Measurement of Ethical Issues in Software Products

Francesco Di Tria
francesco.ditria1@istruzione.it
Ministero dell'Istruzione – Italy

Abstract. Ethics is a research field that is obtaining more and more attention in Computer Science due to the proliferation of artificial intelligence software, machine learning algorithms, robot agents (like chatbot), and so on. Indeed, ethics research has produced till now a set of guidelines, such as ethical codes, to be followed by people involved in Computer Science. However, a little effort has been spent for producing formal requirements to be included in the design process of software able to act ethically with users. In the paper, we investigate those issues that make a software product ethical and propose a set of metrics devoted to quantitatively evaluate if a software product can be considered ethical or not.

Keywords. metrics; algor-ethics; evaluation; validation.

1 Introduction

Ethics in Computer Science is mainly focused on the definition of general guidelines and principles to be followed by people, as computer experts and software engineers, during the design, development, and experimentation processes [2, 7]. Also, groups and organizations issue ethical codes that bind their members [3, 4, 5, 6].

This problem has become emergent in those scientific areas, such as Artificial Intelligence [10], where machines are able to take autonomous decisions. To this end, on February 28th 2020, at the "Rome call for ethics", promoted by the Pontifical Academy for Life, a document that calls attention on the ethical use of artificial intelligence technologies was signed, together with the International Business Machines Corporation (IBM), Microsoft Corporation, and the Food and Agriculture Organization (FAO) [11]. In that document, for the first time appeared the neologism "Algor-Ethics", which is defined as "the ethical use of artificial intelligence according to the principles of *transparency*, *inclusion*, *responsibility*, *impartiality*, *reliability*, *security*, and *privacy*".

It is clear that, nowadays, ethical codes and guidelines do not suffice [22]. A no more valid assumption is that a software tool can be considered ethically valid if the people who designed and developed it followed ethical codes of the groups of membership. Now, software engineers must implement ethical codes directly in software products during the development life cycle, such that a product become ethical by itself (the so-called ethical software) [12].

So, if ethical software is produced, then there is the need of a rigorous verification too. For rigorous verification, we mean formal tools, such as metrics evaluation, that allow to validate the requirements of a product. It is worth noting that this usually happens in other Computer Science fields—such as security or user interface, to name a few—except for ethics, which is always left to the "good will" of the participants.

The contribution of the paper is an inquiry about the characteristics a product tool must possess in order to be considered an ethical software. Then, we examine a method for the verification of the presence of such characteristics, by proposing a set of metrics able to explicitly quantify "how much" a software product is ethical. Due to the novelty of the metrics, these have been subjected to both a theoretical and an empirical validation.

The paper is organized as follows. In Section 2, the basic terminology about metrics is reported in order to clarify the technical concepts used in the rest of the paper. Then, in Section 3, the framework adopted to verify the theoretical soundness of the proposed metrics is shortly explained. Section 4 starts the contribution of the paper with the conceptual background behind the proposed metrics, that are finally presented in Section 5. There, we report also the experimental process for validating the metrics along with our discussion. Section 6 concludes the paper with synthetic considerations that draw future works.

2 Terminology

In what follows, we introduce the main terms related to the measurement theory, in order to clarify some misunderstandings that usually happen in this field, such as the difference between *measure* and *metrics*.

Definition. *Physical quantity*: the attribute of a phenomenon or an entity that can be numerically expressed.□

Examples are body's *weight* and *temperature*. The weight and the temperature are attributes of a physical entity (the human body).

Definition. *Measure*: the value associated to a physical quantity.□

A measure is the result of a measurement process. For example, we may measure the body's temperature and observe that it is equal to 36.

Definition. *Unity of measure*: the standard for the measurement of physical quantities.□

The body's temperature must be expressed according to universal standards, as the measure by itself is not significant. The body's temperature is usually expressed according to the *Celsius degree* unity of measure.

Definition. *Measurement*: a set of operations and activities devoted to observe a measure.□

The measurement of the body's temperature is the observation of the value detected by a thermometer. So, the result of a measurement is a complete information, such as "the body's temperature is equal to 36 Celsius degree". In this context, the attribute and its measure, along with the unity of measure, are provided.

Definition. *Metrics*: the measurement method that aims at establishing quantitative information about an entity or a phenomenon.□

Metrics are also known as *indicators* because, in general, allow to answer to a question; for example: "how is my health state?". To answer this, I may use a *simple* metrics, that basically corresponds to the physical quantity (the temperature), whose measurement is the triple <*physical quantity, measure, unity of measure*>. Alternatively, I may use *composite* metrics that involve the computation of (complex) algebraic expressions. In this case, useful metrics can be "the difference of body's temperature in the last day" or "the mean temperature in the last week".

Example. "How many kilometers do I have to travel to get home from my current location?". To answer the question, I need to compute the following metrics: the number of kilometers to travel. This is an information that numerically extends my knowledge about the problem. To answer the metrics, I need to take a measurement. After the measurement process (by a map, for instance), the resulting measure is 15 Km, expressed according to the *meter* standard unity of measure. In this case, 15 is a numerical quantity that evaluates in Km the space between locations having precise geographical coordinates. That measure is also the value wanted by the metrics, for it provides quantitative information about the distance between my current location and home.□

To summarize, a metrics can be a simple physical quantity or a complex expressions. Both take a measurement, the second require also a computation. The measures are the values detected by the measurement.

Metrics differ by quality, that is, the information they provide have different meaning. Indeed, some values are not numerical and the relationships among numerical data is not so obvious. Therefore, metrics must be placed on a scale, that allows to understand the measurement data. Measurement is based on a scale structure, where each scale holds more properties than the previous.

The measurement scale is:

- *Nominal*. Metrics on this scale are classification labels, with no quantitative information. An example of nominal metrics is *gender*, which classifies people according to Male and Female classes.
- *Ordinal*. In this scale, there is an order among values. However, the difference among values is not given. Metrics on this scale measure non-numeric concepts, like *happiness*. So a level of happiness equals to 2 (that may corresponds to “happy”) is greater than a level of happiness equals to 1 (that may corresponds to “unhappy”), but the difference between 1 and 2 has no meaning.
- *Interval*. Metrics on this scale have both the properties of order and distance. Thanks to these, we can measure the differences among values. Addiction and subtraction between values are allowed. However, we still cannot compute ratios to compare values. An example of metrics on the interval scale is the temperature measured with the Celsius (C) unity of measure. If we have 10° C + 10° C, the result is 20° C. Surprisingly, 20° C is not the double of 10° C. The problem here is that there is not the concept of zero corresponding to “no temperature”.
- *Ratio*. In this scale, we have the concepts of order, distance, and an absolute zero. So, also multiplication and division between values are now allowed. Good examples of ratio metrics include height, weight, length, and duration.

3 Framework for metrics validation

The adopted framework is that explained In [1], which allows to verify the theoretical soundness of novel metrics. Here, we propose an excerpt of that validation theory.

A system is composed of a set of elements and a set of relationships among those elements, whereas elements and relationships are context-dependent. A module is a part of a system. The elements of the module may have outgoing relationships towards elements of the system; similarly, the elements of the module may present relationships coming from elements of the system. First of all, an *empty* module is that having no elements nor relationships. A module m_1 is *included* in another module m_2 if and only if (iff for short, hereinafter) all the elements of m_1 belong to m_2 and all the relationships of m_1 belong to m_2. The *union* of two modules m_1 and m_2 is the module composed of elements and relationships that belong to m_1 or m_2. The *intersection* of two modules m_1 and m_2 is the module composed of elements and relationships that belong to m_1 and m_2. Two modules are disjoint iff they present no elements nor relationships in common. A system is said to be *modular* iff it is partitioned in a set of disjoint modules.

A set of *measurement concepts* are introduced, namely Size (see Table 1), Length (see Table 2), Complexity (see Table 3). All of these have been demonstrated by the authors to be metrics on the ratio scale. Moreover, each of them is characterized by a set of properties that describe their intrinsic characteristics.

Table 1. Size's properties.

Size	
Property	**Description**
Non-negativity	The size of a system is non-negative.
Null value	The size of a system is a null value iff it does not present any element.
Module Additivity	The size of a system is equal to the sum of the sizes of its disjoint modules. Non-additivity happens when merging modules having elements in common. In this case, the size is lower than the sum of the sizes of the modules.

Table 2. Length's properties.

Length	
Property	**Description**
Non-negativity	The length of a system is non-negative.
Null value	The length of a system is a null value iff it does not present any element.
Non-increasing Monotonicity for Connected Components	If *S* is a system and *m* a module of *S* such that *m* is connected component of *S*, adding a relationship in *m* does not increase the length of *S*.
Non-decreasing Monotonicity for Non-connected Components	If *S* is a system and m_1 and m_2 are two modules of *S* such that m_1 and m_2 are separate connected components of *S*, adding a relationship from m_1 to m_2 does not decrease the length of *S*.
Disjoint Modules	If a systems *S* is composed of two disjoint modules m_1 and m_2, then the length of *S* is equal to the maximum between the lengths of m_1 and m_2.

Table 3. Complexity's properties.

Complexity	
Property	**Description**
Non-negativity	The complexity of a system is non-negative.
Null value	The complexity of a system is a null value iff it does not present any element.
Symmetry	The complexity of a system does not depend on the convention used for representing relationships between its elements.
Module Monotonicity	The complexity of a system is not lower than the sum of any two of its modules having no relationships in common.
Disjoint Module Additivity	The complexity of a system *S* composed of two disjoint modules

	m_1 and m_2 is equal to the sum of the complexity of m_1 and m_2.

In order to validate a new metrics, we have to check the metrics against the properties of the measurement concept of the framework. If properties hold, then the new metrics is coherent with that concept. For the sake of simplicity, a metrics having all the Size's properties is a Size metrics. If a new metrics, that to be validated, agrees with any of the measurement concepts, than that metrics can be safely considered on the ratio scale.

Further measurement concepts, namely Cohesion and Coupling, also introduced in the framework, are not considered here, because they are related to modular systems solely.

4 Conceptual analysis

Ethics in Computer Science is always related with security, since most of the security issues, such as privacy, anonymity, and copyright, involve ethical behavior of computer experts and software engineers. A famous distinction is made about hackers' definition, who can be classified as black hat, those acting in non ethical way, and white hat, those acting in ethical way. So, a little confusion and overlapping concepts arise between ethics and security.

We consider ethics as distinct from security. However, common concepts exist. Some examples are illustrated in Figure 1. This exemplification is not exhaustive.

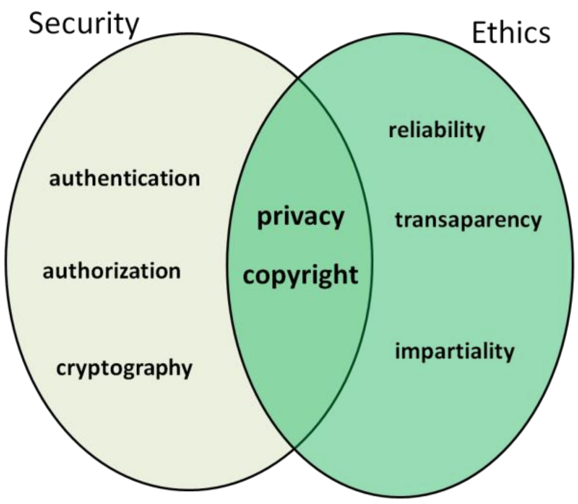

Figure 1. Classification of ethical software.

Of course, a software is secure if it respects all the security issues. In the same way, a software is ethical if it respects all the ethical issues. Indeed, if a software is secure, then it may be ethical or not ethical. Accordingly, if it is ethical, then it may be secure or not secure. So, four classes for software exist (see Table 4).

Table 4. Software classification

	Ethical	Not ethical
Secure	*A*	*B*
Not secure	*C*	*D*

- **Class A**. This class includes those products that are both secure and ethical. Examples are products that motivate the request for user data, protect sensitive data by cryptography, and allow users to delete collected data without having to provide any explanation.
- **Class B**. This class includes those products that are secure but not ethical. Examples are products that track user data in a hidden way and store sensitive data in protected area, where only they can access and read contents.
- **Class C**. This class includes those products that are not secure but ethical. Examples are products that motivate the request for user data and allow users to delete collected data without having to provide any explanation, but do not protect data by cryptography or transfer data as a clear text on the network.
- **Class D**. This class includes those products that are not secure nor ethical. Examples are products that open networks ports without the user's knowledge and allow a remote administration of a device. These are widely-known as back-doors. We can consider a software products in class D as an illegal tool.

4.1 Towards metrics identification

Due to a lack of standard in ethics evaluation, we discuss ethical software issues on the basis of the algor-ethics principles proposed in [11], namely *transparency*, *inclusion*, *responsibility*, *impartiality*, *reliability*, *security*, and *privacy*. Among these, we do not take into account the inclusion, security, and privacy issues. We focus only on the principles related to ethics (the light-green area in Figure 1). Indeed, inclusion is addressed in accessibility evaluation [23], as a part of the user-machine interaction research area. Also, security is a wide research area, provided with standards and protocols. In this area, well-known evaluation metrics already exist [13], usually applied in the so-called penetration tests. At last, privacy is classified in literature also as a security issue [24] and not only as an ethical problem.

The analysis of ethical software principles follows.

1. Transparency

    a. A system should not make decisions without user's permission. This principle of collaboration between users and machines is inspired to the *informed consent* extended to software [14]. This implies a disclosure of information that the system must provide to the users before and during the utilization. Mainly, it should be clear the purpose of the system and, eventually, the applied procedures. In case of risks for the user, for example privacy threats, the system have to present the information in a way the subjects can understand and have to guarantee that the user possess the basic competence to make a rational informed choice. This prevents that vulnerable groups of users may unconsciously suffer of unforeseen risks.

    b. A system should explain the results of elaborations. This means that the output data must be traceable. Today, a great concern is about the use of artificial intelligence, especially in critical fields as medicine or, to be more specific, in diagnosis by images. Neural networks are seen as black-boxes, unable in many cases to explain the results. In this context, If image analysis in medicine becomes totally, or largely, conducted by a neural network, which can establish the actions to be followed, then the doctor becomes a merely executor of the machine's decision, even without a complete level of comprehension. In case of error, who is guilty? Similarly, is a machine-made critical decision socially acceptable? For sure, the society must pretend to understand how a neural network works and how the results have been produced [15].

2. Responsibility

   a. A system should not perform other operations except those that are strictly necessary to conclude an assigned task. For example, saving data should be avoided, if it is not strictly necessary. Why should a cooking application access to our geographical coordinates? Or, why does a game require access to our camera or gallery? This requests, that we usually accept voluntarily without any reservation, hide a not declared purpose. Worst cases are when applications circumvent the security mechanism and still gain access to protected data, even without user acceptance [16].

3. Impartiality

   a. A system should not make critical decisions or produce different output on the basis of personal characteristics of users, such as gender [21] or race [18]. For example, in an insurance company or a bank, can an artificial intelligence algorithm trained on biased data be impartial when deciding whether granting an insurance [17]? An interesting case is discussed in [19]. The case is related to the health system, where the tool was designed to predict healthy needs on the basis of the costs of care. It came up that black people, having the same risk score as white, were much sicker. The underlying reason is that providers spent less resources for the care of black people, and then the algorithm classified them as non in need of care.

4. Reliability

   a. A system should not contribute to foster false information. To this end, a system must act in order to always guarantee trustworthiness. This is especially true in social networks, where non-certified information can be easily deployed and shared among users. Similarly, search engines may produce an arbitrarily-ordered list of search results, due to private interests in promoting certain sites. This is also encouraged by the small number of search engines that don't favor competition [20].

5 Metrics proposal

According to the defined ethical principles, we propose a set of metrics (see Table 5). For each metrics, we included the short name, the long name, its description, and the ethical issue of reference, as discussed above.

Table 5. Proposed metrics.

Name	Long name	Description	Issue
nic	non-informed-consent	Number of decisions made by a system without an informed consent.	Transparency 1.a
nod	non-traceable-data	Number of non-traceable output data.	Transparency 1.b
uno	unnecessary-operations	Number of unnecessary operations.	Responsibility 2.a
nir	non-independent-results	Number of non-independent results	Impartiality 3.a
nci	non-verified-information	Number of non-certified information	Reliability 4.a

All of these metrics are to be, then, normalized according to the number of tested processes. Let us consider the first metrics *nic*. If we observe *m* decisions made without an informed consent over a total of *n* tested decisions, then, for $n \in \mathbb{N}$ and $m = 0, \ldots, n$,

$$nic = \frac{m}{n} \in [0, 1]$$

Therefore, to be more precise, each metrics is a percentage that evaluates the degree (how much) a certain ethical principle is satisfied. It is worth noting that the greater the metrics, the greater is the degree of non-satisfaction.

The last composite metric is *gns*, as the global level of non-satisfaction of ethical principles, given by the arithmetic mean:

$$gns = \frac{nic + nod + uno + nir + nci}{5}$$

Accordingly, the ethics degree of a software product—or *eds*—can be finally computed as

$$eds = 1 - gns$$

5.1 Theoretical validation

Here, we present the theoretical validation of the novel metrics according to the validation framework in Section 3.

The system *S* is a software product. The elements are the processes of *S* and the relationships are the sequential links between processes, which form a task. For example, *S* = <*E*, *R*>, where *E* = {*a*, *b*, *c*, *d*, *e*, *f*} and *R* = {(*a*, *b*), (*b*, *c*), (*d*, *e*), (*e*, *f*)}. We say that there are two main tasks: the first is composed by the processes *a*, *b*, *c*, in that order; the second by the processes *d*, *e*, *f*, in that order.

All the proposed metrics—namely, *nic*, *nod*, *uno*, *nir*, *nci*— agree only with the Size metrics (see Table 6). For this reason, they can be placed on the ratio scale. Also, we note that all further metrics, eventually obtained by composition of these, can be placed on the ratio scale.

Table 6. Validation of the proposed metrics.

	Property	**Metrics**				
		nic	***nod***	***uno***	***nir***	***nci***
Common properties	Non-negativity	X	X	X	X	X
	Null value	X	X	X	X	X
Size	Module Additivity	X	X	X	X	X
Length	Non-increasing Monotonicity for Connected Components					
	Non-decreasing Monotonicity for Non-connected Components	X	X	X	X	X
	Disjoint Modules					
Complexity	Symmetry					
	Module Monotonicity	X	X	X	X	X
	Disjoint Module Additivity	X	X	X	X	X

Now, we discuss the validation process for the *nic* metrics. The same rationale can be repeated for the other metrics, therefore it is not reported. We recap that *nic* stands for "the number of decisions made by the system without an informed consent". As discussed below, *nic* is a Size metrics, for it verifies all the Size's properties, but it is not a Length metrics nor a Complexity metrics.

Common: Non-negativity. The property is verified. The *nic* of a system cannot be negative, because the minimum value is 0, when all the decisions are based on informed consent.

Common: Null value. The property is verified. The *nic* of a system is null when there are no processes.

Size: Module Additivity. The property is verified. When merging modules that do not have elements in common, we expect *nic* to be additive, that is the *nic* of the composite module is equal to the sum of the *nic* of each disjoint module. The *nic* is lower than sum of the *nic* of each module, in case of non-disjoint modules.

Length: Non-increasing Monotonicity for Connected Components. This property is not verified because adding a sequence in a module can increase the *nic* of a system.

Length: Non-decreasing Monotonicity for Non-connected Components. The property is verified because adding a sequence from a process of a module to a process of another module does not decrease the *nic* of a system.

Length: Disjoint Modules. The property is not verified because the *nic* of the composite module is not given by the maximum of the *nic* of each module.

Complexity: Symmetry. The property is not verified, because the sequential relationship is not symmetric, for <*a*, *b*> means that process *a* is followed by *b*, while <*b*, *a*> means the opposite.

Complexity: Module Monotonicity. The *nic* of a system *S* is not lower than the sum of the *nic* of any two of its modules with no relationships in common. Then, the property is verified.

Complexity: Disjoint Module Additivity. The property is verified because the *nic* of a system composed of two disjoint modules is given by the sum of the *nic* of the two modules.

5.2 Empirical validation

The empirical validation is an application of the metrics in a laboratory experiment in order to show the trend of the measurement process. In this case, we tested the composite metrics *gns* for the measurement of global level of non-satisfaction of ethical principles. This metrics, that is based on the arithmetic mean, has been compared against the same composite metrics, computed by the geometric mean.

For the experimental set, we generated random values n_1, n_2, n_3, n_4, and n_5 as the number of the total tested processes for, respectively, *nic*, *nod*, *uno*, *nir*, and *nci*. We recall that these random values are necessary in order to normalize the metrics in the range [0, 1].

Then, we made a set of observations of 5 variables m_1, m_2, m_3, m_4, and m_5 as the number of the positive processes for, respectively, *nic*, *nod*, *uno*, *nir*, and *nci*. With "positive process" we mean a process that did not pass the test. Now, for $i = 1, \ldots, 5$, we varied m_i in the range $[1, n_i]$, by incrementing by 1 a variable at a time.

The result of the empirical validation is depicted in Figure 2. The first thing that appears evident is the not constant trend of the geometric function. Indeed, the geometric mean tend to privilege low values and, then, it increases in a non linear way, as it happens for the arithmetic mean. The linear increase of the function is a good quality for the metrics where each positive case must have the same weight.

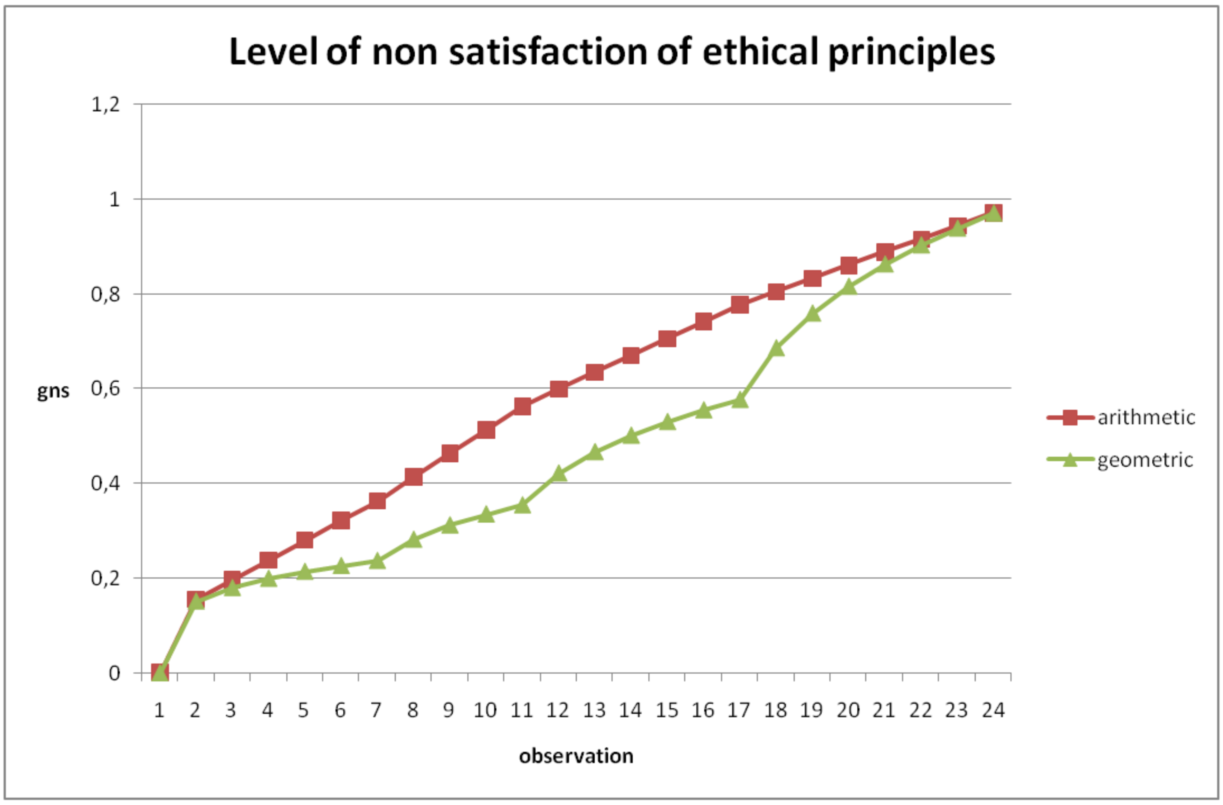

Figure 2. Empirical validation.

6 Conclusion

The use of software products encapsulating specific ethical principles must replace tools designed and developed according to traditional lifecycles: those that aim to pass usability and security tests, for example. Now, there is a call for new professional figures devoted to formally test the tools according to their “intrinsic behavior”. These professionals should apply rigorous validation methods that produce significant and understandable reports, in order certify software’s properties. This is what generally happen when security experts proceed with the penetration tests. Why is the same not actually conducted for ethical issues?

In the paper, we proposed a set of metrics that quantify “how much” a software is ethical. Since the metrics have been theoretically and empirically validated, we deem that can be used for such rigorous tests, aiming at declaring whether a software can be considerate ethical or not. Current research is devoted to introduce further useful metrics that catch other ethical issues, not included in the paper.

References

1. Briand, L. C., Morasca, S., & Basili, V. R. (1996). Property-based software engineering measurement. *IEEE transactions on software Engineering*, *22*(1), 68-86.
2. Singer, J., & Norman, G. V. (2002). Ethical issues in empirical studies of software engineering. *IEEE Transactions on Software Engineering*, 28(12), 1171-1180.
3. ACM Executive Council, ACM Code of Ethics and Professional Conduct, Communications of the ACM, Vol. 36, No. 2, 1993, pp. 99-105

4. Anderson, R. E., Johnson, D. G., Gotterbarn, D., & Perrolle, J. (1993). Using the New ACM Code of Ethics in Decision Making, Communications of the ACM, Vol. 36, No. 2, pp. 98-107.
5. IEEE Board of Directors. IEEE Code of Ethics, IEEE, 1990.
6. IEEE-CS/ACM Joint Task Force on Software Engineering Ethics and Professional Practices. Software Engineering Code of Ethics and Professional Practice, 1998.
7. D. Gotterbarn, K. Miller, & S. Rogerson, Software Engineering Code of Ethics is Approved, Communications of the ACM, Vol. 42, No. 10, 1999, pp. 102-107.
8. Johnson, Deborah G. Computer ethics. *The Blackwell guide to the philosophy of computing and information* (2004): 65-75.
9. Johnson, Deborah G. Computing ethics Computer experts: guns-for-hire or professionals?. *Communications of the ACM* 51.10 (2008): 24-26.
10. Smuha, N. A. (2019). The EU Approach to Ethics Guidelines for Trustworthy Artificial Intelligence. *CRi-Computer Law Review International*.
11. https://romecall.org/. Accessed on May 2020.
12. Karim, N. S. A., Al Ammar, F., & Aziz, R. (2017, September). Ethical software: Integrating code of ethics into software development life cycle. In *2017 International Conference on Computer and Applications (ICCA)*, pp. 290-298. IEEE.
13. Al-Shiha, Reem & Alghowinem, Sharifa. (2019). Security Metrics for Ethical Hacking: Proceedings of the 2018 Computing Conference, Volume 2. 10.1007/978-3-030-01177-2_83.
14. Miller, K. (1998). Software informed consent: docete emptorem, not caveat emptor. *Science and Engineering Ethics*, *4*(3), 357-362.
15. Brady, A. P., & Neri, E. (2020). Artificial Intelligence in Radiology—Ethical Considerations. *Diagnostics*, *10*(4), 231.
16. Reardon, J., Feal, Á., Wijesekera, P., On, A. E. B., Vallina-Rodriguez, N., & Egelman, S. (2019). 50 Ways to Leak Your Data: An Exploration of Apps' Circumvention of the Android Permissions System. In *28th {USENIX} Security Symposium ({USENIX} Security 19)*, pp. 603-620.
17. Garcia, M. (2016). Racist in the machine: The disturbing implications of algorithmic bias. *World Policy Journal*, *33*(4), 111-117.
18. Sandvig, C., Hamilton, K., Karahalios, K., & Langbort, C. (2015, May). Can an algorithm be unethical. In *65th annual meeting of the International Communication Association*.
19. Benjamin, R. (2019). Assessing risk, automating racism. *Science*, *366*(6464), 421-422.
20. Noble, S. U. (2018). *Algorithms of oppression: How search engines reinforce racism*. nyu Press.
21. Zou, J., & Schiebinger, L. (2018). AI can be sexist and racist—it's time to make it fair.
22. Rehg, W. (2015). Discourse ethics for computer ethics: a heuristic for engaged dialogical reflection. *Ethics and Information Technology*, *17*(1), 27-39.
23. Petrie, H., & Bevan, N. (2009). The Evaluation of Accessibility, Usability, and User Experience. The universal access handbook, 1, 1-16.

24. Abi Sen, A. A., & Basahel, A. M. (2019, March). A Comparative Study between Security and Privacy. In 2019 6th International Conference on Computing for Sustainable Global Development (INDIACom), pp. 1282-1286. IEEE.