An Objectified Entropy-based Software Metric for Assessing the Maturity of Use Case Diagrams

Stanisław Jerzy Niepostyn
School of Computer Science & Technologies, University of Economics and Human Sciences in Warsaw, Okopowa str. 59, 01-043 Warsaw, Poland *j.niepostyn@vizja.pl*

Wiktor Bohdan Daszczuk
Institute of Computer Science, Warsaw University of Technology, Nowowiejska str. 15/19, 00-665 Warsaw, Poland *wiktor.daszczuk@pw.edu.pl*

Abstract

Various metrics exist for evaluating UML diagrams, including entropy-based ones like ours, which assess information content. They allow for judging certain features of the design that depend on the information content. This paper proposes the FBS24 use case diagram measure, which should ensure that the software architecture design consists of mature UML diagrams. Detecting an inappropriate (immature, unfinished) Use Case Diagram before the software development phase can stall the entire software development process until a mature UCD is developed. Currently, no indicators (metrics) show the maturity (or lack of applicability) of diagrams. Moreover, in most such software metrics, weights are selected arbitrarily, which leads to numerous anomalies. We show how to construct the measure most resistant to such anomalies. We also show how to check the correctness and usefulness of the constructed measure. As our measure is objective, it is suitable for remote work of distributed teams building IT systems.

Keywords: FBS, software project metric, entropy, normalized entropy, use cases.

1. Introduction

UML is the standard for software design, leading to measures for evaluating projects, like labor intensity and maintenance ease. The Abran procedure for these measures [1] involves creating a meta-project, applying it, and assessing industrial use, but lacks details on tuning metrics. The e-CMDA algorithm [2] systematically builds UML diagrams from context to implementation view using information content measures.

In this paper, we propose the FBS24 (24 - the year of publication) use case diagram measure to ensure that the software architecture design consists of mature UML diagrams. To our best knowledge, currently there are no indicators (metrics) showing the maturity (or lack of applicability) of UML diagrams.

Our approach calculates entropy for functionality (F), behavior (B), and structure (S) in UML diagrams, called the FBS measure (FBS16 – from the year of publication). This measure qualitatively assesses project consistency and completeness based on entropy values: 1 (no information) or <1 (information present). Using the proposed FBS24, the detection of inappropriate (immature, unfinished) Use Case Diagram (UCD) before the software development phase can prevent the entire software development process from being stalled until a mature UCD is developed.

By maturity, we mean a sufficiently large information content of the diagram, understood as the number of elements and connections between them. We do not have a recipe for a minimum maturity level, which the designer must determine. However, we provide a way to compare the maturity of a specific diagram with the designer's pattern.

Maturity is a concept that the reader probably cannot find in the literature. We believe that this notion is useful and sound.

The entropy formula for software architecture, presented in [3], defines the FBS16 metric. The work [2] provides a precise formula for this metric, with UML element weights chosen subjectively based on the authors experience. Weights range from 0 (no influence) to 3 (strong influence) on the software architecture dimension.

However, when assessing industrial diagrams using the FBS metric constructed in this way, it turns out that FBS16 metric anomalies occur for such selected weights of UML diagram element kinds. The higher value of the FBS metric for the diagram containing more elements can result from the difference in concentration, i.e., predominance of elements of one or two kinds.

The proposed FBS24 metric offers a method to objectively determine weights in the functionality dimension of use case diagrams, improving accuracy and resistance to anomalies when evaluating normalized entropy, regardless of UML element count.

While adjusting weights and verifying the FBS24 metric, it turned out that: it helps assess UML diagrams against project standards, distinguishing business, system and implementation views; it identifies inadequate or immature diagrams, such as those overloaded with unnecessary information or unfinished. These properties, confirmed by calculations and observations, highlight the advantages of the FBS24 metric.

2. Related work

Software metrics, dating back to the 1970s, aimed to gauge software quality, workload, and complexity in the IT industry. Despite standards like COSMIC and IFPUG, most lack the foundation to become widely recognized.

Software architecture metrics have gained prominence over source code metrics due to their ability to optimize and evaluate IT systems before code creation. Despite the usefulness of source code metrics like lines of code and cyclomatic complexity, they cannot adequately assess system consistency, completeness, or labor consumption.

Quantitative models often focus on class diagrams, which are crucial in object-oriented software development. In the late 1990s, Marchesi [4] introduced metrics for class and use case diagrams, while Genero [5] later expanded these metrics to assess software architecture completeness. However, these metrics primarily counted element occurrences or connections, limiting their ability to gauge software architecture.

The Multi-Attribute Decision Matrix [6] was among the earliest metrics to estimate software complexity based on information content (entropy). Following this, metrics like AICC [7] (Average Information Content Classification) and CDE [8] (Class Design Entropy) emerged, calculating complexity for structured and object-oriented code, respectively. Notably, while initially proposed for source code, these metrics were later applied to evaluate UML diagrams. It is worth noting that, to our best knowledge, there is currently no work related to the assessment of the maturity of UCDs, which makes FBS24 a unique software metric.

3. FBS24 elaboration rules (based on the Abran procedure)

3.1. The overview of the FBS metric

The accuracy of the metric relies on coefficients linked to specific element kinds, determined by methods described in standards like the Cosmic method [1], IFPUG method [9], or Use Case function points [10], which have limitations: subjective specialist input introduces subjectivity; tailored to individual projects, hindering comparison between projects, levels, or versions.

We aim to standardize UML element weights to create a metric ensuring project independence and comparability across systems. Our approach uses the new FBS24 metric to assess diagram maturity and design quality. The two observed anomalies are: entropy=1 having one element of every used kind; growth of entropy with the increase of the number of elements, due to concentration difference.

We will show how to choose weights for element kinds to create an objective, robust FBS24 metric, reducing both anomalies. This metric will assess UCD maturity and allow project comparisons.

3.2. Calculation of the FBS24 metric

We apply the FBS24 metric to assess the maturity of UCDs, not entire projects. Each element kind is weighted between 0 and 3. Most UCD elements only affect the description of functionality (e.g., Use Case), but there are elements such as Actor that can also describe the structure of the IT system (e.g., the organizational structure of the system). This was taken into account in our calculations, which confirm that UCD describes the system architecture only as a dimension of functionality.

The formula to calculate the FBS24 metric, based on normalized entropy, is identical to that of FBS16:

$$\boldsymbol{E_j = \frac{-1}{ln(m)} \sum_{i=1}^{m} \dot{p}_{ij} \cdot ln(\dot{p}_{ij})} \quad (1)$$

where m is the number of UML element kinds in the model, p_{ij} is the normalized value of the element i in the selected dimension j ($j \in$ {Functionality, Behavior, Structure}). In addition, we assume that for $0 \cdot \ln(0)$, the value is set to 0, and the FBS entropy is set to 1.

Thus: $\boldsymbol{FBS = (E_{functionality}, E_{behavior}, E_{structure})}$, where E_j is given by formula (1). FBS16 metric calculation in 3 steps is provided in [2].

Normalized entropy, ranging from 0 to 1 and non-additive, aids for a value of 1, indicating no information. However, it may not always decrease with more elements, leading to higher FBS metric values in some cases. To achieve a valuable FBS metric, we aim to minimize undesirable cases by carefully selecting UML element weights.

3.3. Building diagram series and calculation of FBS16 metrics

Artificially constructed UML use case diagrams, grouped into series: the *Uc* series consists of a single actor and an increasing number of UCs (Use Cases) and associations; the *UcA* series consists of one UC and an increasing number of actors and associations; the *UcZ* series consists of one actor, one association and a growing number of UCs and relationships between UCs; the *Diag* series are diagrams composed of all types of UML. Figure 1 shows four diagram series, each focusing on specific UML element kinds, helping to identify optimal weights by minimizing incorrect calculations. The optimal set, shown in red, has the fewest errors (non-monotonicity).

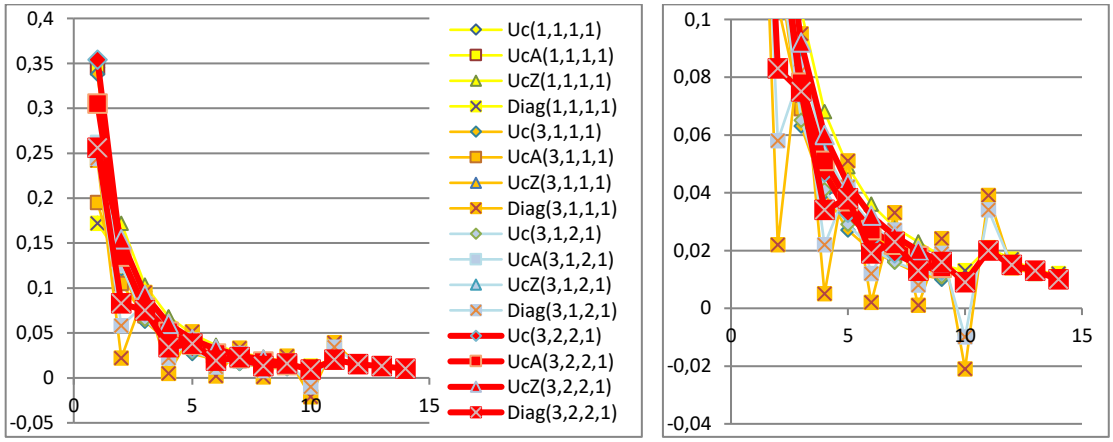

Fig. 1. FBS16 metric values for UML use case diagrams depending on the number of elements (left), and the zoom in area 0-0.1 (right).

4. Implementation of the FBS24

Abran selected coefficients for their equations without specifying a procedure, validating the metric on a subset of diagrams from system documentation and benchmark models. We present a method for objectively selecting FBS24 weights, enhancing resistance to undesirable cases.

We plot all weight cases dependency of FBS.F on the number of elements in Figure

1, showing overall monotonicity. The first step ends with selecting a set of weights whose series group is strictly monotonic, does not face a false lack of information, and presents the fastest drop of value as the number of elements increases (this allows for more clear detection of immature diagrams).

As the preparation for the verification, we defined the needed FBS values for mature diagrams. Our experience in designing IT systems shows that:

- mature Business UCDs have from 24 to 55 elements, with a small number of DirectedRelationship elements (→0), immature have fewer than 20 elements,
- mature System UCDs have from 15 to 80 elements and have more UC elements than Actors elements, immature have fewer than 15 elements or have more Actors elements than UCs,
- mature Implementation UCDs have from 20 to 100 elements and more UC elements than Actors elements, immature have fewer than 20 elements or have more Actors elements than UCs.

Our assumptions align closely with FBS24 metric calculations in industrial diagrams. However, analysis reveals more diagram types than anticipated, stemming from practices in system design. The FBS24 metric excels in measuring diagram maturity, effective for business, system, and implementation diagrams.

4.1. Metric validation on industrial diagrams

The final validation of the FBS24 metric was performed using use case diagrams collected from various IT projects of complex IT systems over the years 2005-2019 - over 2,000 UML diagrams in 18 project repositories as EAP files created in the Enterprise Architect tool by Sparx.

It is worth mentioning that actor diagrams, lacking use cases but featuring actors and connections, were evaluated differently. Their high concentration of actors and connections resulted in lower FBS24 values compared to standard UC diagrams, blurring the boundary between mature and immature actor diagrams.

We prepared the plot of the industrial cases, showing the dependency between an FBS.F and a number of elements of a diagram. We grouped the diagrams according to their structure (Actors diagram / general UC diagram) and design level. The extreme lines in the plot, between which almost all others fall, are *Diag* and *Uc* series obtained in step 1.

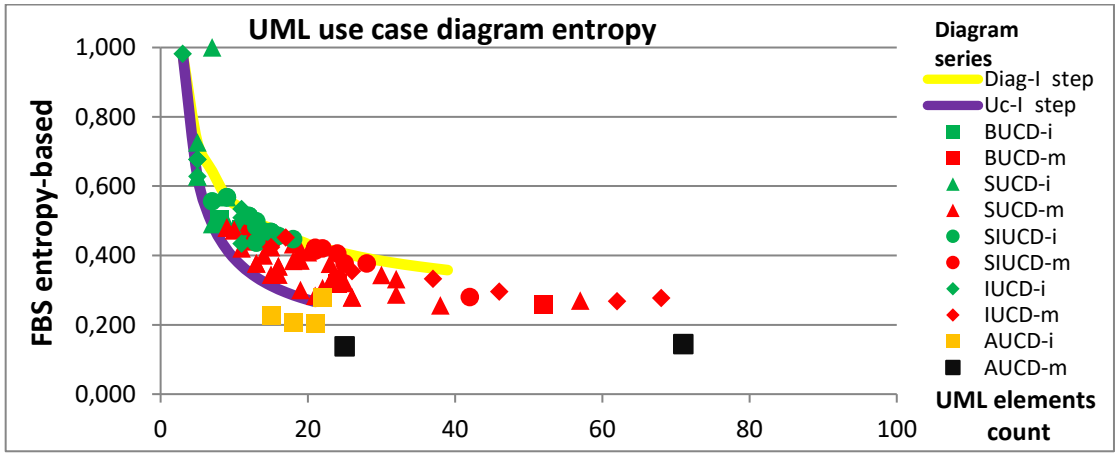

Fig. 2. FBS24 metric values of studied industrial UML UCDs. The abbreviations hidden if letter or two letters are: B-business, S-system, I-implementation, SI-internal, A-actor. The last letter denotes: m-mature, i-immature.

Some diagrams exhibit FBS24 metric values beyond the range defined by the *Diag* and *Uc* series. These diagrams are deemed immature, indicating deficiencies in their construction or usefulness for the project. Comparing their metric values to those of other series helps identify such outliers. Figure 2 shows the thresholds between mature and immature diagrams: for business UCDs 0.4; for system UCDs 0.495; for implementation UCDs 0.423; for internal UCDs have 0.425. For business Actors diagrams there is not

enough data: the threshold is between 0.14 and 0.2.

5. Conclusions

The article introduces entropy-based metrics to evaluate UML diagram information content, especially functionality, focusing on the new FBS24 metric. It quantitatively measures information content to ensure the software architecture design consists of mature UML diagrams. Detecting an inappropriate (immature, unfinished) UCD before the software development phase can stall the entire software development process until a mature UCD is developed. Currently, there are no indicators (metrics) showing the maturity (or lack of applicability) of UML diagrams. A weight adjustment procedure ensures metric independence, reducing false lack of information and non-monotony effects. The metric evaluates diagram maturity across design levels, validated in real-world projects, identifying immature diagrams despite initial assumptions. Some observed phenomena are discussed.

- Lack of information is desirable for structure and behavior dimensions, resulting in a value of 1 regardless of UCD size.
- UCDs with only one kind of elements (UC or UCA) have no information.
- Some UCDs show increased FBS metric values with more elements, indicating reduced information content.
- High concentration in UCDs with few element kinds and occurrences can identify incomplete or unusable diagrams.
- While the FBS metric generally decreases with more elements, cases of non-monotonicity suggest that information content impacts the metric more than element count.

The proposed metric, focusing on element kind concentration, has broader applications such as diagram classification (e.g., business view, system view, implementation view) and actor diagram identification. Our future work will extend this methodology to other metric dimensions: behavior and structure. Once completed, we aim to develop a plugin for the Enterprise Architect framework to measure diagram information content and draw conclusions from its values and comparisons across diagrams.

References

1. Abran, A.: Software Metrics and Software Metrology; Wiley-IEEE Press, ISBN 9780470597200 (2010)
2. Niepostyn, S.J.: Entropy-Based Consistent Model Driven Architecture. In Proceedings of the Photonics Applications in Astronomy, Communications, Industry, and High Energy Physics Experiments, Wilga, Poland, Romaniuk, R.S., Ed.; SPIE, pp. 1–10 (2016)
3. Niepostyn, S.J., Daszczuk, W.B.: Entropy as a Measure of Consistency in Software Architecture. Entropy 25, 328, doi:10.3390/e25020328 (2023)
4. Marchesi, M.: OOA Metrics for the Unified Modeling Language. In Proceedings of the Second Euromicro Conference on Software Maintenance and Reengineering, Florence, Italy, 11 March 1998; IEEE, pp. 67–73 (1998)
5. Genero, M., Piattini, M., Calero, C.: Metrics for Software Conceptual Models; World Scientific, ISBN 978-1-86094-497-0 (2005)
6. Hwang, C.-L., Lin, M.-J.: Group Decision Making under Multiple Criteria; Lecture Notes in Economics and Mathematical Systems; Springer Berlin Heidelberg: Berlin, Heidelberg, Vol. 281; ISBN 978-3-540-17177-5 (1987)
7. Harrison, W.: An Entropy-Based Measure of Software Complexity. IEEE Trans. Softw. Eng. 18, 1025–1029, doi:10.1109/32.177371 (1992)
8. Bansiya, J., Davis, C., Etzkorn, L.: An Entropy-Based Complexity Measure for Object-Oriented Designs. Theory Pract. Object Syst. 1999, 5, 111–118, (1999)
9. ISO IFPUG Functional Size Measurement Method (2009)
10. Karner, G.: Resource Estimation for Objectory Projects. Object. Syst. SF AB (1993)