

FedCSIS_2023_paper_6538.pdf

Software Architecture Entropy

S. Niepostyn,¹ University of Economics and Human Sciences in Warsaw

²
Abstract- In building software architectures, the relations between elements in different diagrams are often overlooked. When constructing software architecture, IT architects more or less consciously however introduce elements that represent the same object instance on different diagrams with similar names. These connections are called consistency rules and are usually not saved in any way in the modeling tool. It was mathematically proved that the application of consistency rules increases the information content of software architecture. The feelings about increasing readability and ordering of software architecture by means of consistency rules have their mathematical rationale. In this article it was carried out a proof of decreasing information entropy while applying consistency rules in the construction of software architecture of IT systems. Therefore, it has been shown that marking selected elements in different diagrams with similar names is therefore an implicit way to increase the information content of software architecture while simultaneously improving its orderliness and readability.

I. INTRODUCTION

²
In building software architectures, the relations between elements in different diagrams are often overlooked. When constructing software architecture, IT architects more or less consciously however introduce elements that represent the same object instance on different diagrams with similar names. ¹These connections are called consistency rules and are usually not saved in any way in the modeling tool. But these implicit ¹consistency rules significantly facilitate reading of such software architecture. Consequently such consistency rules lead to increase of the ¹software architecture information content.

In this article, a proof of decreasing entropy of the architecture model while applying consistency rules between ¹various elements of different diagrams in the construction of software architecture of IT systems was carried out. The uncertainty of software architecture construction is thus reduced when introducing consistency rules and therefore the orderliness and consistency of the entire software architecture increases.

Therefore, marking selected elements in different models using similar names is an implicit way to increase the information content of the software architecture and at the same time to improve its orderliness and readability. This explains use of consistency rules in the construction of software architecture, in order to cut its design time, as compared ¹to the designing without paying attention to consistency rules.

The term "entropy" is used in thermodynamics, probability theory, information theory, or the theory of dynamic systems. However, as Thims [1] pointed out, Shannon's equation regarding information theory has no relation to similar equations used in thermodynamics or statistical mechanics. In the further part of the work the term "entropy" will be understood in the meaning assigned to it by Shannon in accordance with the formula:

$$\text{Entropy} = -\sum_{i=1}^{n_i} (P_i \log_2 P_i) \quad (1) \quad 3$$

where ¹ n_i is the total number of all elements, and P_i are the probabilities of occurrence of a set i of possible elements.

Entropy may be perceived as a measure of uncertainty associated with discrete distribution with appropriate probabilities. In research on UML [2] diagrams, it is assumed that the probability distribution of a given UML element is the quotient of the number of occurrences of a given UML element to the number of occurrences of all UML elements in a UML diagram. For example, the probability distribution for one class and for one attribute is identical and amounts to 0.5 for a UML diagram consisting of only one class (one occurrence) and one attribute (one occurrence).

For a UML diagram composed of only one class, entropy would be equal to 0, because the probability distribution for this one class would be equal to 1.0 (certain event). Hence, a UML diagram with only one class does not contain any ¹information about the given system.

Using the software complexity metrics with entropy, it can be proved that for the same diagrams configuration, the decrease in entropy occurs due to the increase in the number of consistency rules (links) between the elements of these diagrams. Relationships between individual elements of diagrams should be interpreted as displaying the existence of consistency rules between elements. These rules do not introduce additional elements to the diagrams

configuration (relationships between elements of diagrams), but are expressed in practice through similar names of the elements being linked.

The definition of inconsistencies in the models was provided by Spanoudakis and Zisman [3], indicating that the ultimate goal of management consistency is to maintain consistency during the design of the system. However, often researchers, including Straeten [4], indicate that this is not realistic in a real project, where several architects work in parallel. On the other hand, it is worth mentioning that many studies on inconsistencies, along with the area of application of consistency rules ([5], [6], [7]), have come to interesting proposals for defining and applying individual properties of various UML models in the field of software development. In addition, recent publications in this area, Torre ([8],[9]), seem to aim to develop a full list of UML diagram consistency rules according to the new proposals for their classification after Allaki [10].

II. AICC AND CDE SOFTWARE COMPLEXITY METRICS

One of the first metrics based on estimating the information content (entropy) of the software data structure was the AICC [11] (Average Information Content Classification) and the CDE [12] (Class Design Entropy) metrics, which calculated the complexity of the software code, the first referring to structural languages (in particular PL/I) and the other object-oriented languages. It is worth noting that the AICC and CDE metrics have been proposed for estimating the source codes. The proposed mathematical formulas of these metrics were used in their later applications to assess, among others, UML diagrams.

The Average Information Content Classification metrics (AICC) describes the complexity of the software based on the concept of entropy, and was proposed to estimate the complexity of the source code. The AICC metrics includes the following parameters: N_l is the total number of (non-unique) symbols of the language used in the code, and f_i , where $1 \leq i \leq n_l$, is the number of occurrences of the i -th language symbol appearing in the source code.

The formula for calculating the AICC metrics is given below.

$$AICC = - \sum_{i=1}^{n_l} \frac{f_i}{N_l} \log_2 \frac{f_i}{N_l} \quad (2)$$

The interpretation of this metric is that a program with a higher value of a metric should be less complex (complex) than a program with a lower value of this metric. The AICC values are not additive, nor the comparison of its values for two different modules is not meaningful, but already these values, for the same module, could indicate justification for using, for example, some complex software libraries. For large systems, the AICC metric can take values between 1.7 and 5.1. In AICC-based diagrams describing UML diagrams, the elements from which a given diagram can be built are used as language symbols, whereas n_l means the number of groups of elements of the same type.

The Class Design Entropy metrics, like the AICC metrics, has also been proposed for estimating the source code. An identical formula was proposed for calculating the value of this metric as for the AICC metric, where instead of all language symbols, only the names of identifiers appearing in the tested part of the software (class definitions) such as class name, variable, constant, class, and symbols characteristic for a given language are included. On the other hand, symbols characteristic for a given language, such as keywords or operators, are omitted. The proposed CDE metrics includes the following parameters: N_l is the total number of occurrences of identifiers (non-unique) used in the definition of the tested class, and f_i , where $1 \leq i \leq n_l$ is the number of occurrences of the i -th identifier in the definition of this class, n_l is the number of groups identifiers with the same name (unique identifier names) of the class definition being examined. The formula for calculating the CDE metrics, which is identical to (2), is given below, but the symbols used have different meanings.

$$CDE = - \sum_{i=1}^{n_l} \frac{f_i}{N_l} \log_2 \frac{f_i}{N_l} \quad (3)$$

In the diagrams based on the above formula, describing UML diagrams, the name of the UML element of the analyzed diagram is usually taken as software code identifiers, whereas n_1 means the number of groups of UML elements having similar names.

The AICC and CDE metrics have been used in this work to demonstrate that the application of consistency rules, understood as linking elements of identical interpretation in any independent diagram, results in a decrease in its entropy, i.e. an increase in the information content of the model and an increase in its orderliness.

III. PROOF OF THE DECREASE IN ENTROPY WHEN APPLYING CONSISTENCY RULES

Below is proof of the decrease in entropy when linking UML elements of different diagrams using identical names, i.e. it will be proved that $E_{indep} > E_{dep}$, where E_{indep} is the entropy of configuration of independent diagrams, and E_{dep} is the entropy of configuration of dependent diagrams using consistency rules. The formula for the AICC metrics described in (2) will be used, as well as the formula for the CDE metrics, given by (3).

Figure 1 shows two configurations with two UML diagrams. On the left a configuration with independent diagrams, and on the right a configuration with related diagrams. The configuration shown on the left has 4 occurrences of independent UML elements (two occurrences of UML Activity elements named "a" and "c", one occurrence of UML UseCase element named "d" and one occurrence of UML ControlFlow element without a name). Thus, according to (2), parameter N_I is the number of all occurrences of UML elements (4 UML elements), parameter η_1 is the number of all types of elements in the diagram (3 types of elements: UML Activity, UML UseCase, UML ControlFlow), and the number of occurrences individual UML elements are as follows: $f_{UML\ Activity} = 2$ (2 elements of UML Activity); $f_{UML\ UseCase} = 1$ (1 element of UML UseCase); $f_{UML\ ControlFlow} = 1$ (1 element of UML ControlFlow).

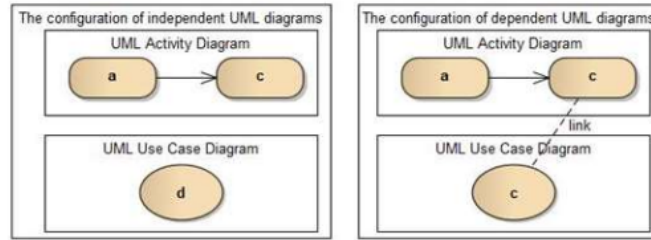


Figure 1. Configuration with two UML diagrams with a minimum number of element.

On the other hand, the configuration shown on the right has 2 independent elements (UML Activity element named "a" and unnamed UML ControlFlow connection) and 1 dependent element (element named "c" appearing as UML Activity element in the top diagram and the same element occurring as UML UseCase in the bottom diagram). The connection drawn with a dashed line and named "link" is not part of the configuration, but indicates the consistency rule. Thus, according to (3), parameter N_I means the number of all UML elements (4 non-unique identifiers, i.e. all UML elements), parameter η_1 is the number of all elements on the diagram having different names (3 unique identifiers: "a", "c", and an unnamed identifier). Moreover the number of occurrences of individual UML elements with different names is as follows: $\hat{f}_a = 1$ (1 UML element with the identifier "a"); $\hat{f}_c = 2$ (2 UML elements with identifier "c"); $\hat{f}_{unnamed} = 1$ (1 UML element with id unnamed).

It is worth noting that unrelated UML elements occur only once in a diagram system, whereas UML elements related in a given configuration occur many times in the form of various UML elements (e.g. the UML element named "c" occurs once as part of the UML Activity, and for the second time as UML UseCase element).

Assuming that the entropy E_{indep} for an unrelated configuration results from the formula for AICC and is given by the formula:

$$E_{\text{indep}} = \text{AICC} = - \sum_{i=1}^{\eta_1} \frac{f_i}{N} \log_2 \frac{f_i}{N}, N \in \mathbb{N}^+, f_i \in \mathbb{N}^+ \quad (4)$$

where the symbol N denotes the number of all elements of the configuration of independent (unrelated) diagrams, the symbol η_1 denotes the number of occurrences of elements of type i , and f_i is the number of occurrences of elements of the i -th of the type $i = 0, 1, 2 \dots$

On the other hand, entropy E_{dep} for the related same configuration results from the formula for CDE and is given by the following formula:

$$E_{\text{dep}} = \text{CDE} = - \sum_{i=1}^{\eta_2} \frac{\hat{f}_i}{N} \log_2 \frac{\hat{f}_i}{N}, N \in \mathbb{N}^+, \hat{f}_i \in \mathbb{N}^+ \quad (5)$$

The symbol N denotes the number of all configuration elements of related diagrams, the symbol η_2 represents the number of occurrences of elements with the given name i , and $\hat{f}_i$ the number of occurrences of the i -th element with the given name $i = 0, 1, 2 \dots$

For the configuration of independent diagrams there is equality: $f_1 = f_2 = \dots = f_{\eta_1} = 1$. By inequalities $1 \leq id \leq \eta_2$ let us indicate the number of elements (in the configuration of related diagrams) such that $\hat{f}_k = 1$. Without loss of generality, we can assume that:

$$\hat{f}_1 = \hat{f}_2 = \dots = \hat{f}_{id} = 1 \quad (6)$$

In addition, equality is satisfied:

$$\sum_{i=1}^{\eta_1} f_i = \sum_{i=1}^{\eta_2} \hat{f}_i = N \quad (7)$$

and inequality $\eta_1 > \eta_2$. Thus:

$$\begin{aligned} E_{\text{indep}} - E_{\text{dep}} &= - \sum_{i=1}^{\eta_1} \frac{f_i}{N} \log_2 \frac{f_i}{N} + \sum_{i=1}^{\eta_2} \frac{\hat{f}_i}{N} \log_2 \frac{\hat{f}_i}{N} = \\ &= - \sum_{i=1}^{id} \frac{f_i}{N} \log_2 \frac{f_i}{N} - \sum_{j=id+1}^{\eta_1} \frac{f_j}{N} \log_2 \frac{f_j}{N} + \sum_{i=1}^{id} \frac{\hat{f}_i}{N} \log_2 \frac{\hat{f}_i}{N} + \sum_{j=id+1}^{\eta_2} \frac{\hat{f}_j}{N} \log_2 \frac{\hat{f}_j}{N} = \\ &= - \sum_{i=1}^{id} \frac{1}{N} \log_2 \frac{1}{N} - \sum_{i=id+1}^{\eta_1} \frac{1}{N} \log_2 \frac{1}{N} + \sum_{i=1}^{id} \frac{1}{N} \log_2 \frac{1}{N} + \sum_{j=id+1}^{\eta_2} \frac{\hat{f}_j}{N} \log_2 \frac{\hat{f}_j}{N} = \\ &= - \sum_{i=id+1}^{\eta_1} \frac{1}{N} \log_2 \frac{1}{N} + \sum_{j=id+1}^{\eta_2} \frac{\hat{f}_j}{N} \log_2 \frac{\hat{f}_j}{N} \end{aligned} \quad (8)$$

Equations (6) and (7) show that:

$$\sum_{j=id+1}^{\eta_2} \hat{f}_j = \eta_1 - id \quad (9)$$

Thus, the above equality can be written as:

$$\sum_{j=id+1}^{\eta_2} \left(\frac{\hat{f}_j}{N} \log_2 \frac{\hat{f}_j}{N} - \frac{\hat{f}_j}{N} \log_2 \frac{1}{N} \right) = \sum_{j=id+1}^{\eta_2} \frac{\hat{f}_j}{N} \left(\log_2 \frac{\hat{f}_j}{N} - \log_2 \frac{1}{N} \right) \quad (10)$$

Because $\forall j \in \{id+1, \dots, \eta_2\}$ then the inequality $\hat{f}_j > 1$ occurs and from monotonicity of the function $\log_2 x$ we get:

$$\sum_{j=id+1}^{\eta_2} \frac{\hat{f}_j}{N} \left(\log_2 \frac{\hat{f}_j}{N} - \log_2 \frac{1}{N} \right) > 0 \quad (11)$$

What ends the proof that $E_{indep} > E_{dep}$..

IV. CONCLUSION

The above evidence shows that for the configuration of diagrams with consistency rules, created by assigning similar names to diagram elements, the entropy value of the system decreases compared to the configuration with the same number of elements, but without consistency rules. Thus, the introduction of consistency rules (links between elements by assigning similar names to some elements) reduces the uncertainty of information, and consequently increases the orderliness and consistency of the whole model. Typically, this method of building software architecture is performed by experienced IT architects. In this article, therefore, a rationale for practices of experienced IT architects was presented.

REFERENCES

- [1] L. J. Thoms, "Thermodynamics ≠ Information Theory: Science's Greatest Sokal Affair," *Journal of Human Thermodynamics*, 8(1), pp. 1-120, 2012.
- [2] —, Unified Modeling Language, Object Management Group, <http://www.omg.org> (25.03.2019), 2015.
- [3] G. Spanoudakis, A. Zisman, "Inconsistency management in software engineering: survey and open research issues," *Handbook of Software Engineering and Knowledge Engineering*, World Scientific Publishing Co., Singapore, pp. 329-380, 2001.
- [4] R. Straeten, "Inconsistency Management in Model-Driven Engineering: An Approach Using Description Logics," Ph.D. dissertation, Vrije Universiteit Brussel, Belgium, 2005.
- [5] J. Ha, B. Kang, "Cross Checking Rules to Improve Consistency between UML Static Diagram and Dynamic Diagram," In *International Conference on Intelligent Data Engineering and Automated Learning - IDEAL 2008*, LNCS, vol. 5326, pp. 436-443, 2008.
- [6] P. G. Sapna, H. Mohanty, "Ensuring Consistency in Relational Repository of UML Models," In *10th International Conference on Information Technology*, pp. 217-222, 2007.
- [7] Y. Shinkawa, "Inter-Model Consistency in UML Based on CPN Formalism," In *13th Asia Pacific Software Engineering Conference*, 414-418, 2006.
- [8] D. Torre, Y. Labiche, M. Genero, "UML consistency rules: a systematic mapping study," In *18th International Conference on Evaluation and Assessment in Software Engineering*, 2014.
- [9] D. Torre, Y. Labiche, M. Genero, M. Elaasar, "A systematic identification of consistency rules for UML diagrams," *Carleton University*, 2015.
- [10] D. Allaki, M. Dahchour, A. En-Nouary, "A new taxonomy of inconsistencies in UML models with their detection methods for better," *International Journal of Computer Science and Applications*, Vol. 12 (No. 1), pp. 48-65, 2015.
- [11] W. Harrison, "An Entropy-Based Measure of Software Complexity," *IEEE Transactions on Software Engineering*, Volume 18, Issue 11, 812.
- [12] J. Bansiya, C. Davis, L. Etzkorn, "An entropy-based complexity measure for object-oriented designs," *Theory and Practice of Object Systems*, Volume 5, Issue 2, pp. 111-118, 1999.

94%

SIMILARITY INDEX

PRIMARY SOURCES

1	www.mdpi.com Internet	1576 words — 64%
2	sciforum.net Internet	212 words — 9%
3	Stanislaw Jerzy Niepostyn, Wiktor Bohdan Daszczuk. "Entropy as a Measure of Consistency in Software Architecture", Entropy, 2023 Crossref	197 words — 8%
4	www.researchgate.net Internet	123 words — 5%
5	link.springer.com Internet	39 words — 2%
6	www.mecs-press.org Internet	36 words — 1%
7	researchspace.auckland.ac.nz Internet	28 words — 1%
8	Malin Kallen, Sverker Holmgren, Ebba pora Hvannberg. "Impact of Code Refactoring Using Object-Oriented Methodology on a Scientific Computing Application", 2014 IEEE 14th International Working Conference on Source Code Analysis and Manipulation, 2014	27 words — 1%

9 Roland Kretschmer, Djamel Eddine Khelladi, Alexander Egyed. "Transforming abstract to concrete repairs with a generative approach of repair values", Journal of Systems and Software, 2021

26 words — 1%

Crossref

10 publikationen.ub.uni-frankfurt.de

Internet

18 words — 1%

11 etd.lib.metu.edu.tr

Internet

17 words — 1%

12 Damiano Torre. "Verifying the Consistency of UML Models", 2016 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), 2016

11 words — < 1%

Crossref

EXCLUDE QUOTES OFF

EXCLUDE BIBLIOGRAPHY OFF

EXCLUDE SOURCES OFF

EXCLUDE MATCHES OFF