Knowledge Discovery from Software Engineering Data: Rough Set Analysis and Its Interaction with Goal-Oriented Measurement

Günther Ruhe

Fraunhofer Gesellschaft
Institute for Experimental Software Engineering
D-67661 Kaiserslautern

Abstract. Knowledge discovery from software engineering measurement data is an essential prerequisite for software quality improvement. Software measurement is increasingly recognized as a device to better understand, monitor, predict, and improve software development and maintenance projects. The paper describes the interaction between goal-oriented measurement and rough set based analysis in the context of experimental software engineering. The gained experiences are based on the application of rough set analysis in practical problems of software engineering.

1 Introduction

Discovery, representation and reuse of software engineering know-how are of increasing importance for improvement in software development. The Quality Improvement Paradigm QIP [1] offers a general framework for performing systematic quality improvement. Software measurement and analysis of measurement data from carefully designed experiments is an essential part therein. Measurement is introduced in software organizations to gain qualitative and quantitative insight into the development processes for producing better products. Software measurement is increasingly recognized as an essential prerequisite to better understand, monitor, predict, and improve software development and maintenance projects. The Goal/Question/Metric (GQM) approach [2],[3] represents a systematic approach for tailoring and integrating the objectives of an organization into a practical mechanism to attain measurable goals in combination with models of software processes, products and quality perspectives.

Rough set theory [13] is a promising analysis approach which has been successfully applied in many real-life problems of various areas, e.g. medicine, pharmacology, business, and banking [16]. So far, not so much is known about applications in software engineering. The objectives of the paper are (i) to overview experiences gained from application of rough set analysis in practical software measurement, (ii) to describe the interaction between goal-oriented measurement and rough set based analysis in the context of experimental software engineering, (iii) to give a formal description of the different kinds of application of rough sets for the analysis problem within goal-oriented measurement, and (iv) to evaluate rough set based analysis from the point of view of knowledge discovery in experimental software engineering.

The remainder of this paper is organized as follows. Chapter two describes the experimental approach of software engineering and the crucial role of goal-oriented measurement therein. In chapter three, some background of rough sets is summarized. Subsequently, different forms of interaction between rough set based analysis and goal-oriented measurement are investigated. In chapter four we summarize some experiences gained from the application of rough set based analysis in the context of software measurement. The final chapter five is devoted to summary and conclusions.

2 Experimental Software Engineering and Goal-Oriented Measurement

2.1 The Quality Improvement Paradigm

The Quality Improvement Paradigm (QIP) is a basic strategy for systematic improvement of well defined aspects of software quality. QIP is based on (i) an appropriate characterization of the environment which provides a context for goal definition and (ii) systematic reuse of obtained experiences by packaging them in structured knowledge. The improvement process due to QIP is defined as an iterative process that repeatedly performs the following six steps [1]:

1. **Characterize:** Understand the environment based upon available models, data, intuition, etc. Establish baselines for the existing business processes in the organization and characterize their criticality. This characterization is based on implicit knowledge or on already available data.
2. **Set goals:** On the basis of the initial characterization of the capabilities that have the strategic relevance to the organization, set quantifiable goals for successful project and organization performance and improvement. Reasonable expectations are defined based upon the baseline provided by the characterization step.
3. **Choose process:** On the basis of the characterization of the environment and of the goals that have been set, choose the appropriate processes for achieving that goal. This includes selection of supporting methods and tools and making sure that they are consistent with the formulated goals.
4. **Execute:** Perform the processes constructing the products. Provide feedback based upon the data that are being collected.
5. **Analyze:** At the end of each specific project, analyze the data and the information gathered to evaluate the current practices, determine problems, record findings, and make recommendations for future project improvements.
6. **Package:** Consolidate the experience gained in the form of new, or updated and refined models and other forms of structured knowledge gained from this and prior projects, and store it in an experience base so it is available for future projects.

2.2 Goal-Oriented Software Measurement

The current state-of-the practice in software measurement is characterized in [14]. While measurement is considered more and more to be an essential background for decision making [17], efficiency of measurement (defined by the number of insights gained from analyzed data related to measurement effort) needs to be improved. Goal-oriented measurement based on the GQM approach was successfully applied in numerous organizations (compare [5], [9], [11]). GQM supports the operational definition of all kinds of measurement goals due to the top-down refinement of goals into metrics via questions. It emphasizes the necessity of collecting context data thereby integrating all essential influence factors. The result of the application of the GQM approach is the specification and implementation of a measurement program for a particular set of issues and a set of guidelines how to interpret the measurement data in the context of the goal. For both objectives, the inclusion of the project team members is a basic principle. Their knowledge and their view of the underlying software processes and products are captured by performing structured interviews and by interpreting measurement results during feedback sessions.

There are different analysis approaches which are applied in the different areas and disciplines of science. Analysis of software engineering data has some additional difficulties resulting from the combination of the following characteristics of software development:

- Software development is a human based technology,
- it has a large number of influence factors (human, process, problem, product, and resource related),
- the development process is very dynamic,
- the requirements of the final product are often changing,
- available data sets are often of small size, and
- available data are often incomplete and uncertain.

In what follows, we will investigate how rough set based analysis can contribute to knowledge discovery from experimental software engineering data.

3 Rough Set Based Analysis in the Context of Goal-Oriented Software Measurement

3.1 Brief Overview on Rough Sets

We use basic concepts and notation of rough sets as introduced in [13]. We assume a data representation called *decision table*. Rows of this table correspond to (software) objects and columns correspond to attributes. For each pair (object, attribute) there is a known value called a *descriptor*. We summarize all these data in the notation of an *information system* which is formally defined as a 4-tuple $S =< U, Q, V, f >$ where

- U is a finite set of *objects*,

- Q is a finite set of *attributes resp. metrics*,
- V_q is a *domain* of the attribute q,
- $V = \bigcup_{q \in Q} V_q$, and
- $f : U \times Q \to V$ with $\quad f(x,q) \in V_q \quad \forall q \in Q \quad \forall x \in U$ is called *information function.*

Let $S =< U, Q, V, f >$ be an information system and let $P \subseteq Q$ and $x, y \in U$ be two objects of U. x and y are called *indiscernible* by the set of attributes P iff $f(x,q) = f(y,q) \quad \forall q \in P$. Every $P \subseteq Q$ generates a binary relation on U called *indiscernibility relation* denoted by $IND(P)$. $IND(P)$ is an equivalence relation for any P and the family of all equivalence classes of this relation is denoted by $U \mid IND(P)$ or $U \mid P$. Equivalence classes of relation $IND(P)$ are called *P-elementary sets* in S.

$Des_P(X)$ denotes a *description* of P-elementary set $X \in U \mid IND(P)$ in terms of values of attributes from P, i.e.,

$$Des_P(X) = \{(q,v) : f(x,q) = v \quad \forall x \in X \quad \forall q \in P\}. \tag{1}$$

Indiscernibility of objects by means of attributes prevents their precise assignment to a set. A *rough set* is a set of objects which, in general, cannot be precisely characterized in terms of the values of the set of attributes, while a pair of a lower and an upper approximation of the collection can do. Assume an information system S, a subset $Y \subseteq U$ of objects and an equivalence relation $IND(P)$ for a subset of attributes $P \subseteq Q$. Then

$$\underline{P}Y = \bigcup \quad \{X \in U \mid IND(P) : \quad X \subseteq Y\} \tag{2}$$

is called *P-lower approximation* of Y. Correspondingly,

$$\overline{P}Y = \bigcup \quad \{X \in U \mid IND(P) : \quad X \cap Y \neq \emptyset\} \tag{3}$$

is called *P-upper approximation* of Y. An information system may be seen as a decision table $\mathcal{DT}$ assuming that $Q = \mathcal{C} \cup \mathcal{D}$ with $\mathcal{C} \cap \mathcal{D} = \emptyset$, where $\mathcal{C}$ is a set of *condition attributes* (related to independent variables), and $\mathcal{D}$, a set of *decision attributes* (related to independent variables). Let $U \mid IND(\mathcal{C})$ be a family of all $\mathcal{C}$-elementary sets called *condition classes* in $\mathcal{DT}$, denoted by $X_i \quad (i = 1, \ldots, k)$. Let, moreover, $U \mid IND(\mathcal{D})$ be a family of all $\mathcal{D}$-elementary sets called *decision classes* in $\mathcal{DT}$, denoted by $Y_j \quad (j = 1, \ldots, n)$.

$$\{Des_{\mathcal{C}}(X_i) \Longrightarrow Des_{\mathcal{D}}(Y_j) : X_i \cap Y_j \neq \emptyset\} \tag{4}$$

is called *$(\mathcal{C},\mathcal{D})$-decision rule* $\{r_{ij}\} \quad \forall i,j$. Rule r_{ij} is *deterministic* in $\mathcal{DT}$ iff $X_i \subseteq Y_j$, and *non-deterministic* otherwise. The (absolute) *strength* of a rule r_{ij} related to a given set of objects is defined as the number of objects matching that rule and is denoted by $Strength(r_{ij})$. The *relative strength* of a rule r_{ij} is defined as the ratio between (absolute) strength and the cardinality of decision class Y_j.

$$RelStrength(r_{ij}) = Strength(r_{ij})/card(Y_j) \quad \forall i,j. \tag{5}$$

3.2 Analysis Problems related to Goal-Oriented Measurement

We consider four analysis problems which are related to GQM-based measurement:

1. Attribute evaluation is devoted to evaluate the importance of particular attributes and to determine minimal subsets of attributes which are able to classify all the objects with the same quality as in the case of using all attributes. Evaluation of attributes includes elimination of redundant attributes from the decision tables.

2. Rule generation aims in describing knowledge discovered in performed experiments. The rules involve the relevant attributes only and explain the taken classification with respect to quality aspects. There can be used different options for computing the rules.

3. Classification assumes that there is a set of rules which is based on performed experiments. The question is to apply the existing rules for classification of new software objects which are measured in the same way as the former objects. Accuracy of prediction clearly depends on size and quality of the data.

4. Experience base update considers the update of the set of generated rules in the case that new objects have been measured or integration of rule sets form different (comparable) experiments is intended.

3.3 Contributions to Goal-Oriented Measurement

Feedback sessions are organized meetings integrating members of the project and the measurement team [9]. The main objective of feedback sessions is to discuss the results of the measurement program and to derive interpretations by the project team from the data collected so far. Rough set based analysis results are used as an essential input for performing interactive interpretation of measurement programs. The idea here is to support human decision making and to facilitate integration of human expert knowledge with knowledge derived by formal method based on experimental data. In accordance to the introduced analysis problems, the following contributions are of importance:

1. Attribute evaluation

- Core attributes give an indication on the most important influence factors in a measurement program.
- Redundant attributes are serious candidates to be eliminated in subsequent measurements.

2. Rule generation

- Deterministic and non-deterministic rules derived from formal rough set analysis are an essential input for feedback sessions. There are four principal cases which my occur with respect to their relationship to established hypotheses:
  - Confirmation of hypotheses,

  - refinement of hypotheses,
  - rejection of hypotheses,
  - formulation of new hypotheses.
- The validity of underlying models and even of collected data has to be investigated again in the case on inconsistencies and in case of experiences completely contradicting the hypotheses. From performed rough set analysis can be derived suggestions for updates of the underlying measurement program. This concerns elimination of attributes on the one side and addition of new attributes on the other side.
- Rough set based rules have been used as foundation for integration of human based expert knowledge with the rules formally derived from experimental analysis. Final rules gained from this integration can be used as decision support in subsequent projects when having similar characteristics.

3. Classification

- Rules offer explanation why certain concepts, decisions or phenomena occurred. This guides subsequent improvement actions.
- Forecast of product and process characteristics such as cost, effort, duration, or quality can be performed.

4. Experience base update

- Rough set analysis is applied from scratch for the new (enlarged) set of objects. However, there is a strong sensitivity between data and resulting rules.
- Integration of different rule sets related to similar objects [12].

4 Evaluation of the Rough Set Approach from an Applications Point of View

4.1 Evaluation Background

We summarize some experiences gained from the application of rough set based analysis in the context of goal oriented software measurement. We performed a series of computations [6], [8], [10], and [15]. In all of them we used the system ProFIT [18] developed at Poznan University of Technology. We studied different practical problems from an experimental software engineering point of view:

1. Determination of essential cost drivers in software projects [15] based on the COCOMO model introduced originally in [4].
2. Module criticality in the context of software maintenance [10].
3. Product and process related flexibility in software development [8].
4. Goal-oriented measurement based evaluation of configuration management [6].

We discuss here some results and experiences gained from application of rough set analysis within these measurement programs.

4.2 Determination of Essential Cost Drivers in Software Projects

The COnstructive COst MOdel to predict effort or cost of software projects was introduced in [4]. We enhance the original approach by performing rough set analysis. The objective is to determine the most essential cost drivers, their preference relation and an explanation how they interact. For that reason, we exploit the original COCOMO data base as used by Boehm. It contains data from 63 completed projects with 23 condition and two decision attributes. Some data were already based on nominal or ordinal scale. All remaining ones were discretized. Among condition attributes we considered:

- Primary programming language used (LANG),
- COCOMO adaption adjustment factor (AAF),
- type of computer used (TYPE),
- application experience (AEXP),
- analyst capability (ACAP),
- programmer capability (PCAP),
- virtual machine experience (VEXP), and
- use of modern programming practices (MODP).

The two decision attributes were

- Kilo delivered source instructions (KDSI), and
- actual man-month required for the project (MM).

We have made a ranking of cost drivers according to their number of occurrences related to all rules obtained from the chosen reduct. Using $' \succ '$ for order relation between cost drivers with $'A \succ B'$ for 'A is a stronger cost driver than B' yields in

$$TYPE \succ AAF \succ ACAP \succ VEXP \succ \{MODP, PCAP\} \succ \{AEXP, LANG\}.$$

4.3 Module Criticality in the Context of Software Maintenance

In [10], rough set based analysis was compared with logistic regression to investigate faultiness of more than hundred modules in the context of software maintenance. The following two activities have been carried out during the transition from two versions of a software product:

- Corrective maintenance: failures reported from customers were being fixed.
- Adaptive maintenance: The product was being transferred from platform OpenVMS/VAX (version 6.0) to platform OpenVMS/Alpha (version 6.1).

Among others, accuracy of prediction was compared with respect to three criteria:

1. *Overall correctness:* Proportion of modules that have been classified correctly.
2. *Faulty module completeness:* Proportion of faulty modules that have been classified as faulty.

3. *Faulty module correctness:* Proportion of modules that have been classified as faulty and were faulty indeed.

For all classes of examples, rough set based analysis performed better with respect to overall correctness and faulty module correctness. On the other side, logistic regression based analysis produced better results with respect to faulty module completeness. Overall, the two techniques look supplementary, in that logistic regression performs better with respect to faulty module completeness, while rough set analysis is better with respect to overall completeness and faulty module correctness.

In [10] a hybrid approach was proposed for integrating the strengths of rough set and logistic regression based analysis. Let $\mathcal{M}$ be the set of all modules. We denote by LR[$\mathcal{M}$] (RS[$\mathcal{M}$]) the application of logistic regression (respectively rough set analysis) on set $\mathcal{M}$. This results in classification of modules into faulty and non-faulty ones respectively in classification rules $\mathcal{R}_{\mathcal{RS}}(\mathcal{M})$. This is described in steps (i) and (ii), respectively. The application of (rough set) classification rules $\mathcal{R}_{\mathcal{RS}}(\mathcal{M})$ on a new set $\mathcal{T}$ is denoted by $\mathcal{R}_{\mathcal{RS}}(\mathcal{M}) \circ [\mathcal{T}]$ as performed in step (iii). Finally, the set of faulty $\mathcal{MF}$ and the set of non-faulty modules $\mathcal{MNF}$ is obtained in step (iv).

Hybrid approach H

(i) $LR[\mathcal{M}] \Rightarrow \mathcal{MF}_1 \oplus \mathcal{MNF}_2$
(ii) $RS[\mathcal{M}] \Rightarrow \mathcal{R}_{RS}(\mathcal{M})$
(iii) $\mathcal{R}_{RS}(\mathcal{M}) \circ [\mathcal{MF}_1] \Rightarrow \mathcal{MF}_3 \oplus \mathcal{MNF}_4$
(iv) $\mathcal{MF} := \mathcal{MF}_3, \quad \mathcal{MNF} := \mathcal{MNF}_2 \oplus \mathcal{MNF}_4$

For validation purposes, we suggested two heuristics to classify those modules that do not match a deterministic rule in the rough set approach:

H1 Modules matching a non-deterministic rule are classified in accordance to that conclusion which is supported by the maximum number of modules related to $\mathcal{M}$ (and thus is more likely to occur). The related approach is denoted by LR*RS.
H2 Modules matching a non-deterministic rule are classified as faulty. The related approach is denoted by LR&RS.

We compare quality of classification from logistic regression, rough sets, and the two hybrid approaches with respect to the three parameters described above. The results obtained from performing leaving-one-test are given in Table 1. We observe that the hybrid approach performs better than both LR and RS. There are small differences between the application of the two heuristics, where the first is more general.

4.4 Product and Process Related Flexibility in Software Development

There is an increasing interest in understanding and subsequently improving flexibility in software development [7]. Flexibility can be roughly understood as

Parameter	LR	RS	LR*RS	LR&RS
Overall completeness	70.8%	93.85%	96.9%	96.1%
Faulty module completeness	94.1%	45.4%	81.2%	90.9%
Faulty module correctness	21.3%	71.4%	81.2%	71.4%

Table 1. Comparison of classification results

the relationship between generated functionality of products and the required effort for achieving this. The motivation for studying flexibility is the strong request to reduce time to market while fulfilling given resource constraints. The following main attributes were studied in [8] to describe influence factors on flexibility:

- Number of involved technical units,
- number of involved IT units,
- degree of usage of external product data base,
- degree of usage of external table,
- number of modified modules in the subsystem,
- degree of reuse of existing components,
- degree of openess for extension,
- usage of standard test cases,
- degree of redesign,
- amount of modifications due to technical requirements, and
- completeness of technical requirements before project start.

There were performed different computations based on (i) different discretizations, (ii) different algorithms (LEM2 and an all rules procedure), (iii) different data sets, and (iv) usage of priorities. We describe here classification results for the leaving-one-out test based on real-world data related to 34 objects/subsystems. The decision attribute (flexibility) had four (ordinal-scaled) values. Most of the attributes were given related to nominal or ordinal scale. For the remaining ones, discretization was done by domain experts. With respect to the 'All rules' option offered by ProFIT we introduced the additional constraint that only rules fulfilling a predefined level α of relative strength are considered. In our case we have chosen $\alpha = 1/3$, i.e., only rules with relative strength at least 1/3 were taken into account. In Table 2 we report classification results based on leaving-one-out test obtained from the application of three approaches to the same data set. Non-deterministic correct (incorrect) classification means the percentage of objects for which more classes were found and the correct class is among (not among) them. The 'Modified all rules' approach generated 22 rules of relative strength greater than or equal to α. Among the three approaches, 'Modified all rules' performed best with respect to correctness of classification.

Algorithm	Modified all rules	LEM2	LEM2 with priorities
Correct classification	50%	35.3 %	50%
Non-deterministic correct	14.7 %	2.9 %	5.9 %
Not classified	11.8 %	29.4 %	2.9 %
Non-deterministic incorrect	5.9 %	0 %	5.9 %
Wrong classification	17.6 %	32.4 %	35.3 %

Table 2. Classification results based on leaving-one-out test.

5 Summary and Conclusions

We have shown that rough set based analysis offers a set of important insights for improvement in software engineering development. Typically, the concluded rules can be used for making predictions on future behavior in software development. Thus, they are a crucial contribution for learning in an experience factory environment [1]. Moreover, the operational rules can be used as decision support for monitoring and controlling ongoing development activities. Finally, from careful analysis of measurement data, inconsistencies in modeling and in the collected data can be detected. This leads to better and more accurate models underlying future investigations. On the other side, goal-oriented measurement delivers considerable support in finding out those attributes which are offering the best chance for knowledge discovery.

There is a number of open research problems which should be studied to enlarge the scope of applications. Among them is the question how to apply rough set analysis in the case of varying sets of attributes within one set of objects. Another questions is related to how to derive the rules. Rough set based analysis exclusively takes into account the values of the investigated attributes. At least three extensions are motivated by practical applications: (i) To define some formalism for interaction with expert knowledge, (ii) to allow also formulation of rules which are true with a predetermined level of confidence, and (iii) to develop rule sets which are stable in case of additional data and experiments.

From the software engineering application point of view, different approaches should be taken into account for analyzing measurement related data. This has the obvious advantage of producing broader and/or more confident conclusions from experimental data. For the application of these different approaches, the underlying assumptions and their evaluated strengths and weaknesses are of great importance. Rough set based analysis with its weak assumptions plays a prominent role therein.

References

[1] V. R. Basili, G. Caldiera, and H. D. Rombach: Experience Factory. In John J. Marciniak, editor, *Encyclopedia of Software Engineering*, volume 1, pages 469–476. John Wiley & Sons, 1994.

[2] V. R. Basili and H. D. Rombach: The TAME project: Towards Improvement-Oriented Software Environments. *IEEE Transactions on Software Engineering*, SE-14(1988), 758-773.

[3] V. R. Basili and D. M. Weiss: A methodology for collecting valid software engineering data. *IEEE Transactions on Software Engineering*, SE-10(1984), 728–738.

[4] B. W. Boehm: *Software Engineering Economics.* Advances in Computing Science and Technology. Prentice Hall, 1981.

[5] European Systems and Software Initiative: Customized Establishment of Measurement Programs (CEMP), December 1995. ESSI Project # 10358, Final Report.

[6] G. Cugola, P. Fusaro, A. Fugetta, L. Lavazza, S. Manca, M.R. Pagone, G. Ruhe, and R. Soro: An Experience in Applying GQM to the Evaluation of the Impact of Configuration Management Technology. submitted to *International Software Metrics Symposium 1997*

[7] H. Günther, H. D. Rombach, and G. Ruhe: Kontinuierliche Qualitätsverbesserung in der Softwareentwicklung- Erfahrungen bei der Allianz Lebensversicherungs-AG (in German). *Wirtschaftsinformatik* 38(1996), 160-171.

[8] S. Hartkopf: Analysis of Software Engineering Data in GQM-Based Measurment: Feedback Sessions with Rough Sets. Diploma Thesis. University of Kaiserslautern, Department of Computer Science, 1997.

[9] B. Hoisl, M. Oivo, G. Ruhe, and F. van Latum: No Imrovement without Feedback: Experiences from Goal-Oriented Measurement at Schlumberger. *Proceedings 5th European Workshop on Software Process Technology*, Nancy October 1996, Lecture Notes in Computer Science Vol. 1149, 167-182.

[10] S. Morasca and G. Ruhe: Knowledge Discovery from Software Engineering Measurment Data: A Comparative Study of two Analysis Techniques. *Proceedings of the 10th International Conference on Software Engineering and Knowledge Engineering*, Madrid 1997.

[11] National Aeronautics and Space Administration:. Software measurement guidebook. Technical Report SEL-84-101, NASA Goddard Space Flight Center, Greenbelt MD 20771, July 1994.

[12] O. K. Ngwenyama and N. Bryson: A Formal Method for Analyzing and Integrating the Rule-Sets of Multiple Experts, *Information Systems*, Vol. 17, No. 1, 1–16, 1992.

[13] Z. Pawlak: *Rough Sets: Theoretical aspects of reasoning about data.* Kluwer Academic Publishers, 1991.

[14] H. D. Rombach, V. R. Basili, and R. W. Selby (editors): *Experimental Software Engineering Issues: A critical assessment and future directions.* Lecture Notes in Computer Science Nr. 706, Springer–Verlag, 1992.

[15] G. Ruhe: Qualitative Analysis of Software Engineering Data Using Rough Sets, *Proceedings of the Fourth International Workshop on Rough Sets, Fuzzy Sets, and Machine Discovery*, Tokyo, November 1996, 292–299.

[16] R. Slowinski. *Intelligent Decision Support: Handbook of applications and advances of the rough set theory.* Kluwer Academic Publishers, 1992.

[17] G. Stark, R. C. Durst, and C. W. Vowell: Using metrics in management decision making. *IEEE Computer*, 27(1994), 42–48.

[18] R. Susmaga and R. Slowinski: Rough Processing of Fuzzy Information Tables (ProFIT). Institute of Computing Science. Poznan University of Technology. Poznan 1996.