BUILDING SECURITY ASSURANCE IN OPEN INFRASTRUCTURES

SECURITY ASSURANCE TAXONOMY AND METHODOLOGY

ABSTRACT:
In order to be able to measure, document and maintain security assurance of telecommunications services, it is necessary to introduce a new general methodology. By discussing the insufficiency of the existing related methodologies, in this approach we first develop a general methodology introducing known related issues and necessary general steps. From this general methodology we derive a six steps operational methodology that can be practically applied in particular by big telecom operators and service providers to gain confidence in the security deployed to protect their service and the associated business revenues.
KEYWORDS: SECURITY ASSURANCE, SECURITY METRIC, SECURITY ASSURANCE PROCESS, AGGREGATION

TABLE OF CONTENT

1 EXECUTIVE SUMMARY

In order to be able to measure, document and maintain security assurance of telecommunications services, it is necessary to introduce a new general methodology. By discussing the insufficiency of the existing related methodologies, we first develop a general methodology introducing known related issues and necessary general steps. From this general methodology we derive an operational methodology that can be practically applied in particular by major telecom operators and service providers to gain confidence in the security deployed to protect their service and the associated business revenues.

This operational methodology, which represents the core of this document, is composed of six main steps. **Each step is addressing specific action(s) in order to build the expected security assurance**:

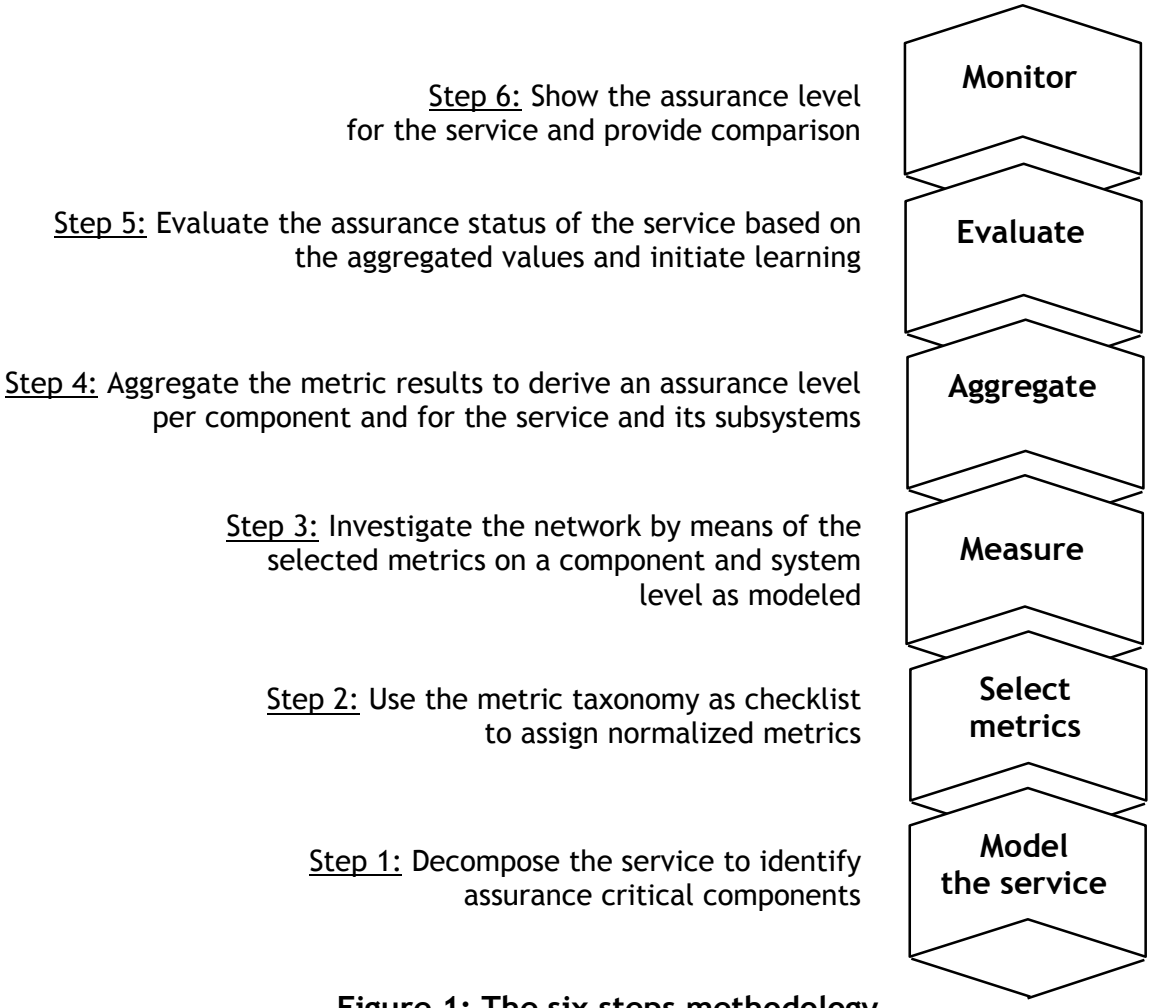


Figure 1: The six steps methodology

The first step concerns **service modelling**. This step is crucial to reflect the assurance needs of the operated services. The modelling allows decomposing the service in order to identify assurance critical components.

The second step addresses the **selection of metrics**. The selection of metrics defines the profile of the estimated security assurance. It requires pertinent choices, especially in selecting the targeted assurance levels of metrics. A low level will provide less confidence but will be easier to implement while a high level of assurance will lead to

higher confidence but will imply a higher effort in deploying, monitoring and maintaining the associated probes. In order to realise this step, a formalised process has been defined transforming the raw measured data into a normalised security assurance level. Five security assurance levels have been defined enabling to express the increasing confidence.

The third step concerns **measurement**. This aspect addresses network investigation by deploying specific probes implementing selected metrics. Choices made in step 2 will directly impact this third step. This step impact is local, i.e. it concerns local measurement of the security assurance.

The normalised measures are **aggregated** in the fourth step to express the assurance level of infrastructure objects. The ultimate purpose of aggregation is to derive an assurance level for the service. During aggregation an analytic model (i.e. various aggregation algorithms) is used to recursively parse the assurance model until the service node (i.e. the root node) is reached. The analytic model calculates the assurance level of an infrastructure object from (a) metrics associated to the infrastructure object and (b) its child objects assurance levels.

The last step provides means to the operator to **monitor** security assurance status both at service and network infrastructure objects, to be able to determine the causes of assurance deviation, and consequently to provide assistance for security management. In order to realise this step, an easy-to-use and centralised monitoring console has been defined that we call the security cockpit. This console provides three main functions: measurement, monitoring and assistance.

The taxonomy and the methodology described in this document have allowed an implementation of an operational framework that can evaluate the security assurance of an operational telecom system from the service perspective, demonstrating its feasibility. Furthermore, as described in Appendix B, materials of this document provide bases to draw standard/best practices document, with the same approach as the Common Criteria from which the initial definition and concepts have been taken in order to build this methodology.

2 INTRODUCTION

2.1 Scope of this document

This document includes a general discussion of issues related to the security assurance evaluation of telecommunications services. To be able to explain methodology choices, this document introduces a consistent terminology.

Using this terminology, this document presents the goal and the motivation for and various problems with the operational security assurance evaluation of telecommunication services. It discusses the issues encountered with such evaluations in modern telecommunications systems, justifies the need for a new methodology and discusses general methodology principles.

Based on some of the discussed principles, this document in particular presents the operational methodology to be implemented in all details. This operational methodology proposes ways to solve major discussed problems. It defines a new methodology based on an innovative approach to the security assurance evaluation in the telecommunication infrastructures, including the whole security assurance evaluation process (data assessment, data processing and result presentation).

2.2 Organisation of this document

This document is organised as follows. The next section (Terminology) introduces a terminology glossary. The terminology section introduces the terms and definitions used throughout this document. The following section (Architecture Overview) presents a high level view on the main idea and outlines the architecture separating the assessed service from the evaluation subsystem.
The section on general aspects of the methodology (Methodology Principles) introduces main scientific, theoretical and practical locks in the project scope, discusses possible solutions and their shortcomings and proposes new approaches.

The section on the chosen operational methodology (Operational Methodology) instantiates a possible approach to security assurance evaluation. It presents made decisions and choices and goes further in the practical details.

The conclusion summarizes the main findings, proposals and choices.
Additional information is available in form of appendices.

3 TERMINOLOGY

3.1 Principles

Security assurance: grounds for confidence that an entity meets its security objectives

Confidence: faith or belief that one will act in a right, proper, or effective way

Trust: firm belief in the reliability, truth, ability, or strength of someone or something.

Complex system: A system that involves numerous interacting agents whose aggregate behaviours are to be understood. Such aggregate activity is nonlinear; hence it cannot simply be derived from summation of individual components behaviour. [Jerome Singer]

Black box: black boxes are complex objects where certain properties of the content are discoverable and certain are fundamentally undiscoverable.

Incident: an event that may represent the materialisation of a threat.

Threats: A potential cause of an incident that may result in harm to a system or an organisation (IS WD 27000)

Vulnerability: Weakness in an asset or group of assets that can be exploited by one or more threats (ISO/WD 27000).

Countermeasure: security services or mechanisms designed to counter a particular threat

Formal: expressed in a restricted syntax language with defined semantics based on well-established mathematical concepts (CC).

Informal: expressed in natural language (CC).

Semi-formal: expressed in a restricted syntax language with defined semantics (CC). These languages are typically graphical notations.

System: A system is here defined as a set of objects together with relationships between the objects and between their attributes related to each other and to their environment so as to form a whole.

3.2 Risk management related definitions:

Residual risk: the risk remaining after risk treatment [ISO/IEC Guide 73:2002]

Risk acceptance: decision to accept a risk [ISO/IEC Guide 73:2002]

Risk analysis: systematic use of information to identify sources and to estimate the risk [ISO/IEC Guide 73:2002]

Risk assessment: overall process of risk analysis and risk evaluation [ISO/IEC Guide 73:2002]

Risk evaluation: process of comparing the estimated risk against given risk criteria to determine the significance of the risk [ISO/IEC Guide 73:2002]

Risk management: coordinated activities to direct and control an organization with regard to risk [ISO/IEC Guide 73:2002]

Risk treatment: process of selection and implementation of measures to modify risk [ISO/IEC Guide 73:2002]

3.3 Architectural

Monitoring system: a system assessing a potential target (object, entity, subsystem, system).

Observed System: a system being assessed, target of assessment.

Assurance Process: the process executed by the parts of BUGYO system in order to give an estimate of the security assurance of the Observed System. Takes input from BUGYO probes and/or BUGYO agents and outputs an estimate of the current security assurance (usually on the BUGYO cockpit).

Cockpit: (Security Cockpit): System from where the BUGYO System can be supervised. Moreover the BUGYO cockpit displays the current SA estimate of the system in operation to the user.

Agents: agents gather data from probes and/or calculate metrics associated with model objects.

Probe: probes perform measurements on the entities of the system under control.

Assurance System: System composed of probes, agents and a cockpit with the purpose of evaluating the Security Assurance of the observed system.

3.4 Observed System

Model: The model is an abstract view of the service to be monitored. The model splits the service into objects to which metrics are associated.

Dependency: a constraining unidirectional relationship between two subsystems A, B of the Observed System such that A depends on B means that A relies on B for the correct execution of its function. If A depends on B and B is not available or not reachable from A, A's function execution is typically incorrect i.e. in practice either incomplete/limited (no unwanted situations can occur, but some wanted situations do not occur; best case) or wrong (unwanted situations can occur; worst case).

Interdependency: a pair wise dependency between two entities, i.e. a direct mutual dependency.

Security function realization: a subsystem (entity, object, sum of connected entities, etc.) of the Observed System implementing a security function.

3.5 Conceptual

Attribute: [ISO 27004] property or characteristic of an entity that can be distinguished quantitatively or qualitatively by human or automated means.

Base measure(s): [ISO 15939] measure defined in terms of an attribute and the method for quantifying it.

Derived measure: [ISO 15939] a measure that is defined as a function of two or more values of base measures.

Security Assurance Assessment: the overall process of measuring, deriving, processing, analysing and evaluating the security assurance of the observed system.

Security Assurance estimate: the Security Assurance of a modelled object as calculated by the BUGYO system.

Security Assurance level: Security Assurance is scaled into five levels from 1 to 5. The level 0 means that no Security Assurance can be asserted.

Aggregation: aggregation functions are used to combine several numerical values to one single numerical value.

Metric: algorithm to estimate a Security Assurance value of a system or any modelled part of it from the input data.

Normalisation: Normalisation is the process in which a derived measure is transformed, by means of an analytic model, into a discrete assurance level that reflects the security assurance status of the infrastructure object according to the assurance taxonomy.

Security Assurance Evaluation: Security assurance evaluation is the processes of analysing the measured state with the expected state by means of automated comparison patterns.

Adaptation: modification of a behavioural tendency by experience (as exposure to conditioning).

4 ARCHITECTURE OVERVIEW

4.1 Main Idea

The main idea behind is to evaluate and monitor the security assurance of an Observed System in operation. The scope is specifically bound to telecommunications services. The approach is explicitly non-intrusive, i.e. all elements are designed to limit their influence on the system that provides the observed telecommunications service as far as possible. The methodical parameter observation and data collection from this system for a timely evaluation is the main project target.

This non-intrusiveness allows adding a dedicated infrastructure (observing system) to the targeted telecommunications service infrastructure in operation (observed system). The main assumption is that the telecommunications service as such can be provided independently of the state of the evaluation system (existent, active or inactive). This view underlines the main idea behind: while it is very difficult to evaluate the security assurance of a complex system in operation, the existence of a fully mastered dedicated infrastructure with a predictable, fully known behaviour allows reliably collecting data necessary for evaluation.

In the proposed methodology we presume that the service provisioning is organised as illustrated in Figure 1. The business relationships of a given organisation define contracts that imply sets of rules. These rules are used in the service plane for service design and dimensioning, both in terms of service's functional (i.e. what it does) and non-functional (i.e. how it does it) properties. The functional properties are available as features at the user interface. The non-functional properties are necessary to maintain the specified operation of the service, e.g. for the reliability, quality and security of the proposed service, for internal needs of the service provider, etc. The non-functional properties have an important impact on the functional properties and are usually captured in form of service level agreements (SLA), including e.g. quality claims (bandwidth, delay, jitter, availability, etc.) but also security claims (SecLA).

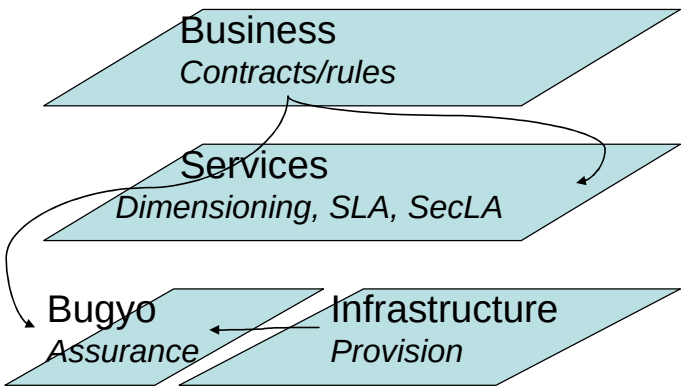


Figure 1 – Abstraction planes in the proposed methodology

The latter are then mapped to an infrastructure capable of providing the requested services with the attached quality and security properties. The ultimate goal of the

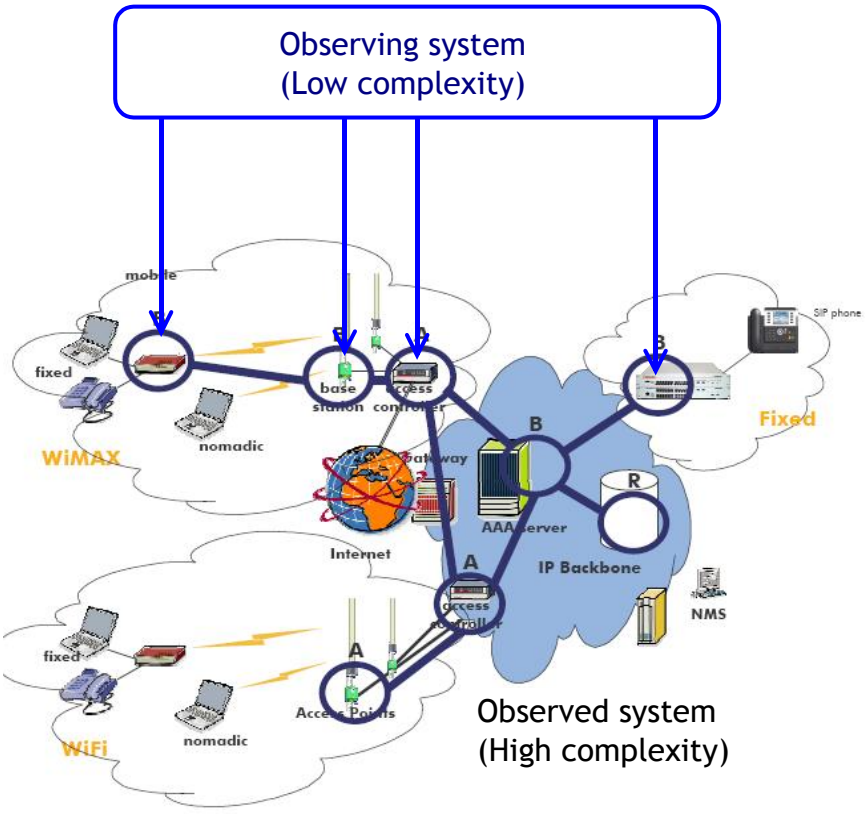
system is to participate in the assurance of the overall compliance of the offered service at any time by giving evidence for the observed system's security assurance.

4.2 The Observed System

The observed system is the original system providing the telecommunication service whose security assurance is to be estimated. This system is represented using modelling techniques.

4.3 The Monitoring System: observing system

The monitoring system is the added system composed of probes, agents and the security assurance cockpit. This monitoring system is dedicated (and usually reserved) to needs and can thus be achieved with very low complexity (simple system) in order to observe the Observed System. The simplicity and the dedicated character permit to develop a very robust and reliable system, which in itself will have a very low failure probability (magnitudes lower) compared to the Observed System. From the point of view of the Observed System, the monitoring system is an almost perfect add-on. This strong reliability requirement caters for sensible statements about the Observed System.



5 METHODOLOGY PRINCIPLES

5.1 Basic Approach

5.1.1 BUGYO Scope and Context

Current telecommunication systems are complex systems. It is generally impossible to decompose the system into building blocks (components) without losing systemic properties available at the system level.

Such a telecommunication system is characterised through a high number of more or less strictly defined interfaces. The interfaces exist at both system levels (usage interface, management interface, service interface, interface to the underlying network, to the subcontractors, etc.) and at component level (horizontal inter-component interfaces – usually protocols, vertical layer interfaces such as SAP interfaces defined by the ISO/OSI model, the Berkeley sockets, but also more abstract interfaces like OSA/Parlay, object APIs, etc).

Unlike some decades ago, the systemic behaviour of currently deployed telecommunication system is reigned by distributed software solutions. In that sense, a telecommunication network can be seen as a distributed software piece. This software is supposed to fulfil both external requirements (service contract compliance, management, etc.) at the system interfaces as well as internal requirements at component interfaces. Consequently and increasingly, a lot of system components are today pure software components with all typical software properties (high configurability and flexibility, external provisioning, high dependency on the executing environment and outer components, possibility of in-runtime creation and destruction of subcomponents).

In that view, both system and component structure and behaviour – i.e. including the architecture of the system – can be treated as reconfigurable in runtime. That would principally prohibit approaches based on an “a priori modelling” of the targeted infrastructure, since that model would necessarily be a program per se which, for a proper real system description, would need all system inputs at any time. A radically different approach would be necessary in that case.

However, the idea is to limit the of the approach applicability to telecommunications services provided by well-defined, almost stable systems (the system shape and behaviour are rarely changed in operation). We presume that the observed system mainly, i.e. in some characteristic systemic traits, has to comply with a preliminary specification. Practically, we thus allow for an important dynamism in system shape and behaviour, but expect that some critical components maintain their aspects. Notably, as the modelling is reflecting business-critical elements, those elements are considered stable enough to be able to express security assurance in order to maintain security. We believe that it describes “classic telecom systems”.

Furthermore, the approach explicitly limits its scope to observation and display so as to minimise interaction with the Observed System.

5.1.2 BUGYO Approach

We presume that an initial Observed System is a system with functional properties as those implied by a given service contract. This purely functional system would not be sustainable; to ensure its correctness (contract compliance at any time, especially concerning the properties of the available functions), it thus necessarily contains a reliability subsystem and a security subsystem.

The reliability subsystem overlaps with the functional system (due to e.g. self-redundancy, resource sharing, etc.) but also contains additional new parts (additional redundant parts, additional control structures and entities, algorithms, etc.) resulting in new systemic properties (e.g. maximum supported load, fault tolerance).

Equally, the security subsystem overlaps with the functional system (security functions relying on network engineering, etc.) but also implies new system properties (e.g. security filtering and the added verifications can cause new service disruptions). Finally, the reliability subsystem overlaps with the security subsystem since security functions can be used to enforce reliability and designed with redundancy for higher availability.

Following its purpose of providing the system administration with a reliable estimate of the current security assurance of the offered service, the targeted approach resides in the overlapping area between reliability and security. However, it must not be forgotten that the Observed System for the system is not the initial functional part, but the resulting sum of the overlap of all parts, i.e. the complex system Σ (see Figure 2).

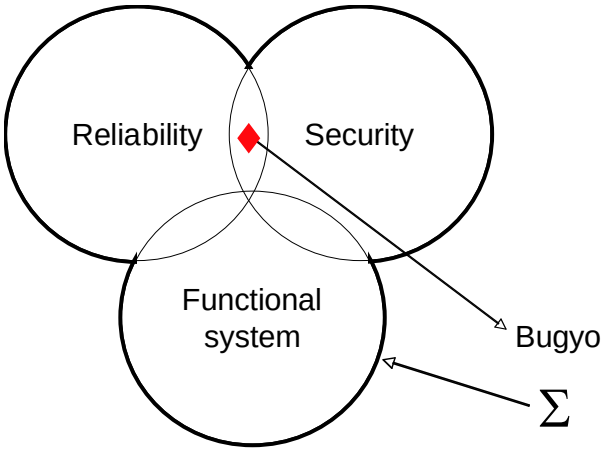


Figure 2 – Scope and position

In the current practice, the reliability subsystem is typically managed from a NOC (network operation centre) and the security subsystem is managed from a SOC (security operation centre). This (rather artificial) separation of Σ in imaginary distinct subsystems expectedly results in an unsatisfactory real system operation. Indeed, problems occurring at the overlap of reliability and security represent an important part of problematic vulnerabilities (usability vs. security discussion, DoS attacks, etc). Confronted with such problems, management staff is often lost in useless policy, responsibility and authority discussions.

The approach can help here by providing the network administration teams with new possibilities, in particular with new integrated administration tools for telecommunication services. By constantly monitoring the security subsystem, the system can detect and report deviations and eventually improve the reliability of the security subsystem, and therefore of the overall service. Security assurance aspect links reliability and security subsystems by providing a sensible estimate for the current security assurance. By giving sensible almost real-time insights in the current system operation modes, it ultimately increases trust in the overall system behaviour.

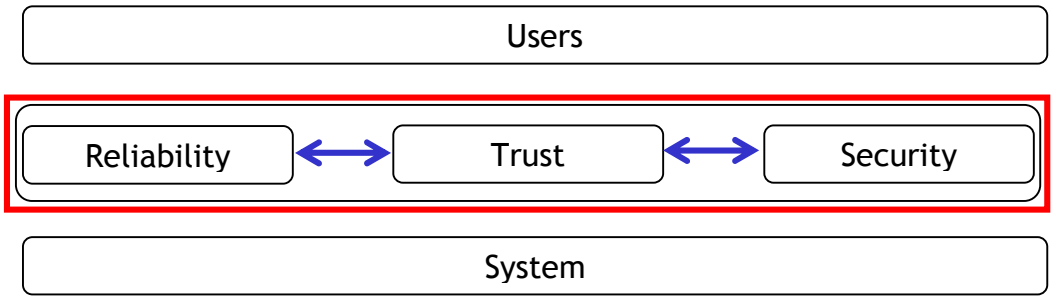


Figure 3 – Linking reliability and security through trust

The major challenge of the BUGYO project is to estimate the current security assurance value of a service delivered by a complex system in operation.

5.2 Problem Definition and Discussion

5.2.1 Security Assurance Definition

We define here the security assurance as the grounds for confidence that an entity meets its security objectives (see the Terminology section; this definition is conform to CC v2.3). The grounds for confidence here translate to verifiable, measurable evidence that we assume you can gather and show to somebody. We can thus gather and interpret data and present proofs that the security objectives are fulfilled. This is the reason why the targeted goals are practically achievable.

5.2.2 Telecom Service's Security Assurance Assessment Problem

In this context, the mentioned "entity" is the service. A service is a distributed application. It thus needs storage, calculation and communication facilities to fulfil its purpose. The security assurance of the service is thus mainly dependent on the security assurance of these parts. However, we know that for a distributed application, the calculation and the storage are actually very reliable. Their security mainly depends on the trust we have into the executing part. The approach is basically works in the trusted environment of a telecommunications provider (one infrastructure). Therefore, the security assurance of the service mainly depends on the security assurance of the communication facilities[1].

From a service perspective, the communication facility is a service session, established over and transported by a telecom infrastructure. Thus, the security assurance of the service is primarily dependent on the security assurance of its sessions. We thus need to deliver proofs for the service-related means of session protection, including all classical security functions (identification, authentication, confidentiality, integrity) and more subtle security functions aimed at the availability (dynamic admission control, compliance checking, IDS, etc.).

The service sessions' assurance properties per se hugely depend on the telecommunications system by which the session is transported. Indeed, the one and same service session running over different infrastructures would need to be assessed to a different security assurance value, depending on the vulnerabilities, provisions and operational assurance of the infrastructure. Thus, to evaluate the security assurance of a service, we generally need to gather evidence data that service's own security mechanisms work as intended and that the environment transports the session as intended.

In practice, this requirement can be relaxed, since for pragmatic reasons we presume that the environment (e.g. the underlying telecommunication system) is stable, e.g. it is designed to be capable of correctly transporting the security sessions, executing the security processes, etc.

The two main difficulties herein are the choice and definition of the "right" model (i.e. a representative model including essential parameters) and, if decomposition in lower

[1] Notice that this is a hypothesis, but it corresponds to the reality of a telecom operator. It does not correspond to e.g. grid calculation or P2P file sharing. There we would need proves that the calculation happens as intended in the remote nodes, that the stored data is the data you search, etc.

complexity level elements is used by the model, the aggregation of the obtained values back to the system's service level security assurance value.

The modelling problem proposes modelling techniques. The parameter choice and the aggregation need profound system knowledge. One problem is to know what matters. The parameter choice and the methodology for that are discussed in [1]. The related problem is how to combine the different parameters of the same level of complexity and to what value (in terms of comparability, stability, frequency, etc.). That is done within a predefined algorithm called metrics. The deployment and definition of metrics is part of the modelling and is discussed in [1]. The problem is further discussed in this chapter. The requirements and expectations on metrics, e.g. their quality and the related processes, are discussed and defined within the operational methodology section of this document.

The second problem is that there is no general form for the aggregation function. Rather it is a function of the input values and the topology relating these sub-entities producing these values. The mentioned topology is the topology from the service point of view. Thus, the aggregation function should be generally different for any system and service, which is however not necessarily the best approach in practice. This is discussed in details in the Aggregation chapter.

5.2.3 Security Assurance Assessment Requirements

The security assurance (SA) assessment requires the following:

- Modelling techniques capturing SA-relevant properties of system structures,
- Methods for measurement (and then assessment) of the SA strength of system entities, regarding the associated set of characteristics,
- A set of measurable SA-related attributes that can be mapped to specified SA metrics,
- A set of SA metrics,
- Methods aggregating the results for individual system entities, possibly including other factors, to system-wide SA values,
- Presentation, evaluation and reporting facilities.

5.3 Aggregation

5.3.1 Introduction

Security assurance of a service is a service-level property. As a rather abstract and high-level property, it is generally not directly measurable on the service level. As explained in previous chapters, the security assurance of the service mainly depends on the realisation of the service by the underlying telecommunications system. Therefore, one often has no other choice but to measure at the realisation level to obtain insights concerning a property on the service level.
The problem is underlined by the fact that the security assurance of the service often makes no contextual sense on lower levels of abstraction such as nodes, communication paths, etc. For instance, one cannot generally assert that the security assurance of the service decreases if the security assurance of some participating (groups of) nodes decreases. Indeed, depending on the nature of the service and its security requirements, this decrease can be relevant or not; it can also be automatically compensated by other elements, etc. For instance, in modern telecommunication architectures, any measured property of the underlying system could principally exist independently of the delivered service. Indeed, it is usually not owned by a service, but shared by different services.

That constitutes one of the central problems with service's security assurance. One needs to know to which degree and through which relation a given measured property of the system delivering the service impacts the service's security assurance. This is the problem, which we here refer to as aggregation.

5.3.2 Aggregation Function

We could thus reformulate the aggregation problem in this context as following. Given a set of chosen measured properties of a telecommunication system delivering a service, what is the function relating these measures one to each other so as to result in a reasonable estimate of service's security assurance?

Since security assurance deals with proofs for the correct security feature execution ("as intended", see Terminology), compliance and reliability of the security countermeasures have a major impact on the assessed value. The compliance to the requirements does not have a structural property as such, and cannot be compensated at the system level. The non-compliance of some parts can sometimes be compensated through e.g. functional redundancy. The reliability for telecommunications services is typically addressed today through an a priori design (dimensioning, pre-provisioning, etc., i.e. mainly through another requirement and compliance set) and through redundancy (of objects, links, but also functions and structures). The redundancy generally has an impact on the security assurance and an aggregation function has to attribute a different treatment to potentially critical elements (like a single point of failure) and to highly redundant elements. The redundancy does have structural properties, as it is highly dependent on the topology of the Observed System. This underlines the fact that in general, the aggregation function has to be constructed by taking into account the service to be delivered (mainly to extract requirements on the service functions and their properties) and the Observed System (to extract requirements and critical properties related to the correct execution on the basis of the implementation).

Due to the high diversity of telecommunication services, different possible realisations and very different topologies and structures encountered today in telecommunication systems, it seems arduous to define one universal aggregation function. The question thus arises how this function can be constructed in the methodology.

5.3.3 The Role of the Modelling

In this scope a very important role has to be attributed to the modelling. The modelling is central to the aggregation function derivation, because its task is to produce a security assurance specific view on a service realisation.

Therefore, if the model is correct (i.e. representative for the task, here security assurance evaluation), then the aggregation function can be derived from the model. If the model is capable of simplifying, then very different architectures, topologies and implementations (usually classes of these) can be mapped to the same model for the same service. The model thus can additionally limit the possible space for the aggregation function.

5.3.4 The Construction of the Aggregation Function

As explained above, in general we can assert that the aggregation function for the security assurance of a service is a function of the produced security assurance service model. A given model can thus be processed so as to derive an aggregation function (by an expert but also automatically).

In the following we present some of different aggregation functions as examples of given security assurance related properties of a service.

5.3.4.1 Max and Union

In the case where the Observed System compensates for an unintended security feature execution at one part of the service realisation, the aggregation function can be a *maximum operator*.

5.3.4.2 Min

If the system part implementing security functions has several critical points, then the security assurance of the provided service will have to be aggregated in the form of the "weakest link", thus resulting in a *minimum operator*.

5.3.4.3 Weighted-sum and Average

If a system consists of security functions each contributing in a different, non-critical manner to the required service security, and if their realisations are implemented by independent Observed System parts, an appropriate aggregation function is the *weighted-sum*. Formally, the SA value of the aggregated node can be described as:

$$SAV_{aggregated} = \sum_{i=1}^{n} w_i SAV_i$$

where n is the number of assessed properties, SAV_i is the SA value of property i, w_i is the corresponding weight percentile ($\sum_{i=1}^{n} w_i = 1$).

A special case of the weight-sum function is the average where the weight percentiles are equal for all elementary nodes. If the weights can be derived dynamically (and are not derived from the model) then the above discussed *min* and *max* operators can also be represented through the weighted-sum operators with appropriate weights:

- $SAV_{aggregated} = SAV_i$ with $SAV_i = min\ (SAV_1, \ldots, SAV_n)$ and $w_i = 1$, $w_j = 0$ ($i \neq j$) for the case of the *min* operator.
- $SAV_{aggregated} = SAV_i$ with $SAV_i = max\ (SAV_1, \ldots, SAV_n)$ and $w_i = 1$, $w_j = 0$ ($i \neq j$) for the case of the *max* operator.

The weighted-sum functions seem to be general and therefore can be used to represent other (not all) aggregation operators, especially if the weights can be derived dynamically.

In general, the resulting aggregation function can be constructed as a combination of max, min and weighted sum functions.

5.3.5 Aggregation Patterns

Patterns describe recurring solutions to common problems in a given context and system of forces. Patterns are a format for capturing insights and experiences of an expert; they are, essentially, representations of knowledge of how particular problems can be solved. The more information a pattern has, the more important structure becomes. Structure leads to uniformity of patterns. Thus, people can compare them easily.

Using standard entities, which have already separately been assessed, can be expected to simplify the process of system security assurance assessment. Likewise, defining and using standard relations is also expected to enhance and simplify system security assurance assessments.

A possible approach for security assurance assessment is therefore to identify known blocks (patterns) in the Observed System's model for which aggregation functions are known a priori. These patterns can be recursively applied by studying the system model.

5.4 Proposed General Methodology

5.4.1 The need to define a new methodology

The most recognised security assurance assessment methodology in the industry is defined by the ISO/IEC 15408 standard [7], also known as Common Criteria (CC) [5][2]. The Common Criteria define security assurance evaluation requirements for the development and design phases. The associated methodology to apply is defined in the Common Evaluation Methodology (CEM) [6] (prior versions of which are available as ISO/IEC 18045 [8]). This methodology is not directly applicable for an evaluation of the security assurance of a system in operation.

Different other methodologies for related subjects (risk assessment, system assessment, system monitoring) exist.

Although the CC methodology as such is not applicable to an operational system, one could use the CC by applying it to data captured from an operational system. However, the underlying Common Criteria paradigm relies upon the strict separation of the evaluated asset and its supposed operational environment. The clearer the separation, the better the CC methodology can be applied. While there usually is such a separation for products, generally it is difficult to draw a clear perimeter of today's telecommunications services. The problem is twofold.

The first part of the problem is domain limit determination. As of today, given all the Byzantine failures and unresolved trust issues, it would be difficult to use an exiting tool on a service spanning over several administrative domains (e.g. Internet). To avoid these issues, in this methodology we pragmatically limit ourselves to one effective authoritative domain, i.e. we presume one of the following situations:

- The whole service runs over one administrative domain (including the implementation system) or
- Full trust can be established to every consumed sub-service or service component by some means out of the project's scope.

The user domain is an exception that deserves a special treatment. As a distributed application, the service spans from the provider over the providing infrastructure until to the consuming entity. If the consuming entity is the end-user, generally at least two distinct administrative domains are involved: user and operator authoritative domains. This is only partly addressed in this project: the main reason for that is that the approach is primarily provider or operator-centric. In the simplest case, the user is part of the provider's organisation (closed, non public network). In a more complicated case, a user-provider contract exists and it is presumed that all main service provisions are part of the providing infrastructure, in particular including security functions and the corresponding mechanisms' realisations. That is why the security assurance accent lies in the providing infrastructure.

[2] Different versions of Common Criteria evolve independently of ISO/IEC standards. Strictly speaking, the latest, 2005, version of ISO/IEC 15408 standard is equivalent to Common Criteria version 2.3. See explication in reference [7]. Common Criteria version 3.1 [5] was published in September 2006 and is expected to be reflected later by a revised ISO/IEC 15408 standard.

However, the problem is not only in the domain limitation. The problem is also in the ongoing virtualisation of the telecommunications services. Most functions today are implemented entirely in software, running on general-purpose processors. The software world is very modular by nature. Indeed, the modularity of software is by design, considered necessary in the software world: modern software engineering hugely relies upon reuse of simple well-tested blocks. Object reuse, copying, creation and destruction in runtime are processes inherent to all modern software. Ultimately, that means that the actual service components find themselves in a cyber layer, a common pool of communicative software modules, functions and interface objects, triggering lengthy branches of other, remote and local, function executions. Under these circumstances, telling what is a service component and what is part of its environment appears more like useless exercise in philosophy. The practical resolution here lies in the choice of the right level of abstraction but the separation remains blurry by nature: at some lower level the components could be shared and interdependent.

We concentrate here on the classic telecommunications infrastructures and do not support an arbitrary level of structure changes (as it would be in modern ICT systems, being in essence distributed software). As explained above, the telecom infrastructures are expected to remain stable from the point of view of business critical services. In this classical view, we could draw a network perimeter, including or excluding entities from monitoring according to their position. Still, the exchanges over this perimeter are diversified and the interactions open a big space for what is possible to achieve. From the other hand, there are internal interfaces that also need to be considered, especially for management and security.

The basic problem with the CC methodology is that it is designed to be applicable in the design and development phases. CEM defines how the system must be developed, but not how to maintain it in the “correct” (i.e. intended) state. Even limiting the possible system morphologies and system’s transformations so as to achieve a clear separation line, still would need a CC methodology extension since new parameters would need to be considered to evaluate the assurance. This new methodology would also need to specify how these parameters could be captured.

Thus, in a very general sense the Common Criteria approach and its associated methodology (CEM) is not directly applicable to the targeted environment. In general, a new methodology is needed.

5.4.2 The Methodology

We apply here a telecommunication service provider centric view. In that view, the provider owns both the underlying infrastructure and the provided service. In other words, the same principal/organisation owns hardware components, the wires and fibres, the mentioned abstract software layer and the service offered upon it. The user side is mostly ignored because the service is provided in a highly asymmetric way. However, the user service interface security – i.e. the service access security – is crucial.

In the first step, the part of the underlying infrastructure involved in the service provision can be seen as the asset to be protected. As mentioned above, the delimitation of the scope is done through the service definitions and through domain boundaries. We presume that a reasonable risk assessment has been carried out on that asset in respect to the service to be provided before the targeted system addition. The risk assessment is understood in the broad sense: it is the set of organisational and

technical procedures that take the system with its services and the managerial limits as input and result in the definition of adequate security measures for the service (internal and external, i.e. at the user interface).

A part of service modelling is carried out with respect to the risk assessment. That modelling defines a system security shape, i.e. a number of security functions with their interdependencies and defined properties.

That modelling and the related deployment of probes in the infrastructure enable BUGYO to monitor the system for potential compliance problems (e.g. function non availability, non conform reconfigurations by the network administration, etc.). This part of monitoring is necessary in any case for the security assurance assessment. The exact methodology for such monitoring, with different precision and rigor levels, the formalisms and the taxonomy, are defined in details as the Operational Methodology later in this document.

However, the boundaries of this asset are fuzzy by definition. This particular asset actually overlaps with its environment through its numerous soft interfaces: with the execution environment through the programming interfaces and with the peers through the protocol interfaces. Because of the built-in system dynamics (even if the boundaries are set), the capabilities of the underlying system and thus the security properties of the provided service could change in time with the system still being compliant to an initial specification. That could be due to different factors, such as actual system load, installed software patches/versions, current system state (currently available redundancy, number of problems), etc. Therefore, the targeted system should generally monitor the current state of the underlying system (parts) related to the security subsystem. The goal is to verify if the system *really* is in the state in which it *can* securely provide the service (and thus, if the system can respect service's security requirements).

If this is necessary (depending on estimated usage, system architecture, system stability and vulnerability, etc.), it can be done in addition using the modelling part [1] that reflects the topology, the architecture and the redundancy of the underlying system from the service point of view. This is however still an experimental topic that needs further research. Alternatively, it can be achieved by hiding the whole state estimation complexity within the deployed metrics and their algorithms using the static dedicated model.

Because of the high dynamicity of the observed values/parameters, the state of the system monitoring process usually needs an adaptation phase: we need to obtain a stable value (which can need some time to converge) and to dynamically set the evaluation thresholds. The adaptation can be realised through learning techniques mentioned later. We could then aggregate the obtained values to a stable system value and check that this value is within some limits.

6 OPERATIONAL METHODOLOGY

6.1 Assurance Taxonomy

The security assurance taxonomy aims to allow for a clear definition of the security assurance levels. Inspired by the *Common Criteria* [5] we define a number of generically comparable assurance levels. As such, the *Common Criteria* [5] indicate that different levels of assurance are possible dependent on **rigor**, **depth** and **scope** of the verification of the security functionalities of a system. We aim to use in the product these parameters but due to our goal of assessing assurance in (close to) real-time for large telecom networks, we need to **add** some **criteria** to determine the assurance level we can assign to a system.

However, before we continue with the specification of the levels we think that it is necessary to clarify our understanding of what assurance levels express. For us, the definition indicates that assurance levels should be built depending on the amount of confidence that they allow and that the defined security mechanisms are working as expected. In our point of view, the growth in confidence between two successive assurance levels is not linear, but rather follows a logarithmic increase of confidence that asymptotically approaches a full confidence but never reaches it – see Figure 4. The reason that it is impossible to reach is twofold. First of all, the full representation of a real system is not possible according to the system theory. And second, the time dependency of assurance evaluation makes it impossible – i.e. a momentary evaluation of all factors of a system is not doable due to latencies.

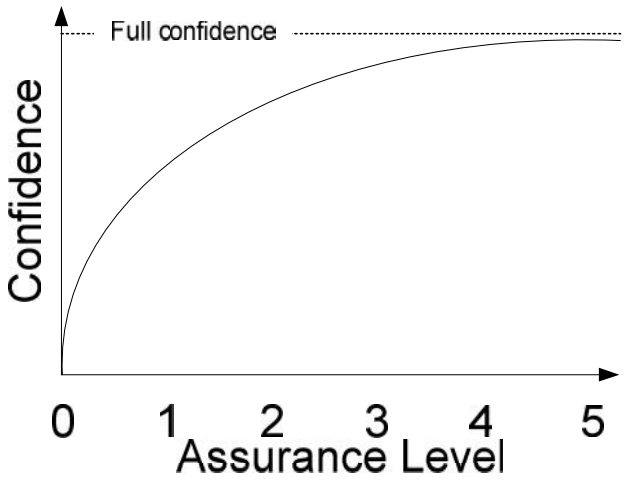


Figure 4 – Assurance level confidence relation

Important to note is that although we assume a logarithmic scale we are not capable to express the exact distance between the assurance levels but only see a qualitative improvement. The indicated assurance level should, however, indicate the expectable real assurance in relation to the spent efforts (i.e. higher levels provide more assurance and are more difficult to achieve).

6.1.1 Assurance levels

By analogy to the *Common Criteria*, we state that the assurance should be defined with a **discrete scale of assurance levels**, and that each level of the scale must **include** the level just above it. Figure 4 already indicates that within product we use five assurance levels. We defined five levels due to the following pragmatic considerations and our perception of experiences with the *Common Criteria*:

- We considered it important to have an odd number of assurance levels so that it is possible to have a medium assurance level (i.e. although only ordinal, the scale is required to have a conceptual middle point).
- The CC evaluation assurance scheme contains seven evaluation assurance levels. However, to our knowledge (almost) none of the system has been evaluated higher than level five. With our scheme, we do not intent to reproduce levels that are not reachable. In general, systems are highly complex and it is unlikely for us that they can satisfy formal evaluation methods. Therefore it did not seem suitable to include levels that are only achievable formally.
- To provide only three assurance levels did not provide enough granularity in our view as our intention is (a) the highest assurance levels should be very hard to achieve (but not impossible) and (b) the lowest assurance level can be achieved by setting up a control system (as it provides already a basic level of confidence).

We therefore derived the following five assurance levels.

Assurance Level	Definition
AL1	Rudimentary evidence for parts
AL2	Regular informal evidence for selected parts
AL3	Frequent informal evidence for selected parts
AL4	Continuous informal evidence for significant parts
AL5	Continuous semi-formal evidence for the entire system

Table 1 – Security Assurance Levels

Our basic postulate is that assurance levels are ordinal (i.e. ordered with an undefined distance between values). A second postulate is that a higher level, we call it B, includes all assurance indications (i.e. requirements) of the level below it, which we call A.

$$A < B \Rightarrow \{A\} \subset \{B\} \qquad \textbf{(Equation 1)}$$

To be complete (and compliant with our intention of an ordinal scale) we have to define an assurance level AL0 with no assurance indication (AL0 is the empty set)

$$AL0=\{\} \qquad \textbf{(Equation 2)}$$

Important to note is that an assurance level is expressed for a specific infrastructure object (service and sub-components).

6.1.2 Assurance Classes

Table 2 represents a summary of the defined ALs. The columns represent an ordinal set of ALs, while the rows represent assurance classes and families of criteria we use in order to express the requirements for the various assurance levels with respect to the assurance level taxonomy defined above. Each number in the resulting matrix identifies a specific assurance component where higher numbers imply increased requirements.

While the presented ALs are defined in BUGYO, other combinations are imaginable under the condition that the ordinal property is preserved. This means for example that a system satisfies AL2 if one or more components satisfy AL3 or higher requirements (Frequency 3). We think that such a combination is possible and meaningful to define if the goal is to stepwise improve towards the next assurance level. Note, that we do not intend using AL2+ or similar, to indicate such (i.e. we do not want a finer granularity that makes it in our view only harder to interpret an AL for the recipient).

Class	Family	Level				
		1	2	3	4	5
CLASS SM: Service Model	SM_VU: Absence of relevant vulnerabilities	1	1	2	2	3
	SM_OR: Unmanaged/managed objects ratio	1	2	2	3	4
CLASS MC: Metric Construction	MC_SC: Scope	1	2	2	3	4
	MC_DE: Depth	1	1	2	2	3
	MC_RI: Rigor	1	2	2	2	3
	MC_RE: Reliability of metric	1	2	2	2	3
	MC_TI: Timeliness	1	2	3	3	3
	MC_FR: Frequency	1	2	3	4	4
	MC_SA: Stability	1	2	2	2	3
CLASS MM: Maintenance management	MM_PM: Probe maintenance	1	1	2	2	2
	MM_OM Infrastructure object model maintenance	1	1	2	2	2

Table 2 – Assurance level Matrix

The subsequent presentation of the assurance classes follows the example set by the Common Criteria. First we describe each class and show which families it contains. Subsequently the families are described. For each family a description, its dependencies and its components are provided. The components are hierarchical and, if not otherwise specified, higher-level components include the lower levels.
(as for Common Criteria, only one class is presented on each page to ease reading and exploitation).

6.1.2.1 CLASS SM: Service Model Class

The service model (as defined in [1] represents the important infrastructure objects that are required to deliver a service. Due to its representative character, the service model is also a continuous documentation facility for the assurance information that concerns the service. Generic properties of the service model therefore influence the overall assurance of the service.

	Families	Description
SM_VU	Absence of relevant vulnerability	Relevant vulnerabilities should not be present in the controlled system.
SM_OR	Unmanaged/Managed Object Ratio	Less unmanaged objects can provide higher confidence in the assurance expression

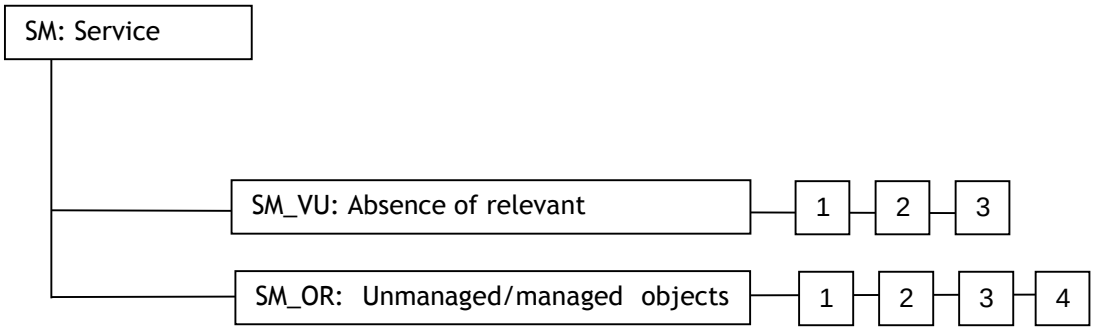
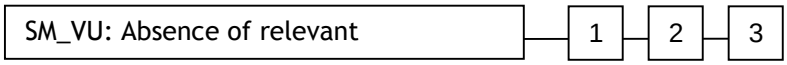


Figure 5 - SM: Service Model class decomposition

SM_VU: Absence of relevant vulnerabilities
The absence of known vulnerabilities is a minimum prerequisite to have any confidence in a service. Due to the automatic character of the monitoring system different degrees of evidence are available which support the claim for vulnerability absence.

Component levelling



Dependency: MC_RI

- ***SM_VU.1 Without further evidence***
  No specific evidence for the presence or absence of vulnerabilities is provided.

  - SM_VU.1.1: During normal operation of the monitoring platform no obvious vulnerabilities were found.
  - SM_VU.1.2: No specific investigation for vulnerabilities was conducted.

- ***SM_VU.2 With informal evidence***
  Informal evidence (scan logs) for the presence or absence of vulnerabilities is provided.

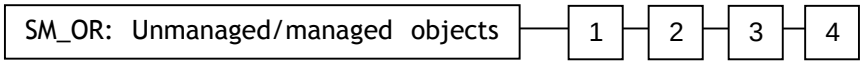
  - SM_VU.2.1: During normal operation of the monitoring platform no obvious vulnerabilities were found.
  - SM_VU.2.2: Investigation for commonly known vulnerabilities was conducted by means of an automated test.

- ***SM_VU.3 With semi-formal evidence***
  Semi-formal evidence (scan logs, analysis documentation) for the presence or absence of vulnerabilities is provided.

  - SM_VU.3.1: During normal operation of the monitoring platform no obvious vulnerabilities were found.
  - SM_VU.3.2: Investigation for commonly known vulnerabilities was conducted by means of an automated test.
  - SM_VU.3.3: Analysis of policies (e.g. configuration policies) for commonly known vulnerabilities was conducted by means of automated analysis tools.

SM_OR: Unmanaged/managed objects ratio
For pragmatic reasons (ease of use, low start thresholds) unmanaged objects are allowed in the system. Unmanaged objects imply that some parts of the service are not measured. Subjective trust in respect to unmanaged infrastructure object's assurance level is used instead. The more important parts of the infrastructure are objectively measured, the higher the confidence for the service assurance can become.

Component levelling



Dependency: No dependencies

- ***SM_OR.1 Parts are assessed***
  - SM_OR.1.1: Parts of the service, formally not estimated regarding their importance, are defined as managed objects.

- ***SM_OR.2 Selected parts of the infrastructure objects are assessed***
  - SM_OR.2.1: A selection of the known important parts of the service, as estimated by an expert, is managed objects. Selected here means that the cumulated weight of the managed infrastructure objects represents more than 0.5.
  - SM_OR.2.2: The depth of the service modelling should reach at least infrastructure objects that are "servers" (e.g. Web Server, Mail Server, Radius server, Application server …).

- ***SM_OR.3 Significant parts of the infrastructure object are assessed***
  - SM_OR.3.1: Significant parts of the service are defined as managed objects. Significant here means that the cumulated weight of the managed infrastructure objects represents more than 0.75.
  - SM_OR.3.2: The depth of the service model should reach at least infrastructure objects that are "OS, application or similar" (e.g. Linux operating system, Apache Web server, SIP phone …).

- ***SM_OR.4 All infrastructure objects are measured***
  - SM_OR.4.1: All modelled parts of the service are defined as managed objects.
  - SM_OR.3.2: The depth of the service modelling should span at least infrastructure objects that are security relevant parts of an "OS, application or similar" (e.g. AC module of the OS, the SSL engine of a Web-Server …).

6.1.2.2 CLASS MC: Metric Construction

Metrics are a core component for the automated assurance monitoring system. Their capabilities influence the assurance evidence they can create. All families are primarily relevant for normalization. Scope, depth and rigor are also relevant for describing probes and interpretation.

	Families	Description
MC_SC	Scope	Broader scope gives more assurance
MC_DE	Depth	More details investigated gives more assurance
MC_RI	Rigor	More formalism gives more assurance
MC_SA	Stability	Stable measures in the evaluation window
MC_FR	Frequency	Frequent and fresh evidence give higher assurance
MC_TI	Timeliness	More recent tools are bound to find more problems and are therefore preferable
MC_RE	Reliability of metric	Higher reliability of the metric (including probes) gives better confidence

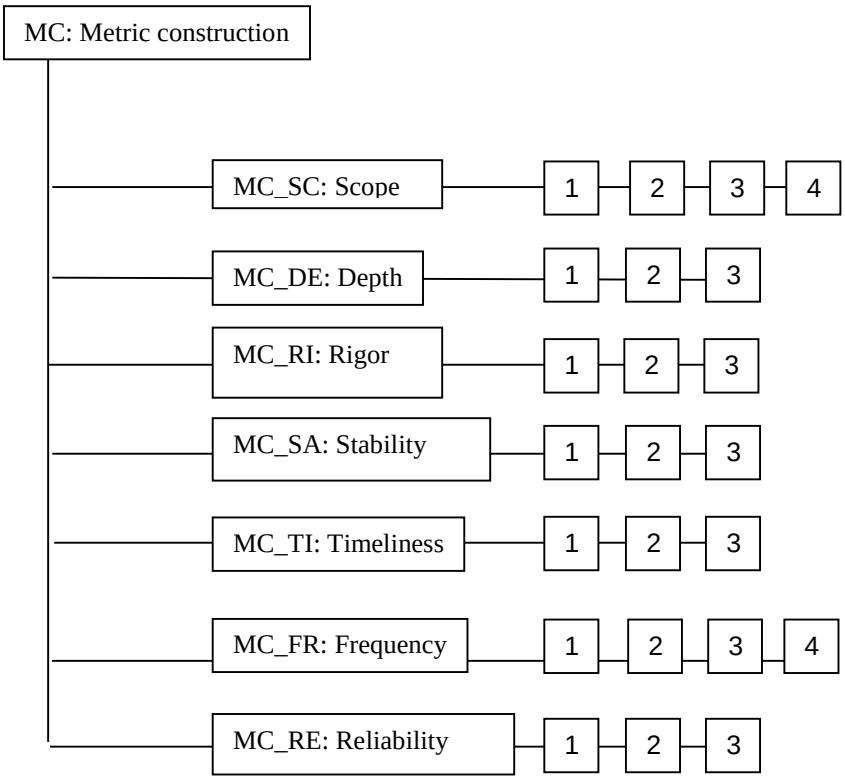
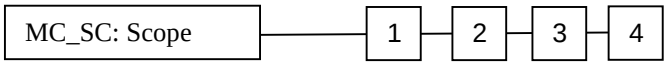


Figure 6 – MC: Metric construction class decomposition

MC_SC: Scope
The more elements of the infrastructure object are measured, the more it can be assumed that the metric result represents the infrastructure object.

Component levelling



Dependency: MC_DE

- ***MC_SC.1 Parts of the countermeasure realization(s) are assessed***
  - SM_SC.1.1: A part not formally estimated regarding its importance, which contribute to the countermeasure's correct functionality, are assessed to be operational (e.g. measured).

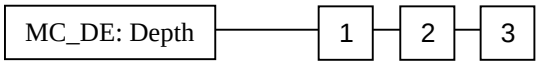
- ***MC_SC.2 Selected parts of the countermeasure realization(s) are assessed***
  - SM_SC.2.1: A selection of the known important parts, as estimated by an expert, which contribute to the countermeasure's correct functionality, are assessed to be operational (e.g. measured)..

- ***MC_SC.3 Significant parts of the countermeasure realization(s) are assessed***
  - SM_SC.3.1: All parts characterized as significant, by an expert, which contribute to countermeasure's correct functionality, are assessed to be operational (e.g. measured).

- ***MC_SC.4 All known parts of the countermeasure realization(s) are assessed***
  - SM_SC.4.1: All parts that can formally be considered as contributing to the countermeasure's correct functionality are assessed to be operational (e.g. measured).

MC_DE: Depth
The deeper the infrastructure object (or parts of it) is assessed, the more confidence of its correct function can be gained.

Component levelling



Dependency: MC_SC

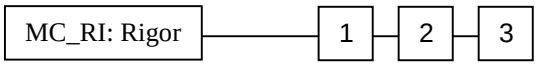
- ***MC_DE.1 The interface is assessed***
  - MC_DE.1.1: All assessed parts are measured via their interfaces.

- ***MC_DE.2 The whole chain down to the hardware is known but only relevant parts are assessed (top down) of it***
  - MC_DE.2.1: The whole chain down to the hardware it is running on is known (documented, maybe outside the service model).
  - MC_DE.2.2: Relevant parts are assessed via their interface.

- ***MC_DE.3 The whole chain down to the hardware is known and assessed***
  - MC_DE.3.1: The whole chain down to the hardware it is running on is known (documented, maybe outside the service model).
  - MC_DE.3.2: The whole chain is assessed via the interface.
  - MC_DE.3.3: For relevant parts (where possible) the assessment also investigates internal states.

MC_RI: Rigor
In this taxonomy we assume that measures reflect reality and could therefore be considered formal. With the metric process these base measures are transformed into an assurance expression. Different degrees of freedom are possible in the transformation process. A second issue concerns the proof that is provided for the correctness of the transformation algorithm.

Component levelling



Dependency: No dependencies

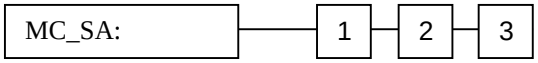
- ***MC_RI.1 unverified***
  - MC_RI.1: The transformation algorithm needs to be neither documented nor proved.

- ***MC_RI.2 informally verified***
  - MC_RI.2.1: The transformation algorithm is documented.
  - MC_RI.2.2: The transformation algorithm is shown to work in an informal way.

- ***MC_RI.3 semi-formally verified***
  - MC_RI.3.1: The transformation algorithm is documented.
  - MC_RI.3.2: The transformation algorithm is shown to work in a semi-formal (analytic) way.

MC_SA: Stability
Under the assumption that assurance levels for services are used to enable the extrapolation of future service security (assurance), results that are continuous are preferable for prediction. Under the assumption that metrics build the base for assurance prediction, it is necessary that they deliver stable results (i.e. they should not fluctuate when the monitored infrastructure objects parameters have not changed (i.e. inexplicable fluctuations) as this could indicate that parameters are treated wrongly (e.g. some parameters are missing, unrepresentative parameters are included).

Component levelling



Dependency: MC_FR, MC_RE

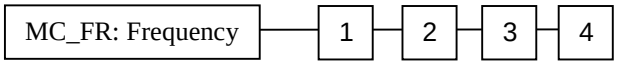
- ***MC_SA.1 stability not evaluated***
  - MC_SA.1.1: The stability of the metric is not evaluated.

- ***MC_SA.2 Perceived stability but no statistical proof***
  - MC_SA.2.1: The stability of the metric is evaluated by human observation (i.e. an expert indicates if the metric fluctuate in a way it should not).

- ***MC_SA.3 Statistical proofed stability***
  - MC_SA.3.1: The stability of the metric is statistically evaluated. Due to the normal fluctuation in the measured infrastructure object, the second order fluctuations (i.e. fluctuation that are not explicable by the changes in the infrastructure object) have to be investigated.

MC_FR: Frequency
In a constantly changing telecom environment the frequency of measuring influences the capability of aggregating and extrapolating the metric data.
Aggregation requires the semantic equivalence of the data. If the time delta between the data is too large the semantic equivalence weakens. A similar impact is expectable when it comes to extrapolation as older values might not reflect recent changes and the extrapolation works on a wrong trend.

Component levelling



Dependency: No dependencies

- ***MC_FR.1 Rudimentary (no frequency defined)***
  - MC_FR.1.1: The frequency with which measures are conducted is not controlled.

- ***MC_FR.2 Regular (at least every month)***
  - MC_FR.2.1: At least every month a measurement is conducted.
  - MC_FR.2.2: At least 12 past measures are available.

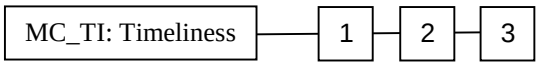
- ***MC_FR.3 Frequent (at least every week)***
  - MC_FR.3.1: At least every week a measurement is conducted.
  - MC_FR.3.2: At least 52 past measures are available.

- ***MC_FR.4 Continuous (at least every day)***
  - MC_FR.4.1:At least every day a measurement is conducted.
  - MC_FR.4.2: At least 100 past measures are available.

MC_TI: Timeliness
As metrics are also mainly software, different versions of them can exist. Under the assumption that more recent versions are improved in some way, it is concluded that a recent version is preferable over an older version.

Note:
- A metric that has changed the way it measures, interprets or normalizes is considered a different metric, and the model has to be updated to reflect the usage of a new metric.
- We consider that a metric remains the same even if a probe or a metric module has changed as long as the modification does not modify its external behaviour.

Component levelling



Dependency: No dependencies

- ***MC_TI.1 Not assessed***
  - MC_TL.1.1: A generic prerequisite for all metrics is that they have to correspond to a major version of the infrastructure object on whose measures they operate. If there is a divergence the assurance is negatively influenced.
  - MC_TL.1.2: The timeliness of a metric is not assessed.

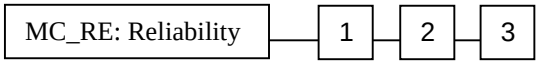
- ***MC_TI.2 Versions prior to the newest version***
  - MC_TL.2.1: A generic prerequisite for all metrics is that they have to correspond to a major version of the infrastructure object on whose measures they operate. If there is a divergence the assurance is negatively influenced.
  - MC_TL.2.2: The metric used is either the latest or the previous version. Within a three month time period a version that is two generations past the recent version is permitted.

- ***MC_TI.3 Most recent released version***
  - MC_TL.3.1: A generic prerequisite for all metrics is that they have to correspond to a major version of the infrastructure object on whose measures they operate. If there is a divergence the assurance is negatively influenced.
  - MC_TL.3.2: The metric used is the most recent version. Within a one month time period a version that is one generation past the recent version is permitted.

MC_RE: Reliability of metric
The reliability of metrics influences the confidence that can be put into the metric result. The reliability of a metric is composed of the reliability of the probes, the interpretation and the normalization algorithm. Criteria for the reliability are:

- In how far the result is reproducible with the same inputs (i.e. in how far the algorithm produces the same results given the same inputs);
- In how far the result is understandable in respect to the provided inputs (i.e. in how far a domain expert could come up with the same result);
- How consistent the result is (i.e. free from contradiction and relatively free form variation).

Component levelling



Dependency: MC_SA

- ***MC_RE.1 Reliability not investigated***
  - MC_RE.1.1: The reliability of a metric is not assessed

- ***MC_RE.2 Perceived reliability***
  - MC_RE.2.1: The metric should be reproducible, understandable and consistent, but there is no proof of reliability.
  - MC_RE.2.2: The reliability is informally assessed (i.e. experts judge the reliability).

- ***MC_RE .3 Proofed reliability***
  - MC_RE.3.1: The metric is proofed to be reproducible, understandable and consistent.
  - MC_RE.3.2: The reliability is semi-formally assessed (i.e. some kind of proof – analytic or argumentative – for reliability is provided.)

6.1.2.3 CLASS MM: Maintenance Management

An automated assurance monitoring system depends on the appropriateness of the components it is constructed from. The appropriateness is preserved by maintenance activities, which are performed on the system. The maintenance activities have to be guided by a maintenance policy.

	Families	Description
MM_PM	Probe maintenance	In which way are the probes maintained
MM_OM	Model maintenance	In which way is the model maintained

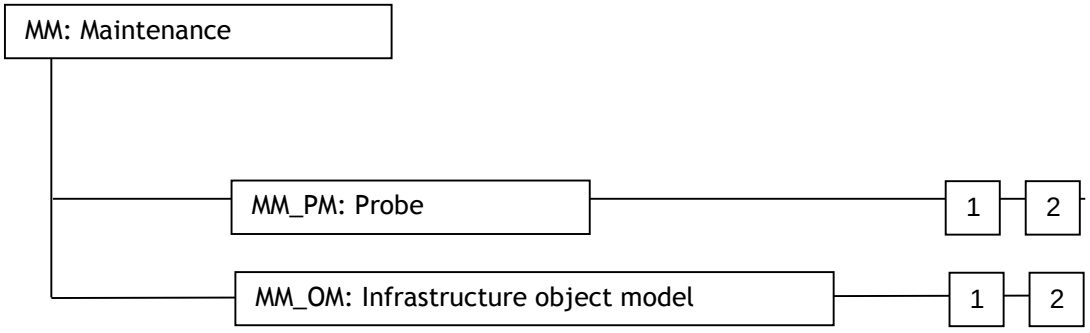
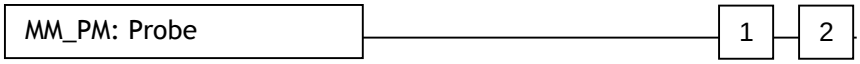


Figure 7 – MM: Maintenance Management class decomposition

MM_PM: Probe maintenance
With continuous assurance monitoring software based automated probes are used to gather measurements. These automated probes must be maintained to preserve their capability to derive evidence.

Component levelling



Dependency: No dependencies

- ***MM_PM.1 Reactive maintenance***
  Maintenance reacts to events.

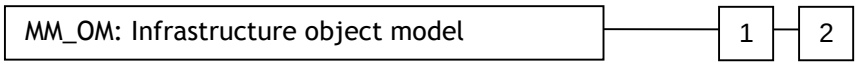
  - MM_PM.1.1: Maintenance is performed ad-hoc when a need is perceived.
  - MM_PM.1.2: In many cases no formal maintenance policy exists.

- ***MM.PM.2 Pro-active (preventive) maintenance***
  Maintenance is conducted regularly

  - MM_PM.2.1: The maintenance policy describes regular maintenance intervals in which the probes are controlled and possibly maintained.
  - MM_PM.2.2: Documentation of the maintenance activities indicates the appropriateness of the probes.

MM_OM: Infrastructure object model maintenance
The infrastructure object model represents the service and is crucial for correct assurance measuring and aggregation. It is therefore important that this model is maintained to preserve the correct reflection of the service.

Component levelling



Dependency: No dependencies

- ***MM_OM.1 Reactive maintenance***
  Maintenance reacts to events.

  - MM_OM.1.1: Maintenance is performed ad-hoc when a need is perceived.
  - MM_OM.1.2: In many cases no formal maintenance policy exists.

- ***MM_OM.2 Pro-active (preventive) maintenance***
  Maintenance is conducted regularly

  - MM_OM.2.1: The maintenance policy describes regular maintenance intervals in which the model is checked for conformity with the service and eventually maintained.
  - MM_OM.2.2: Documentation of the maintenance activities indicates the appropriateness of the model.

6.2 The resulting six step methodology overview

The operational methodology can be described in six main steps. Each step addresses a specific action in order to build security assurance:

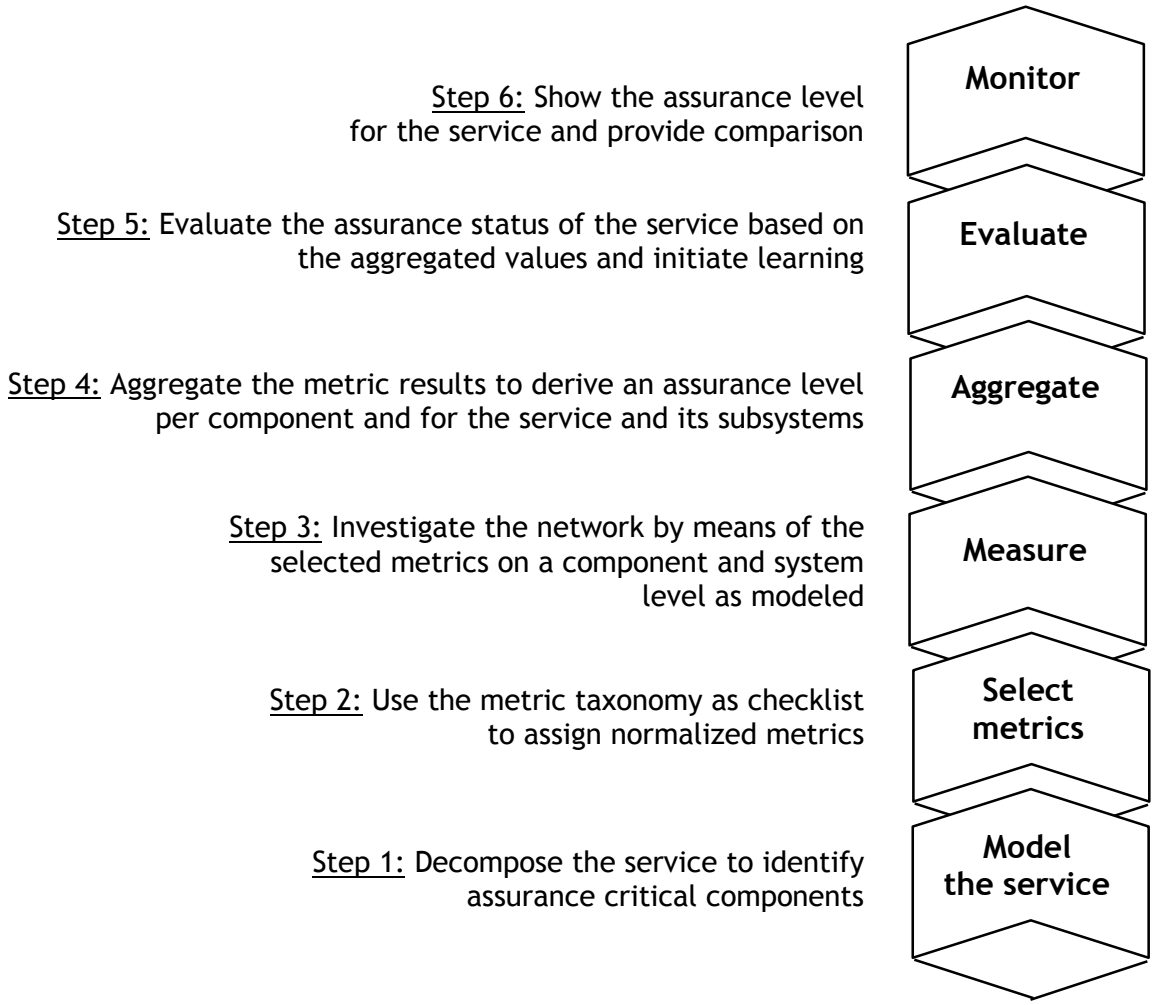


Figure 8 – The six steps methodology

The first step concerns **service modelling**. This step is crucial to reflect security assurance needs of the operated services. The modelling allows decomposing the service in order to identify assurance critical components. In order to realise this phase, a dedicated security assurance model has been defined (BUGYO D1.2 [1]).

The second step is addressing the **selection of metrics**. The selection of metrics will define the profile of the measured assurance. It requires pertinent choices, especially in selecting the targeted assurance level of metrics. A low level will provide less confidence but will be easier to implement while high level of assurance will lead to higher confidence but will imply a higher effort in deploying, monitoring and maintaining the associated probes. In order to realise this step, a formalised process has been defined transforming raw measured data in a normalised security assurance level.

The third step concerns **measurement**. This aspect concerns network investigation by deploying selected probes implementing metrics. Choices made in step 2 will directly

impact this third step. This step impact is local, i.e. it concerns local measurement of the security assurance.

The fourth step enables to **aggregate** these local measurements at service level. This aggregation is derived from the service modelling made in step 1.

A multi-agent platform and a centralised server provide a measurement infrastructure (implementing third step) and multiple aggregation algorithms (implementing fourth step).

The fifth step provides **evaluation** of the assurance status based on the result of the aggregation step. This step is crucial for the operated service as it measures deviation, evolution and by consequence determines management actions that need to be taken in order to maintain targeted security assurance level. In order to provide evaluation means, five assurance levels have been defined enabling the expression of the actual security assurance.

The last step provides means to the operator to **monitor** security assurance status both at service and network infrastructures objects, determine causes of assurance deviation, and consequently provide assistance for security management. In order to realise this step, an easy-to-use centralised monitoring console has been defined: the security cockpit. This console provides three main functions: measure (the security assurance level), monitor (the security assurance level), and assist (to maintain security assurance level).

Each step is precisely defined in the following sections.

6.3 Service modelling

Modelling requires identification of different present entities, classification and establishment of document formats for all entities involved in the service delivery. This is not principally different from any other modelling technique. The scope of this modelling is a logical service-bound abstraction. The product modelling technique, dedicated to security assurance, provides the user with a comprehensive path on how to model the system under study (infrastructure/architecture) without going into unnecessary details (hardware, variables, etc.). It provides guidance, gives a good degree of freedom and allows elimination of service-unrelated entities. The model service viewpoint is refined towards the network architecture.

Only assurance relevant network objects should be modelled. This means that the model does not reflect the whole network but only critical elements that are important and need to be security assured towards operator business continuity. In this respect, the modelling effort should reflect business impact of any failure due to a security concerns but neither infrastructure architecture nor quality of service approach.

Assurance information is explicitly modelled as a tag in every relevant object. The model provides three different views: topology, hierarchy and flows.

In the operational point of view, we suggest to use the hierarchy view of the service security. This view indicates how the infrastructure objects contribute their assurance information towards the service. The first step is to decompose the service into main sub-components.

Due to the fact that we consider assurance aggregation, we can also start adding the first weights to indicate how much importance we assume the different components have towards the super object (i.e. the service). The next refinement step is to start modelling each branch of the service in more detail. At this level, it seems reasonable to start considering metrics and adding them to the hierarchy view. Owing to the hierarchical nature of the modelling, each of these steps can be repeated until the resulting model reaches a sufficiently realistic representation of the service.

The product methodology does not provide any specific method on how to achieve and validate the modelling but some guidance is given below. It assumes that an expert is performing the modelling.

As mentioned in the D2.1 document [3], using results of risk and impact assessment seems a reasonable way to attribute weights and to consider where metrics should be added. A typical risk analysis will result in the identification of risks within the infrastructure along with the assessment of their importance and severity in case of breaches. Location and severity of an impact will lead to the component in the modelling and its relative weight. Countermeasures deployed on each critical element will also lead to the determination of suitable metrics.

Risk assessment leads to, and validates the identification of critical elements.

Impact assessment leads to, and validates the identification of the hierarchical structure of the model and the affectation of related weights.

Countermeasure deployment leads to, and validates the identification of the assurance metrics.

A V-model should be considered in describing the service model and the validation.

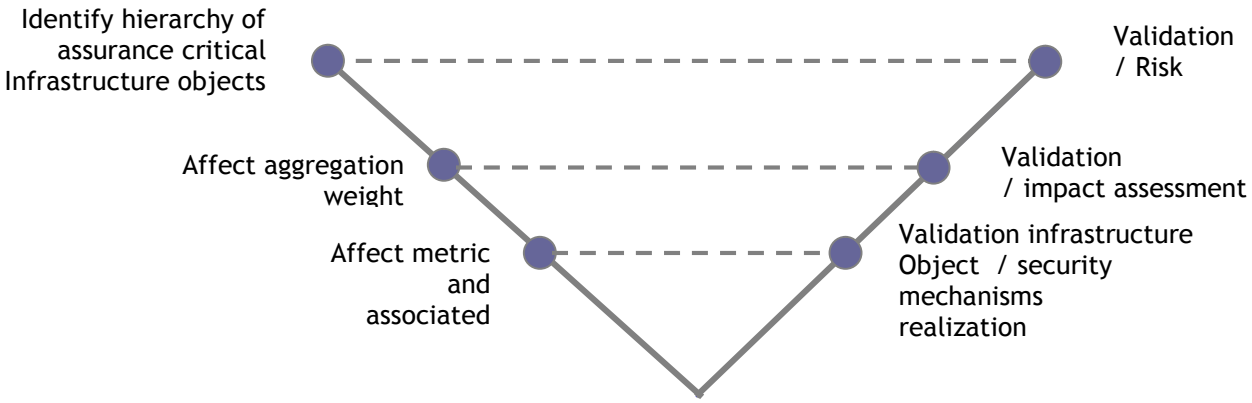


Figure 9 – Modelling and validation

6.4 Metric

Within the product project, a metric is a modelling object defined as the process that allows producing a normalised assurance level for an Infrastructure Object. A description of this process is proposed in the present section.

A metric is based on the measurement of various parts/parameters of security functions implemented on infrastructure objects that compose a network service. One metric is linked to one and only one infrastructure object while one infrastructure object can be linked to several metrics.

In order to design those metrics and to guarantee that they can produce normalised assurance levels, we think it is mandatory to propose a decomposition of the process of a metric. Our proposed decomposition is presented in the Figure 8.

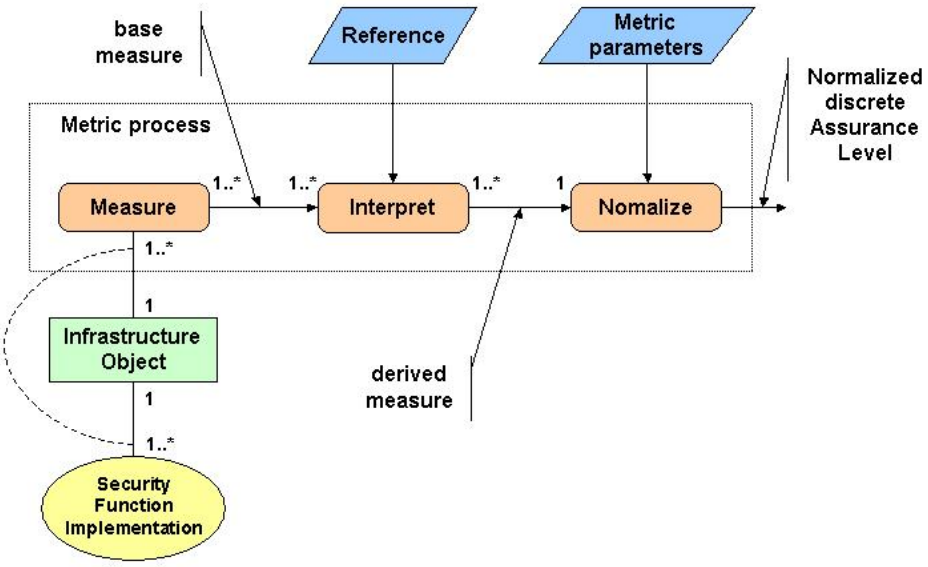


Figure 8 – The proposed metric decomposition

This decomposition helps us to simplify the problem of normalising the raw measurement results or **base measures** (in reference to ISO [2] terminology). By interpreting the interdependent base measures of the metric we produce independent **derived measures** (in reference to ISO [2] terminology). Finally, those derived measures are composed in order to produce a normalised and infrastructure related discrete Assurance Level.

6.4.1 Measure

The first step in the metric is to the process of **measuring** the Infrastructure Object (cf. Figure 9). That means getting **raw data** from the system (an Infrastructure Object) without any kind of comparable format and scale. The measuring stage produces at its output what we call **base measure** in reference to the terminology of ISO [2].

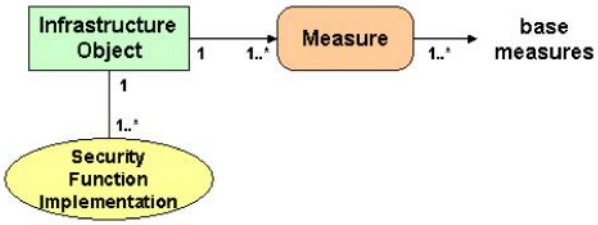


Figure 9 – Measuring the Infrastructure Object

The measurement stage can be performed in two different ways depending on whether the expert that develops the **probe** has a complete knowledge of the security function implementation or not:

- **White box measurement**: in that case, the expert has the complete knowledge of the security function implementation that composes the Infrastructure Object. This knowledge permits to perfectly fit the probe with the implementation and then to collect the most accurate data.
- **Black box measurement**: if the expert does not have the knowledge of the security function implementation specificities, the probe will collect data upon standard implementations. The accuracy of the collected data cannot be certified.

If base measures collected by white box measurement can be considered as more accurate than those collected by black box measurement, the independence of the probe provider has to be taken into account.

Because white-box measurement usually requires the implication of the security function developer in the development of the probe, such measurement must usually be considered as self-measurement. Third-party measurement (i.e. requiring the use of a probe developed by an organisation independent from the security function developer) is preferable to obtain higher assurance.

The base measures that can be produced by off-the-shelf probes, proprietary probes or the system itself need to be interpreted using a **reference** in order to produce meaningful information.

In order to help the process in the following stages of the metric, it is possible to define a classification of the base measures. For instance, we can decide to classify the measures using a three-dimensional basis:

- **Availability** relates to a measurement that monitors whether a counter-measure exists and is available.
- **Conformity** deals with a measurement that controls configuration conformity of a counter-measure.
- **Vulnerability** relates to a measurement that controls the counter-measure is not vulnerable itself.

6.4.2 Interpretation

The interpretation stage takes as its entries base measures on particular Infrastructure Object and reference information in order to produce **derived measures** that will be compared at a higher level in the metric. A derived measure is a complex structure of several values derived from the interpretation of one or more base measures.

Base measures by nature are raw data that are not necessarily using the same result scale or format. In another paradigm, some base measures can be **correlated** (i.e. interdependent) even if they do not relate to the same properties of the security functions parts implemented on an Infrastructure Object. For example, a base measure that gives information on the availability of a security function implementation can be of a major importance compared to base measures that carry compliancy information for the same, or another, security function implementation.

Knowing these facts, we propose to address the interdependency and scaling issues between base measures, by decomposing the interpretation stage into two sub-stages as presented in the Figure 10:

- The first sub-stage of the interpretation allows the **scaling** of base measures. By either comparing to a policy (where expected values are codified) or to a generically available result scale the base measure is induced with meaning.
- Then, the second sub-stage correlates scaled base measures in order to produce synthesised results at the output of the interpretation stage. In correlation synergies, contradictions and dependencies between scaled base measures are exploited to gain higher order information.

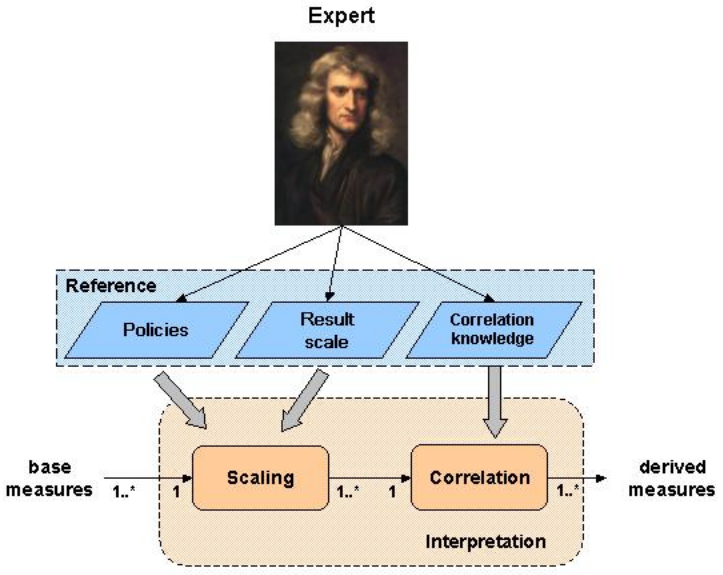


Figure 10 – Interpreting base measures

Carrying the interpretation this way, allows obtaining at the output of the interpretation stage independent (i.e. uncorrelated) and scaled derived measures that can be more easily composed (using the assurance taxonomy) in order to produce a unique result through the normalisation process.

6.4.2.1 Scaling

The primary goal of the scaling is to rationally express correlated results of base measures using a common scale.
In order to perform the scaling, additional reference information is needed. A **reference** can be either composed of **policies** (any kind of rule collection related to the security of the infrastructure, like a *security policy* or a *vulnerability patching policy* for instance) or a default result **scale** (a set of values that enable to assess an impact of the base measure values in terms of security). The design of the scaling process should be devolved to a Security Expert who can balance the base measures, in terms of absolute value and interdependency, with the business implication (a cost for instance) of the results on the global infrastructure and services.

The scaling will be dependent on the application area, the tool and the environmental information. For instance, if the derived measures are produced using a vulnerability-testing tool, it is likely that the result will be scaled by defining several ranks of severity like CERT does. On the other hand, if the base measures are produced by custom-made scripts testing the availability of a counter-measure, it is more likely that the scale will be a binary or ternary one (available, not available and undefined for instance). Other alternatives are scales that are ordinal or rational in nature.

Two things are important when defining the derived measure scales for a metric. First, the Security Expert community must admit a scale, so that the interpretation of the base measures makes sense. Second, the interaction between each level of each scale used to interpret the base measures has to be clearly expressed in the metric definition.

Some examples of scales for specific kinds of derived measures can be found in Appendix A of the document.

6.4.2.2 Correlation

The correlation stage corresponds to the process that expresses the relation between several measures (base measures or measures at the output of the assessment stage) that are in some way correlated. By correlated we mean the information given by a measure can confirm (neutral correlation) or contradict (negative correlation) the information given by another. In order to rationally express the way the measures are correlated, a deep knowledge of the underlying security mechanism is needed. This **correlation knowledge** enables to produce a derived measure based on the input measures. The Security Expert responsible for the design of the metric must be able to provide this correlation knowledge.

6.4.2.3 Example for scaling and Correlation

In order to illustrate how the scaling and the correlation stages can be interpreted, we provide an example.

Our case deals with metrics on two authentication servers both composed of a RADIUS daemon and a LDAP daemon on the same network element. These authentication servers are part of a authentication countermeasure. The authentication servers, representing the decision part of the authentication countermeasure, compose a cluster of two nodes (i.e. two physical servers) providing the same RADIUS authentication service. The cluster uses a watchdog on both nodes to manage the failover strategy. The clustering watchdog, the RADIUS daemon and the LDAP daemon on a cluster node can be

considered as three modules related to the **authentication security function** that is implemented on a particular node. On each node of the cluster, a **metric** is deployed. Several base measures are performed both on the RADIUS daemon instance, the LDAP daemon instance and on the clustering watchdog instance. Some of them give availability information on the part of the authentication countermeasure implemented on the nodes.

Some base measures of the metric on the authentication cluster nodes are related to the RADIUS daemon:

Base measure Id	Description
BM1	The first base measure is a particular vulnerability-checking test performed with a SATAN-like probe. This kind of tool checks the version number of the software through the network and compares it with a vulnerability database. This base measure checks if the RADIUS server is vulnerable to a particular DoS attack, which stops the RADIUS process in case of success. NOTE: For an attack to succeed using the vulnerability checked by the above base measure, the RADIUS software must be of a particular version and the configuration file of the software must have one of its parameters set to a particular value. This kind of information is usually provided in the vulnerability announcement like e.g. CERT alert.
BM2	The second base measure, related to the same DoS vulnerability, is performed on the configuration file of the RADIUS server. It checks if the parameter involved in the vulnerability is set to the specific value that would allow the exploitation of the DoS. For instance, a probe that globally checks the configuration file of the RADIUS daemon can perform this test. This probe could take as input CERT alert information relative to configuration parameters.
BM3	A third base measure is performed by a network IDS probe that detects if an attack is tried on the network to exploit the DoS vulnerability described above.
BM4	A base measure listing the running RADIUS processes. Usually a running RADIUS server has several processes for the RADIUS functionality. One of those processes is dedicated to the monitoring of the others. The monitoring process is designed to restart the others in case of halting.

- **Scaling**

The scale composed of the set of values *{None, Weak, High, and Critical}* is defined so that a total order exists on the values of the set: *None* < *Weak* < *High* < *Critical*. Thus, a predicate function can be defined on this set of values, like *max* and *min*.

Due to the nature of the base measures described above, we can use the same scale for the assessment. The scale chosen here is similar to those used for CERT alerts to value

the threat level of vulnerability. This scale is a set composed of the values described in the following table:

Threat value	Comment
None	No known risk exists
Weak	A theoretical risk exists but practically unlikely
High	A likely risk exists
Critical	A highly probable risk exists

Since these base measures are related to availability of the authentication countermeasure, it is clear that they are interdependent. The four base measures have to be correlated together to give an accurate and reliable measure of the confidence that the authentication countermeasure is available.

- **Correlation**

Depending on the results of the four base measures, the risk that the RADIUS authentication server is unavailable because of a security issue is different. Using the above scale, we are able to illustrate the resolution of the correlation between base measures:

Derived measure Id	Derived measure value	Base measure Id	Base measure value
DM(1,2,3,4)	None	BM1	Installed RADIUS server version is not vulnerable to the DoS
		BM2	Any
		BM3	Any
		BM4	Any
	None	BM1	Installed RADIUS server version is vulnerable to the DoS
		BM2	RADIUS configuration does not allow to exploit the DoS vulnerability
		BM3	Any
		BM4	Any
	Weak		Installed RADIUS server version is vulnerable to the DoS
			RADIUS configuration allows to exploit the DoS vulnerability
			No exploitation attempt detected
			All RADIUS processes are up and running
	High	BM1	Installed RADIUS server version is vulnerable to the DoS
		BM2	RADIUS configuration allows to exploit the DoS vulnerability
		BM3	No exploitation attempt detected

		BM4	The monitoring process is not running
	Critical	BM1	Installed RADIUS server version is vulnerable to the DoS
		BM2	RADIUS configuration allows to exploit the DoS vulnerability
		BM3	Exploitation attempt detected
		BM4	Any

NOTE: in our example, the result of the correlation is a simple value, but it is not necessarily the case.

6.4.3 Normalisation

Normalisation transforms the derived measures, an incomparable assurance expression, into a comparable discrete assurance level. The normalisation is a critical activity within the product methodology and serves the following purposes:

- It allows subsequent aggregation by establishing semantic equivalence for the assurance expressions.
- It takes away the environmental dependency of an assurance expression.
- It makes assurance comparable
- It allows a ranking of different assurance expressions.

We use here a three-step normalisation process as shown in Figure 11. This process aims to encapsulate similar "non-mechanistic" procedures into coherent activities. Subsequently we describe each activity in more detail.

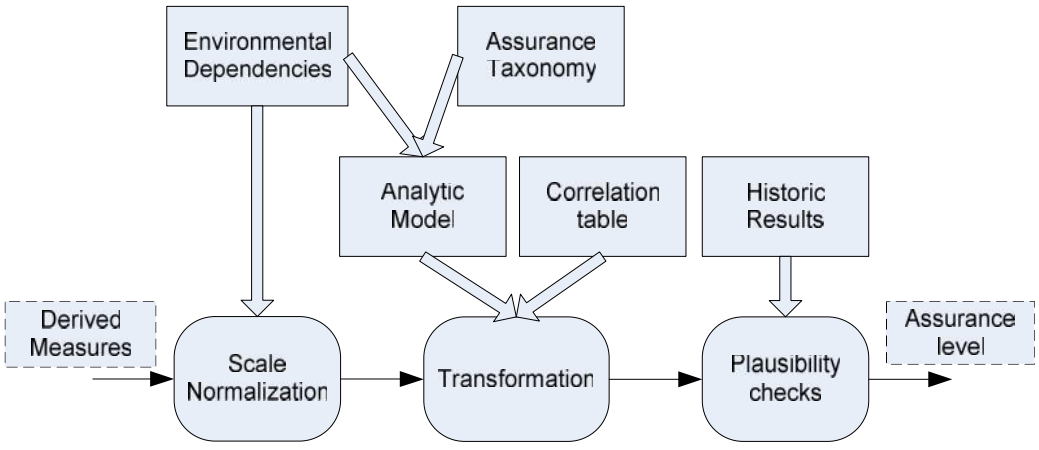


Figure 11 – Normalisation process

The **scale normalisation activity** aims to normalise the incomparable scales, due to environmental dependencies, with which assurance is expressed in derived measures (as output of Interpretation). Due to semantic inconsistencies, keeping the environmental dependency would prevent aggregation and would make it difficult to compare assurance levels. It can therefore become necessary to create an environmentally independent scale spectrum. This spectrum distributes the possible measure values over a normalised scale that is independent from the environment. The normalised scale has to conform to the basic intention of the assurance taxonomy (as presented in section

6.2.1) and provide a "0"-point to mark the total absence of assurance. Noteworthy is that, although we define AL5 as the maximum assurance level, a scale does not need to have an upper limit (i.e. if the AL5 threshold is reached, assurance can still increase).

Example:

We imagine a metric that can measure up to AL5 and will eventually cover a broader spectrum in its scale than one which can measure up to AL3. If we assume that the AL5 capable metric measures between 0-1000 on an "Asterix" scale (i.e. interval scale) and the AL3 capable metric between "A-D" on an "Obelix" scale (i.e. ordinal scale) we can see that the mapping has to be different.

During **transformation** an analytic model is used to derive the assurance level expressed by the metric based on the measurements. The analytic model contains a number of algorithms (e.g. set of rules) that describe which measured values correspond to which assurance level. These algorithms encapsulate a lot of complexity and must be carefully constructed and verified (see below). We consider that the analytic model can take static and dynamic input parameters.

The static parameters are primarily considered during the creation of the analytic model. Important categories are parameters from the environment, the assurance taxonomy and the metrics domain. When creating the analytic model, the first step is to identify the range of assurance levels that shall be covered by the model (i.e. is the metric to be used up to AL5 or up to AL1). It is important here to not only split up the scale evenly (i.e. by building an average), but to take into account the logarithmic structure of the assurance levels as specified in the assurance taxonomy. Environmental information has to be included when deciding the value ranges from the derived measures that are associated to each assurance level. In case further measures are necessary to create the analytic model this has to be considered as feedback that requires reworking the base measurement selection and interpretation design.

To design the algorithms of the analytic model the assurance taxonomy is also an excellent starting point to roughly structure the metric domain parameters. From a practical point, we recommend to enumerate the possible assurance levels for each assurance taxonomy family and fill it with the domain specific parameters, which shall correspond to the intended assurance level (see example below). It is likely that not all families will contain parameters. In addition to the families from the taxonomy, additional metric domain specific categories can become necessary – i.e. result specific mapping. However, rule based approaches are only one option and we think that other approaches, like formal analytic modelling (e.g. stochastic model, differential equation systems or simulation techniques), are imaginable.

Dependent on the aimed assurance level, the analytic model has also to satisfy various demands from the assurance taxonomy that have to be verified after construction[3].

Example:

We would like to create an AL3 capable metric that is based on the Nessus tool. According to the documentation, Nessus provides different categories of checks that allow for classification. Note that the here provided presentation is an example and intentionally neither motivated nor complete

[3] When a formal certification for the metric is sought special focus shall be put on the normalisation and the verification of the normalisation.

Scope

AL1

2 Domain plug-ins are used (enumeration of the plug-ins)

1 Handcrafted probe is used

AL2

5 Domain plug-ins are used (enumeration of the plug-ins)

1 Handcrafted metric is used

AL3

8 Domain plug-ins are used (enumeration of the plug-ins)

1 Handcrafted probe is used

Depth

AL1

Probes are non-DOS and safe-checked

Scans are performed with user rights

AL2

The port scanner used to identify alternative ports for services

Some probes are allowed to be DOS but all are safe-checked

Scans are performed with user rights

AL3

The port scanner used to identify alternative ports for services

Some probes are allowed to be DOS or “unsafe”

Scans are performed with power-user rights

Timeliness

AL1

-

AL2

The most recent plug-ins are installed (difference in last update made)

The latest version of the scanner installed (difference in age)

NMAP in its last version is used

AL3 (as AL2)

Frequency

AL1

A scan is performed once a month

AL2

A scan is performed once a week

AL3

A scan is performed every day

Stability

AL1

25% of the results are false positives

AL2

15% of the results are false positives
The probes deliver within a normal distribution
AL3
5% of the results are false positives
The probes deliver within a normal distribution
Result specific:
AL1
Max. 4 serious and 10 non-serious vulnerabilities are found
AL2
Max. 2 serious and 5 non-serious vulnerabilities are found
AL3
Max. 0 serious and 3 non-serious vulnerabilities are found

The dynamic parameters are received from the many different derived measures. Although it is possible to handle correlation during the model construction, we recommend dealing with correlation during runtime because this allows correlating normalised results with similar algorithms as in aggregation. Naturally this implies a loss of accuracy, as the granularity of correlation is lower. However, we let it open to a metric provider to choose to handle correlation with better granularity in the analytic model (e.g. by means of Bayesian algebra).

The final activity in the normalisation process is a **plausibility check**. This check serves as a quality assurance activity to prevent unexpected derivations due to mistakes in the analytic model. This activity works as an explicit checkpoint to satisfy the stability requirement from the assurance taxonomy. As indicated in the taxonomy, this activity will be primarily relevant for metrics that are to satisfy high assurance levels. This activity is mandatory for AL5 and we recommend using it for AL4 and sometimes for AL3 (but at these levels the historic results are expert derived and not statistical).

The check uses reference variable sets for the assurance levels (as defined during the metric construction) to check that the actual calculation is within expectable limits. We can distinguish between two generic cases:

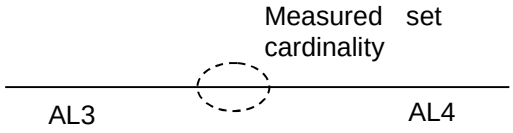
- If the measured variables are the same as the reference set, the same result must be calculated.
- If the measured variables are different to the reference set, an extrapolation, taking the set cardinality as a base, allows ordering. In other words, if the measured set's cardinality is between two reference variable sets, the calculated assurance level shall be in the same variable set.

$$|Reference\ A| < |Measured\ Set| < |Reference\ B| \rightarrow AL_A \leq AL_{Measured} \leq AL_B$$

Equation 3

Example:

Let us assume that the set cardinality (a set cardinal expression for the variable set used during normalisation) is located between the cardinality typical for AL3 and AL4 so shall the analytic model derive AL3 as the assurance expression.



6.4.4 Certified metric

In order to measure the security assurance, the operational product methodology relies on metrics that give an assurance level from base measures of the Infrastructure Objects. Our process to determine an assurance level at metric level uses criteria. Some criteria are related to the intrinsic design quality of the metric, while others are related to the current values of the base measures on which the metric relies.

It is therefore important to have means to evaluate the intrinsic quality of a metric in a **reproducible** and **repeatable** way in order to provide increased confidence in correctly designed metrics. A means to guarantee such properties is therefore to verify the metric process and award certification to a metric if it complies with well-defined certification standards. A metric that has completed a certification (i.e. a compliance verification) becomes then a **Certified metric**.

A **Certification process** is to be defined in order to certify a metric. This process enables to determine for each metric a theoretical maximum assurance level that the metric could deliver if all the base measures are delivering perfect values from the security assurance point of view. This process will guarantee that two results of a metric (i.e. two derived assurance levels) can be aggregated to a higher level.

This process can be compared to the Common Criteria certification process, i.e. to the accreditation according to ISO/CEI 17025:2005 “General requirements for the competence of testing and calibration laboratories” (previously EN 45001) of Commercial Licensed Evaluation Facility Laboratory (CLEF). Each lab that wants to perform Common Criteria evaluation needs to be accredited up to a certain level of Common Criteria by the national CC certification body in relation with national accreditation authority (e.g. COFRAC, COmité FRançais d’ACcréditation, in France). Therefore, for instance some labs are only authorised to perform evaluation up to EAL4 because they have no skills to perform formal testing, etc.

This process is out of scope of this project and should be carried on with a standardisation body in conformance with security and governmental agencies and/or certification bodies.

6.5 Aggregation

6.5.1 Operational aggregation

As stated in the general aggregation part (see section 5.3), the aggregation problem is important in order to know to which degree and through which relation a given measured property of the system delivering the service, will impact the service's security assurance.

The operational aggregation provides a way to aggregate metric outputs (normalized measures, i.e. assurance level) to obtain an assurance level per infrastructure objects and for the service and its subsystems by using achievable and simple aggregation algorithms. In a simplified way, this is addressing aggregation of results AL1-AL5 of each metric for each infrastructure object and towards the service.

The operational aggregation function for the security assurance of a service is a function of the produced security assurance service model, which means that the modelling is central to the aggregation function derivation, since it has to produce a security assurance specific view on a service realisation (see section 6.3).

Note: for specific infrastructure object called unmanaged infrastructure objects, there is no need for aggregation function to determine the security assurance level, as it is already given without (known) metrics.

6.5.2 Operational Aggregation requirements

The objective is to describe achievable product aggregation at the operational level. In other words how to operate the aggregation of the normalized values of the metrics measures taking into account their weights and also the associated subsystems. This mechanism elaborates a processing of all levels (infrastructure objects) until arriving to the top level of the hierarchy, which represents the service.

In order to achieve this objective, operational aggregation functions/algorithms consider the following criteria:

- The aggregation should be associated to one service description;
- It is seen as a sort of static algorithm and only one algorithm at the same time;
- The results for a given service description should be comparable over the time;
- BUGYO framework should not rely on one specific aggregation algorithm;
- The aggregation algorithm should be simple as possible in order to avoid intensive computation;
- The choice of the aggregation algorithm for a given service description should be specified by the expert;
- Only one value at the highest level is displayed.

6.5.3 Proposed Operational Aggregation algorithms

Before proposing any specific operational algorithms, as the aggregation function should be applicable to the product specific assurance model, we want to remind here basic principles of the hierarchical representation of the model. This model has defined weights as in the following table (extracted from D1.2). The weight is in fact representing relative importance of each sub components in the model, or relative

importance of each metrics in determining the security assurance level of an infrastructure object.

Weight type	*Formula*	*Description*
Assurance level weights	$mw + \sum_{i=1}^{n} cw_i = 1$	The weight of the assurance level derived by metrics (mw) plus the weight of the assurance level of the sub-components (cw_i) equals 1
Metric weights	$\sum_{i=1}^{n} mw_i = 1$	The sum of weights of all metric (mw_i) that determine the assurance level of an object equals 1

Table 3 – Weights used in the operational aggregation function

Those properties lead naturally to use a weighted sum as the operation aggregation function. Nevertheless, depending on a given context, we believe that two other algorithms have to be proposed as alternatives. We will describe those three algorithms in the following sections and will compare their outputs using a very simple service assurance model as presented in Figure 12. Aggregation algorithms will determine and indicate the aggregated assurance level ***{AL:?}***. As mentioned before the unmanaged infrastructure object is not concerned by the aggregation, as its assurance level has to be estimated.

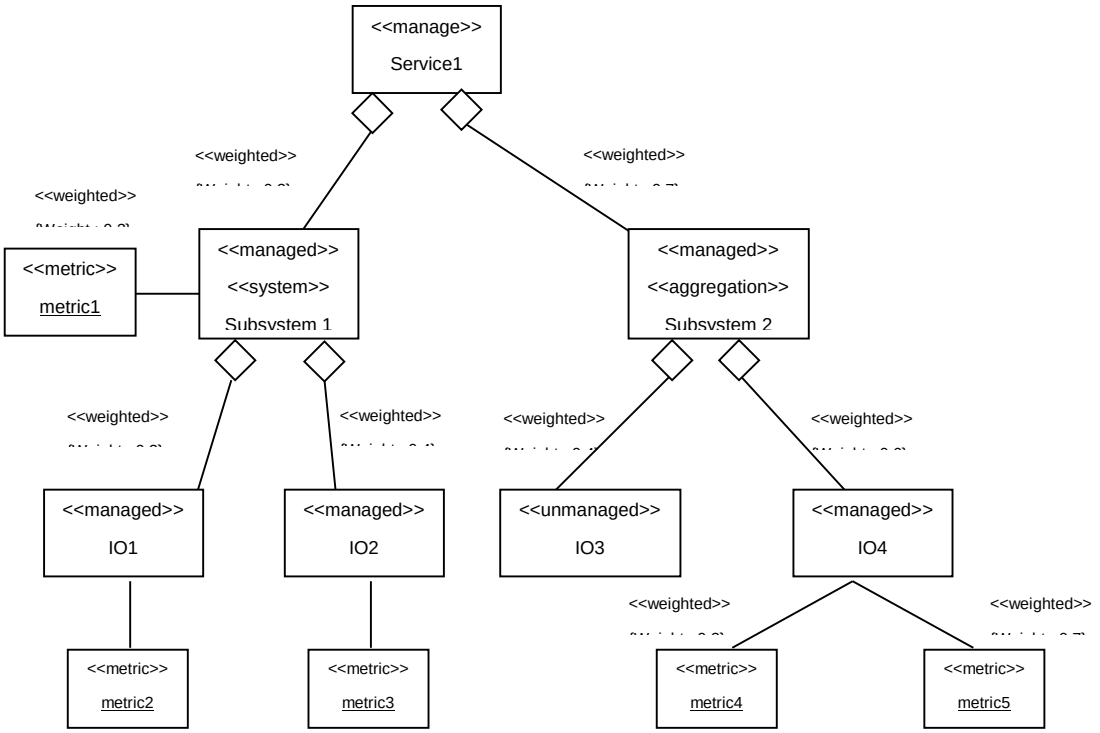


Figure 12 – Service assurance level model

All algorithms, presented below, assume that the actual model is coherent (i.e. properties described in weights table are correct). If there is an error in the model, therefore, the aggregated assurance level might give some non-correct values.

6.5.3.1 Recursive Minimum algorithm

The Minimum is a pessimistic algorithm. Indeed, if the assurance level of any metric is null, the global service assurance level will be also null. This algorithm makes no difference between sub infrastructure objects. Indeed, this algorithm might be requested by some operators, which have very strong security assurance expectations with all metrics at the same level. In this case, the weights are not required in the model. One potential usage of this model can occur when the product is deployed at the first time or when operators want to deploy assurance metrics but without any strong idea on relative importance of each infrastructure object taken part in the service.

The recursive minimum algorithm is:

```
For each service,
      Begin
            AL =  minimum AL of each sub component
                  for each sub component until metric
                        take the minimum
                              of each metric of each sub component
                        end
                  end
      End
```

Based on our example (see Figure 13), this algorithm gives the following service assurance level reaching only level 1.

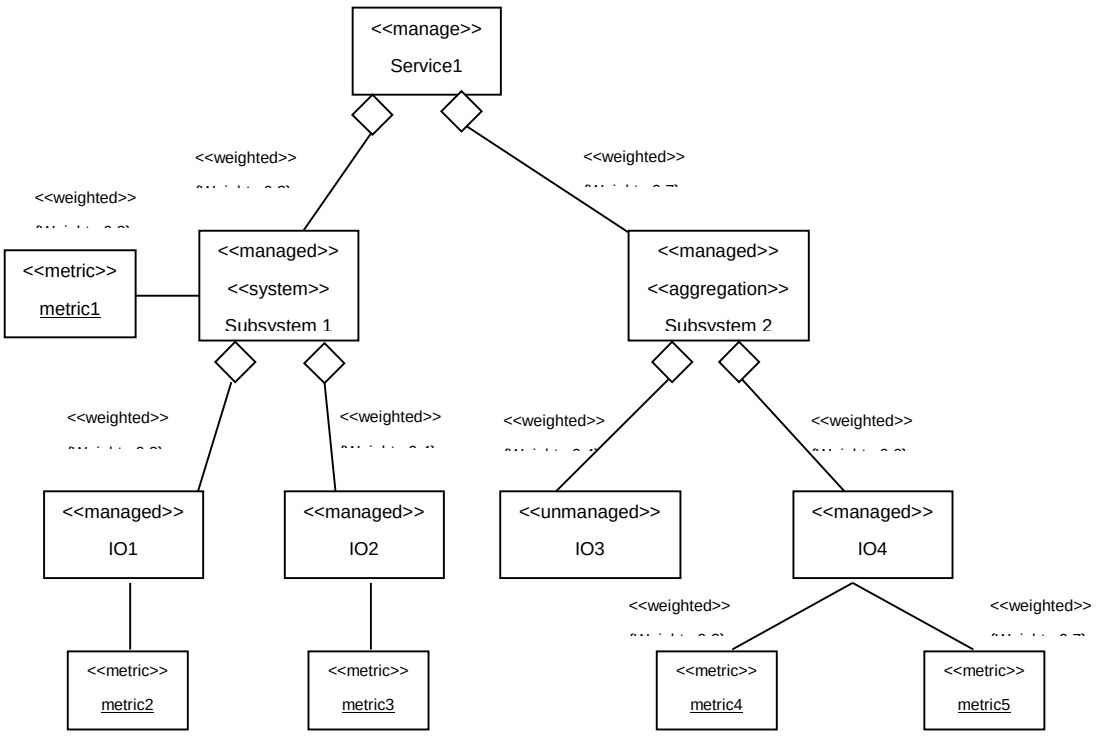


Figure 13 – Instantiated Model (Minimum)

The minimum algorithm is similar to the composition mechanisms for Common Criteria certification. In fact, the assurance level of a composition of several certified product can only be at the minimum of the composed assurance levels.

The main problem of this algorithm is that some changes can be hidden when monitoring the service. If the operator monitor service only at the service level any change above the minimum will not be reflected in the overall service assurance level.

The threshold mechanism described in section 6.8 avoids this major problem.

6.5.3.2 Recursive Maximum algorithm

The Maximum is an optimistic algorithm. Indeed, if the assurance level of any sub infrastructure object is high, the global service assurance level will then be high. In this case, weak assurance levels will be completely ignored. Thus, it is not an optimal way to aggregate assurance levels. In fact, no difference between sub infrastructure objects is done.

The recursive maximum algorithm is presented as follows:

```
For each service,
      Begin
            AL =  maximum AL of each sub component
                  for each sub component until metric
                        take the maximum
                              of each metric of each sub component
                        end
                  end
      End
```

Despite the fact that this algorithm is not optimal, this could be in some cases requested by the operator in order to know which maximum assurance level is reached regarding their infrastructure. This value might be a marketing argument to show that one or more services have made very high assurance level assessment. Of course, this could hide some infrastructure objects with no assurance at all. In that respect, this algorithm, as well as the minimum one, is comparable to the Common Criteria assurance aspect, where assurance can be the maximum on some very limited parts of the infrastructure, and leading the overall assurance level to high values. This mainly happened for Microsoft Windows product where high assurance level is obtained from a very limited part of the product; nevertheless all products claimed high assurance level.

Considering our example, the service reached a level 4 for the assurance level (see Figure 14).

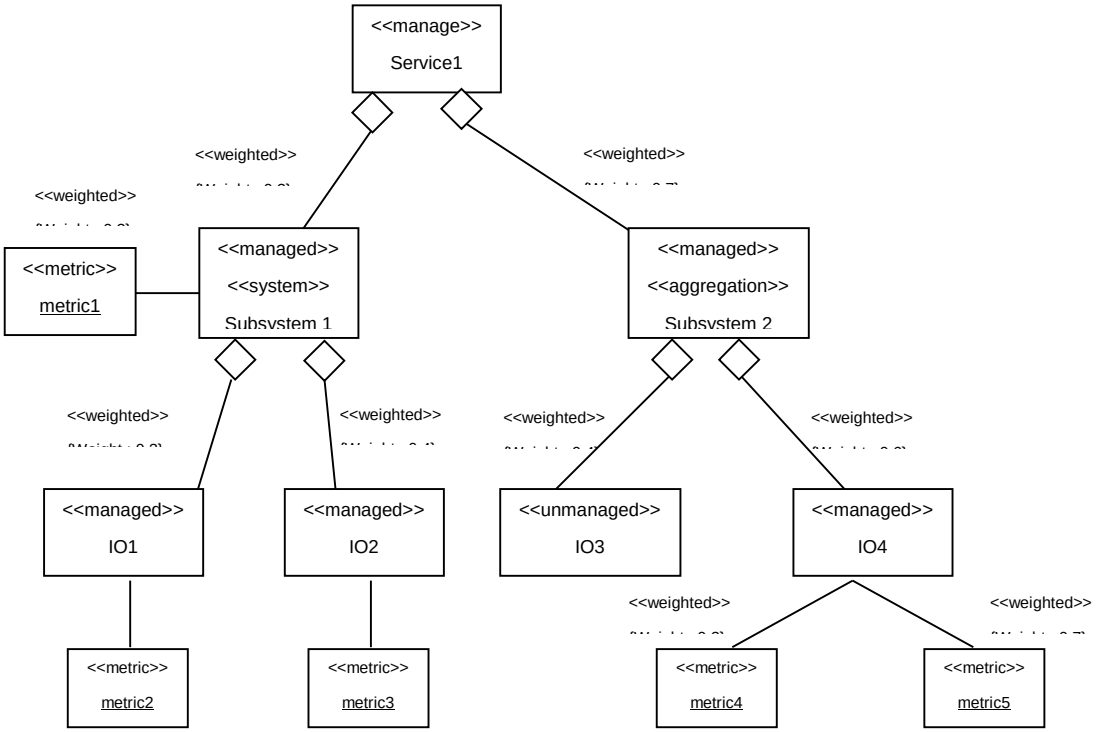


Figure 14 – Instantiated model (Maximum)

The main problem of this algorithm is that the service's monitoring can ignore some assurance level changes.. If the operator monitors only at the service level, any change below the maximum will not be reflected in the overall service assurance level. The threshold mechanism described in section 6.8 avoids this major problem.

6.5.3.3 Recursive Weighted Sum algorithm

The weighted sum algorithm uses weight properties model to calculate the assurance level value at each level of the assurance model of a service and consequently will reflect more precisely assurance needs described in the model.

The actual algorithm calculates at each level, from metrics to service, the weighted sum. For each infrastructure object, the assurance level will be the weighted sum of each metrics assurance level. Each subcomponent assurance level will be the weighted sum of all subcomponents if the actual subcomponent is an aggregation type subcomponent. If the subcomponent is a system type subcomponent, then it will be the weighted sum of each component and of its proper metrics

The recursive Weighted Sum algorithm is as follows:

```
For each service,
      Begin
            AL = Weighted Sum AL of each sub component
                  for each sub component until metric
                        if (subcomponent is aggregation type):
                              compute the Weighted Sum of each
                              subcomponent
                        if (subcomponent is system type):
                              compute the Weighted Sum of each
                              subcomponent and proper metric
                        end
                  end
      End
```

For each level of the model, the calculated (aggregated level) will be a non-discrete assurance level, which in the taxonomy does not represent anything. So, the real assurance level will be the entire part of the calculated assurance level. Nevertheless, this non-discrete level should be propagated and used to calculate aggregation value at the higher level until the service level.

In Common Criteria, it exists a notation to indicate an assurance level augmentation. It is indicated with a + sign such as EAL3+ which means that the level EAL3 has been reached augmented with other assurance. We could have indicated such kind of level for example, if the aggregated value is 3.9, we could have indicated 3+ instead of 3. This option, even if it is interesting, has not been kept for the product framework.

Contrary to the minimum and maximum algorithms, any changes from sub component and service assurance level will immediately impact. Combined with threshold mechanisms described in section 6.8, this algorithm provides a very efficient and sensible algorithm to calculate and detect deviance of the security assurance of the service.

As the aggregation is linked closely to the model, if errors occur or if the weights are not realistic (does not match the reality), the aggregated assurance will not reflect this reality.

If we come back to our example, in our case the service assurance level reaches only 2, as presented in Figure 15.

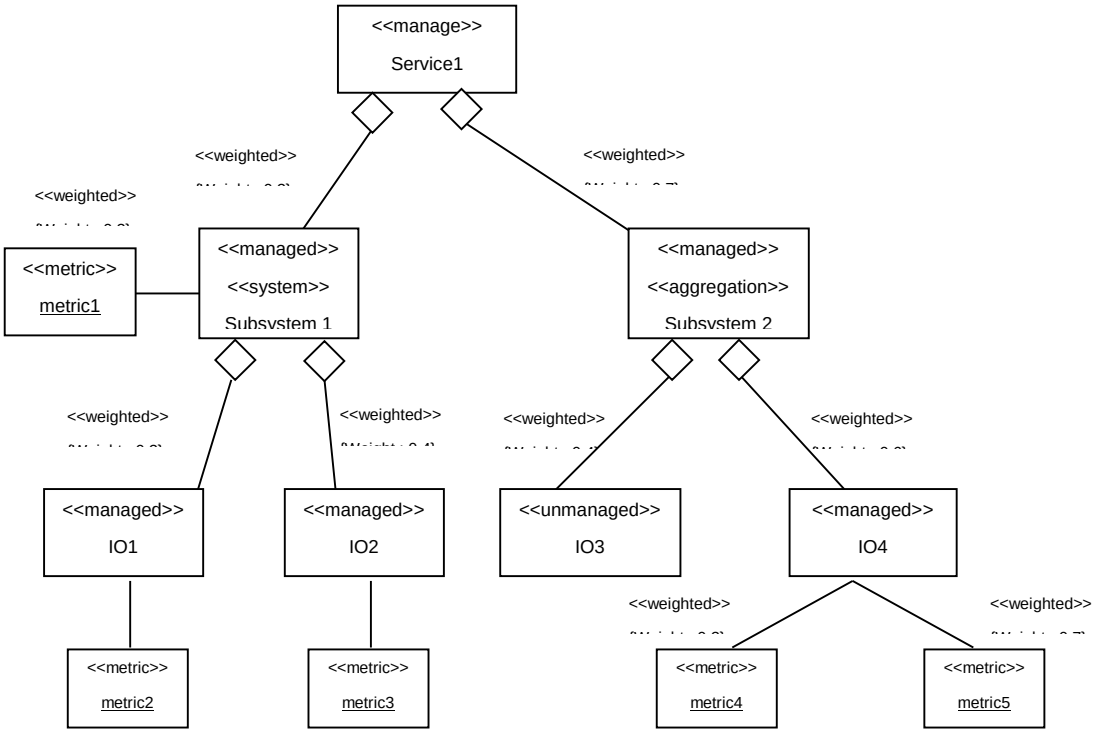


Figure 15 – Instantiated Model (Weighted Sum)

6.5.4 Comparing algorithm

We provide here some comparisons based on some practical criteria (this is not intended to be a scientific comparison but looking to some practical criteria). We are looking at the following aspects:

Simplicity: property that will guarantee that even in large infrastructures and complex models, the aggregation will not require intensive and powerful computing,

Ability to indicate any changes: property that will guarantee that any changes on the assurance level measured by metric will impact infrastructure objects and service level.

Main advantage: Property representing the main advantage of the algorithm.

Main constraint: Property presenting the main disadvantage of the algorithm.

	Min	Max	Weighted Sum
Simplicity	Very simple	Very simple	More complex but does not require powerful computation
Ability to indicate any changes	Only below min	Only above max	Enable any changes
Main advantage	Represent exact assurance	Represent best effort of the operator	Can monitor any minor change in the infrastructure
Main constraint	Required homogeneous metrics and distribution	Required homogenous metric levels and distribution	Need rigorous weight affection in building the model

Table 4 – Comparison of the aggregation functions

We can conclude by stating that the proposed product operational aggregation algorithms are able to compute the service assurance by taking into account the propagation of the level of the metric at the different subsystems.

The optimal algorithm, due to the specific product model, is the weighted sum algorithm. This can be operationally observed due to the fact that the assurance level might change (according to their level) and impact the service level where a single value is kept.

The aggregation procedure gives accurate computed values but the final value is discrete in order to show the possible deviation of the assurance value. For investigation reasons, the product operator can follow these deviations until arriving to the concerned metric.

Nevertheless, minimum algorithms should also be considered by operators, as it will indicate necessary actions if the overall service level could never reach the targeted (theoretical) level.
The same for maximum algorithm that can make sense in situations where operators want to demonstrate which best assurance level had been reached in maintaining optimal security conditions over the time.

6.6 Evaluation

Evaluation is the process step where the automatically derived assurance level is compared against the expected value (nominal/actual value comparison). This activity is performed to allow an automatic analysis and support a human decision maker. To use an automatic system, which is capable of not only comparing the top level system assurance expression but more complex patterns which include lower levels, should allow for more advanced analysis. It is noteworthy that the product does not encourage decision-making but only decision support.

To be able to conduct this comparison, the nominal value has to be defined by an expert and documented so that an automatic evaluation process can be conducted. We therefore define a new (optional) tag (see Table 5) for the “managed infrastructure object" in the model.

Tag	Stereotype	Type	M	Description
Threshold	Managed	data	0,1	This value describes the expected assurance value

Table 5 – Threshold Tag

In course of the automatic evaluation process an algorithm checks if the “managed infrastructure object” has a nominal assurance level defined in its threshold tag and performs a comparison with the actual value. A rule-based decision process then interprets the result of this comparison and in case of a mismatch an alarm is triggered.

The rule-based decision allows for various comparison alternatives. We imagine “simple" rules, which compare only the assurance level of one object with its threshold, and “complex" rules, which take interdependencies (i.e. correlations) into account. Example for a simple rule is the comparison if the actual value is equal or higher then the nominal value whereupon in the negative case an alarm would be raised.

The “complex” rules would perform their comparison either based on a custom rule set (e.g. if object A’s assurance decreases, object B's assurance level threshold must be one higher than the defined or otherwise the alarm must be raised) or based on prefabricated patterns (e.g. the “infrastructure object” provider who knows the critical parts of his system has provided a pattern that describes the interdependencies based on redundancy of security mechanisms that allow a more elaborate alarm scheme).

Apart from the immediate operational evaluation the product system will also provide means to compare trends. For that purpose statistical process control should be used to identify extraordinary fluctuations.

6.7 Presentation

The product system can play two roles. It can be used as a standalone assurance product. In this case, the product needs a specific GUI (security cockpit). It can also be used as a security assurance extension to a global monitoring system. Then, the product system is considered as a security assurance data source for this monitoring system.

Whatever the usage of the product (standalone or integrated), there is a need for administration. For instance, the product administrator should be able to manage the assurance model or to get information about user actions.

The following chapters describe the specific needs and answers for the product system depending on its usage.

6.7.1 Standalone module

As a standalone module, the product system mainly needs to display the monitored services, the assurance indicators (assurance levels and measures) and assistance views (e.g. assurance events). The administration need is described in the Section 6.7.3.

Monitoring views

This chapter deals with monitoring views. For the whole set of descriptions, please refer to the D4.1.1 functional requirements.

BUGYO system users must have a dashboard where the security assurance status of each monitored service is displayed. Figure 16 represents such a dashboard.

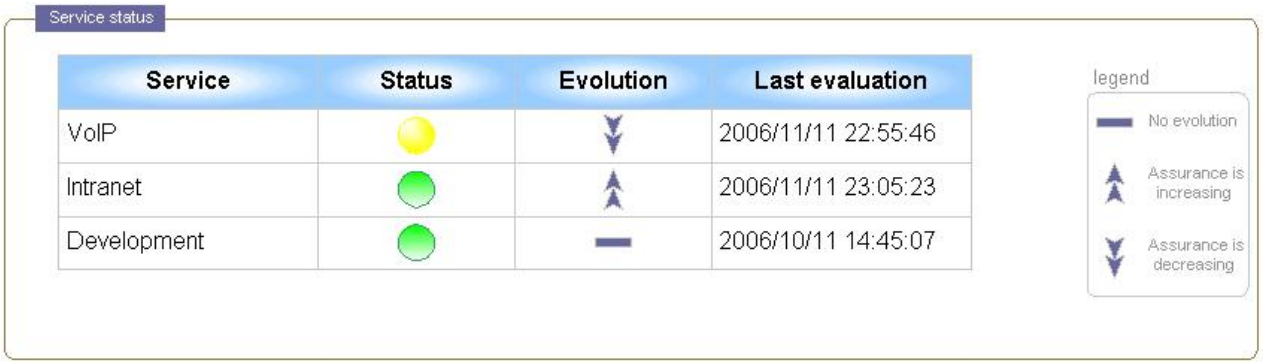


Figure 16 – Monitoring dashboard

For details about a given service, users can select it and a map view appears. The service decomposition (infrastructure objects) can be displayed as well as their assurance level.

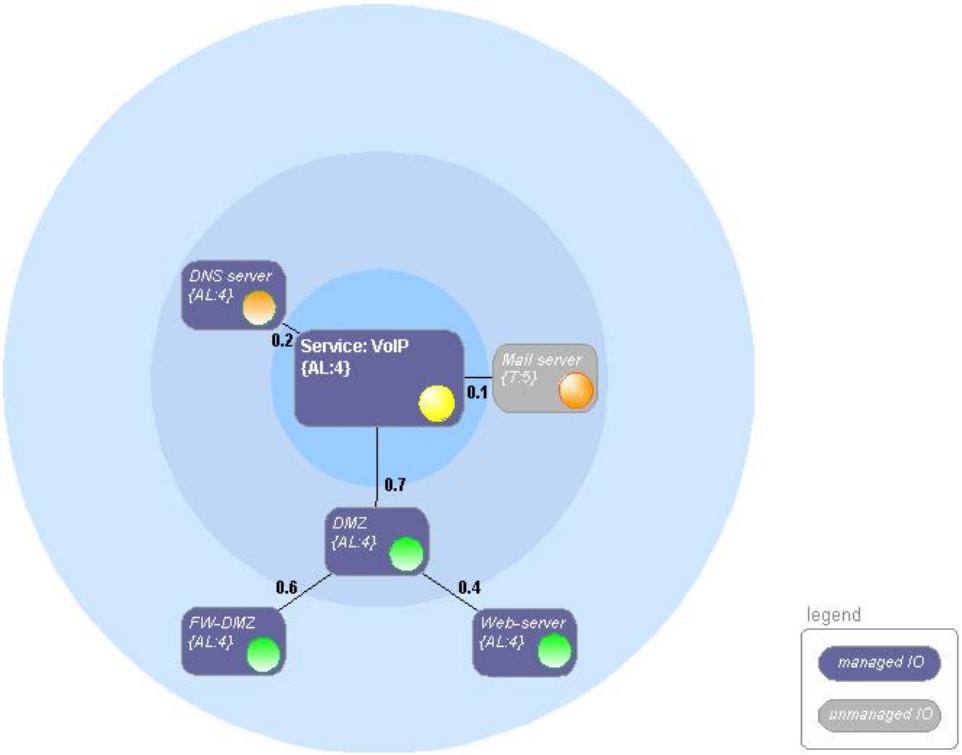


Figure 17 – Service-monitoring view

For details about metrics and base measures, users can select another view. This is important because the monitoring view should not be removed; critical events may occur meanwhile!

Measurement views

The figure below describes the detailed view of an infrastructure object.

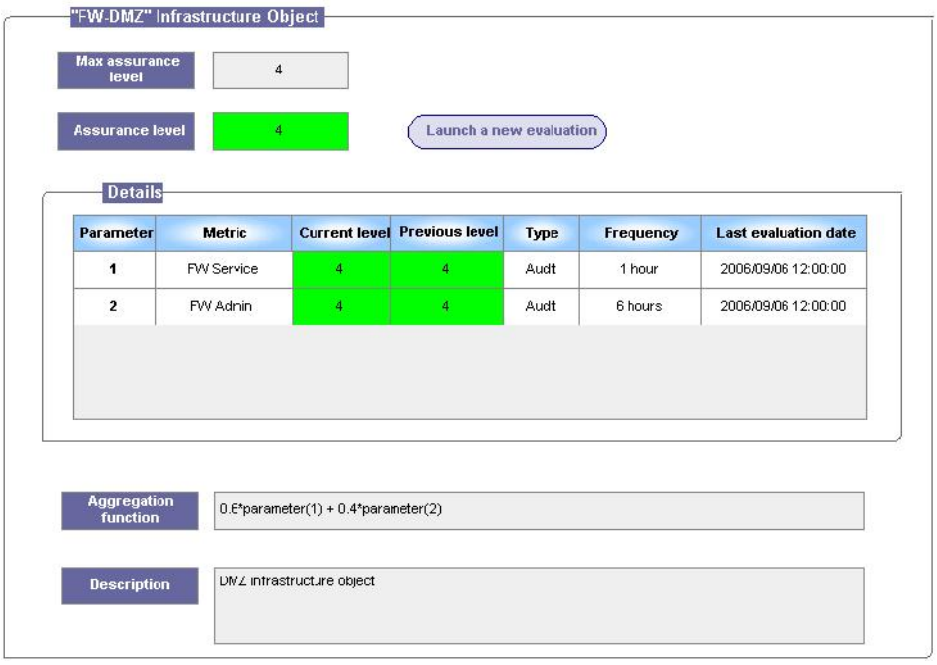


Figure 18 – Metric detail view

There is also a detailed view for measurements (see D4.1.1).

Assistance views

An important feature of the assistance part is the assurance events view. The product system must display assurance events and measure alarms. They are used to detect changes in the monitored services that have severe impacts.

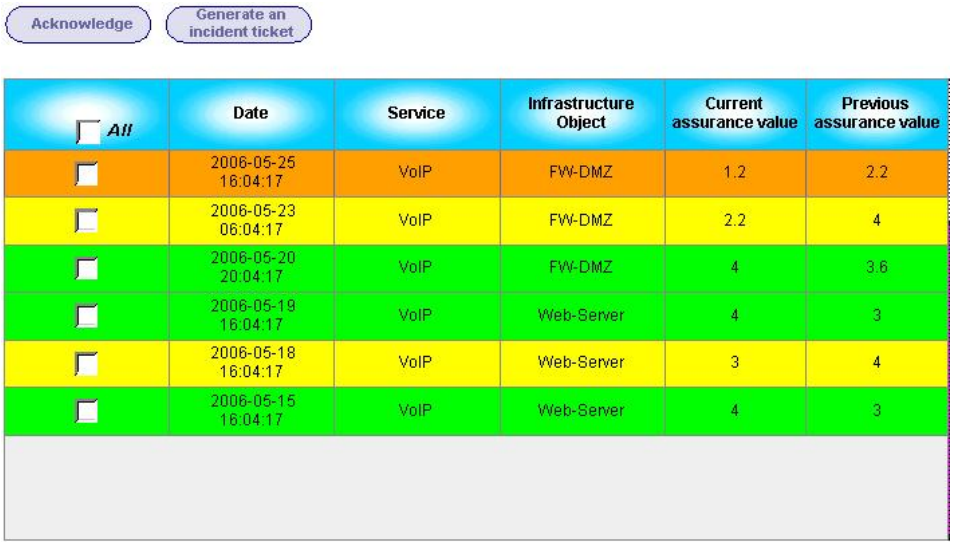


Figure 19 – Assurance event view

This view is detailed in the D4.1.1.

6.7.2 Security assurance extension

As an embedded security assurance module, the product system does not provide any presentation layer (except for administration purpose, see Section 6.7.3). Other modules such as SIM, NMS and alarm management tools directly use the results of the product computations. The product system then must have the ability to propose interfaces to such tools.

The three chapters below describe the possible types of interfaces.

Direct connection to a database

Assurance information is stored in a database.

There are two possibilities:

- The assurance module is connected to the whole system database that contains static (e.g. service models) and dynamic (e.g. measures) security assurance data.
  - This supposes that the connected element knows the structure of the database. This requires a great amount of work to develop APIs. The problem will be to be able to develop such APIs or to make them develop by the external tool editors.
  - A solution could be to set a simple mapping table that presents services, infrastructure objects or metrics and their related assurance indicator.
- The assurance module has its own database and external tools get information from it.

- Same technical issues as previously.

In both cases, this may be dangerous from a security point of view as you authorise different users and even different hosts to connect to a database that contains sensitive information. Such issues can be relatively easily resolved through authorizations, access control, etc.
As external tools cannot presume any change in the infrastructure objects, they must periodically request the whole set of assurance items or make a complex request that will get only items that have changed. Furthermore, the frequency must be quite high because, even if a change in the assurance should not occur often, it could have dramatic impacts on the service. So this solution is not adapted to get asynchronous data of potentially big size (such as assurance events) but is convenient for the retrieval of indicators.

Request/Response mechanism

External tools can also communicate with the assurance module through a request/response protocol. This is a widely used solution that supposes a set of commands (GET, SET, DELETE…) and parameters (Data, answers…). This also supposes that the product system integrates all the needed commands to satisfy external tools. The main users of such an interface are those who need to handle static data (typically GUI).

This obviously requires the implementation of some kind of client-server architecture (BUGYO system as server and external tools as clients). The communication flow must be secured (authentication, integrity and encryption). This is not a real problem. Typically the product GUI could be accessible over HTTPS with a Web browser.

Subscription mechanism

This mechanism is used to provide security assurance information (typically assurance events) to subscribers.
The advantage of this mechanism is that subscribers can define which kind of information they would like to receive. They get information in real-time. Other modules can use this generic provided mechanism. They are able to get information by means supported by the product system (e.g. e-mail).

A security issue is to manage subscribers. The administrator must check if subscribers are always active and the subscription should be made by the administrator and not automatically.

The communication flow must also be secured (authentication, integrity and encryption).

6.7.3 Administration

This is a necessary component of the product system embedded in a security/network solution or as a standalone module.

The product administrator is in charge of the management of:
- The system itself
- The assurance model

- The users and user profiles

The administrator also has to look at log records.

System administration

The product system stores and gets assurance data (e.g. base measures) in a database. The administrator is able to configure parameters relative to the connection to this database.

There is no strong need for a GUI, although it can prove useful.

Assurance model management

From the risk assessment, an expert describes the security assurance model for each monitored service. The product system must have views where this expert can create and manage models. This requires a specific GUI and a data manager.

The administrator is in charge of the deployment of the Metric files needed for the service model building (see D4.1.2 for the file descriptions).

User management

The product system, whatever its use, handles sensitive information. Users who access to the BUGYO must authenticate themselves and a restriction of possible actions must be set by product depending on their profile.

This requires a specific GUI.

Action loggings

A specific GUI may be useful.

6.7.4 Reporting

While the cockpit is displaying real-time information about service security assurance information, it is necessary to be able to produce reports based on evolution of the security assurance for a service, infrastructure objects or metrics.

The following report descriptions are suitable for the product system as a standalone tool (see section 6.7.1). When used as a security assurance extension (see section 6.7.2), security assurance reporting may also be integrated into global reports regarding all aspects of the security. They will not be described here.

6.7.4.1 Profile definition

We define several types of reporting based on recipient profile.

Four profiles are defined:

- **Operations executive management**, who is responsible for operations of a given operator or service provider and may want to be reassured about secure service provisioning;
- **Service operator**, who is in charge of quality of service and eventually customer satisfaction for a given service, may be

interested in knowing the evolution of a particular service or sub domains of service, even at the infrastructure object;
- **Equipment vendor**, who manufactured equipment used for a given service, may want to be informed about the security behaviour of the equipment under operational conditions;
- **Security expert**, who usually is the only one capable of understanding the problems occurring on any specific equipment and/or in a specific security function realisation, needs, in order to decide immediate or longer-term action, reporting about fine-grained measurement done on security realisation in operation.

These profiles will define types of report.

Customised ones may report all data logged in the product system. An example of a customised report might be targeting a sub part of a service regarding a specific function. For example, it might be interesting to look at a remote domain for a service with a special focus on a remote access functions. Such customised reports basically are combined service operator and security expert profiles types of report.

6.7.4.2 Content of a report

Operations executive management report
Output:

Report on Evolution of service's security assurance

Parameters:

- List of services
- Duration
  - Last week
  - Last month
  - Last 6 months
  - From *start_date* to *end_date*

Service operator report
Output:

Report on Evolution of service's security assurance and infrastructure object's security assurance

Parameters:

- Service selection
- Selection of Level of depth of infrastructure object
- Duration
  - Last week
  - Last month
  - Last 6 months
  - From *start_date* to *end_date*

Equipment vendor
Output:

Report on Evolution of one or several infrastructure objects' security assurance

Parameters:

- Service selection

- Infrastructure object(s) selection
- Duration
  - Last week
  - Last month
  - Last 6 months
  - From *start_date* to *end_date*

Security expert report
Output:

Evolution of one or several metrics' security assurance

Parameters:

- Service selection
- Infrastructure object selection
- Metric(s) selection
- Duration
  - Last week
  - Last month
  - Last 6 months
  - From *start_date* to *end_date*

7 CONCLUSION

Security assurance evaluation in a complex system is a subtle and a complicated process. It has been identified, as the major problem to be resolved within the product project and to our best knowledge has never been implemented until now.

7.1 Results summary

This document introduces a consistent terminology, develops a coherent view on the problematic and discusses several issues related or central to the security assurance domain. It establishes a general methodology principles by discussing the shortcomings of the related approaches, by detailing several systemic issues, by presenting the modern telecommunications landscape and by decomposing the problem in several major blocks, thus, reducing the overall complexity. Several possible approaches are discussed where applicable.

These general principles are then instantiated to the operational product methodology under a defined requirements framework. The latter is derived from the operational experiences of project partners. The operational product methodology is to be implemented further, developed and finally deployed by the consortium within the rest of this project.

The proposed operational methodology opts for the introduction of discrete security assurance levels and automatic evaluation of the Observed System and its parts to one of five standardised and, in principle, globally comparable assurance levels. It defines the general approach based on the assumption of a well-designed Observed System and the attachment of trusted metrics to its identified critical instances. It introduces precise rules for how to design, weigh and reassemble different intermediate results (base and derived measures) within the system observation and result interpretation process. Where it is applicable, this methodology follows the ISO methodology and terminology.

In view of the complexity of the main subject of this work, i.e. the operational security assurance evaluation of a running telecommunications service, the introduced proposed product methodology represents one possible approach. This methodology needs to be further evaluated through the operational tests. The deployment within the project will produce first feedback results. However, a further and deeper exploitation seems necessary, possibly in the follow-up projects.

7.2 Limitations and perspective

Indeed, the made choices introduce some room for criticism. For instance, several restrictions apply to this methodology. Since it opts for comparable levels, it is required to perform all measurements on a common scale. This is achieved through a specified normalisation and aggregation processes. However, to truly produce comparable and reliable results, this methodology will need reliable, trusted metrics. In practice, a metrics certification process, comparable to laboratory certification in the Common Criteria, can achieve this. Yet, the metrics will need to be certified per assessed

infrastructure object, which might constitute an important deployment burden and thus needs to be further studied. In general, the metrics certification will need further evaluation per se, once institutionalised. A viable and realistic methodology is necessary in this new scope.

Another point is the ease of use. Although a lot of effort was spent on this specific issue, the online monitoring of the Observed System represents an important complexity as such. Before deploying a product system, it is important to prove that the additional investments in such an infrastructure do result in substantial operational improvements on the one hand and do not introduce any new practically exploitable threats on the other.

However, it should be noted that such an intense system observation is necessary in every possible security assurance methodology and even more broadly in every system health estimation methodology. That is why our methodology strictly separates the pure monitoring/data collection from the interpretation/evaluation parts. From our experiences, we believe that such an observation is essential and should thus be further promoted, regardless of the precise aim. Therefore in this sense, the product represents an important and extensible platform, capable of serving as proof of concept for several related open system questions.

The taxonomy and methodology described in this document allow implementation of an operation framework that can evaluate security assurance of an operational telecom system from the service perspective. Furthermore, as described in appendix B, materials of this document provides bases to draw standard/best practices document, with the same approach than the common criteria from which initial definition and concepts have been taken in order to build this methodology.

8 REFERENCES

[1] BUGYO Consortium – "Management Reference Model of Targeted Services Infrastructure", BUGYO Deliverable D1.2 version 1.2, Oct. 26, 2006.

[2] ISO/IEC 15939:2002 – "Software engineering -- Software measurement process", JTC 1/SC 7, 25th Jan. 2006.

[3] BUGYO Consortium – "Assessment of Existing Standards, Methodologies and Tools", BUGYO Deliverable D2.1, version 1.1, Oct. 26, 2006.

[4] BUGYO Consortium – "Report of Selection of Telecommunications Services to Target", BUGYO Deliverable D1.1, version 1.0, Sep. 30, 2005.

[5] CC, Common Criteria for Information Technology, Parts 1 – 3, version 3.1, September 2006.

[6] CEM, Common Criteria for Information Technology, Evaluation Methodology, CCMB-2006-09-004, version 3.1, Revision 1, September 2006.

[7] ISO/IEC 15408, Information Technology – Security Techniques – Evaluation Criteria for IT Security, version 2005, equivalent to Common Criteria version 2.3: ISO/IEC 15408-1:2005 "Part 1: Introduction and general model", ISO/IEC 15408-2:2005 "Part 3: Security functional requirements"; ISO/IEC 15408-3:2005 "Part 3: Security assurance requirements".

[8] ISO/IEC 18045, Information technology – Security Techniques – Methodology for IT Security Evaluation, version 2005: equivalent to Common Criteria CEM version 2.3.

9 APPENDIX A: (INFORMATIVE) DERIVED MEASURE SCALE EXAMPLES

Three types of derived measures are defined by the operational methodology:

- **Availability** relates to measures that monitor whether a counter-measure exists and is available.
- **Conformity** relates to measures that control the compliance of a counter-measure to a predefined policy (the Reference).
- **Non vulnerability** relates to measures that controls if the counter-measure is not vulnerable (the Reference is the vulnerability database).

Normalised measurement results are required.

9.1 Availability derived measures

OVAL 5.0 definitions (http://oval.mitre.org/) can be used for the normalisation of the availability derived measures.

Required base measures: counter measure response to a stimulus

Value	Description
true	the service is available.
false	the service is not available.
unknown	the measurement has been performed but the results are not available. (1)
not evaluated	a choice was made not to perform the measurement. (2)
error	there was an error either collecting information or in performing analysis. (3)
not applicable	the request is not valid???

Table 6 – Derived measures scale for availability

(1) In terms of implementation: the agent answers that the probe does not provide results.
(2) In terms of implementation: the agent answers that the measurement cannot be performed because no probes exist to perform the measurement.
(3) In terms of implementation: the agent does not answer.

9.2 Conformity derived measures

OVAL 5.0 definitions (http://oval.mitre.org/) can be used for the normalisation of the conformity derived measures.

Required base measures: counter measure characteristics

Value	Description
true	the characteristics fulfils the requirements
False	the characteristics do not fulfil the requirements
unknown	the measurement has been performed but the results are not available. (1)
not evaluated	a choice was made not to perform the measurement. (2)
error	there was an error either collecting information or in performing analysis. (3)
not applicable	the requirements to be evaluated are not valid on the given platform. For example, trying to collect Linux RPM information on a Windows system.

Table 7 – Derived measures scale for conformity

(1) In terms of implementation: the agent answers that the probe does not provide results.
(2) In terms of implementation: the agent answers that the measurement cannot be performed because no probes exist to perform the measurement.
(3) In terms of implementation: the agent does not answer.

9.3 Non vulnerability derived measures

OVAL 5.0 definitions (http://oval.mitre.org/) can be used for the normalization of the Non-vulnerability derived measures

Required base measures: counter measure vulnerability and its score

Value	Description
true	No vulnerability has been found
False	Vulnerabilities have been found.
unknown	Measurement has been performed but the results are not available. (1)
not evaluated	A choice was made not to perform the measurement. (2)
error	There was an error either collecting information or in performing analysis. (3)
not applicable	Requirements to be evaluated are not valid on the given platform. For example, trying to collect Linux RPM information on a Windows system.

Table 8 – Derived measures scale for non-vulnerability

(1) In terms of implementation: the agent answers that the probe does not provide results.

(2) In terms of implementation: the agent answers that the measurement can not be performed because no probes exist to perform the measurement.
(3) In terms of implementation: the agent does not answer.

9.3.1 Vulnerability scoring

CVSS (http://www.first.org/cvss/cvss-guide.html) defines a metric as a constituent component or characteristic of a vulnerability that can be quantitatively or qualitatively measured.

These atomic values are clustered together in three separate areas: a base group, a temporal group, and an environmental group.

The **base group** contains all of the qualities that are intrinsic and fundamental to any given vulnerability that do not change over time or in different environments.

The **temporal group** contains the characteristics of a vulnerability that are time-dependent and change as the vulnerability ages.

Finally, the **environmental group** contains the characteristics of vulnerabilities that are tied to implementation and environment.

The final adjusted score represents the threat a vulnerability presents at a particular point in time for a specific environmental condition. The metric groups are shown below.

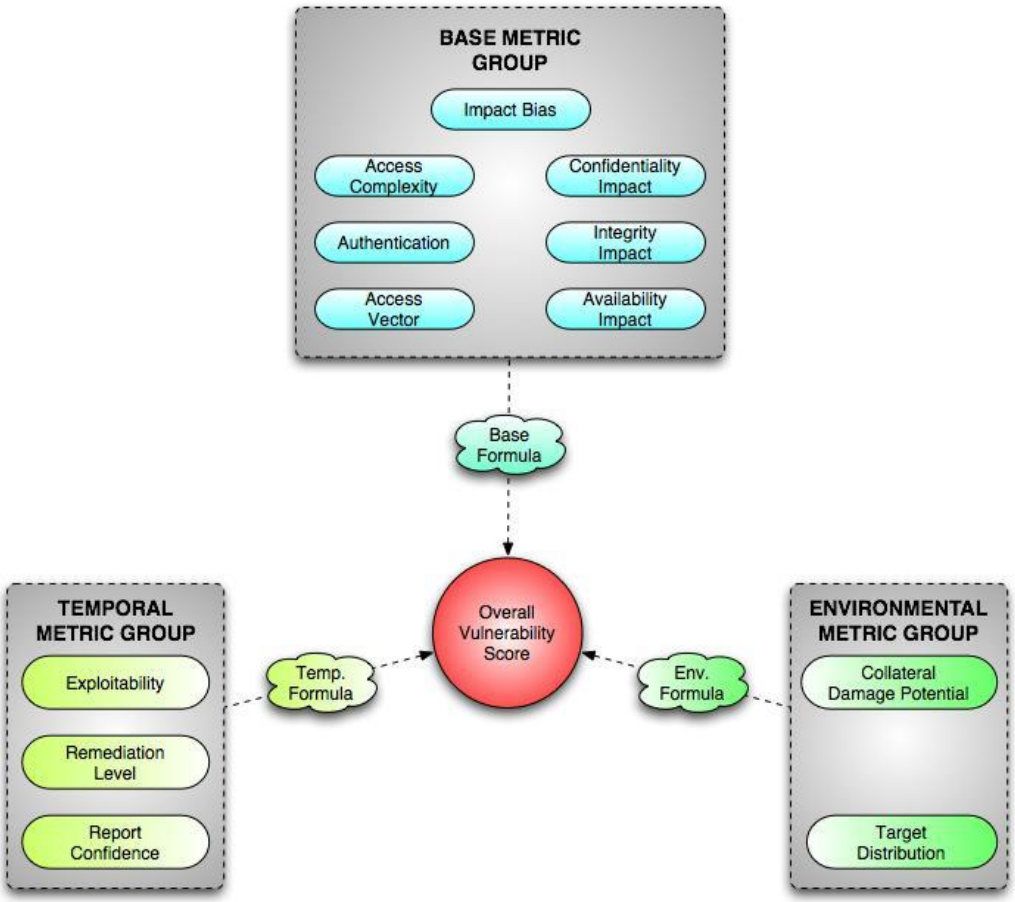


Figure 20 – CVSS scoring system

CVSS proposes the following scoring formulas:

9.3.2 Base Metric Scoring

The base score provides the foundation for the overall vulnerability score. The most significant metrics in the scoring process are the three impact metrics. These metrics dictate the overall effect the vulnerability will have on targeted systems and therefore have the strongest bearing on the final score. Base metric score for a vulnerability is typically set by the vulnerability scanner vendors.

Base Metric Formula

```
AccessVector     = case AccessVector of
                   local:          0.7
                   remote:          1.0

AccessComplexity = case AccessComplexity of
                   high:           0.8
                   low:            1.0

Authentication   = case Authentication of
                   required:       0.6
                   not-required:    1.0
```

```
ConfImpact      = case ConfidentialityImpact of
                  none:           0
                  partial:        0.7
                  complete:       1.0

ConfImpactBias  = case ImpactBias of
                  normal:         0.333
                  confidentiality: 0.5
                  integrity:      0.25
                  availability:   0.25

IntegImpact     = case IntegrityImpact of
                  none:           0
                  partial:        0.7
                  complete:       1.0

IntegImpactBias = case ImpactBias of
                  normal:         0.333
                  confidentiality: 0.25
                  integrity:      0.5
                  availability:   0.25

AvailImpact     = case AvailabilityImpact of
                  none:           0
                  partial:        0.7
                  complete:       1.0

AvailImpactBias = case ImpactBias of
                  normal:         0.333
                  confidentiality: 0.25
                  integrity:      0.25
                  availability:   0.5

BaseScore = round_to_1_decimal(10 * AccessVector
                          * AccessComplexity
                          * Authentication
                          * ((ConfImpact * ConfImpactBias)
                          + (IntegImpact * IntegImpactBias)
                          + (AvailImpact * AvailImpactBias)))
```

9.3.3 Temporal Metric Scoring

The temporal score adjusts the base score by including factors that may change over time. The temporal score will be less than or equal to the base score; that is, the temporal metrics serve only to reduce the base score by a maximum of 33%. This is shown at the end of the scoring section.

Temporal metric score for a vulnerability is typically set by the vulnerability scanner vendors.

Temporal Metric Formula

```
Exploitability  = case Exploitability of
                  unproven:          0.85
                  proof-of-concept:   0.9
                  functional:        0.95
                  high:              1.00

RemediationLevel = case RemediationLevel of
                  official-fix:      0.87
```

```
                    temporary-fix:      0.90
                    workaround:         0.95
                    unavailable:        1.00

ReportConfidence = case ReportConfidence of
                    unconfirmed:        0.90
                    uncorroborated:     0.95
                    confirmed:          1.00

TemporalScore = round_to_1_decimal(BaseScore * Exploitability
                                   * RemediationLevel
                                   * ReportConfidence)
```

For BUGYO, it can be assumed that the temporal score is the score for each found vulnerability.

The environmental metric score for a vulnerability must be set by the end-user of the vulnerability scanner.

For BUGYO, it can be considered that the environmental factors can be introduced at the level of the model.

9.4 Metric value

In Table 9 we consider that the metric is certified at level 3 (i.e. the maximum value for the metric is 3).

The values in the lines correspond to the base measures values required to achieve the metric value mentioned in the corresponding column.

Base measures	Metric Value		
	1	**2**	**3**
Availability	True	True	True
Conformity		True	True
Non-vulnerability			True

Table 9 – Metric value table