ETSI TR 103 437 V1.1.1 (2020-11)

USER;
Quality of ICT services;
New QoS approach in a digital ecosystem

Reference
DTR/USER-0045

Keywords
ICT, QoS, quality, service, SLA, user

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Contents

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (https://ipr.etsi.org/).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

Foreword

This Technical Report (TR) has been produced by ETSI User Group (USER).

Modal verbs terminology

In the present document "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

Introduction

Nowadays, the user is at the centre of the architecture. He has access to any service, from any network and by any means, from anywhere, all the time.

1 Scope

The experience of the "Covid-19" recently faced has played a role of accelerating and making users enter definitively into "the digital era".

On the user side, this meant that in their daily life teleworking, online shopping and a lot of vital and key information was shared through online social networks. Considering the needed digital services, often used for the first time, the user became aware of the importance of quality of service and the many factors which contribute to it.

On the supplier side, the role of digital transformation manager was created, directly linked to general management.

Among the new paradigms, the "As-a-Service" is the main driver to support digital transformation. The user wishes to obtain a personalized service whatever the place they are and whatever their means of access with the corresponding QoS. The user expects this is also provided "As-a-Service", meeting their needs and not a "best effort" delivery.

The present document proposes a new approach to implement a QoS adapted to the digital ecosystem, with both views from the user side and from the supplier side.

2 References

2.1 Normative references

Normative references are not applicable in the present document.

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] ETSI TS 102 827 (V1.1.1) (2008-08): "GRID; Grid Component Model (GCM); GCM Interoperability Deployment".

[i.2] ETSI TS 102 828 (V2.1.1) (2010-03): "GRID; Grid Component Model (GCM); GCM Application Description".

[i.3] ETSI TS 102 829 (V1.1.1) (2009-03): "GRID; Grid Component Model (GCM); GCM Fractal Architecture Description Language (ADL)".

[i.4] ETSI EG 202 009-1 (V1.3.1) (2014-12): "User Group; Quality of telecom services; Part 1: Methodology for identification of indicators relevant to the Users".

[i.5] ETSI EG 202 009-3 (V1.3.1) (2015-07): "User Group; Quality of ICT services; Part 3: Template for Service Level Agreements (SLA)".

[i.6] IETF RFC 2617: "HTTP Authentication: Basic and Digest Access Authentication".

[i.7] Lloyd V. ITIL Continual Service Improvement: "The Stationery Office (TSO)". 23 Aug 2011. ISBN: 9780113313143.

[i.8] ISO/IEC 20000-1:2018: "Information technology -- Service management -- Part 1: Service management system requirements".

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the following terms apply:

InMonitor: component that intercepts incoming service, stores the non-functional information about the requests, which are then transmitted (unchanged) to the functional component, via the corresponding internal interfaces

micro-service: basic and simple service (with SoA properties) that be combined for the composition of services as expected by the User

NOTE: The basic concept behind this term is that each service performs a unique feature (e.g. for security, "authentication" is a micro-service, for discovery, "find" is a micro-service).

OutMonitor: component that intercepts and stores outgoing service requests

profile: information template (model) to provide or to access to personalized services

QoSControl: component that makes the necessary metric analysis and calculations to evaluate the behaviour of the service component and its conformity with the contract

quality of service: ability of a service to respond by its characteristics to the different needs of its users or consumers (AFNOR)

service: immaterial performance that can be composed, manifestly displayed and which, in a pre-defined condition of use, is a source of value for the consumer and the supplier (ISO/IEC 20000-1 [i.8])

3.2 Symbols

For the purposes of the present document, the following symbols apply:

InMonitor	Input Monitor
OutMonitor	Output Monitor
QoSControl	Control of the QoS

3.3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

API	Application Program Interface
CPU	Central Processing Unit
FCAPS	Fault, Configuration, Accounting, Performance, Security
GCM	Grid Component Model
GDPR	General Data Protection Regulation
HMI	Human Machine Interface
ICT	Information & Communication Technology
IMS	IP Multimedia Subsystem
IoT	Internet of Things
ISO	International Organization for Standardization
ITIL	Information Technology Infrastructure Library
LED	Light Emitting Device
OpenIMS	Open Infrastructure Management System
QoE	Quality of Experience
QoS	Quality of Service
RAM	Random Access Memory
SLA	Service Level Agreement
SLO	Service Level Objective
SOA	Service-Oriented Architecture
TTM	Time To Market

IT	Information Technology
IP	Internet Protocol
IS	Information Systems

4 User in a digital ecosystem

4.0 New user provisioning approach

In a digital ecosystem the provisioning of a wide range of services depends on the orchestration of heterogeneous, distributed software components, which can be owned by different service providers and operate over diverse networks. In such a context, designing and providing value-added services, ensuring their nominal quality levels with service deployment, provisioning, monitoring and management becomes increasingly difficult. Provider resources are shared by all clients. In the cloud computing context, the outsourcing introduces the need of Service Level Agreement (SLA). How the mapping between the user demand and the provider supply can be performed?

To answer this problem, the proposed new approach aims to express the user requirements and the provided services with the same model. The main advantages of this approach are the modelling and the overall management of digital ecosystem behaviour founded on a new integrated service component that distinguishes itself through a "self-control" property based on QoS and the "As-a-Service" concept.

4.1 User profile

A good digital user experience is based on always online services, easily accessible anytime and everywhere, on demand, in real-time, and available in self-service along with a fast helpdesk service response. For the user, that means a good level of flexibility and control of his digital environment.

In a digital ecosystem the user, according to his level of experience:

- gets the service automatically; or
- makes his choice in the catalogue for a composition of service according to the QoS requested.

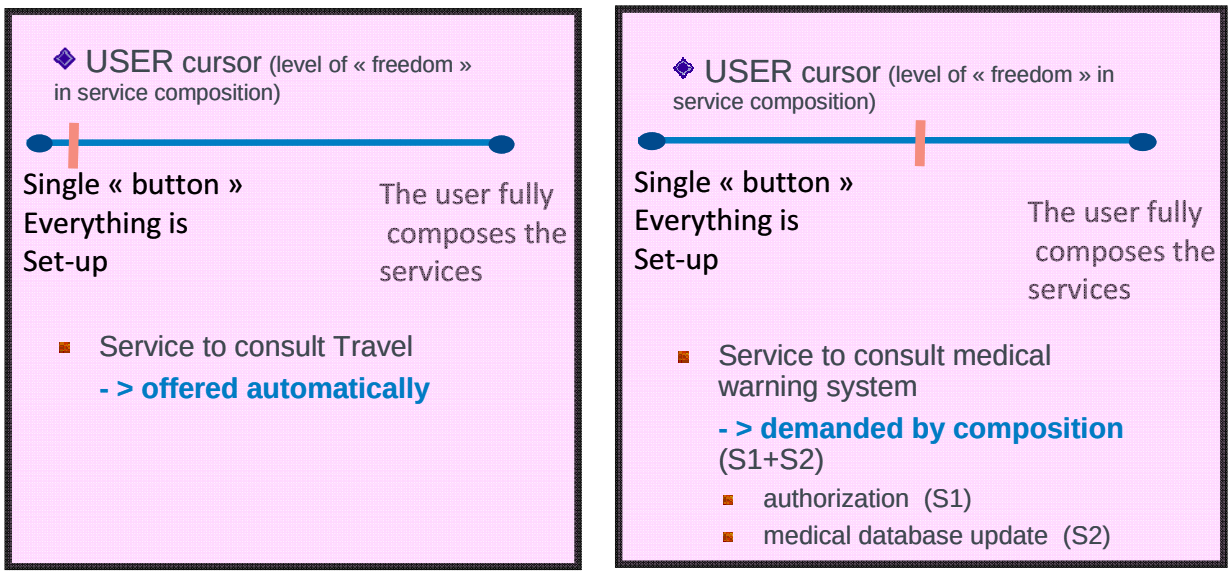

Figure 1: User profile

In this context the user services composition can be based on setting, user profile, HMI, QoE and the degree of service security (authentication, authorization, confidentiality, cryptography, etc.).

The user profile provides the image of the user defining the user's preference, possibilities and constraints, in a structured and uniform format. This profile provides an easy access to all necessary data and relevant selection of each service component according to the user's preference. Each service composition proposed by the provider should be linked to a user service session.

4.2 Requirements: Service Level Agreement (SLA)

A Service Level Agreement (SLA) is an agreement formally negotiated between two parties.

The SLA serves as a means to formally documenting the service(s), performance expectations, responsibilities and borders between cloud service providers and their users. It aims to managing service quality through the customer experience life cycle. This means managing service quality beyond the in-use phase of the life cycle to include sales, provisioning, in-use phase and service termination aspects.

The objective of the end-to-end QoS is to build and maintain the adequate service over a dedicated user session while respecting SLA and QoS constraints (Figure 2).

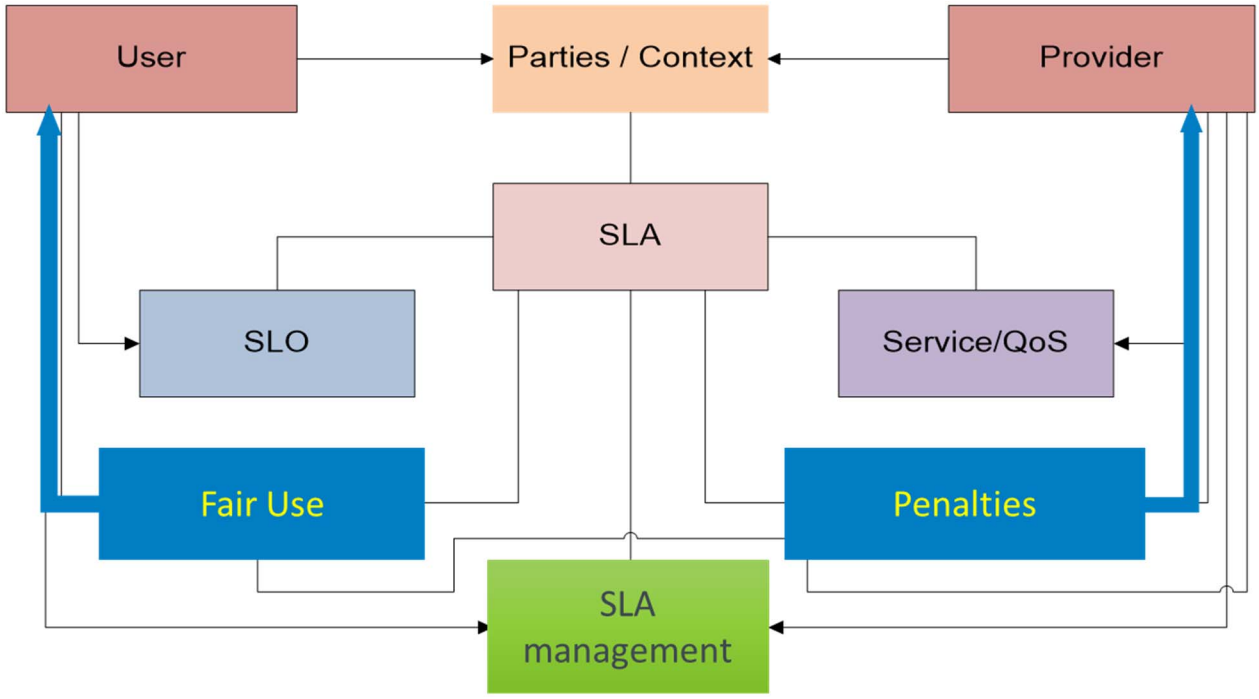

Figure 2: End to end service: conformity with SLA

The SLA parties represent the contracting entities of an SLA contract.

The SLA can be described in two parts:

- The users request their requirements, i.e. SLO and obligation, corresponding to the demand.
- The offer by Cloud provider with the guarantees provided (QoS associated to services offers, penalties) corresponding to the supply.

On the user side, a SLO aims at expressing the user needs. For example: service is available 7/7 and 24/24, access time to the application < 1 s in 90 % of cases, a processing time < 2 s if the number of requests per second < 1 000 in 90 % of cases. The user has the obligation to check the correct functioning of the service.

On the provider side, the services offered are twofold: usage and management. In accordance with the proposed model, every service component integrates a QoS control. Four criteria are proposed [i.4] to describe the behaviour (QoS): availability, integrity, time, and capacity.

From the provider point of view, the objective is to meet the required properties based on customer requirements and needs. In practice, many providers offer the same services that differ in their quality of service levels, price, and in the way, they are created, deployed, and managed. Therefore, the request and the offer should be entirely and explicitly documented and guaranteed by the Service Level Agreement (SLA). The approach presented in ETSI EG 202 009-3 [i.5] allows to model the SLA personalized where user requirements and provider offers converge on a QoS contract.

5 Service offered

5.0 As-a-Service environment

Digital ecosystem, cloud computing and Internet of Things (IoT) are promising to build a new ecosystem where everything is provided "As-a-Service". The "As-a-Service" is the main driver to support digital transformation, that can be translated by "flexibility in the service of business" respecting the quality.

The enterprise should prepare for these changes, which may also require a redesign of its core business, because what can really lead to success is the design of the customer experience. Quality experiences are based on the customer empathy, business analysis, and cognitive technologies, all of which can lead to a successful business strategy. It also means having the ability to meaningfully engage customers and employees, no matter where they are.

This inevitably incorporates the effective use of operational and virtual IT models, which include dynamic provisioning of the infrastructure, the ability to automatically scale it up, applications in the form of micro-services, and ultimately "cloudification" of the digital ecosystem.

The services are built through the composition of services that exist today in the enterprises or can be provided by providers. This approach should be based on:

- designing "As-a-Service";
- building the service by composition; and
- managing it (based on decision-making information).

To comply with the QoS principles, a service should fulfil five important points:

- to be defined through a contract;
- to be evaluated itself through criteria;
- to be measured through significant parameters at each level of visibility;
- to formalize the non-functional aspects of each action;
- to be aggregated for an end-to-end flow.

5.1 Service definition

The service provider is responsible for the creation of a service, to document the functional descriptions in the directory and to provide the interface. Depending on the level of "freedom wanted by the user" (according to Figure 3) the composition may be called on in an autonomous way (step by step) or globally (only the final result is provided to the user).

Figure 3: Service provider

The most important concepts in digital environments are QoS and service composition. This is expected to offer the maximum number of services among a large set of providers. This means that the following question should be answered: what can be offered in term of composition (by construction)?

The concepts to achieve maximum provider agility to provide the highest user level of "freedom" in the service composition are:

- "As-a-Service", micro service, service profile.
- QoS and API (Application Programming Interface) for each micro service and composition.

5.2 "As-a-Service" properties

5.2.0 "As-a-Service" in the new architecture

To better understand the expectations of service creation and management, it is necessary to situate them in the Internet of Things or Cloud Computing architecture.

The properties of "As-a-Service" components have to comply with a set of requirements. These properties are spread among the following categories related to:

- The definition of the structure and the formal descriptions of service components, i.e. the nodes themselves.
- The definition or design of the service logic and functional architecture of service components, i.e. the interactions between the service components.
- The management of the service components.

Models / Features		as-a-Service
Structure	Cohesion	✓
	Reuse	✓
	Abstraction	✓
	Invariance	✓
	Statelessness	✓
	Mutualization	✓
Interactions	Loose coupling	✓
	Invocation	✓
	Composition	✓
Management	Description	✓
	Registration	✓
	Exposition	✓
	Auto-management	✓
	Ubiquity	✓

Figure 4: "As-a-Service" properties

These properties are necessary to design components "as a service" so that from the catalog components can be chosen and assembled easily. In particular, the structure properties like "stateless" and "mutualization", the "loose link" property and those of "self-management and ubiquity" management are very important.

5.2.1 Properties related to service structure

Properties related to the service structure are:

- Reuse;
- Mutualization;
- Cohesion;
- Abstraction;
- Invariance;
- Statelessness; and
- Composition.

Reuse: the possibility of reuse is needed to simplify the software development of services that meet the new needs. Services designed based on this approach and properties (As-a-Service) would be reusable, thanks to the generic character of their interfaces (usage, control and management). A service component should be reusable to build different services, in different compositions and different environments.

Mutualization: a service provider should offer the same service component instance As-a-Service to multiple users. Mutualization stands here for multi-tenancy. Service components should support multi-tenancy in order to be invoked by multiple users requiring the offered service either simultaneously or not. This reinforces the statelessness and the loose coupling features. Mutualization requirement calls for a need for loose bindings or connections between service components to ensure the capacity to provide multiple users and answer multiple service requests autonomously. Thus, mutualization will help realizing minimum functional coupling and loose coupling between functions.

Cohesion: service components should be consistent. The service logic offered should be relevant and recognized as a meaningful business service for potential customers. The service rendered by the component should find all its functionalities in a logical way internally. This feature could also be called: self-sufficiency, autonomy or even functional decoupling.

Abstraction: beyond service descriptions that should appear on service catalogues and SLAs, service components should abstract the internal service logic from outer service environments.

Invariance: a service component should have an identical structure, that would not vary from a level to another in a hierarchy of service components. That means that the structure is invariant when scalability and elasticity are needed.

Statelessness: a service is stateless if it processes each received request as an independent transaction without any relationship with the previous ones. A service component should then neither keep information regarding its state or its processing state nor handle information about previous requests. If it maintains its state for a long period, it will lose the "loose coupling" feature, its availability for other incoming requests and even its possibility to scale.

Composition: service components should be able to be chained as elementary entities (primitive or composite components) to create a service. They should be effective service composition participants, regardless of the size and complexity of the composition. This composition requirement feature is verified only if all the features described are verified.

Each component should handle data coming only from outside its area of responsibility i.e. from other service components so that its functional behaviour does not use data received from previous invocations. For that, it is needed to rely on transactions in unconnected mode that define well-specified formats of in-requests and out-responses. In such context, the component structure with its interfaces would help. It is also needed to delegate information handling and state management to external entities. This feature is crucial as it impacts the independency of a service component and thus the possibility to include it in compositions that need to be dynamic.

5.2.2 Properties related to service interactions

Properties related to interactions are:

- Loose Coupling;
- Invocation;
- Interconnexion.

Loose Coupling: service components should have no predefined sequence between them and should maintain relationship with minimized functional coupling.

Invocation: a service component should be accessible and invoked based on service requirements in SLAs (invocation interface, function or service and QoS level). Service contracts are spread between three types: syntactical contract (service interface, function, service or process name, input/output parameters and structural constraints), semantic contract (informal description of the function or service with service use rules and constraints), and service-level contract (defines the service commitments, i.e. QoS and SLA parameters like time to access, to process, to response).

Interconnection: it means that the service has all the connections for its operation.

5.2.3 Properties related to service management

Properties related to service management are:

- Description;
- Exposition;
- Registration;
- Auto-management;
- Ubiquity.

Description: service components should be describable based on meta-data in an independent manner from their implementation specificity. The formal description should have a logical and meaningful structure.

Exposition: this feature includes cohesion, description, registration and invocation. Exposition is providing the functional and non-functional description of service components as well as their inherent QoS level offered through catalogues on service portals to allow a third-party actor to select and/or build a service based on his profile and competences.

Registration: service components should be able to be registered in a Domain Registry. This registration can be made through a publication (publish) of its service offering, QoS level and state. It should also be able to discover its environment through service discovery.

Auto-management: service components should be able to monitor and control their behaviours (non-functional aspects) using autonomic management approaches. Placing the monitoring of QoS very close around each service component and business logic helps to detect exactly a malfunctioning component.

Ubiquity: is the high equivalence between service components. Service components should be defined and described based on their core function and QoS level they offer (the QoS values). According to this definition, Service components may be grouped into communities of ubiquitous or identical service components where service components of a community provide the same service even if their business codes or algorithms are different, with the same QoS level. This feature goes with scalability issues, as the service provider may decide to scale the service by adding ubiquitous service components. It also enables higher service availability to find the requested service with the desired QoS level as this gathering in ubiquitous components community is an approach to set redundancy schemes. These requirements apply to the software design of services on both, functional and non-functional aspects. Their generic nature allows a service architect to apply them on any service.

All these properties allow to have a service component in the digital world.

5.3 Interfaces

The service component becomes a fully-fledged entity which is called upon through its user interface. But to be integrated into the system, it needs a management interface and a control interface in order to interact with its environment. This is why the new QoS approach includes interfaces dedicated to QoS control or compliance and to service management. The service component therefore has three types of interfaces:

- usage;
- control; and
- management (Figure 5).

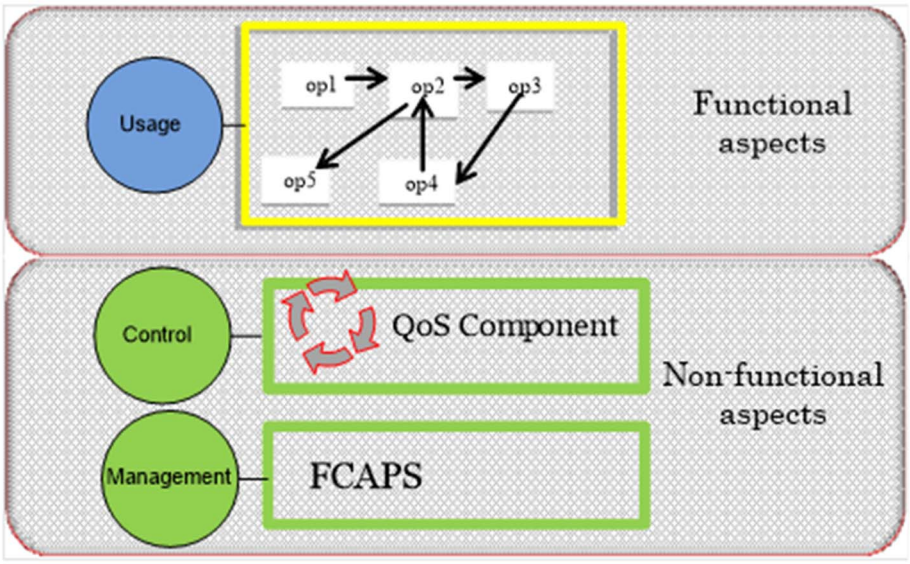

Figure 5: New way to design services

The usage interface (in blue in Figure 5) includes the processing functions (service operations: op1, op2, etc.) that can be performed by the service component. One interface performs invocation for the next service of the chain, transmitting its current result for further treatment or for exploitation of the final result.

The control (non-functional aspects) interface (in green in Figure 5) provides mechanisms for conveying the self-control information to the manager in charge of processing QoS violation events. Its outputs InContract notifications as long as the behaviour conforms to the contract, otherwise it triggers an OutContract notification.

The management interfaces (in green in Figure 5) contain the necessary management mechanisms for FCAPS functionalities.

A new approach to design a software component "As-a-Service" should be applied in order to make it compliant with SOA (*Service-Oriented Architecture)* and micro-services service design requirements. This new design approach involves five steps shared between functional and non-functional aspects:

- Step 1: To structure.
- Step 2: To integrate.
- Step 3: To self-control.
- Step 4: To design As-a-Service.
- Step 5: To describe in catalogue.

5.4 Functional aspects

This section describes how to design a software component As-a-Service in order to make it compliant with SOA and micro-services service design requirements. The proposed approach allows to cover progressively the required properties for its design. Design approach involves five steps.

Step 1: To structure

In a system where a service is available through the network and where the service is considered as a node of an architecture, this service should be structured. The structuring applies to services according to two aspects: the functional aspect representing the offered functionality and the non-functional aspect containing control functionality representing the automation and policies serving the management and control functionality. A component needs to have control and management interfaces. That is proposed to adopt the GCM structure. The resulting component structure at this step will be transformed to become a component including a management membrane with two interfaces: a control interface and a management interface. The membrane is proposed in GCM and standardized by ETSI TS 102 827 [i.1], ETSI TS 102 828 [i.2] and ETSI TS 102 829 [i.3]. Thus, the service is represented with a functional part, i.e. the business content, and a non-functional part, i.e. the membrane (Figure 6).

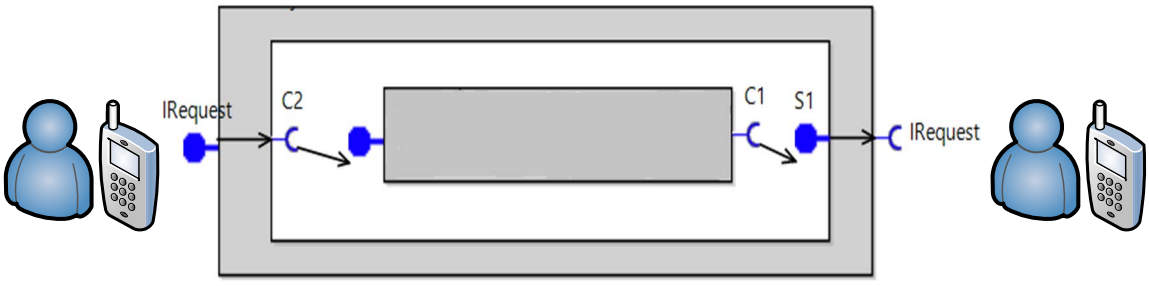

Figure 6: Component representation

Step 2: To integrate

To integrate a component into a global service environment, there is a need to link it to other components. Therefore, it is proposed to integrate the service through a definition of its functional and non-functional interfaces. It is related to the links that the interfaces provide. They allow the component to be invoked (functional) and managed (non-functional). Moreover, these interfaces allow to realize compositions based on the desired organization. This latter can be hierarchical, distributed or even centralized. Consequently, the service component is integrated in its service environment with a business content with external interface representing the functional part, and a membrane for the non-functional aspects, with management and control interfaces to be linked to other components and to communicate within the service environment.

5.5 Non-functional aspects: QoS

The non-functional aspects are handled by the component membrane. The component QoS in the membrane plays a role of interceptor. For all the component services, incoming service requests are tested, and then the functional content of the component transmitted via the corresponding internal interfaces. The structure of the self-control service component allows us to precisely specify the non-functional information flow.

Step 3: To self-control

For the control aspect, it is proposed to embed a QoS Control agent (Figure 7) in order to introduce the needed autonomic aspect in an environment that is meant to scale. The aim of introducing autonomic control of components is to enforce monitoring mechanisms that collect information concerning the behaviour of the component in order to control the respect of the Service Level Agreement (SLA) and QoS level and react in case of non-compliance. Thus, at this stage, the service becomes a self-controlled service. This should be achieved thanks to a recursive service architecture, where a service may comprise a set of self-controlled service components. Therefore, the service can be integrated within a global self-controlled service architecture.

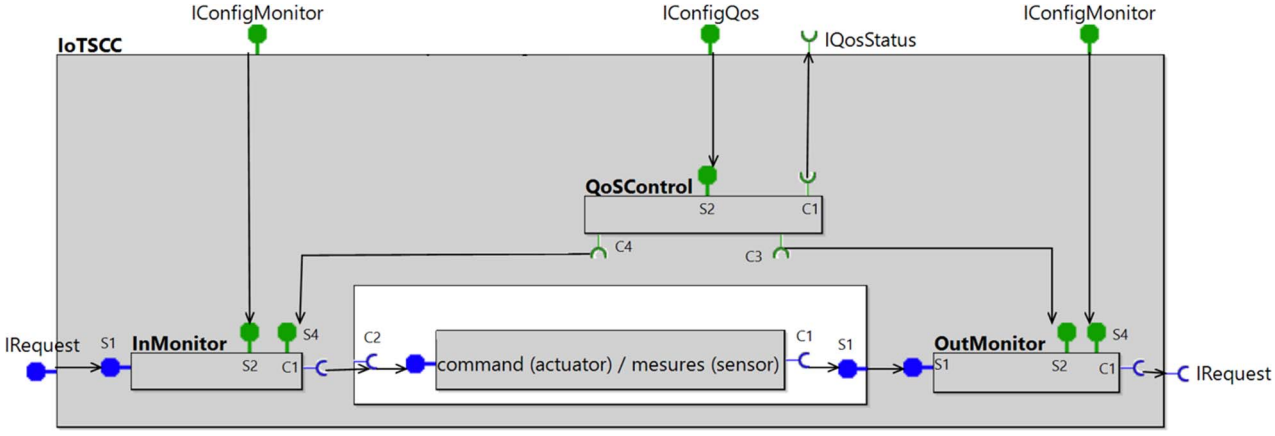

Figure 7: Self- control representation

Step 4: To design As-a-Service

This step aims at ensuring that the offered service component can be added, removed or composed with other services, without crashing the whole organization, i.e. the global service architecture. The "As-a-Service" design should allow flexibility in composing service offers and thus service customization, adaptability of services and solutions, as well as "on-the-fly" deployments. For this purpose, a set of properties needs to be satisfied for designing a "simple" service as an "As-a-Service" service. The service design needs to rely on the following main properties: statelessness, loose coupling and reuse.

As seen hereabove any service has a functional and also non-functional aspect that is the QoS.

6 QoS evaluation: New approach

6.0 QoS evaluation model

In the process of continuous improvement of services ITIL (Information Technology Infrastructure Library) standard recalls that three basic requirements are needed to be fulfilled to achieve an effective management. This includes the following sayings about measurements control and management [i.7]:

"You cannot manage what you cannot control.
You cannot control what you cannot measure.
You cannot measure what you cannot define."

The ITIL definition declines in the following aspects:

1) "To manage" is to control.

2) "To control" is to measure.

3) "To measure" is to define.

QoS evaluation should be declined as follows (Figure 8):

- "To define" answers the question what to measure and represents the metrics.
- "To measure" represents the perform of the measurement.
- "To control" represents the QoS control ("in/out contract").
- "To manage" represents FCAPS (Fault, Configuration, Accounting, Performance, Security) functionalities of ISO (International Organization for Standardization) management model.

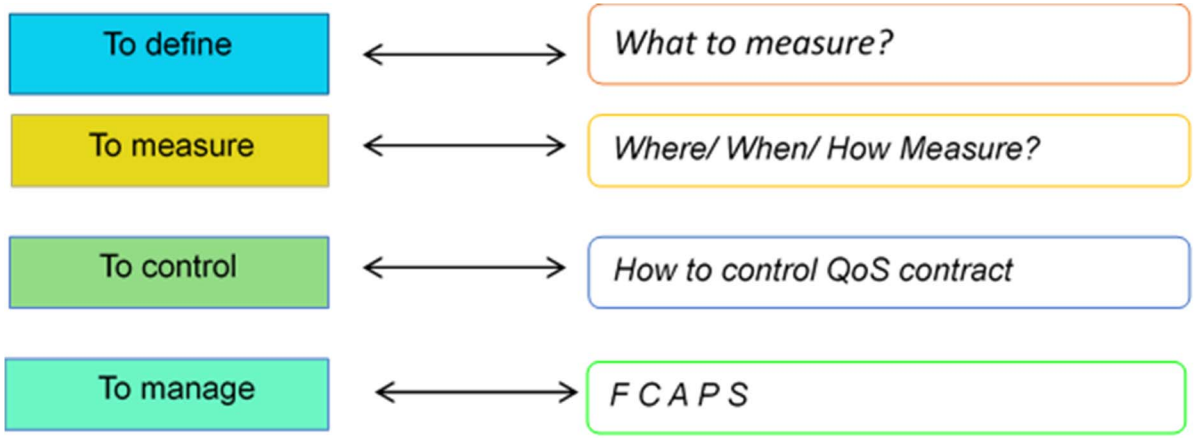

Figure 8: From metrics to management

The present document shows the model proposed to define the evaluation of the QoS requested by the user when choosing the service. The notion of QoS criteria (clause 6.1) is introduced to aggregate the measurable data (clause 6.2) and allow the calibration of services (clause 6.3).

6.1 Measurable requirements: QoS criteria

To describe the behaviour of the components and permit homogeneous QoS management, a generic QoS model has been defined. Four criteria are proposed to evaluate the QoS:

- **Availability:** Time and spatial transparency: Process and transfer information every time the user produces it and for as long as their generation lasts.
  Availability represents the accessibility rate of the service component (for example: accessibility rate).

- **Integrity:** Semantic Transparency: Processing and transferring is done in full and without changing its content.
  Integrity represents the capacity to run without alteration of information (for example: error rate).
- **Time:** Distance Transparency: Treat and transfer without changing the inherent time relationship to the information generated.
  Time represents the time required for request processing (for example: response time).
- **Capacity:** Transparency at source: Processing and transferring the amount of information generated instantly
  Capacity represents the maximum load the service component can handle (for example: processing capacity).

For each criterion, values are defined as:

- design;
- current; and
- threshold values.

The quality of service rendered by the service component should by presented in the three processes of the life cycle:

- design (THINK);
- deployment (BUILD); and
- operating (RUN).

Figure 9 shows the QoS values that it performs at each life cycle phase:

- In the design phase the QoS is evaluated such as the design value by calibration.
- In the deployment phase the QoS is compared between the QoS offered value and QoS demanded value.
- In the provisioning phase, the QoS is evaluated, according to the state of the current resources, if it can answer the query with the required QoS.
- In the operating phase, the QoS control of current values takes continuously the state of service component resources.

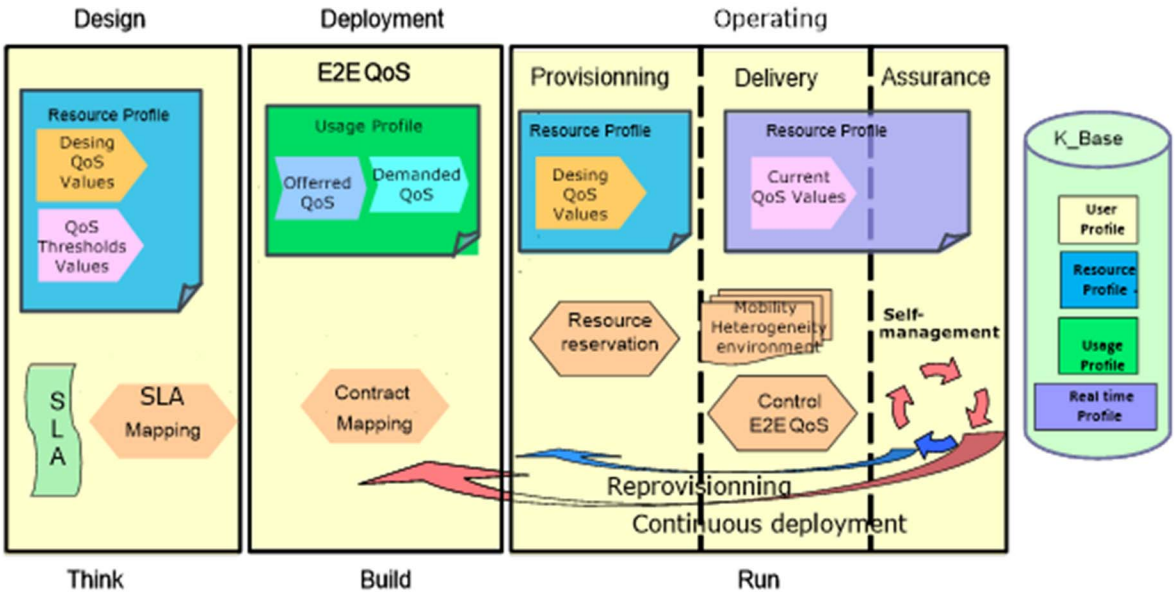

Figure 9: Lifecycle and QoS Values

The values defined for control should be obtained during the calibration phase.

EXAMPLE: The resources (e.g. CPU, RAM) needed for design values.

6.2 The measure

6.2.0 The measure model

Service providers usually offer services to their customers with certain Quality of Service (QoS) guarantees specified in Service Level Agreements (SLAs). To make this possible, providers need to know the capabilities of the services they offer.

With the growing sophistication of applications running on infrastructures, monitoring and control will become an indispensable feature. Monitoring is a prerequisite for an efficient control.

The measure is detailed through these four questions:

1) What to measure?
2) When to measure?
3) Where to measure?
4) How to measure? (Figure 10).

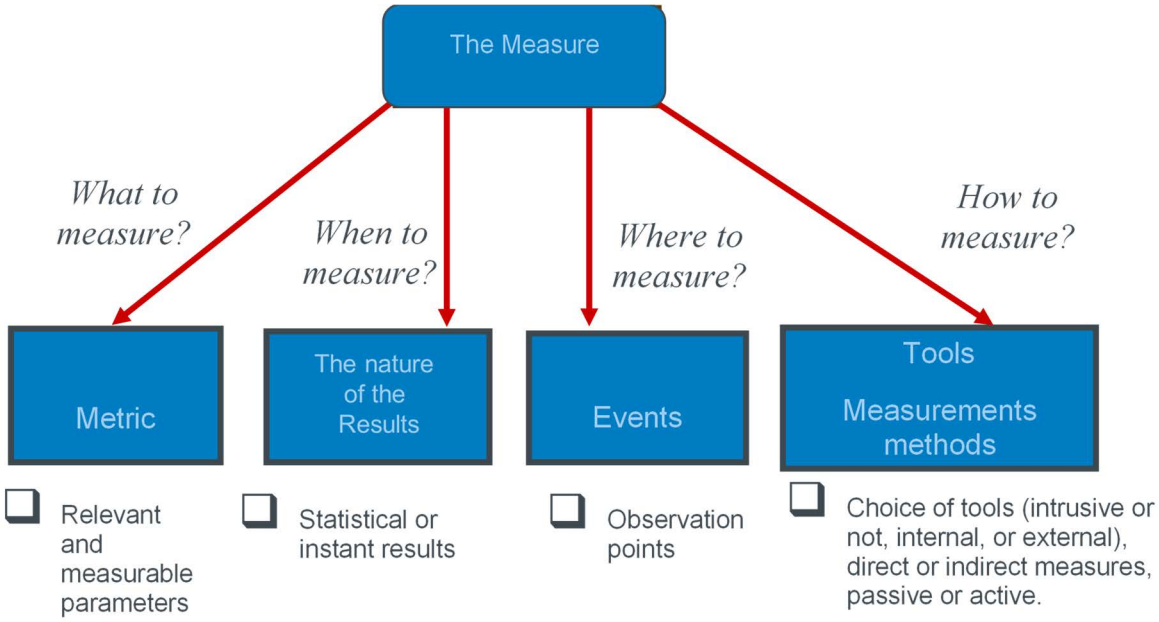

Figure 10: The measure

6.2.1 What to measure

All measurable data, which are the basic data which will allow after calculation to evaluate the criteria of QoS, like the parameters of the system (CPU and RAM) or those of the network protocols or any other significant indicator.

The QoS criteria will be evaluated, for example, using the following data:

- **Availability:** should be used to measure rejected requests when running the service to get the availability rate.
- **Integrity:** measure queries that have been altered by the service's execution of the service to get the error rate.
- **Time:** increase the time at the entrance of the service and the time to leave the service to get the time to run.
- **Capacity:** count the number of requests processed to get the processing capacity.

The number of incoming or outgoing requests (queries, packets, primitive) and their timestamp are recorded. The QoSControl agent periodically ask the monitor for their records. It can detect if a request has been processed by the business component or not and can compute the number of processed/unprocessed requests and the processing time by subtracting the out and in time stamps. The QoSControl agent can compute other metrics like for example, the availability of the component or the number of processed requests by minutes and consider richer model like moving averages. In a normal situation, it checks compliance with the SLA by comparing the result with a reference threshold and sends an in or out contract signal.

6.2.2 When to measure?

It all depends on the nature of the desired response.

If statistical data are needed to make the mathematical evaluation of the behaviour of the system or the analysis of the effectiveness of the communication (performance) or to collect information on the state of the system (configuration), then the measurement will be done all the time continuously.

Whereas, if it is needed to analyse a point event discrete values will be collected.

6.2.3 Where to measure?

The measurement falls under the activity of "monitoring" which is a task of structural monitoring and application supervision which falls to the administrator of the ecosystem (network + IT platform). Based on previously established control points (observation points), it essentially consists in ensuring that the data flow is in line with the SLA, in order to proactively remedy the problems affecting the behaviour of the system.

The observation points depend on the type of activity monitored such as these:

- **Commercial activities:** which analyses the data associated with the commercial processes for a better competitive advantage.
- **Data security:** which guarantees the protection and security of files, data and personal information profiles.

For the QoS control described in this document, it is at the level of each "As-a-Service" element observed and a detailed view of the criteria and their variations can hence be obtained.

6.2.4 How to measure?

Basically, a digital ecosystem should be including a generic monitor that measures the number of arriving requests, the number of erroneous or rejected requests, as well as the input and output time. To obtain these values "counters" and "timestamp" using the system time are needed. More advanced monitoring functionalities could be designed for checking.

In the new approach, the "Monitor As-a-Service" should be a software component as simple as possible.

The measurement component is cut-through with the user interface. It allows observation of the input and output of the functional component.

At the input, the InMonitor records information when the functional component is requested. On output, the OutMonitor records information on the response of the functional component.

The use of two measurement components makes it possible to have precise numerical values on the input and the output of the service.

InMonitor and OutMonitor will perform the measurements but will not analyse the metrics or make decisions.InMonitor records information about the solicitation of the functional component. OutMonitor records information about the functional component's response.

The use of two monitoring components allows for having precise numerical values on the input and output of the service.

InMonitor and OutMonitor will perform the measurements but will not analyse the metrics or make decisions.

6.3 Calibration

6.3.1 Calibration method

- Unit tests (to get design values), self-testing.
- Support tests (to get threshold values), self-testing.
- "Reference" tests to see the consequence of the environment, to refine the first two values (for use): with network or without network, in different contexts.

The auto-testing allows for defining the values of QoS offered (or expected).

In order not to interfere with the overall operation, incoming queries can be placed in a queue (the time needed for the auto-test to end up). Stored queries are processed and sent to the "Business"(functional) component.

In the self-test procedure, during design phase, *it is common practise* to compute the offered/nominal QoS and the threshold values from which the business component stops responding by gradually increasing the numbers of requests. The obtained QoS are given on resources conditions because they depend on their environment. Reference tests can also be processed by modifying the resources to highlight the effect of the environment on the measures.

6.3.2 Calibration result: QoS design and threshold values.

Calibration should be done for each service, as well as for service composition.

To support self-management of resources, three values are defined for each quality of service criterion:

- The design value (QoS offered) is the value determined during the design phase of the service.
- The current value is the value monitored over the life of the service.
- The threshold value represents the limit that the criterion should not exceed in order for the component to ensure the expected processing of demands.

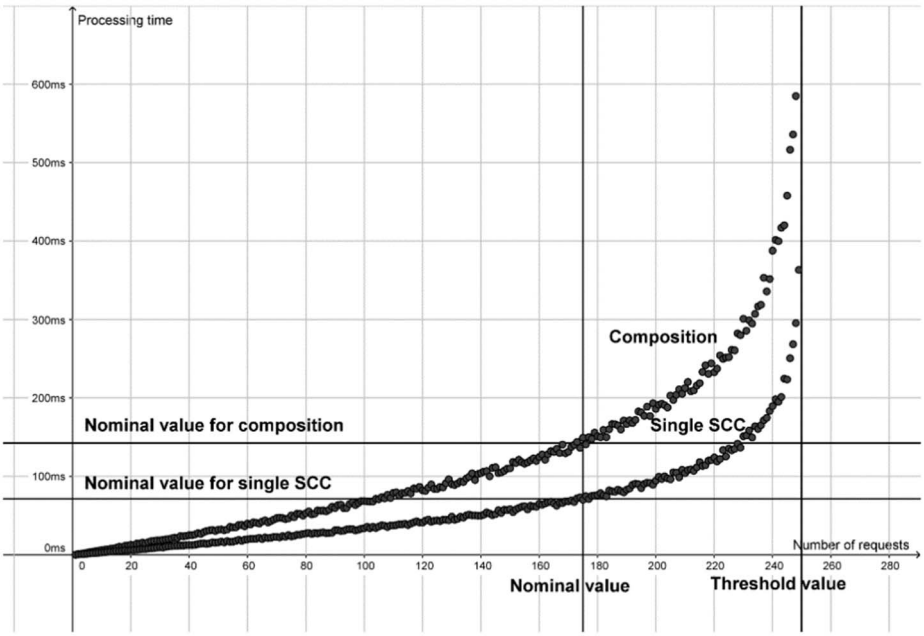

Figure 11: Digest Access Authentication and authorization to service calibration (example)

The first service component performs authentication of users based on Digest Access Authentication (challenge-response codes) (IETF RFC 2617 [i.6]).

The second service component is responsible for checking the user rights towards performing some actions.

The user should provide the right "response" code in his request to prove it is authenticated otherwise the component sends the message "401 Unauthorized" asking him to authenticate first.

NOTE: service tasks could be of any nature and can cover a lot of domains as computer vision systems and image processing, signal processing, web services, internet of things services, etc.

The threshold value is determined as the value from which the service component stops responding (Figure 11). Here for 250 requests.

The service provider chooses **a nominal value**, which may be defined, for example, at 70 % of the threshold value: 146,6 ms for 175 requests (Figure 11).

If the composition has been entirely calibrated (as is the case here) then it can be put in a catalogue too.

Secondly, the calibration of a composition includes an authentication and authorization service. As previously mentioned, to design an application or service, the provider chooses multi-tenant components in providers' catalogues, based on the specified nominal/offered QoS and thresholds values. The provider calibrates the composition with the same technique as the one described to obtain the nominal QoS and threshold value of the full composition.

By repeating the operation and by increasing at each time the number of requests, the provider can compute the average processing time for a given physical resources level:

- CPU usage in OpenIMS (Open IP Multimedia Subsystem) and IMS-as-a-service (Figure 12); and
- RAM in OpenIMS (Open IP Multimedia Subsystem) and IMS-as-a-service (Figure 13).

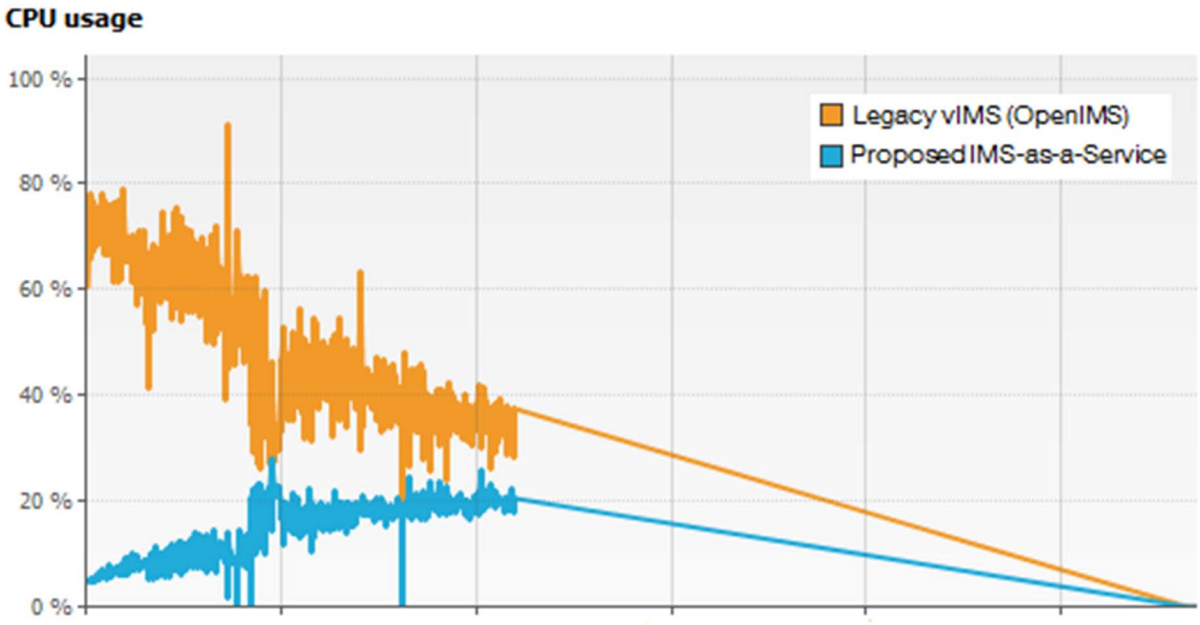

Figure 12: Resource consumption (CPU) of authentication service

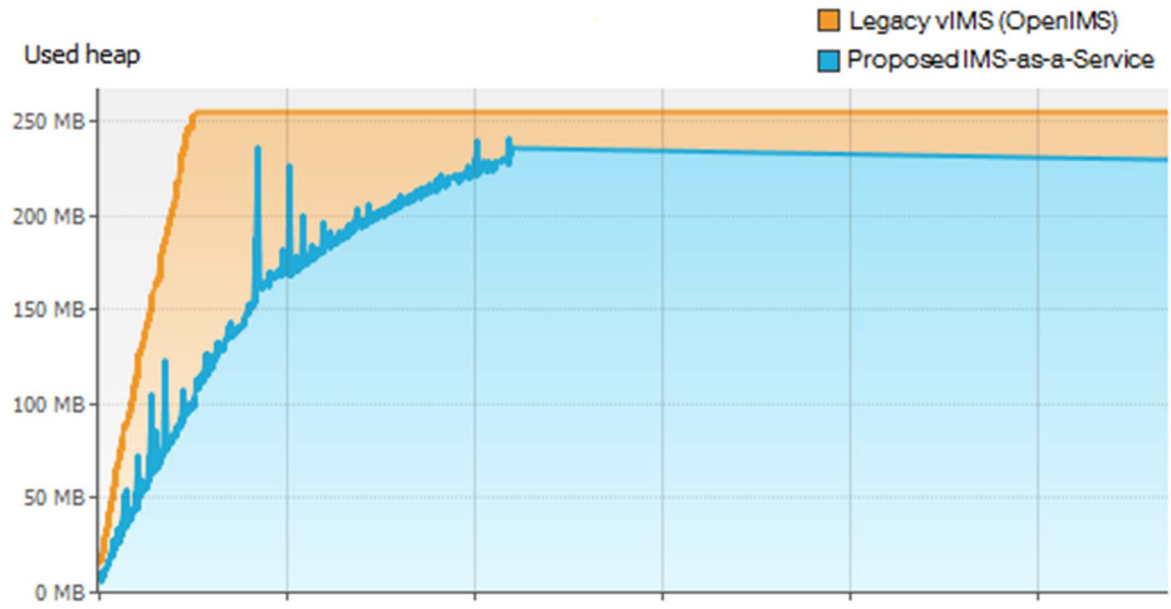

Figure 13: Resource consumption (RAM) of authentication service

7 Requested service

7.1 Service composition

The new QoS approach makes it possible to control the QoS compliance of each service component. The component As-a-Service should be designed to meet the properties of "As-a-Service" detailed in the preceding paragraphs.

The users can design their own service by choosing and assembling the elementary components. This composition would thus be customizable and flexible by adding, replacing and deleting service elements according to the needs of the users. The composition of a service consists in generating a global service by composing or by chaining a set of service components "As-a-Service".

How to make the composition also "As-a-Service"?

This will be done by construction, because with each assembly, the designer (Provider or User) gets a new service (Figure 14) with its own QoS component, its input monitor and its output monitor.

And consequently, the new architecture will consolidate the functionality As-a-Service and will provide the interoperability and flexibility needed by users, through the composition of services.

Sensors and actuators are also offered As-a-Service in this ecosystem and can interact with other services. They will serve as the basis for human interaction with the digital ecosystem.

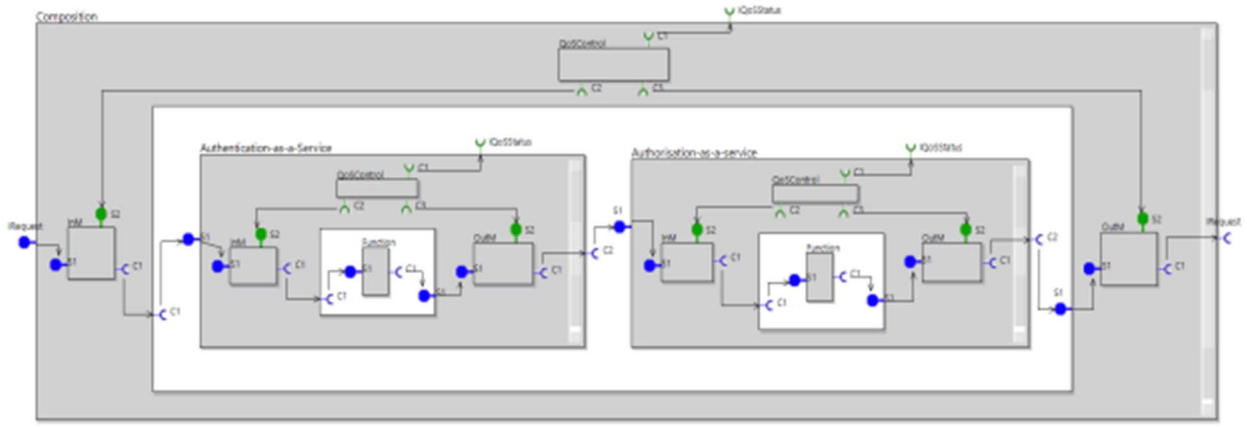

Figure 14: User Service composition

7.2 User end-to-end service

As seen above, the overall service of the user is an "As-a-Service" element composition which should respect the end-to-end QoS. Therefore, a monitor is at the user terminal and another at the application server to check compliance with the SLA.

To meet the requirements of the digital ecosystem the end-to-end personalized service should be delivered according to the user profile and with the following properties:

- flexibility;
- dynamicity; and
- sensitivity to QoS/QoE (Figure 15).

In most solutions the provider tends to improve the user subjective satisfaction (QoE). In some areas, however, the end-to-end QoS remains essential and, from a user's viewpoint, is the most relevant. In a critical situation, as aeronautics or healthcare, processing time is important. The time between the measures done by the sensors and their representation on the screen needs to be controlled. If the processing time is too long, the displayed data no longer represents the reality, which can put the user at risk. The users can have a good perceived QoE (fluidity, responsiveness, etc.) but if the end-to-end QoS (processing time) is too long, the information displayed on the screen may be obsolete, creating a delay between the screen and the reality, while maintaining a good apparent fluidity.

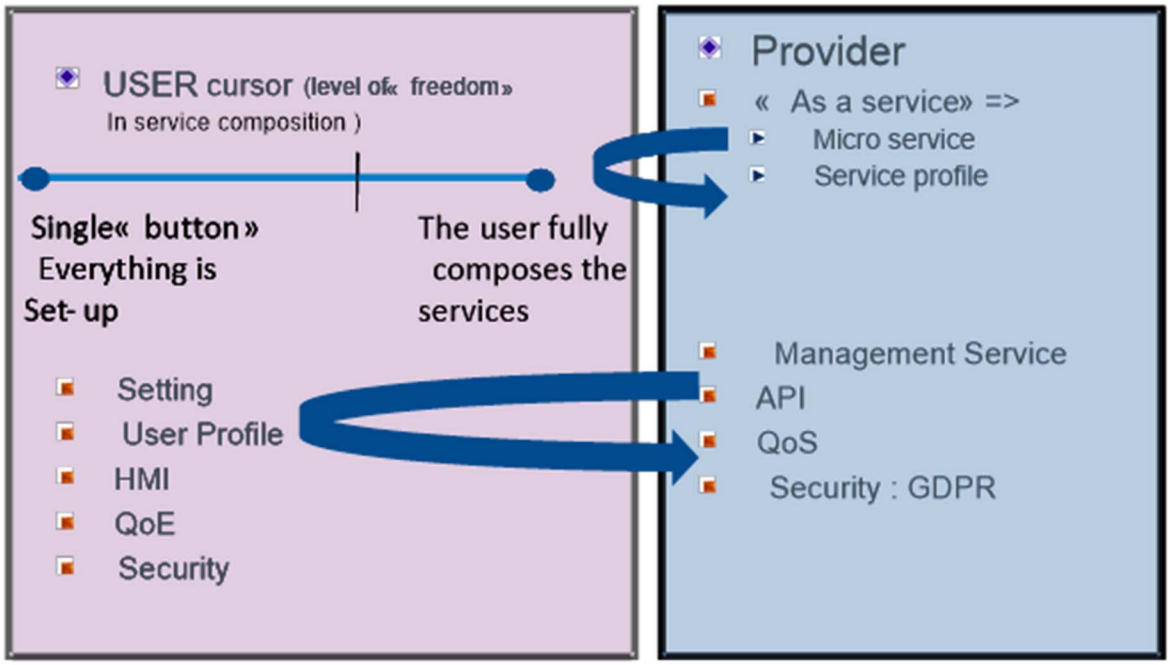

Figure 15: Requested service and Provider service composition

The availability monitoring will ensure the actual availability of each hardware and software component of the system and should allow a monitoring of the user experience, which improves the user comfort by monitoring the flow exchanged with the system.

The monitoring of the "time" criterion then represents the end-to-end response time, which measures the response times to user requests on applications.

8 Catalogue

8.1 The role of the catalogue

The services presented in the supplier catalogues allow for comparing the services of the suppliers. Each service is described through functional and non-functional (QoS) specifications:

- The catalogue is a showcase for reusable components.
- The provider chooses them according to their QoS.
- For the user, each service is proposed with his offered QoS.
- The user can compare the services of different providers.

The role of the catalogue is the following:

- Issue for the supplier: provision of service with added value in phase with the market (TTM) relying on Information Systems (IS), organization and technical aspects (API, QoS). The catalogue is the focal point of the ecosystem, it is a trade-off between technical and economic viability.
- Issue for the User: easier access for customizing his request according to his location, his agenda, his preferences and means.

In the design of As-a-Service approach, Step 5 aims "to describe in catalogue":

- A Service needs to be correctly described by the service provider and visible by users that would request it. For that, there is a need for service description and service registry using formal processes. These two properties allow for establishing a service catalogue. To design an application or a service, the architect selects multi-tenant services in the provider's catalogue. Using the QoS-based As-a-Service model, the selection would be based on the specified offered QoS and thresholds values. At run-time, in order to introduce agility and eliminate static configurations, a service has to be invoked through standardized API. A service would include interfaces dedicated to QoS compliance control, service control and service programming.

- The catalogue is a showcase for reusable components. The provider chooses them according to their QoS. But as mentioned hereabove, in a context of components re-utilization, it is appropriate to know the offered QoS and the needed resources to provide this QoS. Indeed, for the same functionality, different algorithms and treatments may be used and therefore different QoS are provided. The consumed resources are not the same. That is why, the provider's catalogue is filled with calibrated components.

- Each service is given with its offered QoS and the associated resources conditions. A component located at the lower layer depends on hardware resources (CPU, RAM).

8.2 Example: Automatic Number Plate Recognition System

This use-case deals with an automatic Number Plate Recognition system that logs and controls the access of the vehicles (Figure 16) to a parking.

It uses high-resolution digital cameras and LED illuminations technologies.

As a vehicle arrives at a checkpoint (barrier, gate, weighbridge), the system automatically captures and recognizes in real time the license plates. It queries the database to determine if the vehicle has the authorization to proceed, and to get more information about the vehicle (model, type, owner). According to the vehicle authorization, the system opens the gate or the barrier or play an audible or visual alarm to an operator. It accelerates and secures the access control and tracking of the vehicles.

It is designed to:

1) Automatically recognize the plate number of vehicles.

2) Manage access permissions by license plate, vehicle type, time slots.

3) Execute actions automatically according to the license plate detected.

This system is built on four independent services:

1) Photo taking.

2) Photo transformation (light improvement for example).

3) Character recognition.

4) Access to the database.

Each service can run on a separated CPU like the small single-board computer Raspberry Pi computer for example.

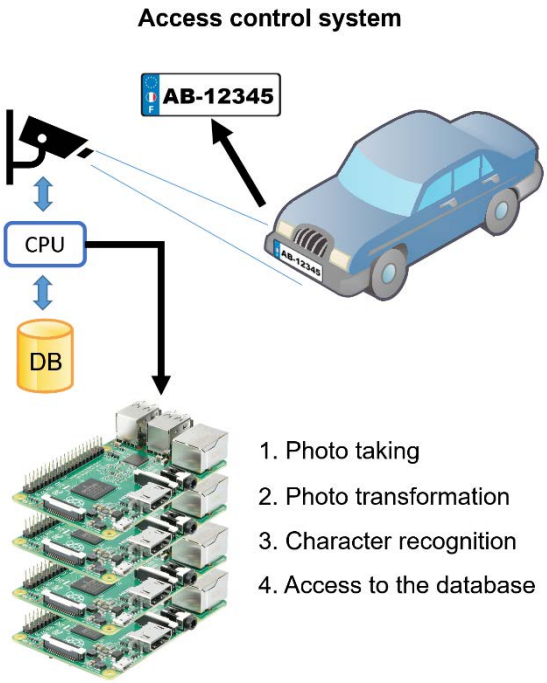

Figure 16: Access control service composition

The service composition in adequacy to the QoS requested by user is summarized as follows:

1) At the beginning, the four service components should be calibrated. Each service is thus calibrated on a Raspberry Pi card:

   a) Photo taking represents the calibration of the image capture service.

   b) Photo transformation represents the photo transformation service.

   c) Character recognition represents the character recognition service.

   d) Access to the database represents the database access service.

By increasing the number of requests, the provider determines the threshold value from which the service component stops responding. The provider computes the offered QoS and the threshold value from which the service component stops responding. He chooses a nominal value, which may be defined, for example, at 60 % of the threshold value. After the calibration of the component is completed, it can be put in the provider catalogue.

2) Secondly, the service provider creates his catalogue by putting into it the four preceding calibrated components. The provider's catalogue is filled with calibrated components. Each component is given with its offered/nominal QoS.

3) Thirdly, to design the application, the architect chooses multi-tenant services in providers catalogues, based on the specified nominal/offered QoS and thresholds values. The catalogue is a showcase for reusable components. The architect chooses them according to their QoS.

For the same functionality, different algorithms and treatments may be used and therefore different QoS are provided. The consumed resources are not the same.

In this example (Figure 17), the provider chooses the four preceding detailed services and build a composition.

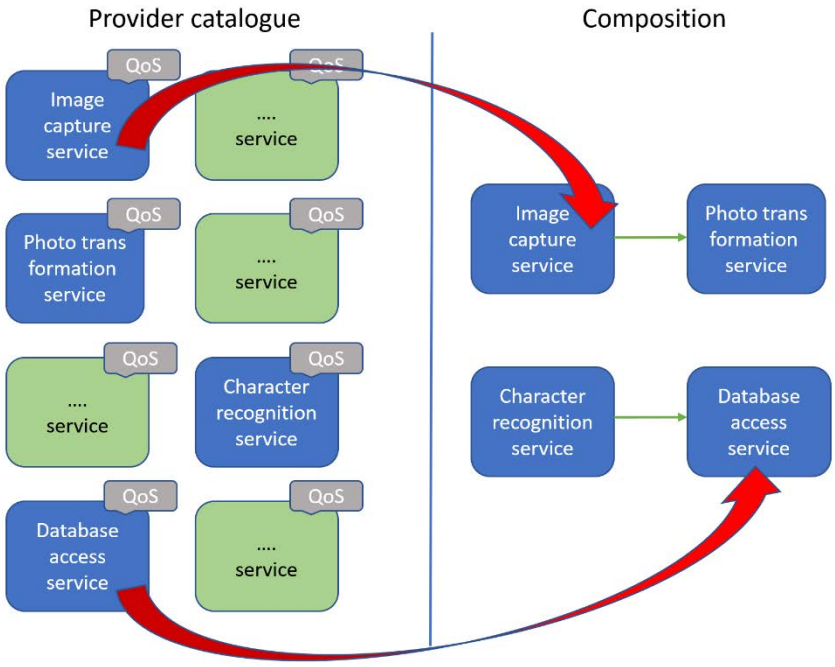

Figure 17: Service composition from the catalogue

4) Finally, SLA management actions will be proposed to ensure the adequacy of its nominal QoS to the requested QoS (SLO).

9 Use cases: service composition in medical warning system

This use case can easily be extended to IoT environments where processing time is crucial or when human decision-making is necessary, especially in critical and urgent situations in which QoS should be controlled (Figure 18).

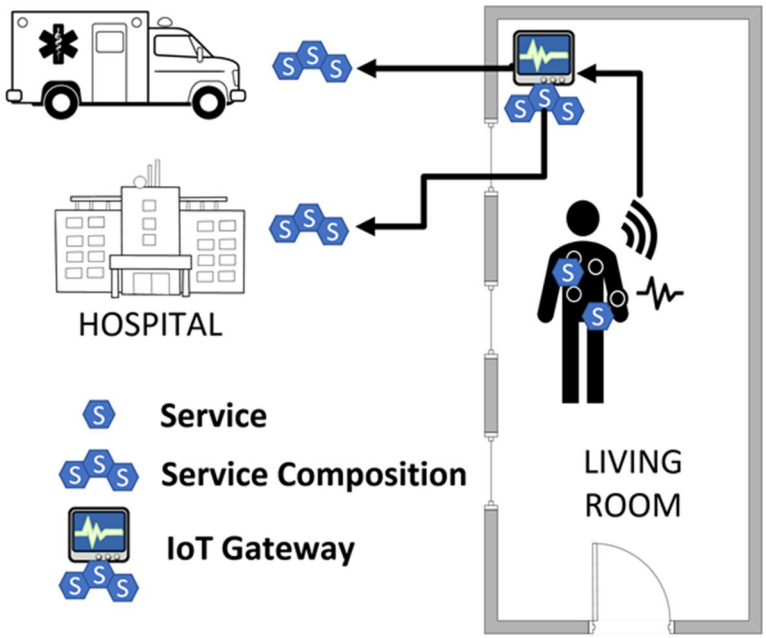

Figure 18: Service composition in medical warning system

Healthcare connected medical devices and applications are already creating an Internet of Medical Things which is contributing to better health monitoring and preventive care.

Due to the importance of observing the medical state of patients who are suffering from acute diseases, especially cardiovascular diseases, a continuous remote patient monitoring is essential.

With the help of wearable wireless sensors, a service-based system can provide a continual access to the medical parameters of a patient. IoT Gateways are located in every room in the house in a way to follow the patient. They are equipped with computational capacity. They monitor the current state of the patient and provide a means to predict future medical condition via machine learning methods and artificial intelligence algorithms. They are able to contact the rescue teams according to the type of emergency detected and to notify the nearest hospital of the arrival of the patient.

Due to the constant incoming data in a continuous medical monitoring, the system may encounter problems such as latency in system response, data transmission and computations related to data analytics. QoS has to be controlled from end-to-end. The processing time of the services chain from end-to-end has to be controlled (Figure 18).

For any of these applications, failures might lead to serious injury (including on a large scale). As the number of objects, including sensors and actuators, increases, IoT becomes more and more complex and should be controlled, especially in these critical domains. This implies to compose and structure an application with controlled service components. Services have first to be designed, calibrated and provided, with an offered QoS, in a digital catalogue.

The methodology is summarized as follows:

- To monitor his health, the user needs a personalized application.
- This application is built As-a-Service composition.
- Services are offered by providers like those who are integrated in the IoT gateway (prediction of future medical condition via machine learning methods and artificial intelligence algorithms) or provided by hospitals.
- These services have been calibrated and their nominal/offered QoS is known.
- Services are thus composed/linked to form the health application requested by the user.
- The QoS from end-to-end (processing time) is known in advance and can be controlled.

Annex A:
Change History

Date	Version	Information about changes
07-2020	<1>	First version

History

Document history		
V1.1.1	November 2020	Publication