Towards Validating Risk Indicators Based on Measurement Theory

Ayşe Moralı
Distributed and Embedded Security Group
University of Twente
Enschede, The Netherlands
ayse.morali@utwente.nl

Roel Wieringa
Information Systems Group
University of Twente
Enschede, The Netherlands
roel.wieringa@utwente.nl

***Abstract*—Due to the lack of quantitative information and for cost-efficiency purpose, most risk assessment methods use partially ordered values (e.g. high, medium, low) as risk indicators. In practice it is common to validate risk scales by asking stakeholders whether they make sense. This way of validation is subjective, thus error prone. If the metrics are wrong (not meaningful), then they may lead system owners to distribute security investments inefficiently. Therefore, when validating risk assessment methods it is important to validate the meaningfulness of the risk scales that they use. In this paper we investigate how to validate the meaningfulness of risk indicators based on measurement theory. Furthermore, to analyze the applicability of measurement theory to risk indicators, we analyze the indicators used by a particular risk assessment method specially developed for assessing confidentiality risks in networks of organizations.**

***Index Terms*—security; risk assessment; measurement**

I. Introduction

IT risk metrics are designed to monitor, analyze and manage risk related attributes of system entities, e.g. impact of unauthorized disclosure of a certain information asset. Risk experts commonly measure these attributes with partially ordered values (e.g. high, medium, low). This is due to the unavailability of sufficient and complete event data and also to the limited availability of resources (time, money) to do the risk assessment. To determine risk, risk assessment (RA) methods aggregate these values with *merge operators*. A merge operator defines a qualitative ordering over each possible measurement combination. We furthermore call the values aggregated with merge operators *risk indicators*. Risk indicators are also partially ordered values.

In practice, risk indicators are validated by asking stakeholders' commitment. This way of validation is based on subjective opinions of stakeholders, but not on analysis of meaningfulness of the indicators. If risk indicators are not meaningful, they may mislead system owners at their security investment decisions. For instance, in an extended enterprise this may mean over investing in service level agreements or obtaining a contract that provides a lower security level than the system requires. Therefore, it is important to validate the meaningfulness of the risk indicators. This means testing first the meaningfulness of each indicator and then the meaningfulness of each merge operator.

Testing the construct validity and aggregation validity of checklist-based methods is usually trivial. However, as the complexity of an RA method increases, the number of risk indicators and merge operations increases as well. Consequently, validating the meaningfulness of the risk indicators that sophisticated RA methods use becomes challenging.

The goal of this paper is to use measurement [10] theory investigate how to validate the meaningfulness of risk indicators and merge operators. In particular, the following two questions will be investigated: (Q1) Is measurement theory applicable to such a resource-constrained critical field such as RA? and (Q2) Is measurement theory based validation of risk indicators repeatable?

In this paper we present a road map for validating the meaningfulness of risk indicators. We furthermore analyze the applicability of the measurement theory to risk indicators by testing the indicator used by the confidentiality risk assessment method CRAC++ [9].

Although meaningfulness of software measures is analyzed by many researchers, e.g. [2], [4]–[6], analyzing the validity of IT security risk measures is relatively new. Research in this field focuses on how to define meaningfully security measures, e.g. [5], and how to test their relevance for IT security risk, e.g. [7], rather than how to validate the measures proposed by available methods. To the best of our knowledge this is the first approach that aims to validate the meaningfulness of IT security risk indicators based on measurement theory.

II. Measurement Theory

Most of these concepts of measurement theory are introduced by Stevens [10] and used in software engineering context first by Baker et al. [1].

A *metric* is a value that results from measuring a certain attribute of an entity being investigated. From here on we refer to the value as *measurement*. It facilitates non-subjective decision making and improves performance through time efficient mapping of value to scales.

An *entity* is a logical or physical object, such as an application or a work station, or an event, such as an incident [5].

This research is supported by the research program Sentinels (http://www.sentinels.nl). Sentinels is being financed by Technology Foundation STW, the Netherlands Organization for Scientific Research (NWO), and the Dutch Ministry of Economic Affairs.

An *attribute* on the other hand is a feature of an entity, such as the impact of an incident. By assigning values to attributes one has to consider the representation theory of measurement. This means that the assignment of values must correspond to all empirical relations about the entities and their attribute.

Depending on the empirical relations among the measurements Stevens [10] classifies them in four main *scale types*, i.e. nominal scale, ordinal scale, interval scale, and ratio scale. Scale types define the arithmetic operations one can use to analyze values that are mapped to certain scales (Table I).

TABLE I
CLASSES OF MEASUREMENT SCALES.

Scale Type	**preserve order**	**magnitude**	**admissible arithmetic operations**
Nominal scale	no	no	none
Ordinal scale	yes	monotonic increasing	none
Interval scale	yes	positive linear	+,-
Ratio scale	yes	positive geometric	+,-,*,/

A *nominal scale* defines a classification schema that consist of mutually exclusive and jointly exhaustive categories. There is no order among these categories, therefore none of the possible arithmetic operations are applicable to the measurements from this scale. This type of scale can be used for instance to categorize attackers according to their capabilities.

An *ordinal scale* allows arranging the variable categories in sequences with a monotonically increasing magnitude. However there is no precise information on the magnitude of the differences between these variables. Therefore, arithmetic operations are still not applicable to attributes belonging to this scale. Most of the risk measurements, such as criticality of an IT-component classified as marginal–minor–moderate–catastrophic, belong to this scale.

An *interval scale* extends the ordinal scale by indicating the exact differences. The magnitude of the difference among the variables of this scale is positive linear. Accordingly, addition and subtraction are permitted arithmetic operation among the variables. Risk indicators for which quantitative information is available (such as number of IT-components and number of instances of an information asset) belong to this scale.

A *ratio scale* extends the interval scale by allowing to capture positive and negative values. The magnitude between the variable categories of this scale are positive geometric. Thus, values that belong to this scale type can be transformed by multiplying each value by a constant. All arithmetic operations are permitted among the variables of this scale.

Herrmann [5] differentiates among three types of measurements: ratio, proportion, and percentage. A *ratio* measurement (different from ratio scale) compares two quantities that are from distinct populations. If they are from the same population then the measurement is of type *proportion*. *Percentage* converts proportion to terms of per hundred units.

Herrmann [5] argues that to prevent subjectivity all measurements should be collected according to the same quantification process. Fenton and Pfleeger [3] differentiate between two kinds of quantifications processes: direct quantification and calculation.

Direct quantification assigns values to attributes of entities directly, e.g. quantifying the number of incidents. In this way it maps the *facts* of the empirical world to the *indicators* of the mathematical world. Whereas, *calculation* is an indirect quantification, which involves aggregating several attributes of an entity, to gain new knowledge on that entity. For instance to provide additional knowledge "expected impact of an incident" on entity "impact", one may aggregate the penalties to be paid, recovery costs and image loss.

Fenton and Pleeger [3] argue that a statement involving measurement is *meaningful* if its represents some real world phenomena and doe not change after applying any admissible transformations. Kitchenham et al. [6] elaborate on this statement by arguing that for a measurement to be meaningful the correlation among the facts should be preserved also among the indicators (construct validity), and no other operations of scales than the permitted ones should be performed on indicators (aggregation validity).

does a measurement has a truth value? It is not a proposition. I would say that a measurement is an object in the mathematical world (e.g. a number, or a symbol) that represents a phenomenon (e.g. an attribute of an entity) in the real world. The requirement on calculations seems to be that the output of a calculation must be a mathematical objects that represents some relevant real world phenomenon.

Furthermore, there are four key characteristics that a construct should obey to be meaningful: accuracy; precision; completeness; and correctness. *Accuracy* shows how well individual or average measurements agree with a true value. Repeatability of accurate measures are captured with *precision*. In other words, precision shows how well identically performed measurements agree with each other, or how small the variance is. Precision analyzes the variabilities in the test environment, such as human error or not paying attention to detail. *Completeness* analyzes whether a measurement measures what it is intended to measure. It depends on the completeness and coherence of the measurement process. Finally, *correctness* analyzes if the data is collected exactly as it is described by the method. The more formal a measurement process is defined the more correct results it delivers.

III. VALIDATION ROAD MAP

In this section we present a road map for validating the meaningfulness of the risk indicators of complex RA methods. In the following we describe the activities of the steps that the validation road map consists of.

a) Step 1: Eliciting information: The aim of this step is to elicit a list of entities, a list of attributes of these entities (risk indicators) that the RA method measures, and the merge operators with which the RA method "calculates" some of the indicators.

b) Step 2: Analyzing Construct Validity: The aim of this step is to analyze the meaningfulness of risk indicators that are identified in step 1. In this step we first determine to which scale type (i.e. nominal scale, ordinal scale, interval

scale and ratio scale) each indicator belongs, then analyze how well each risk indicator satisfies the key characteristics of meaningful measurements (i.e. accuracy, precision, validity and correctness).

c) Step 3: Analyzing Aggregation Validity: Finally we analyze the meaningfulness of the merge operations that are identified in Step 1. This means we identify possible arithmetic operations each merge operator is allowed to use based on the scale of the measurements that are aggregated.

Commonly risk is presented as a single value that is aggregated of two entities, e.g. likelihood and impact. For instance if two constructs are impact and likelihood with respective construct measures high and medium, then applying a merge operator to combine these two measures may define the measurement of risk construct as high. A merge operator is necessary because it is not possible to combine qualitative values, with arithmetical functions. There is a partial order between qualitative values of each construct measurement. This order needs to be respected by defining merge operators. Accordingly by comparing two risk values $r_1 = (l_1, i_1)$ and $r_2 = (l_2, i_2)$, we can say that r_1 is superior to r_2 if and only if $l_1 \geq l_2$ and $i_1 \geq i_2$.

IV. Application

In this section we demonstrate how the three steps can be applied to CRAC++ method.

A. Eliciting information from CRAC++

Based on the formalizations of CRAC++ we elicited 11 risk indicators (as presented in Table II) and two merge operations.

TABLE II
Risk indicators of CRAC++

Indicator ID	Attribute	Entity
I01	confidentiality value	information asset
I02	number of instances	information asset
I03	homogeneity	information asset
I04	number of capabilities	threat agent
I05	number of conditions	vulnerability
I06	level of mitigation	countermeasures
I07	impact	information asset
I08	total impact	IT component
I09	ease of exploiting vulnerabilities	vulnerability
I10	ease of accessing one component	threat agent
I11	protection level of a component	IT component

Indicators I01, I02, I03, I04, I05 and I06 are direct quantifications. I01, *confidentiality value*, indicates the business criticality of entity information asset from the perspective of the system owner. I02, *number of instances*, is the attribute of entity information asset and measures the number of instances that belong to an information asset, e.g. if client data is an information asset, and there are information records that belong to 100 clients, then a number of instances of client data is 100. I03, *homogeneity*, reflects the presence of a correlation between the numbers of instances of an information asset that gets disclosed by an incident and the impact of the disclosure. For instance, "social security numbers" are homogeneous, since the damage due to the loss of one hundred social security numbers is larger than the damage due to the loss of a single social security number. Conversely, an information asset is nonhomogeneous if the damage due to the disclosure of one instance is as big as the damage of the disclosure of all instances. For instance, if the login credentials of one user get disclosed, the damage to the company is basically the same as if the credentials of 100 users with equal access rights would be disclosed. I04, *Number of capabilities*, is the number of necessary attributes (out of a predefined set of attributes, e.g. hacking skills and physical access) that a threat agent is estimated to have. I05, *number of conditions*, is the number of necessary attributes (from the same set of attributes as in capabilities) to exploit a certain vulnerability of IT architectural components. I06, *level of mitigation*, is the level indicating how well a countermeasure mitigates a vulnerability.

Impact (I07) and total impact (I08) are indirect quantifications (calculations) that are achieved by applying respectively $\odot$ merge operation on I01 and I02, and $\oplus$ merge operation on two distinct variables of I07. I07, *impact*, is the loss related with unauthorized disclosure of instances of one information asset on an IT component. I08, *total impact*, indicates the impact of accessing all information assets available on an IT component.

Finally, risk indicators I09, I10 and I11 are direct quantifications that are measured in relation to other measurements. The measurement type of I09 and I10 is proportion, and the measurement type of I11 is ratio. I09, *ease of exploiting vulnerabilities*, indicates the ease of a potential attacker exploiting a vulnerability. I10, *ease of accessing one component*, is the likelihood of a threat agent disclosing information assets on one component exploiting the easiest vulnerability. Finally, I11, *protection level of a component* is how well an IT component is protected against the easiest unauthorized access.

B. Analyzing Construct validity of CRAC++

To analyze the construct validity of the CRAC++ method we determine the scale type that each risk indicator belongs to.

If quantitative information is not available, then CRAC++ measures *confidentiality value* in terms of ordered variables (e.g. *high - medium - low*). These values are determined based on the relative business criticality of information assets. So, they are used to rank the criticality but do not indicate the size of the interval between variables. Assuming that information asset i_1 is more business critical than information asset i_3. CRAC++ models this relation by representing the confidentiality value of information asset i_1 with ordinal variable *high* and the confidentiality value of information asset i_3 with the ordinal variable *medium*. Since the confidentiality value preserves the relation that is observed among the facts (more business critical assets have higher value and vise versa) we claim that the confidentiality value is a meaningful construct.

In the ideal case (when risk assessor has complete knowledge on the information assets) the *number of instances* belong to the interval scale. For cost-efficiency reasons we measure it with the ordinal scale.

Homogeneity consists of two mutually exclusive and jointly exhaustive categories, *homogenous* and *nonhomogenous*. Therefore, we say that it is of type nominal scale.

Capability and conditions are of nominal scale. Accordingly, both *number of capabilities* and *number of conditions* are from interval scale.

CRAC++ estimates the *level of mitigation* with ordinal scale values (e.g. *1/3 – 2/3 – 3/3*). If there is a certain mitigation mechanism in place that is recommended in best practices then the mitigation level is $1/3$, if there is a mitigation mechanism that mitigates worse than the best practices recommend then the mitigation level is $2/3$, other wise (no mitigation mechanism in place) $3/3$. The order of the entities reflect the empirical mitigation level of the security mechanism but does not preserve the size of the intervals.

CRAC++ calculates *impact* by aggravating confidentiality value, number of instances and homogeneity. Impact belongs to ordinal scale.

Total impact builds on impact. Since impact is of the ordinal scale, and total impact is calculated by aggregating the impact variables, the variables of the total impact are also of the ordinal scale.

The CRAC++ method estimates *ease of exploiting vulnerabilities* by calculating the *ratio* of number of capabilities the attacker has over the number of capabilities required for exploiting the vulnerability. There is an AND relation among the owned capabilities and necessary capabilities. Furthermore, there is a positive geometric magnitude between the variables of ease of exploiting vulnerabilities. The ease increases as the ratio of owned to necessary capabilities increases. Thus, ease of exploiting a vulnerability is of type ratio scale.

A further risk indicator is the *ease of accessing one component*. This indicator belongs to the scale type ratio.

The last indicator that we analyze here is the *protection level of a component*. This indicator is calculated by comparing ease of accessing components for different threat agents. Since ease of accessing one component belongs to scale type ratio, so does this indicator.

We summarize these findings of Step 2 in Table III.

TABLE III
SCALES TYPES OF THE RISK INDICATORS OF CRAC++.

Risk Indicator	Scale type	possible values
I01: confidentiality value	ordinal scale	high–medium–low
I02: number of instances	ordinal scale	all–single–none
I03: homogeneity	nominal scale	homogeneous–nonhomogeneous
I04: number of capabilities	interval scale	natural numbers
I05: number of conditions	interval scale	natural numbers
I06: level of mitigation	ordinal scale	1/3–2/3–3/3
I07: impact	ordinal scale	null, high, medium, low, very high
I08: total impact	ordinal scale	null, high, medium, low, very high
I09: ease of exploiting vulnerabilities	ratio scale	positive rational numbers
I10: ease of accessing one component	ratio scale	positive rational numbers
I11: protection level of a component	ratio scale	positive rational numbers

Next we analyze how well each risk indicator satisfies the four key characteristics (i.e. accuracy, precision, validity and correctness).

All risk indicators, accept confidentiality value (I01) and level of mitigation (I06), obey the accuracy property. If quantitative information on I01 and I06 was available, these indicators would also obey the accuracy property. Here, to minimize the possibility of disagreement, one may use coarse-grained variables.

There is space for human error only by determining (I10) the ease of accessing one component. I10 is determined by comparing the ease of alternative attack propagation paths, which are formed based on physical and logical connections among IT components. Here, if a logical connection is overlooked this may lead to missing a propagation path and consequently achieving a wrong variable. Thus, I10 does not satisfy the precision goal.

Only I07 and I08, impact and total impact, do not satisfy the completeness goal. This is due to the trade-off between cost of assessment and completeness. The more complete a method is the longer it takes to conduct it. CRAC++ addresses this trade-off by considering only the confidentiality value of the information assets and ignoring other measures that affect the impact, e.g. loss of image or penalties to be paid. Total impact is affected by this optimization, because it is aggregated based on impact.

All of the indicators of CRAC++ satisfy the correctness goal. This is due to the high level of formalization CRAC++ provides.

C. Analyzing Aggregation validity of CRAC++

In the following we discuss the meaningfulness of the merge operators that are elicited at step 1. These operations formalize the constructs validated at step 2. Due to page limitations we can not present the formalizations here but the interested reader may find them in the technical report [8]. Such formalizations aim to facilitate testing the aggradation validity.

CRAC++ defines $\odot$ operator to determine the *impact*. $\odot$ aggregates the number of instances and the confidentiality values of the information asset, and formalizes impact. Assuming that the entities of confidentiality value are *high – medium – low* and the entities of number of instances are *all – single – non*, the $\odot$ operator behaves as shown in Table IV.

TABLE IV
BEHAVIOR OF THE $\odot$ OPERATOR.

$\odot$		Number of instances		
		all	**single**	**none**
Conf. value	**high**	very-high	high	null
	medium	high	medium	null
	low	medium	low	null

We determined in step 2 that the variables of both confidentiality values and number of instances are of ordinal scale. To recall from Section II the ordinal scale preserves order and is monotonic increasing. Since the $\odot$ operator satisfies these properties as well, we claim $\odot$ is aggregation valid.

CRAC++ aggregates *total impact* by incrementally applying the $\oplus$ operator on impact values until all available information assets are considered. The behavior of $\oplus$ operator with the impact values from Table IV is presented in Table V. This operation formalizes the calculation of total impact.

TABLE V
BEHAVIOR OF THE $\oplus$ OPERATOR.

$\oplus$		**Impact**				
		very-high	**high**	**medium**	**low**	**null**
Impact	**very-high**	very-high	very-high	very-high	very-high	very-high
	high	very-high	very-high	high	high	high
	medium	very-high	high	high	medium	medium
	low	very-high	high	medium	medium	low
	null	very-high	high	medium	low	null

The scale class of impact is ordinal and this scale preserves order and is monotonic increasing. Since the $\oplus$ operator satisfies these properties as well, we claim that $\oplus$ is aggregation valid.

CRAC++ calculates *ease of accessing one component* by first multiplying the ease of exploiting vulnerabilities of the component with the level of mitigation estimation, and then applying the MAX operator on the achieved measurements. The first aggregation is valid because both of the involved constructs are of ratio scale and at this scale it is possible to multiply the entities. The second aggregation is also valid because it is applicable to variables with positive linear magnitude, and the magnitude of ratio scale extends positive linear magnitude.

A further construct that is calculated by applying the MIN/MAX operator is the *protection level of a component*. It is calculated by applying the MIN operator on the elements of the set containing measurements of ease of accessing the component for all threat agents. Since ratio scale extends ordinal scale and monotonic increasing magnitude is a property of ordinal scale, this operation is also valid.

V. CONCLUSION

In order to avoid potentially expensive mistakes in risk assessment, the meaningfulness of risk indicators as well as merge operators need to be validated, based on measurement theory. In this paper we adapt approaches from the field of software engineering to IT security RA, and present a systematic method for validating model based RA methods.

Our analysis shows that measurement theory is applicable to RA (Q1). However, due to unavailability of quantitative values for some of the risk measurements, some subjectivity may be involved in assigning values. We do not intend to address this aspect of RA here, because although many RA methods can work with quantitative values, resource constraints ensure that these methods are not used in practice. To prevent inconsistency such methods usually define rules for assigning values. We tested our validation method on CRAC++ using measurement theory. Since the method provided formalizations of the risk indicators, we could apply the method easily. To answer Q2 (repeatability) we plan to apply measurement theory to other methods that use qualitative values as well.

REFERENCES

[1] A.L. Baker, J.M. Bieman, N. Fenton, D.A. Gustafson, A. Melton, and R. Whitty. A philosophy for software measurement. *J. Syst. Softw.*, 12(3):277–281, 1990.

[2] L. Briand, K.E. Emam, and S. Morasca. On the application of measurement theory in software engineering. *Empirical Software Engineering*, 1:61–88, 1996. 10.1007/BF00125812.

[3] N.E. Fenton and S.L. Pfleeger. *Software metrics: a rigorous and practical approach*. International Thomson Computer Press, 2 edition edition, 1996.

[4] R. Harrison, S.J. Counsell, and R.V. Nithi. An evaluation of the mood set of object-oriented software metrics. *Software Engineering, IEEE Transactions on*, 24(6):491–496, 1998.

[5] D.S. Herrmann. *Complete Guide to Security and Privacy Metrics: Measuring Regulatory Compliance, Operational Resilience, and ROI*. Auerbach Publications, 2007.

[6] B. Kitchenham, S.L. Pfleeger, and N.E. Fenton. Towards a framework for software measurement validation. *Software Engineering, IEEE Transactions on*, 21(12):929 –944, 1995.

[7] N. Mayer, E. Dubois, R. Matulevicius, and P. Heymans. Towards a measurement framework for security risk management. In *Proceedings of the Workshop on Modeling Security (MODSEC08)*, pages 151–160. IEEE Computer Society, 2008.

[8] A. Morali and R. J. Wieringa. Risk-based confidentiality requirements specification for outsourced it systems (extended version). Technical Report TR-CTIT-10-09, 2010.

[9] A. Morali and R.J. Wieringa. Risk-based confidentiality requirements specification for outsourced it systems. In *Proceedings of the 18th IEEE International Requirements Engineering Conference (RE 2010), Sydney, Australia*. IEEE Computer Society, 2010.

[10] S.S. Stevens. On the theory of scales of measurement. *Science*, 103(2684):677–680, 1946.