of classification performance of the proposed rule based classifiers provided estimates of classification accuracy around 63%. Analysis of the detailed results presented in table 3 showed that it was easier to recognize class 3 (high effort or large program size) while discriminating between classes 1 and 2 (low and medium costs) was more difficult.

5. Introducing the dominance based rough set approach (DRSA) allowed us to discover new elements in the available data. Assuming the appropriate preference orders on criteria domains, we identified four inconsistent objects. Two of them, coded as 12 and 60, are difficult cases for both decision attributes. As a consequence unions of classes could not be precisely described, although the quality of classification was still high.

6. With the DRSA it was easier to identify the core of attributes and the number of reducts was much smaller than for the classic rough sets.

7. The sets of dominance based rules were larger (it was partly a consequence of considering 4 unions instead of 3 classes). Anyway these rules were shorter and more differentiated. Although the average rule strength for DOMLEM rules seems to be similar to the strength of MODLEM rules, the variance of their strength is different. There are more weaker (and specific) rules and only a few very strong ones (for two larger unions), while MODLEM rule strengths are located closer to an average value with a smaller dispersion.

8. The classification accuracy of DOMLEM rules were smaller then MODLEM ones. This could be caused by not sufficiently advanced classification strategy, which should solve more ambiguous case of multiple matching to conflicting unions.

9. The comparative study showed that the rules discovered by the MODLEM algorithm were among the best classifiers. The similar results were obtained by C4.5 induced decision trees (so also symbolic knowledge representation) and the bagging (which is a specialized approach to increase the classification accuracy, but in our case did not help too much). One should notice that popular neural networks provided worse results.

To sum up, the proposed rough sets and rule approach offers compact rule sets, which could be used as acceptable classifiers for predicting software project costs. The syntax of obtained rules is easy for human inspection, which is not a case for many of previously used approaches. Moreover, the rough sets analysis enables to identify the most influential cost factors. Comparing our result to previous study, e.g. [26], we could comment that we do not require any pre-discretization that change the domains of attribute (in this way we can then apply the preference orders on original criteria values). Moreover, we applied new rule algorithms that provided smaller sets of rules having acceptable classification accuracy - although it should be noticed that the direct comparison with the previous results is not possible because of the lack of information on Ruhe's preprocessing of data.

Our original contribution is also taking into account preference orders of attribute domains. The use of the dominance based rough sets approach allowed us to discover new type of inconsistency. According to our best knowledge it is the fist case study of applying this method to software engineering data.

Coming to final remarks, our approach tackles the issue of providing human interpretable patterns, which have good classification ability and could be helpful for project leaders to better understand the previous experience and possibly support their decisions while managing software development for new situations. However, it is necessary to remember limitations of using any learning methods in this kind of applications - see our discussion in section 2.

Finally, there are still some methodological open problems. For instance, in a case of DRSA it is necessary to make more research on advanced classification strategies using dominance based rules. Moreover, it would be fruitful to develop new methods for identification of interaction of rule elementary conditions and their importance for the object classification - some attempts have already been undertaken by studying, so called, fuzzy measures and conjoint analysis [10].

Acknowledgments:
The author thanks Jerzy Nawrocki for encouraging him to study problems of software engineering data and Adam Trendowicz for discussions on applying machine learning methods to cost estimation.

Appendix: List of cost factors
The specification of attributes according to COCOMO models is divided into the following categories:
(Prod) - product atributes (3)
(Comp) - computer attributes (5)
(Pers) - personnel attributes (6)
(Proj) - project attributes (4)
(oth) - factors not included in COCOMO (1)

The detailed description of attributes:
RELY - (Prod) required software reliability;
DATA - (Prod) size of database;

CPLX - (Prod) product complexity;
AAF - (oth) adaptation adjustment factor for size instructions that were adopted instead of newly developed;
TIME - (Comp) execution time constraint;
STOR - (Comp) main storage constraint;
VIRT - (Comp) virtual machine volatility (virtual machine = hardware and software under which works analysed system);
TURN - (Comp) computer turnaround time;
TYPE - (Comp) type of computer [maxi, midi, mini, micro];
ACAP - (Pers) analyst capability;
AEXP - (Pers) applications experience;
PCAP - (Pers) programmer capability;
VEXP - (Pers) virtual machine experience;
LEXP - (Pers) language experience;
CONT - (Pers) personnel continuity [low, nominal, high];
MODP - (Proj) use of modern programming practicies;
TOOL - (Proj) use of software tools;
SCED - (Proj) required development schedule;
RVOL - (Proj) requirements volatility.

References

[1] Basili V.R., Rombach H.D.: The TAME projects: Towards Improvement-Oriented Software Environments. *IEEE Transactions on Software Engineering*, SE-14, 1988, 758–773.

[2] Bazan J., Skowron A., Synak P.: Discovery of decision rules from experimental data. In: Lin T. Y., Wildberger A. (eds.), *Soft Computing: Rough Sets, Fuzzy Logic, Neural Networks, Uncertainty Management, Knowledge Discovery*, Simulation Councils, Inc., San Diego CA 1995, 276–279.

[3] Blaszczynski J., Greco S., Slowinski R.: Multicriteria classification using decision rules. In: *Proceedings of the 3rd Int. Conf. on Decision Support for Telecomunication and Information Society*, Warsaw, September 4-6, 2003; (an extended version accepted for publication in European Journal of Operational Research).

[4] Boehm B.W.: *Software engineering economics*, Englewood Cliffs, NJ: Prentice-Hall, 1981.

[5] Boehm B.W.: The COCOMO 2.0 Software Cost Estimation Model. *American Programmer*, Englewood Cliffs, July 1996, 2–17.

[6] Briand L.C., Basili V.R., Thomas W.M.: A Pattern Recognition Approach for Software Engineering Data Analysis. *IEEE Transactions on Software Engineering*, **18** (11), 1992, 931–942.

[7] Greco S., Matarazzo B., Slowinski R.: The use of rough sets and fuzzy sets in MCDM. In: Gal T., Stewart T., Hanne T. (eds), *Advances in Multiple Criteria Decision Making*, Kluwer, Dordrecht, 1999, chapter 14, pp. 14.1-14.59.

[8] Greco S., Matarazzo B., Slowinski R.: Multicriteria classification. In: Klosgen W. Zytkow J.M. (eds), *Handbook of Data Mining and Knowledge Discovery*, Oxford University Press, Cambridge, MA, 2002, 318–328.

[9] Greco S., Matarazzo B., Slowinski R., Stefanowski J.: An algorithm for induction decision rules consistent with the dominance principle. In: Ziarko W., Yao Y. (eds), *Rough Sets and Current Trends in Computing*, 2nd Int. Conference RSCTC 2000 Banff, October, 2000, LNAI vol. 2005, Springer Verlag, 2001, 304–313.

[10] Greco S., Matarazzo B., Slowinski R., Stefanowski J.: Importance and interaction of conditions in decision rules. In: J.Alpgini, J.Peters, A.Skowron, N.Zhong (eds.), *Rough Sets and Current Trends in Computing*: Proceeding of the 3rd Int. Conference RSCTC 2002, Spinger Verlag, LNAI no. 2475, 2002, 255–262.

[11] Grzymala-Busse J.W.: LERS - a system for learning from examples based on rough sets. In: R. Słowiński, (ed.) *Intelligent Decision Support*, Kluwer Academic Publishers, Dordrecht 1992, 3–18.

[12] Grzymala-Busse J.W.: Managing uncertainty in machine learning from examples. In: *Proc. 3rd Int. Symp. in Intelligent Systems*, Wigry, Poland, IPI PAN Press, 1994, 70–84.

[13] Grzymala-Busse J.W., Stefanowski J.: Three approaches to numerical attribute discretization for rule induction. *International Journal of Intelligent Systems*, 16 (1), 2001, 29–38.

[14] Kemerer C. F.: An empirical validation of cost estimation models. *Communications of ACM*, **30** (5), 1987, 416–429.

[15] Klosgen W. Zytkow J.M. (eds): *Handbook of Data Mining and Knowledge Discovery*, Oxford University Press, Cambridge, MA, 2002.

[16] Komorowski J., Pawlak Z., Polkowski L. Skowron A., Rough Sets: tutorial, In: Pal S.K., Skowron A. (eds.), *Rough fuzzy hybridization. A new trend in decision-making*. Springer-Verlag, Honkong, 1999, 3–98.

[17] Leung H., Zhang F.: Software Cost Estimation. Chapter in: *Handbook of Software Engineering and Knowledge Engineering*. World Scientific Pub. Co., 2002. Another version also available as chapter 23 in Estimation of software projects, R.S. Pressman & Associates Inc. Downloadable Reference Library.

[18] Mair C., Kadoda G., Lefley M., Phalp K., Schofield C., Shepperd M., Webster S.: An investigation of machine learning based prediction systems. *Journal of Systems and Software*, **53**, 2000, 23–29.

[19] Michalski R.S., Bratko I, Kubat M. (eds.): *Machine learning and data mining*, John Wiley & Sons, 1998.

[20] Mitchell T.: *Machine Learning*, Mac-Graw Hill, Boston, 1997.

[21] Pawlak Z.: *Rough sets. Theoretical aspects of reasoning about Data*. Kluwer Academic Publishers, Dordrecht/Boston/London, 1991.

[22] Peters J., Ramanna S.: Towards a Software Change Classification: A Rough Set Approach. *Software Quality Journal*, **11**, 2003, 121–147.

[23] Polkowski L.: *Rough sets: mathematical foundations*. In: Series: Advances in Soft Computing, Heidelberg, Physica Verlag, 2002.

[24] Quinlan J. R.: *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo CA, 1993.

[25] Ramanna S.: Rough Neural Network for Software Change Prediction. In: Alpigini J. *et al.*, editors, *Proceeding of the 3th Int. Conference on Rough Sets and New Trends in Computing* RSCTC 2002, Penn-Valey, LNAI 2475, Springer-Verlag, 2002, 602–609.

[26] Ruhe G.: Qualitative Analysis of Software Engineering. In: Tsumoto S. *et al.* (eds.), *Proceeding of the 4th Int. Workshop on Rough Sets and Fuzzy Sets and Machine Discovery*, Tokio University Press, 1996, 293–299.

[27] Ruhe G.: Rough set based data analysis in goal oriented software measurement. *Proceedings of the Third Symposium on Software Metrics*, Berlin, March 1996, IEEE Computer Society Press, 10–19.

[28] Slowinski, R., Stefanowski, J.: 'RoughDAS' and 'RoughClass' software implementations of the rough set approach. In: Slowinski R. (ed.), *Intelligent Decision Support. Handbook of Applications and Advances of Rough Set Theory*. Kluwer Academic Publishers, Dordrecht/ Boston/ London, 1992, 445-456.

[29] Skowron A.: Boolean reasoning for decision rules generation. In: Komorowski J., Ras Z. (eds.), *Methodologies for Intelligent Systems*, LNAI 689, Springer-Verlag, Berlin, 1993, 295–305

[30] Stefanowski J.: The rough set based rule induction technique for classification problems. In: *Proceedings of 6th European Conference on Intelligent Techniques and Soft Computing* EUFIT 98, Aachen 7-10 Sept., 1998, 109–113.

[31] Stefanowski J.: On rough set based approaches to induction of decision rules. In: Polkowski L., Skowron A. (eds.), *Rough Sets in Data Mining and Knowledge Discovery*, vol. 1, Physica-Verlag, 1998, 500–529.

[32] Stefanowski J.: *Algorithims of rule induction for knowledge discovery*. (In Polish), Habilitation Thesis published as Series Rozprawy no. 361, Poznan Univeristy of Technology Press, Poznan, 2001.

[33] Stefanowski J., Vanderpooten D.: Induction of decision rules in classification and discovery-oriented perspectives. *Int. Journal of Intelligent Systems*, **16** (1), 2001, 11–28.