How to Tell Apart the Good from the Bad: Setting Thresholds in Software Engineering

Luigi Lavazza(1)
Dipartimento di Scienze Teoriche e Applicate
luigi.lavazza@uninsubria.it

(1) This material was developed jointly with Prof. Sandro Morasca

Contents

Topics
The context
The problem
Proposal 1: slope-based thresholds
Proposal 2: optimistic-pessimistic approach
Proposal 3: fault-proneness H-index
Final considerations
The context

- Like in all engineering disciplines, Software Engineering practitioners need to manage the quality of software products and processes
  - monitor
  - control
  - evaluate
  - improve
  - . . .
- In this presentation, we focus on the faultiness of software modules as the quality of interest
A bit of terminology

- **Faultiness**
  - Presence of at least one fault in a module.

- **Software module**: a “piece” of software
  - A subsystem, a class, a procedure, etc.

---

The context

- Quantitative information helps managing quality

- **Measures**
  - **Internal**, depending only on the software itself
    - Code measures: size, complexity, coupling, etc.
  - **External**, depending also on elements of the external world
    - Faultiness (depending on specifications)
    - Maintainability (depending on the required changes)
    - …
Internal measures are not interesting by themselves

- The manager gets code measures, but he does not know how to interpret them.
- Note: even a smart manager who knows the meaning of RFC does not know what values of RFC are “good” and what values are “bad”.
  - This is typical of internal measures.

External measures are interesting

- The manager wants that only good quality code is released.
- Faultiness is what practitioners are really interested in for decision making along the software lifecycle
  - allocating V & V resources
  - controlling the production process
  - assessing the quality of the software under construction

The context

- Unfortunately, faultiness cannot be measured based on the code only.
  - E.g., given a module, how can you “measure” if it is faulty or not?

- We need to estimate faultiness
  - We can use our knowledge about the module, i.e., the values of its internal measures

- But how?

Hypothesis one: estimates are based uniquely on internal measures

- The test set
  - The data to be estimated
  - Every point in the plot is a module

[Graph showing faultiness with points labeled as 'unknown' and 'Known (Measurable)']
We need a threshold for estimating

- Under the current hypothesis, estimates are based uniquely on internal measures
  - E.g., RFC, response for a class
- We need a threshold $T$ such that
  - Modules whose RFC measure is greater than $T$ are classified faulty
  - Modules whose RFC measure is not greater than $T$ are classified not faulty

Problem: how do we define threshold $T$?
- let’s consider a few possibilities …

Midpoint Threshold
Midpoint Threshold

Upper Fourth Threshold: too Optimistic?
Lower Fourth Threshold: too Pessimistic?

Mean Threshold
Median Threshold

Mean + Standard Deviation Threshold
[Erni and Lewerentz]
(Mean + Standard Deviation Threshold) \times 1.5 \text{ [Lanza and Marinescu]}

Hypothesis one: estimates are based uniquely on internal measures

What's Faultiness got to do with it?
Hypothesis one: estimates are based uniquely on internal measures

- Do we get good results (i.e., accurate estimates) with this strategy?
  - Not really.
    - We shall see some experimental results at the end of the presentation.

Bad results could be expected. If you try to estimate fault-proneness based on a measure that is known to be related to fault-proneness, but without taking into consideration how it is correlated to fault-proneness, your guess could easily be wrong!

Hypothesis two: Use Internal Measures and Faultiness Data

- A faultiness estimation model can be built on top of
  - a fault-proneness estimation model
  - a fault-proneness threshold

A common practice in many fields. E.g., widely used in mechanical maintenance, or in medicine.
Quality models available!

- Models relating internal measures (CBO, WMC, RFC, etc.) to external quality (e.g., fault-proneness) are (often) available.

- These models “transform” internal measures with no practical meaning into meaningful indications.

Models of fault-proneness

- Independent variable(s):
  - One or more internal measures
    - E.g., RFC, CBO, ...
  - Dependent variable:
    - The quality of interest
      - In our case, fault-proneness

- Why fault-proneness instead of faultiness?
  - A model estimates the probability of faultiness, i.e., fault-proneness
Binary Logistic Regression (BLR) models of fault-proneness

\[ fp(x) = \frac{e^{\text{logit}(x)}}{1 + e^{\text{logit}(x)}} \]

- \( \text{logit}(X) \) is a linear function
  - univariate case: \( c_0 + c_1X \)
    - \( X \) is the internal measure
  - multivariate case: \( c_0 + c_1X_1 + c_2X_2 + \ldots \)
    - \( X_1, X_2 \ldots \) are internal measures

BLR model

[Graph showing the logistic function with a curve that approaches 1 as the input increases.]
In this presentation we consider only models with increasing fp. Decreasing models are possible. Other models can be used, like, the Probit Binary Regression (PBR)

\[ fp(z(X)) = \Phi(z(X)) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z(X)} e^{-t^2/2} dt \]

The resulting model is S-shaped, much like the BLR model.

In this presentation we shall use only BLR models.
- What we shall see here can be usually extended easily to other types of models.
How to build models

- Assuming that we have proper data
  - E.g., a spreadsheet with
    - A row for each module
    - A column for each internal measure
    - A column for faultiness
  - We need a statistical tool to compute the model.
- I suggest R
  - Open-source and free
  - Supported by a huge community
  - There are books and documentation available
  - Provides a wealth of statistical tools
    - To make sure that the models found are statistically significant
    - To test their “goodness”
      - Hosmer test, likelihood ratio test, ...

How to use a model?

- How do we use the knowledge that an internal measure is related to the probability of faultiness?
Threshold on a Fault-proneness Curve

We need a threshold $p_t$, which indicates the maximum acceptable value for fault-proneness $p_t = f_p(T)$.

When CBO of a module $M$ grows greater than $T$ the manager should start some activity to improve $M$, because its probability of being faulty is beyond the maximum acceptable risk.

Threshold on Fault-proneness

We need a threshold $p_t$, which indicates the maximum acceptable value for fault-proneness.
Threshold on Independent Variable

\[ pt = fp(T) \]

Positives, i.e., faulty

Negatives, i.e., not faulty
Estimated

Estimated positives

Estimated negatives

False negatives

Estimated negatives, But are actually positive
False positives

Estimated positives, but are actually negative

True negatives

Negative modules estimated negatives
How good is a model?

- We need to evaluate how good is a model, i.e., how accurate are its estimates.
- Informally, we want
  - Many true positives and true negatives
  - As few as possible false negatives and false positives.
Estimated/Actual Faultiness Contingency Tables

We need to check how close estimated faultiness is to actual faultiness.

<table>
<thead>
<tr>
<th></th>
<th>Non-faulty</th>
<th>Faulty</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimated</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Non-faulty</td>
<td>TN</td>
<td>FN</td>
<td>EN</td>
</tr>
<tr>
<td>Faulty</td>
<td>FP</td>
<td>TP</td>
<td>EP</td>
</tr>
<tr>
<td>Total</td>
<td>AN</td>
<td>AP</td>
<td>n</td>
</tr>
</tbody>
</table>

Accuracy indicators

- **Precision**: proportion of estimated positives that are actually positive
  \[ Precision = \frac{TP}{EP} \]

- **Recall**: proportion of actual positives that are estimated positives
  \[ Recall = \frac{TP}{AP} \]

- **F-measure**: harmonic mean of Precision and Recall
  \[ F_{\text{measure}} = \frac{2}{\frac{1}{Recall} + \frac{1}{Precision}} \]
Contents

Topics

- The context
- The problem
  - Proposal 1: slope-based thresholds
  - Proposal 2: optimistic-pessimistic approach
  - Proposal 3: fault-proneness H-index
- Final considerations

The real problem

- How should we choose pt (hence, T)?
Some possible thresholds

- \( fp = 0.5 \) (Fifty)
  - a theoretical threshold, used for no prior knowledge, same value no matter the application or discipline

- \( fp = \frac{AP}{n} \) (All)
  - This is the proportion of faulty modules in the entire data set
    - Useful to evaluate the accuracy of the proposed thresholds
      - It is the value you get with a constant logit

Why not using Fifty or All thresholds?

- Fifty does not use any knowledge about the actual modules.
  - If \( AP/n \) is 0.1 and you use 0.5 thresholds, you are going to have a lot of false positives
  - If \( AP/n \) is 0.9 and you use 0.5 thresholds, you are going to have a lot of false negatives

- \( AP/n \) could be a reasonable choice. Unfortunately, \( AP \) is not known at estimation time.
Contents

Topics

The context
The problem
Proposal 1: slope-based thresholds
Proposal 2: optimistic-pessimistic approach
Proposal 3: fault-proneness H-index
Final considerations

Slope-based Thresholds

A first proposal is applicable when we want to identify “early symptoms” of possible faultiness

The problem

- The manager wants that only good quality code is released.
- He wants to get some evidence that lets him take action as soon as the quality of a module under development becomes “not good enough”.

We have a model

- The model is built as shown before, based on data from previous developments (e.g., of previous releases).
Yes, but … how to use the model?

What does the model tell us?

Unsafe flat zone: here modules are probably faulty (fp is close to 1)

Safe flat zone: here modules are probably not faulty (fp is close to 0)
Slope-based thresholds

- When a module is created its CBO is zero.
- Then, while the module is being implemented, CBO increases over time.
- We want to identify “early symptoms” of possible faultiness.
- Idea: we need to constrain CBO to be less than a value \( \text{CBO}_{\text{MAX}} \) where small variations of CBO imply large variations of \( \text{fp(CBO)} \).

We get out of the safe zone when the slope increases “too much.”

The basic idea

We get out of the safe zone when the slope increases “too much.”
Where to set the threshold?

- When should the manager start worrying?
  - At $t_1$, fault proneness is comfortably close to zero.
  - At $t_4$, the slope is close to maximum, and fault proneness is already quite high.
- What about $t_2$ and $t_3$?

Our proposals

- Goal: identify a threshold based on "early symptoms" of faultiness

- Basic observations
  - $fp(x)$ looks rather "flat" for small values of $X$
    - even fairly large variations in $X$ imply small variations in fault-proneness
  - As $x$ increases, $fp(x)$ reaches a value past which
    - it departs very fast from the flat low-risk area
    - actually, it increases very fast

- Idea: set the threshold where the slope starts to become too steep
  Based on geometric properties of models
    - Maximum convexity
    - Fraction of maximum slope
Proposal 1: Maximum Convexity (MC)

- At the beginning the slope/direction of $f_p(X)$ changes very slowly
- At the end the slope/direction of $f_p(X)$ changes very slowly too
- But, in between the slope/direction of $f_p(X)$ changes much faster
- We define the threshold as the value $x_{MC}$ of $X$ in which $f_p(X)$ changes slope/direction the fastest

Maximum convexity (MC)

- Slope is measured by $f_p'(X)$
- Slope change is measured by $f_p''(X)$, i.e., convexity
- Since we are looking for the point where $f_p''(X)$ is maximum, $x_{MC}$ is such that $f_p'''(x_{MC}) = 0$
- Beware $x_{MC}$ is not necessarily where $f_p(X)$ is steepest or even “too steep”
Proposal 2: Fraction of Maximum Slope

- It might be too late to wait until the curve has reached maximum slope $f_{\text{max}}^\prime$.
- Define the threshold as the point $x_{\text{MS}}$ such that $f_{\text{MS}}^\prime(x_{\text{MS}})$ is a fraction $r$ of $f_{\text{max}}^\prime$.

$$f_{\text{MS}}^\prime(x_{\text{MS}}) = r f_{\text{max}}^\prime$$

- The value of $r$ is set by the practitioners, based on their goals.
- Via empirical studies, we found that $r = 0.5$ is a reasonable choice.
  - Hence, we look for $x_{\text{MS}/2}$, where the slope is half the maximum value.

Half maximum slope (MS/2)

- Max slope occurs when $fp = 0.5$ (too late!).
- Half max slope: Halfway between the safe area and the unsafe area (where $fp$ is already high, and small increase of $x$ results in large increase of $fp$).
- Min slope tends to zero. The safe flat area has slope close to zero.
MC threshold values

\[ x_{MC} = \frac{1}{c_1} \left( \ln(2 - \sqrt{3}) - c_0 \right) \approx -\frac{1.55 + c_0}{c_1} \]

\[ fp(x_{MC}) = \frac{1}{2} - \frac{\sqrt{3}}{6} \approx 0.2113 \]

The maximum convexity is always positioned where \( fp = 0.2113 \)

MS/2 threshold values

\[ x_{rMS} = \frac{1}{c_1} \left( \ln \left( \frac{1 - \sqrt{1 - r}}{1 + \sqrt{1 - r}} \right) - c_0 \right) \]

\[ fp(x_{rMS}) = \frac{1}{2} - \frac{\sqrt{1 - r}}{2} \]

When \( r = 0.5 \):

\[ x_{MS/2} \approx -\frac{0.7656 + c_0}{c_1} \]

\[ fp(x_{MS/2}) \approx 0.1464 \]

The slope is always half the maximum when \( fp = 0.1464 \)
BLR Thresholds: MS/2

BLR Thresholds: MS/2, MC
For this specific dataset.

The proposed thresholds are more risk-averse than both Fifty and All.
What about PBR models?

- Results of the mathematical analysis:
  - For any BLR model, maximum convexity occurs at the same values of fp.
  - For any BLR model, half maximum slope occurs at the same values of fp.
  - For any PBR model, maximum convexity occurs at the same values of fp.
  - For any PBR model, half maximum slope occurs at the same values of fp.

Fault-proneness values per type of model and type of threshold.

<table>
<thead>
<tr>
<th>Model</th>
<th>MS/2</th>
<th>MC</th>
</tr>
</thead>
<tbody>
<tr>
<td>PBR</td>
<td>0.1195</td>
<td>0.1587</td>
</tr>
<tr>
<td>BLR</td>
<td>0.1464</td>
<td>0.2113</td>
</tr>
</tbody>
</table>

- The values in the table above apply to all BLR and PBR models.
Empirical study

- We used real-life datasets hosted on the PROMISE repository, with data on
  - module actual faultiness
  - several independent variables
- We carried out T-time K-fold cross-validation
  - 10-time 10-fold cross-validation for larger datasets
  - 5-time 5-fold cross-validation for smaller datasets
- For each fold, we built statistically significant univariate BLR and PBR models for all internal attribute measures
- We computed overall average Precision, Recall, F-measure

Accuracy indicators

- Precision: proportion of estimated positives that are actually positive
  \[ Precision = \frac{TP}{EP} \]
- Recall: proportion of actual positives that are estimated positives
  \[ Recall = \frac{TP}{AP} \]

Recall indicates how risk-averse is the estimate. Recall=1 means that all actual positives are estimated positive.

- F-measure: harmonic mean of Precision and Recall
  \[ FM = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} \]
Berek Dataset: Average F-measures with BLR

<table>
<thead>
<tr>
<th>var</th>
<th>All</th>
<th>0.5</th>
<th>MC</th>
<th>MS/2</th>
</tr>
</thead>
<tbody>
<tr>
<td>WMC</td>
<td>0.80</td>
<td>0.79</td>
<td>0.74</td>
<td>0.65</td>
</tr>
<tr>
<td>CBO</td>
<td>0.82</td>
<td>0.77</td>
<td>0.83</td>
<td>0.81</td>
</tr>
<tr>
<td>RFC</td>
<td>0.88</td>
<td>0.88</td>
<td>0.88</td>
<td>0.88</td>
</tr>
<tr>
<td>CA</td>
<td>0.81</td>
<td>0.77</td>
<td>0.78</td>
<td>0.75</td>
</tr>
<tr>
<td>CE</td>
<td>0.73</td>
<td>0.69</td>
<td>0.81</td>
<td>0.77</td>
</tr>
<tr>
<td>LOC</td>
<td>0.91</td>
<td>0.91</td>
<td>0.91</td>
<td>0.88</td>
</tr>
<tr>
<td>MOA</td>
<td>0.69</td>
<td>0.69</td>
<td>0.52</td>
<td>0.54</td>
</tr>
<tr>
<td>CAM</td>
<td>0.69</td>
<td>0.69</td>
<td>0.75</td>
<td>0.75</td>
</tr>
<tr>
<td>AMC</td>
<td>0.73</td>
<td>0.73</td>
<td>0.70</td>
<td>0.68</td>
</tr>
<tr>
<td>Max CC</td>
<td>0.71</td>
<td>0.69</td>
<td>0.65</td>
<td>0.64</td>
</tr>
</tbody>
</table>

- There seems to be no best threshold: no threshold maximizes FM for all models.

Berek Dataset: Average Recall with BLR

<table>
<thead>
<tr>
<th>var</th>
<th>All</th>
<th>0.5</th>
<th>MC</th>
<th>MS/2</th>
</tr>
</thead>
<tbody>
<tr>
<td>WMC</td>
<td>0.75</td>
<td>0.69</td>
<td>0.81</td>
<td>0.94</td>
</tr>
<tr>
<td>CBO</td>
<td>0.88</td>
<td>0.75</td>
<td>0.94</td>
<td>0.94</td>
</tr>
<tr>
<td>RFC</td>
<td>0.94</td>
<td>0.88</td>
<td>0.94</td>
<td>0.94</td>
</tr>
<tr>
<td>CA</td>
<td>0.81</td>
<td>0.75</td>
<td>0.88</td>
<td>0.94</td>
</tr>
<tr>
<td>CE</td>
<td>0.75</td>
<td>0.63</td>
<td>0.94</td>
<td>0.94</td>
</tr>
<tr>
<td>LOC</td>
<td>0.94</td>
<td>0.94</td>
<td>0.94</td>
<td>0.94</td>
</tr>
<tr>
<td>MOA</td>
<td>0.56</td>
<td>0.56</td>
<td>0.75</td>
<td>1.00</td>
</tr>
<tr>
<td>CAM</td>
<td>0.75</td>
<td>0.69</td>
<td>0.94</td>
<td>0.94</td>
</tr>
<tr>
<td>AMC</td>
<td>0.75</td>
<td>0.69</td>
<td>0.88</td>
<td>0.88</td>
</tr>
<tr>
<td>Max CC</td>
<td>0.63</td>
<td>0.56</td>
<td>0.75</td>
<td>0.94</td>
</tr>
</tbody>
</table>

- MS/2 maximizes Recall for all models. It is the best threshold with respect to recall.
- MC provides similar performance (it is a bit less risk-averse)
Results for all datasets, with BLR
Best model for each dataset

<table>
<thead>
<tr>
<th>Project</th>
<th>var</th>
<th>n</th>
<th>AP/n</th>
<th>F-measure max thresholds</th>
<th>recall max thresholds</th>
</tr>
</thead>
<tbody>
<tr>
<td>intercafe   1</td>
<td>CBO</td>
<td>27</td>
<td>0.15</td>
<td>0.86 0.5</td>
<td>0.75 All 0.5 MC 0.5</td>
</tr>
<tr>
<td>jivy-1.1</td>
<td>LCOM</td>
<td>111</td>
<td>0.57</td>
<td>0.80 MC 0.5/2</td>
<td>1.00 MC 0.5/2</td>
</tr>
<tr>
<td>lucene-2.2</td>
<td>NPM</td>
<td>247</td>
<td>0.58</td>
<td>0.79 MC 0.5/2</td>
<td>1.00 MC 0.5/2</td>
</tr>
<tr>
<td>lucene-2.4</td>
<td>RFC</td>
<td>340</td>
<td>0.60</td>
<td>0.75 MC 0.5/2</td>
<td>1.00 MC 0.5/2</td>
</tr>
<tr>
<td>mcrunchomosci</td>
<td>MaxCC</td>
<td>27</td>
<td>0.37</td>
<td>0.89 MS 0.5/2</td>
<td>1.00 MS 0.5</td>
</tr>
<tr>
<td>phoenix</td>
<td>LCOM</td>
<td>26</td>
<td>0.77</td>
<td>1.00 MC 0.5/2</td>
<td>1.00 MC 0.5/2</td>
</tr>
<tr>
<td>pfftranlator</td>
<td>LCOM</td>
<td>33</td>
<td>0.45</td>
<td>0.81 MC</td>
<td>1.00 MS 0.5</td>
</tr>
<tr>
<td>poi-1.5</td>
<td>LCOM</td>
<td>237</td>
<td>0.59</td>
<td>0.76 0.5</td>
<td>1.00 MC 0.5/2</td>
</tr>
<tr>
<td>poi-2.5</td>
<td>WMC</td>
<td>385</td>
<td>0.64</td>
<td>0.83 0.5</td>
<td>1.00 MC 0.5/2</td>
</tr>
<tr>
<td>poi-2.5</td>
<td>NPM</td>
<td>385</td>
<td>0.64</td>
<td>0.83 0.5</td>
<td>1.00 MC 0.5/2</td>
</tr>
<tr>
<td>poi-2.5</td>
<td>LCOM3</td>
<td>385</td>
<td>0.64</td>
<td>0.83 0.5</td>
<td>1.00 MC 0.5/2</td>
</tr>
<tr>
<td>poi-3.0</td>
<td>RFC</td>
<td>442</td>
<td>0.64</td>
<td>0.82 0.5</td>
<td>1.00 MC 0.5/2</td>
</tr>
<tr>
<td>poi-3.0</td>
<td>CE</td>
<td>442</td>
<td>0.64</td>
<td>0.82 tr</td>
<td>1.00 MC 0.5/2</td>
</tr>
<tr>
<td>sklebagd</td>
<td>WMC</td>
<td>20</td>
<td>0.60</td>
<td>0.92 MC</td>
<td>1.00 MC 0.5/2</td>
</tr>
<tr>
<td>szylykafucha</td>
<td>CBO</td>
<td>25</td>
<td>0.56</td>
<td>0.89 MC 0.5/2</td>
<td>0.90 MC 0.5/2</td>
</tr>
<tr>
<td>velocity-1.4</td>
<td>RFC</td>
<td>196</td>
<td>0.75</td>
<td>0.92 MC 0.5/2</td>
<td>1.00 MC 0.5/2</td>
</tr>
<tr>
<td>workflow</td>
<td>RFC</td>
<td>39</td>
<td>0.51</td>
<td>0.77 MC 0.5/2</td>
<td>1.00 MC 0.5/2</td>
</tr>
<tr>
<td>xences-1.4</td>
<td>CBO</td>
<td>588</td>
<td>0.74</td>
<td>0.96 0.5</td>
<td>1.00 MC 0.5/2</td>
</tr>
<tr>
<td>valan-2.5</td>
<td>NOC</td>
<td>803</td>
<td>0.48</td>
<td>0.70 MC 0.5/2</td>
<td>1.00 MC 0.5/2</td>
</tr>
<tr>
<td>zuzel</td>
<td>RFC</td>
<td>29</td>
<td>0.45</td>
<td>0.80 MC</td>
<td>0.92 MC 0.5/2</td>
</tr>
<tr>
<td>kalkulator</td>
<td>AMC</td>
<td>27</td>
<td>0.22</td>
<td>0.80 0.5</td>
<td>0.67 All 0.5 MC 0.5</td>
</tr>
<tr>
<td>wospomaginpi</td>
<td>MGA</td>
<td>18</td>
<td>0.67</td>
<td>1.08 MC 0.5/2</td>
<td>1.00 MC 0.5/2</td>
</tr>
</tbody>
</table>

- MS/2 always maximizes Recall (and often also FM)
- MC achieves similar results

---

Results for PBR

- For PBR models we got very similar results.
Summary of results

- MC and MS/2 have
  - almost always better Recall than the other thresholds
  - often better F-measure than the other thresholds
- The introduced thresholds are
  - suitable for identifying “early symptoms” of possible faultiness of a module
  - derived from properties of the fault-proneness model
  - computed automatically
  - quite accurate in terms of Recall and often F-measure too

Conclusions

- If you have a BLR or PBR model q(x) that relates an interesting external quality q to some internal measure x
- You can use the following thresholds on q

<table>
<thead>
<tr>
<th>Model</th>
<th>MS/2</th>
<th>MC</th>
</tr>
</thead>
<tbody>
<tr>
<td>PBR</td>
<td>0.1195</td>
<td>0.1587</td>
</tr>
<tr>
<td>BLR</td>
<td>0.1464</td>
<td>0.2113</td>
</tr>
</tbody>
</table>

These values apply to any q and any x!

- According to our experimental results, you maximize the number of actually positive modules that are estimated positives, while you still get relatively few negative modules that are estimated positives.
- This means that you get an excellent trade-off between
  - the effectiveness of the development and maintenance effort
  - the costs of quality improvement
  - the costs of using faulty software
Conclusions

USE MS/2 or MC thresholds!
- Minimize risk
- Optimize use of resources
- Models and thresholds can be computed automatically

Contents

Topics
The context
The problem
Proposal 1: slope-based thresholds
Proposal 2: optimistic-pessimistic approach
Proposal 3: fault-proneness H-index
Final considerations
The context

- In this case, we consider what happens at the end of the coding phase:
  - You have a bunch of new modules and have to decide which of these modules deserve “special treatment” (e.g., code inspection) because they are likely faulty.
  - Modules developed in the past --whose faultiness is known-- are the training set
  - New modules --whose faultiness is unknown-- are the test set.

Conventional Approach

1. A fault-proneness model is derived from the training set
2. The model is used to estimate the test set
   - To this end, a threshold on fp can be set
     - based on local considerations
     - as \( \frac{A^\text{fp}_{\text{trainingSet}}}{N_{\text{trainingSet}}} \)
3. When actual faultiness data on the test set become available the accuracy of the estimates can be computed.
The optimistic-pessimistic approach\textsuperscript{2}

The test set is used to build two models:
- An optimistic one
- A pessimistic one

- Where the models agree, you can be reasonably confident that the obtained classification is right.
- When the models disagree, you should better consider the faultiness of the module in question “uncertain”

\textsuperscript{2}Luigi Lavazza and Sandro Morasca, “Identifying Thresholds for Software Faultiness via Optimistic and Pessimistic Estimations”, ESEM 2016

Building the optimistic model

- All the modules in the test set are considered as not faulty
  - This is an optimistic assumption!
- You make the union of the training set and the test set
- You build a BLR model as usual

The resulting model is optimistic, because of the initial optimistic assumption.
Building the pessimistic model

- All the modules in the test set are considered as not faulty
  - This is a pessimistic assumption!
- You make the union of the training set and the test set
- You build a BLR model as usual

The resulting model is pessimistic, because of the initial pessimistic assumption.

Optimistic Model Threshold and Optimistic Estimated Faultiness Model

- Select a threshold for the optimistic model and build an optimistic estimated faultiness model
Pessimistic Model Threshold and Pessimistic Estimated Faultiness Model

- Select a threshold for the pessimistic model and build a pessimistic estimated faultiness model

Where to place fp thresholds?

- As usual, we have to decide where to place thresholds for fault-proneness.
Possible fp thresholds

- Several thresholds are possible:
  - Pessimistic threshold: fraction of modules that are known to be positive
    $$t_{pess} = \frac{AP_{trainingSet}}{n_{trainingSet} + n_{testSet}}$$
  - Optimistic threshold: fraction of modules that are known to be positive or are unknown
    $$t_{opt} = \frac{AP_{trainingSet} + UK}{n_{trainingSet} + n_{testSet}}$$
  - Neutral threshold:
    $$t_{neut} = \frac{AP_{trainingSet}}{n_{trainingSet}}$$

- Note that $$t_{opt} > t_{neut} > t_{pess}$$

UK is the number of unknown, i.e., $$n_{testSet}$$

Intersections

We use $$x_{pp}$$ as the threshold for the pessimistic model and $$x_{oo}$$ for the optimistic model
Optimistic and Pessimistic Thresholds and Models

Classification using the optimistic-pessimistic approach

- Modules are classified as follows:
  - $x \leq x_p \Rightarrow$ negative
  - $x \geq x_o \Rightarrow$ positive
  - $x_p < x < x_o \Rightarrow$ undecided
Other estimation approaches

- The reference approach
  - The test set is classified based on the model derived from the training set

- The pessimistic model approach alone
  - \( x \leq x_p \Rightarrow \text{negative} \)
  - \( x > x_p \Rightarrow \text{positive} \)

- The optimistic model approach alone
  - \( x \leq x_o \Rightarrow \text{negative} \)
  - \( x > x_o \Rightarrow \text{positive} \)
Comparison

- We compared the classification obtained using the optimistic-pessimistic approach with the classifications obtained using other approaches.
- Note: when considering the optimistic-pessimistic approach, only classified modules were considered in the computation of the accuracy indicators.

Data sets

- We used 48 real-life datasets hosted on the PROMISE repository
- We carried out 10-fold cross-validation
- We almost always obtained the best results with
  - $x_p = x_{pp}$ and $x_o = x_{oo}$, or
  - $x_p = x_{po}$ and $x_o = x_{oo}$
Results of the comparison

![Box plot](chart.png)

Conclusion

- By means of the traditional approach you get quite variable results, because modules in the grey zone are classified as either faulty or not faulty anyway.
- With the optimistic-pessimistic approach the modules in the grey zone are not estimated, thus avoiding many classification errors.
- Note: if a module is in the grey zone of the CBO models, it could very well be out of the grey zone of the RFC model …
Contents

Topics

The context
The problem
Proposal 1: slope-based thresholds
Proposal 2: optimistic-pessimistic approach
Proposal 3: fault-proneness H-index
Final considerations

Proposal

A new approach to building an estimated faultiness model based on the definition of the Fault-proneness H-Index, an extension to the H-index

Basic idea

- the H-Index identifies the most important papers of a researcher
- the Fault-proneness H-Index identifies the most fault-prone modules in a set of modules

Advantage

- we do not need to set a threshold ourselves, but the threshold is derived from the data

---

H-index computation

- Order absolute frequencies \( af(z) \) in decreasing order
- Set \( z = 0 \) as the initial value of the H-Index
- Increase the value of \( z \) by 1 as long as \( af(z) \geq z \)
- The value of the H-index is the last value \( z \) such that \( af(z) \geq z \)
- The value of \( h \) can be found at the intersection of two functions
  - \( af(z) \), which is decreasing
  - \( z \), which is linearly increasing

---

**My H-index**

**Rank**  **Title**  **Citations**  **Authors**  **Journal/Book**
1  SPAD: An environment for software process analysis, design, and enactment  30  S. Bandinelli, A. Fuggetta, C. Ghedini, L. Lavazza  Software Proc.
3  A conceptual basis for feature engineering  26  C. Red Turner, A. Fuggetta, L. Lavazza, A. Wolf  The Journal
5  Combining UML and formal notations for modeling real-time systems  19  L. Lavazza, G. Quaroni, M. Venturini  ACM/IEEE.
7  The architecture of SPAD: 1 process-centred SE  16  S. Bandinelli, M. Braga, A. Fuggetta, L. Lavazza  Lecture Notes.
10  Providing automated support for the GQM measurement process  12  L. Lavazza  EEE Software.
14  Combining Problem Frames and UML in the Description of Software Requirements  10  L. Lavazza, V. Del Bianco  IATEF 2006.
15  System/C2-based model-driven design for embedded systems  9  Riccobene, Scandurra, Bocchi, Rosti, Lavazza  R/TECS.
18  A UML-based approach for reengineering problem frames  9  L. Lavazza, V. Del Bianco  EE Seminar.
19  A case study in COSMIC functional size measurement: The rice cooker revisited  9  L. Lavazza, V. Del Bianco  Software Pr.
20  Automated support for process-aware definition and execution of measurement plans  9  L. Lavazza, Baralde  ICSE 2005.
22  Requirements based estimation of change costs  9  L. Lavazza, V. Del Bianco  ACM.

---

Luigi Lavazza - Università dell'Insubria
ICSEA 2016
# My H-index

<table>
<thead>
<tr>
<th>Title</th>
<th>Authors</th>
<th>Journal/Book/Conference</th>
</tr>
</thead>
<tbody>
<tr>
<td>A framework for the assessment of OSS</td>
<td>50-50</td>
<td>Information Engineering</td>
</tr>
<tr>
<td>The GOODSTEP Project: General Object-Oriented Database for Software</td>
<td>GOODSTEP Team</td>
<td>Information Engineering</td>
</tr>
<tr>
<td>Engineering</td>
<td></td>
<td></td>
</tr>
<tr>
<td>An experience in process assessment</td>
<td>43</td>
<td>Proceedings of the 9th Int.</td>
</tr>
<tr>
<td>Model based functional size measurement</td>
<td>51</td>
<td>Software Process and Products</td>
</tr>
<tr>
<td>Combining Problem Frames and UML in the Description of Software</td>
<td>32</td>
<td>IEEE Software Standards</td>
</tr>
<tr>
<td>Requirements-based estimation of change costs</td>
<td>22-22</td>
<td>IEEE Software Standards</td>
</tr>
<tr>
<td>An investigation of the users' perception of OSS quality</td>
<td>21-21</td>
<td>International Conference</td>
</tr>
<tr>
<td>A survey on open source software trustworthiness</td>
<td>23-23</td>
<td>IEEE Software Standards</td>
</tr>
<tr>
<td>Managing software artifacts on the Web with Juxtapose</td>
<td>21-21</td>
<td>Conference and Workshops</td>
</tr>
<tr>
<td>Quality of Open Source Software: The QualISO Trustworthiness Model</td>
<td>19-19</td>
<td>Conference and Workshops</td>
</tr>
<tr>
<td>H-index Graphical Representation</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[Graph showing H-index with axes labeled Z and X]
This is the $af=z$ line.
**Fault-proneness H-Index Computation**

- Order the modules in decreasing order of estimated fault-proneness FP.
- Set $z = 0$ as the initial value of the FPH-Index.
- Increase the value of $z$ by $1/n$ as long as $FP(x_m) \geq z/n$.
- The value of $fph$ is the last value of $FP(x_m)$ for which $FP(x_m) \geq z/n$ holds.
- The value of $fph$ can be found at the intersection of two functions:
  - $FP(x_m)$, which is decreasing with $z$.
  - $z/n$, which is linearly increasing.
Fault-proneness H-Index Graphical Representation

ICSEA 2016 - 107 - Setting Thresholds in Software Engineering
Results

- The H-index-based estimation technique
  - has generally higher values of Recall
  - has generally lower values of Precision
  - has generally higher values of F-measure when the weight of Recall is comparatively high

Contents

- The context
- The problem
- Proposal 1: slope-based thresholds
- Proposal 2: optimistic-pessimistic approach
- Proposal 3: fault-proneness H-index
- Final considerations
Estimates based uniquely on internal measures

- Do we get good results (i.e., accurate estimates) with this strategy?
- Not really.
- Let’s see some experimental results.


Let us consider the proposal by Erni & Lewerentz (or by Lanza and Marinescu)

\[ T_{\text{low}} = \mu - \sigma \]
\[ T_{\text{high}} = \mu + \sigma \]

Where \( \mu \) is the mean and \( \sigma \) is the standard deviation

- The threshold do not depend on faultiness data, but just in internal measures.
- What happens when we take into consideration faultiness data?
- Let’s see how the thresholds behave in fault-proneness models.
Estimates based uniquely on internal measures

- $f_p(T_{low})$ is excessively low!

- $f_p(T_{high})$ is excessively high!

$T_{low}$ is excessively low: by doubling it you our estimates improve in every respect:

- TN increases substantially
- TP and FN do not change
- FP decreases substantially
Estimates based uniquely on internal measures

- Even though the low thresholds for RFC and WMC are computed in the same way, they give very different results.
  - If you use $\text{RFC}_{\text{low}}$ you get $fp < 0.01$
  - If you use $\text{WMC}_{\text{low}}$ you get 0.1 (circa)

For these models. The difference could be larger!

Conclusions

- There are many different ways of setting thresholds
- I would recommend methods based on information about internal measures and faultiness information
- Which one is the best?
  - Time will tell . . .
  - Does a “best” method really exist?
Future work

- A lot . . .