AN ADAPTABLE SOFTWARE PRODUCT EVALUATION METRIC

Timm Euler
Computer Science VIII
University of Dortmund
D-44221 Dortmund, Germany
email: euler@ls8.cs.uni-dortmund.de

ABSTRACT
Software product evaluation is a tool to choose among alternative software products for an intended application, but can also assist the developers of the software in judging the relevance of their product and in identifying missing functionalities. The suitability of software products is dependent on the intended application scenario and environment, which is why quantitative, objective metrics for product evaluation are difficult to find. This paper presents an adaptable method of quantifying product characteristics, and includes an application of the method in tool selection, for knowledge discovery (data mining) tools. The method can be made objective by providing precise evaluation plans or scenarios. It is adaptable by the evaluator to reflect different levels of detail, and/or different prioritisations of criteria, in different evaluation scenarios. Two possible levels of details are illustrated based on the example evaluation.

KEY WORDS
Software Product Evaluation, Software Evaluation Metrics, COTS Comparison

1 Introduction

There are many aspects of software which can be evaluated. A useful distinction can be made between the development of a software and its actual use as a *product*. The main evaluations concerning the development of software assess the quality and correctness of the source code; this is usually called *testing*. A higher-level type of evaluation assesses the development *process* in an institution, to see whether it follows certain standards that make the process controllable and repeatable [1].

The present work is concerned with software product evaluation, also referred to as COTS (commercial off-the-shelf) software evaluation. Since the intended employment scenario determines which aspects of the software product are relevant for an evaluation, it is difficult to find general, objective criteria for the evaluation. This paper presents a methodology to identify application-dependent, basic criteria that extends previous work [2], and contributes an adaptable scheme, or metric, of quantifying the degree to which the criteria are fulfilled. The adaptability of the method can reflect different weightings (prioritisations) of the criteria as well as different granularities of detail. The latter allows to present the same evaluation on different levels of detail, so that the presentation can be adapted to different audiences like technicians or decision makers. The method is exemplified by an evaluation of software tools in the area of data mining, or knowledge discovery in databases (KDD), addressing several methodological deficiencies that previous work on evaluating KDD tools showed. The effect of changing the presentation level is illustrated by two evaluation tables that are based on the same data, but employ different granularity levels.

After section 2 describes related work, section 3 presents the evaluation methodology proposed in this paper. Section 4 then applies the methodology to data mining tools and presents a quantitative and objective evaluation of such tools, on two levels of detail. Finally, section 5 concludes the paper.

2 Related Work

Software product evaluation aims at determining the extent to which a software product fulfills a number of specified characteristics [3]. The characteristics of software *quality* are defined in an international standard, ISO/IEC 9126, entitled "Information Technology—Software Product Evaluation—Quality Characteristics and Guidelines for their Use", developed in 1991 and slightly modified several times afterwards. It defines six main characteristics of software quality, each with several subcharacteristics. These characteristics can be the subject of an evaluation of a software product. While the standard aims at comprehensiveness, [4] and others have pointed out that different users of a product may have rather different quality requirements, and that it may be difficult for an organisation to determine the level and type of quality required in a specific situation.

Importantly, the evaluation itself should also follow a standard procedure in order to be as objective as possible, and in particular to be reproducible. To this end another standard was published in 1999, the ISO 14598 standard, entitled "Information Technology—Software Product Evaluation". It introduces four *phases* that the evaluation process should follow:

1. Establish evaluation requirements: The purpose of the evaluation, and the types of products to be evaluated, must be identified in this phase. Most importantly,

a *quality model* is set up, which lists the characteristics that are agreed to bear an influence on the quality. The ISO 9126 quality characteristics provide a useful guide, or a checklist, for the identification of quality-related issues in a particular evaluation, but the ISO 14598 standard also allows other categorisations of quality that are more appropriate under the given circumstances. ISO 14598 explicitly states that there are no established methods for producing software quality specifications.

2. Specification of the evaluation: Since the ISO 9126 characteristics are not directly quantifiable, metrics that are correlated with them have to be established. The term "metric" is used in ISO 14598 not in the usual mathematical sense, but refers to a quantitative scale and a method which can be used for measurement. The word "measure" is used to refer to the result of a measurement (the term "score" is also used in this work). According to ISO 14598, every quantifiable feature of software that correlates with a characteristic from the quality model can be used as a metric. For every metric, a written procedure is needed that prescribes the assignment of measured values to it, to achieve objectivity.

3. Design of the evaluation process: An evaluation plan is produced that specifies the required resources, e.g. people, techniques or costs, and assigns them to the activities to be performed in the last phase.

4. Execution of the evaluation: Measurements are taken and scores computed as fixed in the evaluation plan.

In [3] a critical review and some refinements of this process can be found. In particular, the importance of establishing and prioritising the goals of an evaluation, and of involving all stakeholders of the evaluation in this, are stressed. The first aspect is reflected in the present work by the adaptability of the presented methodology, while the second is irrelevant because the methodology is independent from concrete applications.

A new standard, ISO 25000, entitled "SQuaRE—Software Product Quality Requirements and Evaluation" is currently being developed to combine ISO 9126 and ISO 14598 [5].

The present work addresses mainly the identification of a quality model, and the specification of metrics for its quality characteristics (first and second phase above). Previous work on identifying suitable quality models for COTS software [6] (the first phase above) has refined the ISO 9126 standard, while the approach presented here is more empirical, and inspired by [2]. The latter authors suggest to identify those features of a technology that distinguish it from existing technologies. Such distinctive features are called "technology deltas". This approach evaluates a product with emphasis on its relation to competitive products. Section 3 takes up this idea. [2] stresses that the technology deltas should be evaluated in well-defined, sharply focused usage contexts, because then the extent to which a technology delta supports a given context can be evaluated. A limitation of this approach is that it can lead to a focus on *functional* quality characteristics as defined in ISO 9216.

Regarding evaluation techniques, [7] argues for the use of weighted checklists, where the presence or absence of a number of agreed features is indicated and integrated into an overall score. Checklists are easy to customise and are a transparent, reproducible method of evaluation. A problem is the choice of items on the list, that is, the identification of the quality model (first phase above); [7] argues that the only way to make this choice less subjective is to document and justify it extensively. The technology delta method is another way to increase objectivity, as it is an empirical method.

As regards metrics (see the second phase above), obviously no internal, source-code related metrics can be used for COTS products [8]. Previous research on product metrics has mainly concentrated on such internal metrics (e.g. [9, 10]). Research on COTS evaluations has concentrated on process-oriented aspects [11, 6] but has not established quantitative metrics, except for [12, 8]. In [12], rather general metrics are given, only some of which are external, but require much effort to measure (such as the percentage of design goals met by the finished product). In contrast, in [8] an approach similar to the one in this paper seems to be used, but no details are given, nor any examples from a concrete evaluation project.

3 An Adaptable Product Evaluation Metric

In the following, the proposed methodology is developed in four subsections, following the four phases of the standard product evaluation process introduced in section 2.

3.1 Establishing evaluation requirements

The ISO 14598 standard requires the specification of the purpose of the evaluation, the type of products to be evaluated, and the quality model in this phase. This work assumes that the purpose of the intended evaluation is to gain a detailed, yet clear picture of the strengths and weaknesses of one or more available software tools in a specific area. The type of products to be evaluated is restricted to finished software products and can be further limited in a concrete instantiation of this evaluation process (e.g. section 4).

The quality model is derived empirically in this work. Basic evaluation aspects are identified using the technology delta method described in section 2, by comparing *features* of different products. Another source for features can be a thorough analysis of necessary features in a particular application domain, as described in [13]. This results in lists indicating the presence or absence of the *distinctive* features in the products. Typically, such features will be very

fine-grained or detailed. For example, one word-processing tool might offer automatic enumeration for item lists, while another might only offer manual enumeration. Such a level of detail provides a maximum amount of information, but is usually not appropriate to gain clear overviews of tools, though it could be appropriate for the developers of such software to identify missing features in their products. Therefore section 3.2 introduces an adaptable aggregation method.

Non-functional characteristics are less amenable to the technology delta method, and other methods to complete the quality model may be useful (e.g. [6]). Yet many non-functional characteristics can be broken down into particular features of a product if the employment circumstances are known.

3.2 Specification of the evaluation

Having found the quality model (the basic, boolean features) in the previous phase, it must now be assigned a metric, in the sense defined in ISO 14598 (see section 2). A simple metric would assign a boolean value to each identified distinctive feature, indicating either its presence or absence. However, many small groups of features are usually related in a rather natural way. For example, all features related to enumeration of item lists in word-processing tools (like automatic indentation, availability of different "bullets", etc.) could be grouped to reflect the strength of the tool in the area of enumeration lists. Therefore, such naturally related features are grouped together in this work, and each group forms a *criterion*. The *n-of-m* metric is used to indicate the strength of a tool with respect to such a criterion: m is the number of features grouped together for this criterion, and n is the number of features that are present in the given tool. Thus each n-of-m criterion could be transformed into m boolean criteria. A simple score can be assigned to each tool under each criterion, which is the real value $0 \leq n/m \leq 1$. To increase the adaptability of this method, each of the m features can be weighted before the score for the criterion is computed. To allow for single features that cannot be grouped, m may be 1.

This method allows much flexibility concerning the groupings of the basic features. For a quick overview or superficial comparison, only the more important features can be used, or larger inherently related groups can be formed. This corresponds to larger average values of m. For detailed surveys, more fine-grained criteria can be used, so that the list of criteria is longer but the average value of m is lower. These options are illustrated using the example application in section 4.

The measures for several single criteria can be combined to more integrated scores by building weighted sums, where the sum of the weight coefficients should be 1.0. For example, to assess the strength of a tool in text formatting, all criteria related to this can be evaluated and combined to a single value. If desired, a single global score could be computed for every tool to get a ranking of the tools, though such a ranking would hide many aspects that the detailed score list can provide.

The methodology described here results in objective criteria, with reproducible results, if a written procedure that prescribes how to identify the presence or absence of each feature in a criterion is given (see also section 4).

A limitation to this methodology may be that, when applied to a different type of software products, not all technology deltas might correspond to boolean features that can easily be grouped. Some features, such as performance-related features, require a real-valued, continuous scale. However, such metrics can be mapped to the real interval $[0..1]$ easily, which makes them easily combinable with n-of-m metrics. A more serious limitation is that different n-of-m criteria can result in identical values when evaluated, although the respective values of n and m are different. It is not clear whether the fulfillment of 2 out of 4 features of a criterion "means" the same strength as the fulfillment of 4 out of 8 features. However, to compare a number of software products under any given criterion, of course the same value of m is always used.

3.3 Design of the evaluation process

Based on the list of basic features to evaluate, an evaluation plan can be set up that serves to fix in writing the exact procedures for finding out whether a feature is fulfilled by a given tool or not. If past experiences (evaluations) based on the same list of features are available, it is suggested to sort the list of features, in descending order by the degree of fulfillment of the features in the tools examined previously. In this way, the evaluation plan will check the most common features first, focusing on more and more specific and rare features as the evaluation proceeds. Resource-activity assignment depends on the concrete evaluation scenario, thus nothing is said about it in this general methodology.

3.4 Execution of the evaluation

Executing the evaluation consists of following the execution plan, taking the measurements required by the criteria, and documenting them, as exemplified in section 4.

4 An Application Example

The evaluation methodology from the previous section was applied in the field of Data Mining, or Knowledge Discovery in Databases (KDD), in order to demonstrate its practical usefulness. Previous work on comparing KDD tools [14, 15, 16, 17] shows a number of methodological deficiencies, in that no evaluation standards are used, no written evaluation procedures ensure the reproducibility of results, the list of criteria is not justified in a systematic way, and no flexible quantification of measurements is given. Thus the presented approach led to a quantitative and reproducible evaluation of KDD tools for the first time.

Six KDD tools were used for this evaluation: MiningMart is the result of a research project [18], while SPSS Clementine, Prudsys Preminer, IBM Intelligent Miner, SAS Enterprise Miner and NCR Teradata Warehouse Miner are commercial products. See [19] for brief descriptions of these tools. This evaluation concentrated on all aspects related to data processing and models of this process. The goal of the evaluation was to give an *overview* of the strengths and weaknesses of various KDD tools in the processing area. The technology delta method is particularly useful for such a purpose. Its application led to 123 basic boolean features. Such a detailed list of boolean features counteracts the aim of the evaluation stated above, while on the other hand, it can serve developers of such tools as a checklist of functionalities they might want to include. Yet, for decision making, for instance tool selection, more aggregated views of the results are needed.

Tables 1 and 2 show two different groupings into higher-level criteria. Table 2 is suggested as appropriate for actual evaluation projects; the criteria it uses are explained in a technical report [19]. In contrast, table 1 is more suitable for final decisions, public presentations, or more superficial overviews. It aggregates all boolean features into fewer criteria with larger values of m. Some of these latter criteria are described below, to give a flavour. However, since this is not a paper about KDD, more detailed discussions are omitted. All scores in both tables are built using equal weights for each boolean feature, because a weighting would be application-dependent and no particular application scenario was intended to be modelled in this evaluation. However, various weighting schemes, in particular linear combinations of boolean features, are easily applicable to the scores presented here, so that the evaluation can be adapted to user preferences or certain application scenarios.

The discussion of criteria below is a condensed version of the evaluation plan (see section 3.3), in that it prescribes (at least to KDD experts, who are familiar with the terminology) how to check the single boolean features that form a criterion. For each criterion of table 1, a few examples of boolean features included in it are given below, and also the list of criteria in table 2 that it subsumes. A more elaborate description [19] makes the evaluation as objective as possible by prescribing how to decide whether a feature is fulfilled in a given tool or not, and by providing a use case that is small yet comprehensive enough to check all criteria in a few hours.

Data handling A KDD tool must at least be able to import data from flat files in arbitrary formats, and relational databases using ODBC. Further, the user must be able to specify the place of processing: in the database or on hard disc. Data must be inspectable at all points in the process, and so on (criteria 1, 2, 7, 8 and 19 from table 2).

Caching To enable the handling of large data sets, tools must not attempt to store data sets in temporary files or database tables unless the user has determined that this be done. To enable informed decisions by the user, intermediate data set sizes must be estimated before the data set is computed, must be available after it is computed, and every intermediate data set must be clearly linked to the operator that produced it (criteria 3 through 6).

Data Modelling This criterion subsumes aspects concerning the explicit representation of data sets in the tool, such as data types, relations between data sets, statistical information, and so on (criteria 9 through 16, 21 and 27). This kind of information must not only be recognised automatically, but must also be available for the user when choosing parameters or setting up formulas that depend on this information.

Process Modelling This criterion collects features regarding the explicit representation of the KDD process, including its automatic adaptation to changed input and others. It subsumes criteria 17, 19, 20, 25, 28-30, 35, 37 and 45. For example, it is checked that the data flow can be modeled as an unrestricted directed acyclic graph, or that this graph can be partitioned into meaningful subparts by the user (chunking). These chunks must be nestable (this is one basic, boolean feature).

Execution A KDD process must be executable from the tool that models it. This criterion reflects a tool's strength in supporting this by logging, displaying the current status, etc (criteria 31-34).

Operators The list of processing operators available in a tool, together with a few special options, is reflected in this criterion. It unifies criteria 22-24, 26 and 36.

Mining Some aspects around the actual data mining are also related to data processing [13]. This is reflected in criteria 36 and 38-44. They concern the data flow that is needed for experiments with and evaluations of the mining algorithm, plus some administrative issues.

5 Conclusions

This paper has addressed software product evaluation and provided an adaptable, quantitative and objective methodology to identify lists of criteria and evaluate software products based on the criteria. The n-of-m metric was introduced for this purpose, and demonstrated using an example product evaluation in the field of KDD, addressing a number of methodological deficiencies that previous evaluations showed. The metric is adaptable to various purposes, weighting schemes and levels of detail. The methodology is applicable to any product evaluation, and easily applicable in practice.

References

[1] Mark C. Paulk, Charles V. Weber, Bill Curtis, and Mary Beth Chrissis. *The Capability Maturity Model: Guidelines for Improving the Software Process*. Addison-Wesley, 1995.

No	Name	m	MM	Clem.	Prem.	IBM	SAS	NCR
1	Data Handling	14	0.43	0.79	0.57	0.43	0.64	0.36
2	Caching	7	0.71	0.43	0.43	0.71	0.43	0.71
3	Data Modelling	25	0.76	0.88	0.36	0.6	0.68	0.32
4	Process Modelling	28	0.39	0.57	0.5	0.21	0.43	0.14
5	Execution	18	0.39	0.28	0.5	0.28	0.39	0.11
6	Operators	22	0.73	0.63	0.32	0.36	0.36	0.45
7	Modeling/Deployment	9	0.22	0.56	0.44	0.44	0.44	0.33

Table 1. Higher-level evaluation table. The value of n/m is given for each criterion and each tool.

[2] Alan W. Brown and Kurt C. Wallnau. A Framework for Systematic Evaluation of Software Technologies. *IEEE Software*, 13(5):39–49, 1996.

[3] Teade Punter, Rob Kusters, Jos Trienekens, Theo Bemelmans, and Aarnout Brombacher. The W-Process for Software Product Evaluation: A Method for Goal-Oriented Implementation of the ISO 14598 Standard. *Software Quality Journal*, 12(2):137–158, 2004.

[4] R.J. Kusters, R. van Solingen, and J.J.M. Trienekens. User-Perceptions of Embedded Software Quality. In *Proceedings of the STEP97 Conference*. IEEE Computer Society Press, 1997.

[5] Witold Suryn, Alain Abran, and Alain April. ISO/IEC SQuaRE. The Second Generation of Standards for Software Product Quality. In *Proceedings of the 7th IASTED International Conference on Software Engineering Applications*, Marina del Rey, CA, USA, 2003.

[6] Juan P. Carvallo, Xavier Franch, Gemma Grau, and Carme Quer. On the Use of Quality Models for COTS Evaluation. In *Proceedings of the International Workshop on Models and Processes for the Evaluation of COTS Components (MPEC)*, Edinburgh, UK, 2004.

[7] Teade Punter. Using Checklists to Evaluate Software Product Quality. In *Proceedings of the 8th European Software Control and Metrics Conference (ESCOM)*, Berlin, Germany, 1997.

[8] Regina Colombo and Ana Guerra. The Evaluation Method for Software Products. In *Proceedings of the International Conference on Software and Systems Engineering and Their Applications (ICSSEA)*, Paris, France, 2002.

[9] Jean Mayrand and Francois Coallier. System Acquisition Based On Software Product Assessment. In *Proceedings of the International Conference on Software Engineering (ICSE)*, 1996.

[10] Michelle Cartwright and Martin Shepperd. An Empirical Investigation of an Object-Oriented Software System. *IEEE Transactions on Software Engineering*, 26:786–796, 2000.

[11] Neil A.M. Maiden, Cornelius Ncube, and Andrew Moore. Lessons Learned During Requirements Acquisition for COTS Systems. *Communications of the ACM*, 40(12):21–25, 1997.

[12] Krishnan Rangarajan, N. Swaminathan, Vishu Hedge, and Jacob Jacob. Product Quality Framework: A Vehicle for Focusing on Product Quality Goals. *Software Engineering Notes*, 26(4):77–82, 2001.

[13] Timm Euler. Modelling Data Mining Processes on a Conceptual Level. In *Proceedings of the 5th International Conference on Decision Support for Telecommunications and Information Society*, Warsaw, Poland, 2005.

[14] Dean W. Abbott, I. Philip Matkovsky, and John F. Elder IV. An Evaluation of High-end Data Mining Tools for Fraud Detection. In *IEEE International Conference on Systems, Man, and Cybernetics*, San Diego, CA, 12–14. Oct. 1998. IEEE.

[15] Michael Goebel and Le Gruenwald. A Survey of Data Mining and Knowledge Discovery Software Tools. *ACM SIGKDD Explorations*, 1(1):20–33, 1999.

[16] Ken W. Collier, Donald Sautter, Curt Marjaniemi, and Bernard Carey. A Methodology for Evaluating and Selecting Data Mining Software. In *Proceedings of the 32nd Hawaii Int. Conference on System Sciences*, 1999.

[17] Christophe G. Giraud-Carrier and Olivier Povel. Characterising Data Mining Software. *Intelligent Data Analysis*, 7(3):181–192, 2003.

[18] Katharina Morik and Martin Scholz. The MiningMart Approach to Knowledge Discovery in Databases. In Ning Zhong and Jiming Liu, editors, *Intelligent Technologies for Information Analysis*. Springer, 2004.

[19] Timm Euler. Criteria for KDD Tool Evaluation. Technical report, Fachbereich Informatik, Universität Dortmund, 2005. http://www-ai.cs.uni-dortmund.de/PERSONAL/euler.html.

No	Name	m	MM	Clem.	Prem.	IBM	SAS	NCR
1	Data formats	6	0.33	0.83	0.5	0.5	0.66	0.33
2	Data handling	3	0.33	1.0	0.66	0.66	0.66	0.33
3	Caching control	2	1.0	1.0	0.5	1.0	0	1.0
4	Caching size estimation	1	0	0	0	0	0	0
5	Automatic caching	2	0.5	0	0	0.5	0.5	0.5
6	Caching transparency	2	1.0	0.5	1.0	1.0	1.0	1.0
7	Data inspection	1	1.0	1.0	1.0	0	1.0	1.0
8	Attribute import	3	0.66	0.66	0.66	0.33	0.33	0.33
9	Conceptual data types	1	1.0	1.0	0	1.0	1.0	0
10	Type recognition	5	0.8	1.0	0.4	0.8	0.8	0.2
11	Flexibility of type mapping	3	1.0	1.0	0	0.66	1.0	0
12	Robustness of type mapping	1	1.0	1.0	1.0	1.0	0	1.0
13	Data char. recognition	6	0.66	0.66	0	0.66	0.66	0.33
14	Data char. deployment	1	0	1.0	1.0	0	1.0	0
15	Attribute roles	4	0.75	1.0	0.75	0.5	1.0	0.75
16	Attribute matching	2	1.0	1.0	1.0	0.5	0	0.5
17	Metadata inference	10	0	0.2	0.1	0.1	0.2	0
18	Checking wellformedness	4	0.25	0.5	0.75	0	0.25	0.25
19	Empty data sets recognition	1	0	0	0	0	1.0	0
20	Metadata propagation	5	1.0	1.0	1.0	0.2	0.8	0.2
21	Relations between concepts	1	1.0	0	0	0	0	0
22	Operator transparency	2	0.5	0	0	0	1.0	0.5
23	Availability prim. operators	6	1.0	1.0	1.0	0.83	0.33	0.33
24	Availability conv. operators	10	0.8	0.5	0.1	0.2	0.3	0.6
25	Attribute derivation support	2	0	1.0	1.0	0.5	0.5	0
26	Iteration attribute derivation	3	0	0.33	0	0	0	0
27	Data independence	1	1.0	1.0	0	0	0	0
28	Representation of data flow	1	1.0	1.0	1.0	0	1.0	0
29	Support for chunking	2	1.0	0.5	0	1.0	1.0	0
30	Graph structure	1	1.0	1.0	0	1.0	1.0	1.0
31	Execution transparency	7	0.29	0.14	0.43	0.29	0.43	0.14
32	Execution automation	3	0	0.33	0.66	0	0	0
33	Execution administration	7	0.57	0.29	0.43	0.29	0.57	0
34	Execution in background	1	1.0	1.0	1.0	1.0	0	1.0
35	Export transparency	1	1.0	0	1.0	0	1.0	1.0
36	Editing flexibility	1	1.0	1.0	0	1.0	0	1.0
37	Visual graph arrangement	1	0	1.0	1.0	0	0	0
38	Splitting training and test set	1	1.0	1.0	1.0	1.0	1.0	1.0
39	Model evaluation	1	0	1.0	1.0	1.0	1.0	1.0
40	Cross validation	1	0	0	0	0	0	0
41	Automatic parameter tuning	1	0	0	0	0	0	0
42	Nestability of control operators	1	0	0	0	0	0	0
43	Export of models	1	1.0	1.0	0	1.0	1.0	0
44	Post-processing	1	0	0	0	0	0	0
45	CRISP support	1	0	1.0	0	0	0	0
46	PMML support	2	0	1.0	1.0	0.5	0.5	0.5

Table 2. Finer granularity evaluation table. See [19] for details about the criteria. $m = 1$ indicates boolean (ungrouped) criteria.