Multi-layered, objective, optimized evaluation of strategies over static code analysis tools

László Pitlik, Pitlik Marcell, MY-X team

Abstract: It is well-known: evaluations could ever be derived based on arbitrary, intuitive, instinctive, subjective steps (c.f. evaluation of performances of Students in schools). Each evaluation is subjective where the anti-discriminative expectation is not proved before: Is it possible to declare that each compared object might have the same evaluation value? On the other hand: general-problem-solving-oriented artificial intelligence solutions make possible to manage multi-layered evaluation criteria/dimensions in an objective, optimized way. These context-free solutions can also be used in case of the evaluation of strategies over static code analysis tools. This paper demonstrates the steps of the objective, optimized strategy and compares the evaluation results of the subjective and objective methods.

Keywords: (online) similarity analysis

# Introduction

The starting points of this paper are listed under references especially called as focused materials. The direct impulse (to write about recommendations about evaluation of static code analysis tools) came from the SISY-paper. The other listed paper can be seen as a kind of background information set. The authors of the original SISY-paper presents an object-attribute-matrix (OAM: see. Table V. – in the original paper) where the objects are different tools & strategies like (tools: SonarCube, FindBugs, PMD, Checkstyle / strategies: S1.1, S1.2, S2.1, S2.2 – “1. Running a static analysis tool over a completed software project – S1. / 1.1. Starting the static analysis tool after correcting all defects – S1.1. / 1.2. Running the static analysis tool after correcting any defect – S1.2. / 2. Running a static analysis tool during software project development – S2. / 2.1. Starting static analysis tools by external tool calling during software project development – S2.1. / 2.2. Running static analysis tools by including tools in the development environment in which the software project is developed – S2.2.” In the paper there are references with details “how the proposed strategies were identified and elaborated on the defined steps for the implementation of these strategies in the static code analysis review cycle.”

The original SISY-paper presents analyses basically in a single-layered way – it means only one attribute is analysed at the same time: e.g.

* “Fig. 2. (of the original paper) Average time required (ATR) to eliminate one defect overall tools according to the identified strategies” and
* “Fig. 3. (of the original paper) Percentage of false positive and false negative (FNR, FPR) results in the total number of defects overall tools according to the identified strategies”.

Based on these two figures, it seems to be trivial, that a multi-layered evaluation should be involved and Fig. 4. (of the original paper) “Comparison of strategy implementation speed results and percentage of false positive and false negative results in the total number of identified defects” presents a kind of 2 dimensional view and it comes to the conclusion: “Based on the presented results, it can be concluded that strategies that are faster to implement over the tools used to generate a larger number of defects identified as FPR and FNR and strategies that are slower to implement generate a smaller number of erroneous results.

The aim of the original paper is: “…not to compare tools, but to compare strategies, …”.

The SISY-paper says finally: “The answer to research question RQ1 (= Is there a strategy for implementing the static code analysis process that gives the best results?) is that if both criteria are considered together, it is not possible to label one of the strategies as the strategy that gives the best results. It is concluded that the choice of strategy depends on the purpose of using the static code analysis tool.”

# Problem identification

Assumed, that the Table V (of the original paper) is given for a decision-making-process, and we follow the principle of KNUTH (= science/knowledge is what can be transformed into source code, each other human activity is a kind of artistic performance) then we need a robot being capable of deriving the tool & strategy “that gives the best results” (see RQ1).

In order the create this decision-support-robot, we have to prove whether the objects (tools\*strategies) in the Table V (of the original paper) might be evaluate with the same evaluation value (Y0) based on the available result-components (Xi)?

# The anti-discriminative, optimized approach for multi-layered evaluation

This challenge to manage needs an anti-discriminative approach where the virtual (seemingly not-available) dependent variable (Y0) is a constant value in case of each object. The mathematical challenge is: which kind of (as far as possible: Solver-based) function (or knowledge representation form) is capable of approaching the constant value of Y0 in case of each object in un unlimited way of the approximation fitting? The solution should be Solver-based – that means we have an objective function (a kind of aggregated difference of the constant value and the estimations per object) and this objective function should approximate the target value of 0 (difference). If the difference between the constant Y-value and the estimations is zero in case of each object, then we can speak about a kind of equilibrium = each object has the same evaluation value.

In such a case, the robot can not derive a single winner (tools and/or strategies). On the other hand, the objective winner is always the object with the most estimation value (if the equilibrium is not given and each estimation can be seen as valid 🡨 see later).

# The process of an anti-discriminative analysis

This chapter demonstrates the steps of the anti-discriminative analyses based on the Table V. (of the original paper) in a form making possible of the needed reproducibility (details see: <https://miau.my-x.hu/miau/277/sisy2021_paper7.xlsx>).

## Alternative problems

The above presented multi-layered evaluation steps needs a clarified vision concerning the direction of the attributes. The direction of the attributes can reflect customized objectives:

* It is possible to create analytical tools, where the suspicions are always massive but the amount of them is therefore low,
* and vice versa…

On the other hand: it is possible the build a benchmark object where no checks are available – therefore FPR=0% and FNR=100%.

These alternative interpretations lead to scenarios:

## First step – building scenarios

FIG Nr1-2 (of this paper) present the data asset (management) from the raw data to the scenarios:



FIG Nr1. (of this paper): Involve OAM from the original paper (on the left side) – source: own presentation (see: OAM-sheet of the referenced own XLSX-file)

 Legend of FIG Nr. 1. (of this paper)



FIG Nr2. (of this paper): Scenarios (source:own presentation (see: OAM-sheet of the referenced own XLSX-file)

Scenario-parameter (2\*ˇ2:

* direction of attribute of ND with 2 options: (0) the more the more, (1) the less the more
* existence of a benchmark-object with 2 options: yes or no

## Ranking

The FIG Nr.2 (of this paper) demonstrates the conversion of the raw data with well-known units (like piece, secundum, percent) into ranking values based on the given directions (0 or 1) of the scenarios. The ranking is a relevant transformation for the similarity analyses: the ranking makes possible

* to exclude the mostly different units of the raw date
* to define staircase function for ranking values (X) and components of Y0 where the excluded units (a kind of standardization makes possible to aggregate – in this case to add – the estimated components of Y0 concerning each Xi

## Involving online Solver-based engines

In automated cases, a C-URL-call would also be possible, but here and now, the models got derived in a manual way. (The detailed description of the similarity analysis is not part of this paper.)

Calculation details can be seen in the referenced XLSX-file: sheets “1”, “2” “3”, “4”

Each scenario has 2 sub-models:

* a direct model where the directions are the same ones as in the sheet of OAM
* an inverse model where each basic direction of a scenario is inverted (0🡨🡪1)

The 2 sub-models make possible to derive for each object a kind of validity: if an inverse parameter set can not lead to an inverse result for each object, then objects with non-symmetric sub-results are invalid. It is important to declare now: each object could realize a valid estimation value in case of each scenario.

Estimations as a kind of simple database can be found in the sheet “results”.

# Similarity analysis – as such

Similarity analysis is a Solver-based approach. It can be used in offline version based on Excel-Solver in limited volume because of the standard Excel-licence - and it can be used in online form: <https://miau.my-x.hu/myx-free/> (provided by the My-X Team). The similarity analysis can also be called as COCO: component-based object comparison for objectivity. The COCO-family has 3 basic (not chained) versions: COCO-Y0 (for anti-discriminative modelling), COCO-STD (for production functions), and COCO-MCM (for explorative modelling). Based on these basic elements a lot of chained model-logics can be developed (e.g., STEP-IX, COCO-SWOT, etc.).

The input is always an OAM with integer values: in case of Xi, there are ranking values, in case of Y constant values or real values.

The outputs are estimation values for each object, genetic potential (the sum of the first steps), sum of Y(real) and sum of Y(estimated), and the values of the staircase function for each attribute (see references about similarity analysis).

# Hermeneutics / Model interpretations

It is important here and now to repeat the aim of this paper: We have to create a robot for decision support who is capable of deriving the best object/tool/strategy – so far, the parameters of the scenarios are given:



FIG Nr.3 (of this paper): Evaluation of the tools (source: sheet “REPORTS” of the XLSX-file with details)

Interpretation of the FIG Nr3:

* Checkstyle seems to be the best of, if ND=max (independent from the existence of the defined benchmark.
* It is to highlight that the benchmark can be seen as a better one than Findbugs or PMD (see scenario\_3).
* If ND=min, then PMD has a disadvantageous position compared to SonarCube where the benchmark and the further tools has a kind of normative evaluation value of 1000 (scores, points).
* Has the decision maker still no parameters for scenarios, then Checkstyle should be preferred by the robot expert.

Interpretation of the FIG Nr4:

* S1.1 seems to be the best of (independent from the existence of the defined benchmark and/or ND-preferences).
* It is to highlight that the benchmark can be seen as a better one than S2.\* (see scenario\_3).
* If ND=min, then S1.2 has a disadvantageous position.
* Has the decision maker still no parameters for scenarios, then S1.1 should be preferred by the robot expert.



FIG Nr.4 (of this paper): Evaluation of the strategies (source: sheet “REPORTS” of the XLSX-file with details)

Interpretation of the FIG Nr.5:

* Checkstyle-S11 is the best object – if the decision maker has still no parameters for scenarios.
* Scenario1: Checkstyle-S11 is the best object.
* Scenario2: SonarCube-S11 is the best object (Checkstyle-S11 has the silver medallion, and no further object can be identified with valid positive evaluation value (it means over the norm value of 1000).
* Scenario3: Checkstyle-S11 is the best object.
* Scenario4: SonarCube-S11 is the best object (see scenario2).
* The benchmark is close to the norm value of 1000 – it means there are objects with lower evaluation value



FIG Nr.5 (of this paper): Evaluation of the objects (source: sheet “REPORTS” of the XLSX-file with details)

# Conclusions

The role of ND is relevant. The ND-parameter is responsible for winning of tools (Case ND(max): winner is CheckStyle / Case ND(min): winner is SonarCube).

The evaluation values of strategies are independent of the ND-parameter (S11 is the winner in all cases).

If we have a benchmark, then real tools and strategies can have less value than the benchmark-object.

The existence of the benchmark (of an artificial/virtual object) has no relevant impact: it is norm-like, and not the looser (as might be expected). Therefore: it is very important to model the aggregated Goodness of static code analysis tools before testing real solutions. The term-creation challenge concerning a Goodness-Term can not be substituted through naïve, subjective, arbitrary, partial evaluations. These subjective ones do not lead to a KNUTH-robot-expert.

Robot experts for decision support can be derived in an automated way based on similarity analyses for quasi arbitrary scenarios if an appropriate OAM is given.

# References

Focused/discussed materials:

* D. Nikolić, D. Stefanović, D. Dakić, S. Sladojević and S. Ristić, "Analysis of the Tools for Static Code Analysis," in Proceedings of 20th International Symposium Infoteh-Jahorina, Jahorina, 2021
* Danilo Nikolić, Teodora Lolić, Sara Havzi, Darko Stefanović, Dušanka Dakić, “Evaluation Of Strategies Over Static Code Analysis Tools”, SISY 2021

Materials about the recommended approach:

* MY-X team: Interpretations about the KNUTH-principle: [https://miau.my-x.hu/miau2009/index\_tki.php3?\_filterText0=\*knuth](https://miau.my-x.hu/miau2009/index_tki.php3?_filterText0=*knuth)
* József Varga, Gyöngyi Bánkuti, Rita Kovács-Szamosi, “Analysis of the Turkish Islamic banking sector using CAMEL and Similarity Analysis methods”, Acta Oeconomica 70/2, 2020, <https://akjournals.com/view/journals/032/70/2/article-p275.xml>
* Gyöngyi Bánkuti, Similarity analysis, ERASMUS staff mobility presentation, 2018, <http://miau.my-x.hu/miau/237/eszek/COCO_Eszek_2018_04_24.ppt>
* László Pitlik, Idea Breeding Farm, 2014, <https://miau.my-x.hu/miau/196/My-X%20Team_A5%20fuzet_EN_jav.pdf>