Approaches for Software Metrics Threshold Derivation: A Preliminary Review

TINA BERANIČ and MARJAN HERIČKO, University of Maribor

Knowing the reliable software metrics threshold can contribute to product quality evaluation and, consequently, increase the usefulness of software metrics in practice. How to derive software metrics thresholds is a topic of many researchers, either proposing new approaches, or verifying existing methods on different practical projects. A literature review was conducted to summarise the available knowledge in this area. The aim was to find approaches for software metric threshold derivation and extract relevant information, such as programming language for which an approach was proposed and tested, what metrics were derived, and if approaches are automated with supporting tools. Data extracted from 14 primary studies reveal that the majority of studies present new approaches and approaches usually apply statistical analysis for threshold derivation. Primary programming language for which metrics are derived is Java, followed by C, C++ and C#. We also came across some tools used for threshold derivation, but in practise their use was not detected. Metrics derived, purpose of metrics derivation and other information are also summarised in the tables bellow.

Categories and Subject Descriptors: D.2.8 [**Software Engineering**] Metrics; D.2.9 [**Software Engineering**] Management-Software quality assurance (SQA)

General Terms: Software Metrics Threshold Derivation

Additional Key Words and Phrases: Software Metrics, Thresholds, Reference Values, Object-oriented, Literature Review

1. INTRODUCTION

To deliver software product in required quality, it is crucial to address and manage the quality assurance domain properly. Software metrics can be used for reflection of the qualitative characteristics of software modules in a quantitative way [Arar and Ayan 2016], presenting a control instrument in a software development and maintenance process [Alves et al. 2010]. Software metrics assess software from different views and, therefore, belong to different metrics groups, but overall, reflect the internal quality of software systems [Arar and Ayan 2016]. The usefulness of metrics without knowing their reference values is very limited, due mainly to interpretation difficulties [Ferreira et al. 2012]. To overcome the above-mentioned difficulties it is important that reliable reference values of software metrics are available.

Thresholds are heuristic values that are used to set ranges of desirable and undesirable metric values for measured software and, furthermore, used to identify anomalies which may be an actual problem [Lorenz and Kidd 1994]. Threshold values explain if a metric value is in the normal range [Ronchieri and Canaparo 2016] and, consequently, provide a scale for paying attention to threshold-exceeding components [Arar and Ayan 2016]. There are many approaches available to compute thresh-

Author's address: T. Beranič, Faculty of Electrical Engineering and Computer Science, University of Maribor, Smetanova 17, 2000 Maribor, Slovenia; email: tina.beranic@um.si; M. Heričko, Faculty of Electrical Engineering and Computer Science, University of Maribor, Smetanova 17, 2000 Maribor, Slovenia; email: marjan.hericko@um.si.

In: Z. Budimac (ed.): Proceedings of the SQAMIA 2017: 6th Workshop of Software Quality, Analysis, Monitoring, Improvement, and Applications, Belgrade, Serbia, 11-13.9.2017, Also published online by CEUR Workshop Proceedings (http://ceur-ws.org, ISSN 1613-0073)

old values. Anyway, as summarised by [Arar and Ayan 2016], threshold definition methodology should meet the following requirements: (1) It is built on representative benchmark data and not on opinions; (2) Data should be analysed statistically; (3) It is repeatable and clear [Alves et al. 2010; Sánchez-González et al. 2012].

The aim of the presented paper is to gather available knowledge regarding software metrics threshold derivation techniques. The first step in a literature review is to find already available related studies. [Ronchieri and Canaparo 2016] present a preliminary mapping study in the area of software metric thresholds in which they aim to present the identified state of current affairs. Two main research questions were formed, dealing with identification of the currently most important papers in software metrics threshold research community, and with the meaningful aggregation of those papers. During their review, they observed three main domain problems: (1) Unclear explanation of the method for selecting the technique that calculates threshold; (2) A direct application of the metric threshold values to different code context; (3) A lack of objective analysis of calculated thresholds [Ronchieri and Canaparo 2016]. Anyhow, the extracted domain specific data comprise the study topic and categorisation, number and type of analysed projects, types of derived metrics and programming language and computed threshold values. According to our knowledge, this is the only available review study dealing with the software metrics threshold values. This was also confirmed by [Ronchieri and Canaparo 2016].

The previously mentioned mapping study presented the starting point in our literature review. We addressed different research questions, so different extraction categories were defined. Since our search provided some additional articles that [Ronchieri and Canaparo 2016] did not examined, some extraction categories remain the same. Also, the search resources were expanded in our review.

The article is organised as follows. After the Introduction that presents an insight into the theoretical background and related work, Section 2 presents the performed literature review. Basic data extracted from primary studies are presented in this section. Section 3 explains and describes available approaches for software metrics threshold derivation, together with their connections and expansions. Extracted software metrics used in primary studies and details about derived threshold values are presented in Section 4. The last chapter concludes the article providing the discussion and future work opportunities.

2. A PRELIMINARY LITERATURE REVIEW OF THE RESEARCH AREA

Without knowing the threshold values of software metrics, their usefulness is limited. As stated by [Ferreira et al. 2012; Alves et al. 2010], threshold values for most metrics are still unknown. Nevertheless, numerous approaches for metrics derivation exist; there are many research opportunities that can be addressed aimed at improving domain knowledge. The performed literature review presents an insight into previous research in the area.

Based on our research purpose, four main research question were formed:

. **RQ1:** What approaches and methods are used for software metrics threshold derivation?
. **RQ2:** What is the purpose of threshold derivation?
. **RQ3:** For which software metrics thresholds are derived?
. **RQ4:** What tools are available for metrics‘ collection and threshold derivation?

Based on the research questions, the following search string was formed: *(”software metrics” OR ”source code metrics”) AND (”threshold” OR ”reference values”)*. With it, the search was conducted in selected digital libraries. Search results by data sources and final number of selected studies after applying inclusion and exclusion criteria are presented in Table I. Among the applied inclusion criteria

Table I. Digital Libraries, Search Results and Selected Studies

Electronic Base	URL	Number of Results	Selected Studies
ScienceDirect	https://www.sciencedirect.com	8	2
ACM Digital Library	https://dl.acm.org	8	4
IEEE Xplore	https://ieeexplore.ieee.org	119	10
Scopus	https://www.scopus.com	76	12
Web of Science	https://webofknowledge.com	32	5

are that the study is available in selected digital libraries and is available in English, the study is from software engineering domain and that approaches for threshold derivation are presented or if they are just validated, the used metrics should be explained. Where available and appropriate, the search was limited to abstract, title and keywords. Since some search tools are looking across different digital libraries, some search results are listed in multiple categories.

In the end, 18 primary studies were selected for detailed analysis. Among them, four articles were excluded from the data extraction process. [Veado et al. 2016; Foucault et al. 2014; Sousa et al. 2017] are presenting developed tools for threshold derivation and [Vale and Figueiredo 2015] are presenting a tailored approach for threshold derivation for use in the context of software product lines. Finally, the data was extracted from 14 primary studies.

In Table II, basic information is presented about the selected primary studies. The first one is used programming language for which software metric thresholds were derived. As can be seen, the majority of studies derive threshold values for the Java programming language, while reference values for other programming languages, specifically C, C++ and C# are presented in 4 studies. Threshold values for other programming languages were not detected. The second information matter was the purpose of metric derivation. The majority of studies derive threshold to find or predict defects and errors in programmes, while some of them compute reference values for code smells detection purposes.

Table II. Basic Information About Primary Studies

Study	Programming Language	Purpose of metrics derivation
[Alves et al. 2010]	Java, C#	Maintainability level
[Arar and Ayan 2016]	Java	Fault prediction
[Benlarbi et al. 2000]	C++	Fault prediction
[Boucher and Badri 2016]	Java	Fault prediction
[Ferreira et al. 2012]	Java	Detecting design flaws
[Fontana et al. 2015]	Java	Code smell detection
[Foucault et al. 2014]	Java	Risky classes
[Herbold et al. 2011]	Java, C#, C++, C	Evaluate quality attributes
[Hussain et al. 2016]	Java	Fault prediction
[Mihancea and Marinescu 2005]	Java, C++	Detecting design flaws
[Oliveira et al. 2014b]	Java	Internal quality
[Shatnawi 2010]	Java	Error risk level
[Shatnawi et al. 2010]	Java	Error risk level
[Yamashita et al. 2016]	Java	Defect risk

The process of metric computation or threshold derivation is sometimes automated with tool support. As already mentioned, [Veado et al. 2016] present a TDTool for threshold derivation and [Oliveira et al. 2014a] present an RTTool for the same purpose. While TDTool supports any kind of metrics, RTTool compute threshold only for class level metrics that have heavy-tailed distribution. [Sousa et al. 2017] propose a FindSmells tool for smells detection in software systems based on software metrics and their thresholds. While linked to threshold computation, tools for metrics derivation was listed in some articles. The tools that were used in the selected primary studies are Conecta [Ferreira et al. 2012], ckjm tool [Arar and Ayan 2016], Borland Together [Shatnawi 2010], Moose platform and VerveineJ [Oliveira et al. 2014b] and SIG Software Analysis Toolkit [Alves et al. 2010].

3. APPROACHES FOR THRESHOLD DERIVATION OF SOFTWARE METRICS

3.1 Novel Approaches

Different derivation approaches are proposed among the articles about metrics thresholds. According to [Fontana et al. 2015], approaches can be based on observation, statistical analysis, metrics analysis and machine learning. Detected approaches from primary studies are summarised and presented below in chronological order of publications.

[Erni and Lewerentz 1996] present a statistical method based on [DeMarco 1986] statement talking about judging metric values based on comparison to similar projects. They compute the average (μ) and standard deviation (σ) of metric values of one software system, on the assumption that the metrics are distributed normally. The lower ($T_{min} = \mu - \sigma$) and the higher ($T_{max} = \mu + \sigma$) thresholds can be produced based on calculated values. Values that are outside the interval $\mu \pm \sigma$ are considered as outliers.

A method called Tuning Machine is presented by [Mihancea and Marinescu 2005]. The method is proposing a solution dealing with the problem of threshold values that should be used by a detection strategy. The new method for establishing proper threshold values is proposed with the help of a genetic algorithm.

Identifying the threshold values using Receiver Operating Characteristic (ROC) curves is presented by [Shatnawi et al. 2010]. They claim that meaningful and useful threshold values must be associated with design factors of interest. Therefore, their identification of metric thresholds is associated with class error probability, since their goal is to identify faults. ROC, a diagnostic accuracy test [Zweig and Campbell 1993], was used previously to make decisions in the diagnostic area in Radiology [Hanley and McNeil 1982] and clinical medicine [Zweig and Campbell 1993]. Since medicine represents a field in which metrics and their thresholds represent crucial information [Ferreira et al. 2012], many approaches now used in software engineering have roots in the medical domain. When calculating reference values of object-oriented metrics, several ways can be followed. However, [Shatnawi et al. 2010] are making the first attempt to use it for identifying object-oriented metric threshold values. They analyse three versions of Eclipse software, also using the error data from Bugzilla. The errors were then divided into three categories (Low, Medium, High) and statistical analyses were conducted to identify metric values that could classify Eclipse classes into different error categories (binary and ordinal).

Existing approaches that derive metrics threshold values systematically do not take into account the different statistical properties of software metrics [Alves et al. 2010]. For that reason, a method that derives metrics thresholds values based on empirical measurement is presented by [Alves et al. 2010]. It considers different data distribution and scales of source code metrics, and it is also resilient against outliers. For the presented method, which contains six steps, the benchmark of 100 object-oriented Java and C# systems were used for defining metrics reference values.

A statistical approach for threshold derivation with the help of a logistic regression model is presented by [Shatnawi 2010]. Logistic regression validates metrics and constructs a threshold model. The connection was checked between single metric and fault probability. For significant metrics thresholds, values were investigated with the use of the Bender method [Bender 1999], that has its source in the medical domain. Bender [Bender 1999] proposed two methods for finding possible thresholds: Value of an Acceptable Risk Level (VARL) and lower Value of an Acceptable Risk Gradient (VARG).

Machine learning techniques were used in the approach by [Herbold et al. 2011]. They presented a programming language independent data driven method for threshold value calculation. Methodology can also be used for efficiency optimisation of existing metric sets and for the simplification of complex classification models.

[Ferreira et al. 2012] aimed at defining threshold values for object-oriented software metrics. Thresholds were derived by analysing the statistical properties of the data obtained by a large collection of different size and domain projects. After data were gathered, the values used most commonly in practice were identified. 40 projects were analysed and reference values for six metrics were calculated, taking into account the relevant data distribution.

Since it was shown that thresholds are connected to the context of a project, the approach by [Foucault et al. 2014] aims to overcome the mentioned drawback by presenting a method that can compute the threshold for any given context. They select software projects and entities randomly and follow the [Chidamber et al. 1998] principle that considers that a metric threshold can be derived from a quantile of the distribution of metric values. The input is quantile and estimation of corresponding thresholds is computed using a process that relies on the bootstrap statistical approach.

The Bender approach [Bender 1999] was used as a basis by [Hussain et al. 2016]. They proposed a model for metrics threshold derivation with use of non-linear functions, described with logistic regression coefficient.

3.2 Extensions and validations of Existing Approaches and Replicated Studies

An empirical method for extracting relative threshold for software metrics of real systems based on benchmark data is presented by [Oliveira et al. 2014b]. The absolute metrics threshold can be complemented and the definition of relative threshold introduced: *p% of the entities should have $M \leq k$.* M is the source code metric calculated and p is the percentage of entities that upper limit k should be applied to. Values of p and k are calculated.

[Fontana et al. 2015] also present a benchmark based data driven approach for deriving metrics reference values. Similar to [Oliveira et al. 2014b], the presented approach is inspired by [Alves et al. 2010]. The main goal is to design a method that can be applied on metrics that do not represent ratio values, typically ranging in the interval [0,1]. The method takes into account the statistical properties of obtained data of 74 open source projects.

Logistic regression was used by [Benlarbi et al. 2000], investigating the connection between faults and software metrics. Two models were constructed, one with threshold values and the other without. The models were then compared looking for connection.

[Arar and Ayan 2016] replicated the study by [Shatnawi 2010] and used a broader range of datasets and metrics, trying to resolve the problem regarding project dependent threshold values, so that other projects could benefit from the derived thresholds. Another replication study was published by [Yamashita et al. 2016] using the method by [Alves et al. 2010] is used.

The techniques based on the benchmark based method [Alves et al. 2010] and ROC curves [Shatnawi et al. 2010] were compared and investigated by [Boucher and Badri 2016]. Both methods were assessed as fault predictors and compared with the help of machine learning algorithms, which are often used as a fault-proneness prediction.

4. SOFTWARE METRIC THRESHOLDS

A number of different metrics were used in the selected primary studies and different threshold values were derived. Which software metrics in particular are presented Table III. Metrics are presented used either in the derivation or validation process. Most commonly, primary studies use the CK (Chidamber & Kemerer) object-oriented metrics [Chidamber and Kemerer 1994], consisting of six metrics: WMC

(Weighted Methods Per Class), DIT (Depth of Inheritance Tree), NOC (Number of Children), CBO (Coupling between Object Classes), RFC (Response For a Class), LCOM (Lack of Cohesion of Methods). Together with them, some other metrics are also used. Table III also gives information about the form in which the derived threshold values are presented. Usually, reference values are presented as a single numerical value, sometimes in the form of intervals. Rarely, primary studies present just approaches, and do not provide any concrete threshold value.

Table III. Software Metrics Addressed in Primary Studies and Corresponding Thresholds

Study	Metrics	Presented Thresholds
[Arar and Ayan 2016]	WMC, CBO, RFC, LCOM, Ca, Ce, NPM, AMC, MAX_CC, AVG_CC, LOC	Yes (numeric)
[Alves et al. 2010]	SIG quality model metrics (e.g. LOC, MCCabe, fan-in, number of methods number of parameters)	No (risk intervals)
[Benlarbi et al. 2000]	CK metrics (without LCOM)	Yes (numeric)
[Boucher and Badri 2016]	CK metrics and SLOC	Yes (numeric)
[Ferreira et al. 2012]	LCOM, DIT, coupling factor, afferent couplings, number of public methods, number of public fields	Yes (interval good, regular, bad)
[Fontana et al. 2015]	ATFD, WMC, NOAP + NOAM, WMC LOC, CYCLO, MAXNESTING, NOAV CINT, CM, CC and some others	Yes (interval low, medium, high)
[Foucault et al. 2014]	NOA+NOM, CBO	No
[Herbold et al. 2011]	VG, NBD, NFC, NST, WMC, CBO, RFC, NORM, NOM, LOC, NSM	Yes (numeric)
[Hussain et al. 2016]	DIT, CA, LCOM, NPM	Yes (numeric)
[Mihancea and Marinescu 2005]	ATFD, WMC, TCC, WOC, NOPA, NOAM	No
[Oliveira et al. 2014b]	NOM, LOC, FAN-OUT, RFC, WMC, LCOM, PUBA/NOA	Yes (numeric)
[Shatnawi 2010]	CK metrics	Yes (numeric)
[Shatnawi et al. 2010]	CK metrics	Yes (numeric)
[Yamashita et al. 2016]	LOC, number of methods, fan-in, McCabe	Yes (interval)

Another point of interest was also the area of software metrics‘ correlation and combination. The correlation of metrics, meaning combining semantically related metrics, were not detected. On the other hand, the metric combination was presented only for purpose of composing rules for code smell detection [Mihancea and Marinescu 2005; Fontana et al. 2015].

5. CONCLUSION

As presented in literature, thresholds are crucial for software metrics‘ interpretation and, consequently, for increasing their use in practice [Ferreira et al. 2012]. Many derivation approaches can be found in the literature, each having its own characteristics. Deriving software metric thresholds is an important research topic ,which was confirmed with the performed literature review and selected primary studies. Findings of the review are summarised in Table I, II and III, presenting answers to the proposed research questions.

In the context of RQ1, we searched for available software metrics derivation approaches, which are presented in chapter 3. First, approaches that were proposed for the first time are presented, like [Alves et al. 2010; Erni and Lewerentz 1996; Ferreira et al. 2012; Shatnawi 2010], after them, if available, the validation or extension of those approaches can be seen. The majority of primary studies derive thresholds for error finding purposes, some of them also for code smell detection, regarding the answer on RQ2. With RQ3, we were looking for software metrics for which thresholds were derived. As can be seen in Table III, authors often use CK object-oriented metrics, adding other software metrics. The last research question, RQ4, was aimed at identifying tools, either for metric calculation or threshold derivation. Among primary studies, tools are proposed like TDTool [Veado et al. 2016], RTTool [Oliveira et al. 2014a] and FindSmells tool [Sousa et al. 2017] are proposed.

According to our findings, possible research directions can be highlighted. The presented approaches for software metric derivation that are, nowadays, usually used to find and predict errors, defects or deficiencies in software modules, can be used in the context of threshold derivation for code smell detection. Also, derived thresholds aimed at code smell detection can be compared to ones derived for fault prediction. Since different programme languages are not represented widely among primary studies, thresholds can be derived also for other languages and compared to already derived thresholds for Java. The use and comparisons of presented threshold derivation tools can be done, assessing and comparing their derived threshold values. Finally, tools can also be used in the context of code smell detection.

Acknowledgements
The authors acknowledge the financial support from the Slovenian Research Agency under The Young Researchers Programme (SICRIS/SRA code 35512, RO 0796, Programme P2-0057).

REFERENCES

T. L. Alves, C. Ypma, and J. Visser. 2010. Deriving metric thresholds from benchmark data. In *2010 IEEE International Conference on Software Maintenance*. 1–10.

Ömer Faruk Arar and Kürşat Ayan. 2016. Deriving thresholds of software metrics to predict faults on open source software: Replicated case studies. *Expert Systems with Applications* 61 (2016), 106 – 121.

Ralf Bender. 1999. Quantitative Risk Assessment in Epidemiological Studies Investigating Threshold Effects. *Biometrical Journal* 41, 3 (1999), 305–319.

S. Benlarbi, K. El Emam, N. Goel, and S. Rai. 2000. Thresholds for object-oriented measures. In *Proceedings 11th International Symposium on Software Reliability Engineering. ISSRE 2000*. 24–38.

A. Boucher and M. Badri. 2016. Using Software Metrics Thresholds to Predict Fault-Prone Classes in Object-Oriented Software. In *2016 4th Intl Conf on Applied Computing and Information Technology/3rd Intl Conf on Computational Science/Intelligence and Applied Informatics/1st Intl Conf on Big Data, Cloud Computing, Data Science Engineering (ACIT-CSII-BCD)*. 169–176.

S. R. Chidamber, D. P. Darcy, and C. F. Kemerer. 1998. Managerial use of metrics for object-oriented software: an exploratory analysis. *IEEE Transactions on Software Engineering* 24, 8 (Aug 1998), 629–639.

S. R. Chidamber and C. F. Kemerer. 1994. A metrics suite for object oriented design. *IEEE Transactions on Software Engineering* 20, 6 (Jun 1994), 476–493.

T. DeMarco. 1986. *Controlling Software Projects: Management, Measurement, and Estimates*. Prentice Hall PTR, Upper Saddle River, NJ, USA.

K. Erni and C. Lewerentz. 1996. Applying design-metrics to object-oriented frameworks. In *Proceedings of the 3rd International Software Metrics Symposium*. 64–74.

Kecia A.M. Ferreira, Mariza A.S. Bigonha, Roberto S. Bigonha, Luiz F.O. Mendes, and Heitor C. Almeida. 2012. Identifying thresholds for object-oriented software metrics. *Journal of Systems and Software* 85, 2 (2012), 244 – 257. Special issue with selected papers from the 23rd Brazilian Symposium on Software Engineering.

F. Arcelli Fontana, V. Ferme, M. Zanoni, and A. Yamashita. 2015. Automatic Metric Thresholds Derivation for Code Smell Detection. In *2015 IEEE/ACM 6th International Workshop on Emerging Trends in Software Metrics*.

Matthieu Foucault, Marc Palyart, Jean-Rémy Falleri, and Xavier Blanc. 2014. Computing Contextual Metric Thresholds. In *Proceedings of the 29th Annual ACM Symposium on Applied Computing (SAC '14)*. ACM, New York, NY, USA, 1120–1125.

J A Hanley and B J McNeil. 1982. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology* 143, 1 (1982), 29–36.

Steffen Herbold, Jens Grabowski, and Stephan Waack. 2011. Calculation and optimization of thresholds for sets of software metrics. *Empirical Software Engineering* 16, 6 (01 Dec 2011), 812–841.

S. Hussain, J. Keung, A. A. Khan, and K. E. Bennin. 2016. Detection of Fault-Prone Classes Using Logistic Regression Based Object-Oriented Metrics Thresholds. In *2016 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*. 93–100.

Mark Lorenz and Jeff Kidd. 1994. *Object-oriented Software Metrics: A Practical Guide*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

P. F. Mihancea and R. Marinescu. 2005. Towards the Optimization of Automatic Detection of Design Flaws in Object-Oriented Software Systems. In *Ninth European Conference on Software Maintenance and Reengineering*. 92–101.

Paloma Oliveira, Fernando P. Lima, Marco Tulio Valente, and Alexander Serebrenik. 2014a. RTTool: A Tool for Extracting Relative Thresholds for Source Code Metrics. In *Proceedings of the 2014 IEEE International Conference on Software Maintenance and Evolution (ICSME '14)*. IEEE Computer Society, Washington, DC, USA, 629–632.

P. Oliveira, M. T. Valente, and F. P. Lima. 2014b. Extracting relative thresholds for source code metrics. In *2014 Software Evolution Week - IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE)*. 254–263.

E. Ronchieri and M. Canaparo. 2016. A preliminary mapping study of software metrics thresholds. In *ICSOFT 2016 - Proceedings of the 11th International Joint Conference on Software Technologies*, Maciaszek L., Cardoso J., Cabello E., van Sinderen M., Maciaszek L., Ludwig A., and Cardoso J. (Eds.), Vol. 1. SciTePress, 232–240.

R. Shatnawi. 2010. A Quantitative Investigation of the Acceptable Risk Levels of Object-Oriented Metrics in Open-Source Systems. *IEEE Transactions on Software Engineering* 36, 2 (March 2010), 216–225.

Raed Shatnawi, Wei Li, James Swain, and Tim Newman. 2010. Finding Software Metrics Threshold Values Using ROC Curves. *Journal of Software Maintenance and Evolution: Research and Practice* 22, 1 (jan 2010), 1–16.

Bruno L. Sousa, Priscila P. Souza, Eduardo Fernandes, Kecia A. M. Ferreira, and Mariza A. S. Bigonha. 2017. FindSmells: Flexible Composition of Bad Smell Detection Strategies. In *Proceedings of the 25th International Conference on Program Comprehension (ICPC '17)*. IEEE Press, Piscataway, NJ, USA, 360–363.

L. Sánchez-González, F. García, F. Ruiz, and J. Mendling. 2012. A study of the effectiveness of two threshold definition techniques. In *16th International Conference on Evaluation Assessment in Software Engineering (EASE 2012)*. 197–205.

G. A. D. Vale and E. M. L. Figueiredo. 2015. A Method to Derive Metric Thresholds for Software Product Lines. In *2015 29th Brazilian Symposium on Software Engineering*. 110–119.

Lucas Veado, Gustavo Vale, Eduardo Fernandes, and Eduardo Figueiredo. 2016. TDTool: Threshold Derivation Tool. In *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering (EASE '16)*. ACM, New York, NY, USA, 24:1–24:5.

K. Yamashita, C. Huang, M. Nagappan, Y. Kamei, A. Mockus, A. E. Hassan, and N. Ubayashi. 2016. Thresholds for Size and Complexity Metrics: A Case Study from the Perspective of Defect Density. In *2016 IEEE International Conference on Software Quality, Reliability and Security (QRS)*. 191–201.

M H Zweig and G Campbell. 1993. Receiver-operating characteristic (ROC) plots: a fundamental evaluation tool in clinical medicine. *Clinical Chemistry* 39, 4 (1993), 561–577.