A novel reconfigurable by design highly distributed applications development paradigm over programmable infrastructure

D2.4 - Use Cases and Performance Evaluation and Validation Methods

Editors:	K. Tsagkaris (WINGS), N. Koutsouris (WINGS), P. Gouvas (UBITECH)
Contributors:	N. Havaranis, E. Tzifa, A. Sarli (WINGS), E. Fotopoulou, C. Vassilakis, A. Zafeiropoulos (UBITECH), M. Repetto (CNIT), S. Kovaci, T. Quang (TUB), A. Rossini (SINTEF), J. Sterle (UL), S. Siravo (MAGGIOLI), G. Kioumourtzis, E. Charalampous (ADITESS), L. Porwol (NUIG)
Date:	27/11/2015
Version:	1.00
Status:	Final
Workpackage:	WP2 –ARCADIA Framework Specifications
Classification:	Public

ARCADIA Profile

Grant Agreement No.:	645372
Acronym:	ARCADIA
Title:	A NOVEL RECONFIGURABLE BY DESIGN HIGHLY DISTRIBUTED APPLICATIONS DEVELOPMENT PARADIGM OVER PROGRAMMABLE INFRASTRUCTURE
URL:	http://www.arcadia-framework.eu/
Start Date:	01/01/2015
Duration:	36 months

Partners

NUI Galway OÉ Gaillimh	Insight Centre for Data Analytics, National University of Ireland, Galway	Ireland
SINTEF	Stiftelsen SINTEF	Norway
berlin	Technische Universität Berlin	Germany
cnit	Consorzio Nazionale Interuniversitario per le Telecomunicazioni	Italy
Univerza v Ljubljani	Univerza v Ljubljani	Slovenia
UBITECH ubiquitous solutions	UBITECH	Greece
WINGS ICT Solutions	WINGS ICT Solutions Information & Communication Technologies EPE	Greece
GRUPPO Maggioli	MAGGIOLI SPA	Italy
aditess Advanced Integrated Technology Solutions & Services	ADITESS Advanced Integrated Technology Solutions and Services Ltd	Cyprus

Document History

Version	Date	Author (Partner)	Remarks
0.10	17/07/2015	K. Tsagkaris, N. Koutsouris (WINGS)	**Preparation of Table of Contents (ToC)**
0.20	30/07/2015	N. Koutsouris, N. Havaranis (WINGS)	**Methodologies for the performance evaluation and the validation of use cases – Section 2**
0.30	28/08/2015	E. Fotopoulou, P. Gouvas, C. Vassilakis, A. Zafeiropoulos (UBITECH),M. Repetto (CNIT), S. Kovaci, T. Quang (TUB), J. Sterle (UL), S. Siravo (MAGGIOLI), G. Kioumourtzis, (ADITESS)	**Introduction and scenario description for each use case – Sections 3, 4, 5**
0.40	14/09/2015	E. Fotopoulou, P. Gouvas, C. Vassilakis, A. Zafeiropoulos (UBITECH),M. Repetto (CNIT), S. Kovaci, T. Quang (TUB), J. Sterle (UL), S. Siravo (MAGGIOLI), G. Kioumourtzis, (ADITESS)	**Elaborated description of the scenarios, key performance indicators, validation – Sections 3, 4, 5**
0.50	06/10/2015	E. Fotopoulou, P. Gouvas, C. Vassilakis, A. Zafeiropoulos (UBITECH),M. Repetto (CNIT), S. Kovaci, T. Quang (TUB), J. Sterle (UL), S. Siravo (MAGGIOLI), G. Kioumourtzis, (ADITESS)	**Micro-services and detailed service graph for each use case – final description of the scenarios and the performance evaluation parameters – Sections 3, 4, 5**
0.60	22/10/2015	K. Tsagkaris, N. Koutsouris, N. Havaranis, E. Tzifa, A. Sarli (WINGS), E. Fotopoulou, P. Gouvas, C. Vassilakis, A. Zafeiropoulos (UBITECH),M. Repetto (CNIT), S. Kovaci, T. Quang (TUB), A. Rossini (SINTEF), J. Sterle (UL), S. Siravo (MAGGIOLI), G. Kioumourtzis, E. Charalampous (ADITESS), L. Porwol (NUIG)	**Final version of the deliverable –Editing of Sections 1 and 5, Version for Internal review**
1.00	27/11/2015	K. Tsagkaris, N. Koutsouris, N. Havaranis, E. Tzifa, A. Sarli (WINGS), E. Fotopoulou, P. Gouvas, C. Vassilakis, A. Zafeiropoulos (UBITECH),M. Repetto (CNIT), S. Kovaci, T. Quang (TUB), A. Rossini (SINTEF), J. Sterle (UL), S. Siravo (MAGGIOLI), G. Kioumourtzis, E. Charalampous (ADITESS), L. Porwol (NUIG)	**Update based on comments from internal review - Final version**

Executive Summary

This deliverable sets the cornerstone in the development of the three use cases of the ARCADIA project. It includes all the necessary information for the successful implementation, validation and performance evaluation of the proposed scenarios, which will ensure the delivery of strong proof-of-concept demonstrations. Three discrete domains are included in the use cases, targeting at demonstrating and validating the ARCADIA framework for tackling a wide set of distributed applications deployment challenges.

The use cases are going to be developed based on the ARCADIA Framework and thus numerous challenges are addressed like e.g. the creation of service graphs consisted of ARCADIA components and links among them, the adaptation of existing software components based on the ARCADIA development paradigm. In this context, within D2.4, the specification of the ARCADIA components that are involved in each use case is provided, as well as their role and their interconnection according to the designed service graphs. The requirements denoted or the preferences suggested on behalf of the developers as well as the policies imposed on behalf of the services provider are detailed and going to be taken into account towards the deployment of the proposed distributed applications.

Furthermore, specific parameters are defined as key performance indicators along with a set of acceptance criteria, separately for each use case, in order to be used towards the performance evaluation phase of the use cases. Based on the evaluation results, a set of insights and suggestions with regards to the applicability and the optimal usage of the ARCADIA framework are going to be extracted.

Table of Contents

1 Introduction 6
1.1 Purpose and Scope 6
1.2 Relation with other WPs 6
2 Energy Efficiency vs Quality of Service (QoS) trade-off 6
2.1 Use Case Description 6
2.2 Service Graph Description 11
2.3 Service Provider Policies 14
2.4 Key Performance Indicators and Acceptance Criteria 14
2.5 Programmable Infrastructure 16
3 High Performance Survivable Communications in Distributed IoT Deployments 17
3.1 General description of the use case 17
3.2 Use case targets and preconditions 20
3.3 Detailed description of the use case 27
3.4 Use case service graph 30
3.5 Validation 32
3.6 Performance evaluation 34
4 Security and Privacy Support in the FIWARE Platform 34
4.1 Introduction 34
4.2 Use case description 34
4.3 Use case service graph 40
4.4 Service Provider Policy 42
4.5 Validation and KPI 42
4.6 Programmable Infrastructure 43
5 Performance evaluation and validation framework 44
5.1 Performance Evaluation Criteria 44
5.2 Performance Evaluation and Validation Process 50
6 Conclusions 50
Annex I: References 51

1 Introduction

1.1 Purpose and Scope

This deliverable elaborates on the three use cases that are going to be deployed within the ARCADIA project. As a document, it constitute the basis for the research and development activities that are going to take place in WP5. In addition to the detailed description of the use cases, it describes the methods for validating and evaluating the performance per use case. It has to be noted that the information contained in this document will not be necessarily static, but it may be updated during the progress of the project, in order to incorporate new knowledge and to adapt to any design or implementation modifications.

The main purpose of this deliverable is to communicate a clear and common understanding of the tasks that are required to be accomplished so as to implement, validate and evaluate the ARCADIA use cases. The various functionalities that are envisioned to be developed and demonstrated, the expectations regarding performance and Quality of Experience, the context and the infrastructure in which each of the applications will be used and tested, are aspects that are addressed in the following sections.

More specifically, a dedicated section is provided per use case, providing the description of the scenario considered in each use case, the need for adopting the ARCADIA framework, the components and service graph that is going to be developed and deployed, the policies denoted on behalf of the services provider and the programmable infrastructure that is going to be used. Furthermore, the key performance indicators along with their acceptance criteria are provided per use case.

1.2 Relation with other WPs

The definition of the use cases is based on the specification of the ARCADIA framework as it is documented in D2.3 of the project as well as the specification of the ARCADIA Context Model as it is documented in D2.2. Exploitation of the artefacts envisaged to be produced within the project for tackling the challenges identified per use case is envisaged. Furthermore, the material provided in this deliverable is going to be used mainly towards the implementation and evaluation of the use cases in WP5 as well as within WP6 for communication, dissemination and exploitation purposes.

2 Energy Efficiency vs Quality of Service (QoS) trade-off

2.1 Use Case Description

Energy efficient operation of IT infrastructures entails an optimal balance between performance and power consumption. That means an application should only be assigned the optimal amount of computing/processing resources to fulfil its quality requirements, taking into account the considered trade-off. Unfortunately, the workload continuously changes with a certain degree of unpredictability, and resources cannot be provisioned in real-time. To avoid violations of Service Level Agreements (SLAs), usually more resources than necessary are provided, thus lowering the overall efficiency. This use case will show how the ARCADIA framework could be used to design better energy-aware resource management schemes, by exploiting the application's structure and components.

The use case considers a video streaming application, which includes transcoding capability. This scenario is representative of a multimedia content distribution service; where a variable number of users request their live or on-demand videos and have stringent expectation for both Quality of Service (e.g., seamless flow of the stream) and Quality of Experience (e.g., quick start of the streaming in response to their request). The main purpose is to show how the ARCADIA framework can be exploited to respect SLAs and to provide the expected Quality of Service/Experience while reducing the energy consumption of the programmable infrastructure.

2.1.1 Background

Video applications can be generally classified into two categories: (i) on demand streaming, and (ii) live streaming. Each one video category poses its own set of service requirements and design issues. The most important performance measure, for instance, for both streaming (on demand) stored video and live streaming is average throughput. Latency, is a more stringent requirement in live video streaming compared to video on demand streaming.

Video Transcoding

With the increasing current use of mobile and internet technologies, the diversity of user terminals and video encoding schemes, video files need to be processed in such a way that they can better fit with the user requirements. Therefore, a video file needs to be encoded in a proper format and quality so that it can better meet these requirements. This procedure is called video transcoding. Video transcoding, however, requires extensive processing power and large investments for hardware procurement that make it difficult for most video service providers to afford. And apart from the extensive processing requirements and the hardware cost, power consumption is another major factor as it constitutes a large portion of operational expenses for running and maintaining the infrastructure.

Adaptive HTTP Streaming

HTTP streaming is a solution that has been extensively deployed in practice for streaming multimedia content. Adaptive streaming solutions are deployed in order to provide the highest video quality that is closer to the user receiving capacity [1] and [3]. Current efforts have concentrated on adaptive streaming by using TCP as the transport protocol instead, which is called Dynamic Adaptive Streaming over HTTP (DASH), which is similar to Apple's HTTP Live Streaming (HLS). In DASH, several copies with different quality of the video are encoded and stored on a server. Video files with higher quality require higher receiving rates. The client dynamically requests chunks of video segments of a few seconds in length from the different versions. When there is adequate bandwidth, the provider upgrades the client to the next better QoS, providing better experience. When the available bandwidth is reduced, the client is downgraded to the QoS class that matches its requirements. Current efforts in DASH are carried out by the Moving Picture Expert Group (MPEG). MPEG recently developed a standard for streaming multimedia over Internet. The standard is known as MPEG-DASH (ISO/IEC 23009-1) [4].

Content Delivery Networks (CDNs)

Today, many video service providers are distributing on-demand multi-Mbps streams to millions of users on a daily basis. YouTube, for example, with a huge library of millions of videos, distributes hundreds of millions of video files around the world [5]. In order to meet the challenge of distributing massive amounts of video files, major video provider companies make use of CDNs. A CDN manages servers in multiple geographical locations, stores copies of videos along with other Web content in its servers, and attempts to redirect each user request to a CDN location that is closer to user, increasing user quality experience.

2.1.2 Business scenario

The use case considers a media Service Provider (video on demand or news provider) that offers media services to its clients. This scenario is representative of a multimedia content distribution service; users request their live or on-demand videos and get the content encoded according to their preferences, their devices, and the current network conditions.

Roughly, the layout of the use case streaming application will be made of media storage, transcoding mechanisms and media servers. Video service providers upload compressed video files via a service interface (e.g. web-based) to cloud transcoding services. Transcoding changes the video format, resolution, and quality and creates a number of output video files with the same content. Video files are distributed via CDNs. Video clients can request video files that match their quality and format criteria via a service interface (e.g. web-based). Figure 1 shows the conceptual architecture of the video transcoding and streaming application in terms of the logical components described so far.

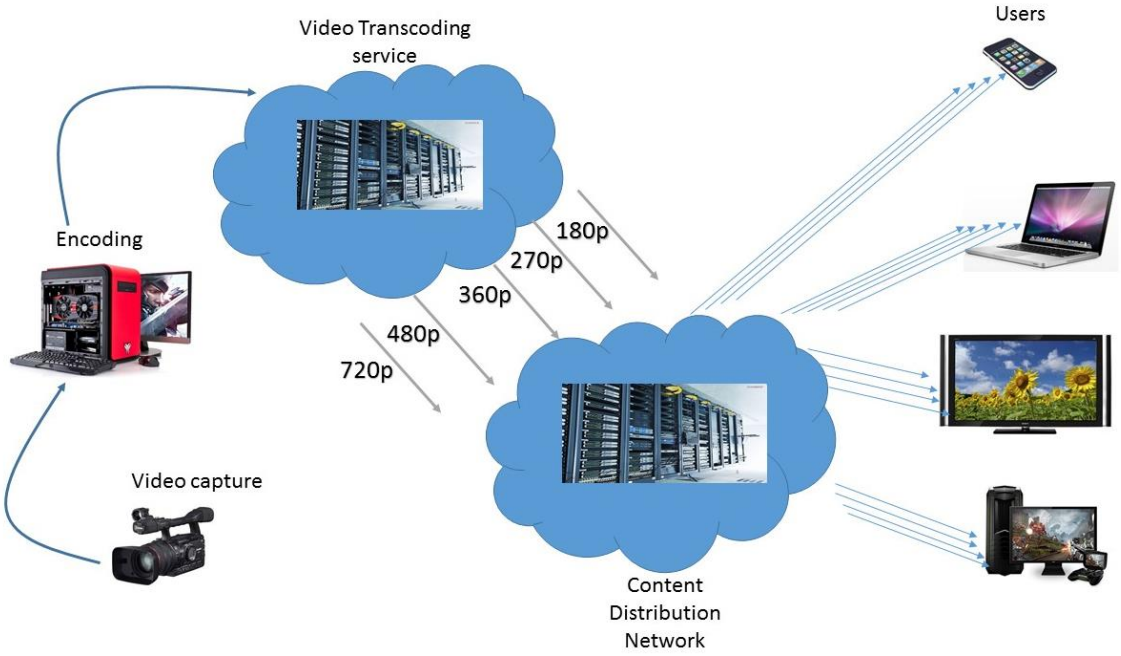


Figure 1. Video transcoding and delivery illustration.

The video streaming service is a typical example of cloud application with high variability in workload and stringent QoS/QoE requirements. Indeed, users are likely to request content off of work hours; further, high peaks of requests are usually expected for popular events like sport matches. Given the large variance in the processing load, the design of a video streaming application should account for scalability. Since every request from users can be processed independently, scaling out (i.e., horizontally) is a perfect technique in this scenario. There are two elements that need to be scaled in and out, namely video streaming and video transcoding components. Video streaming functionality should be always available and should respond with minimal latency, otherwise the user could be annoyed by the unresponsive service and could look for something else. Video transcoding must process large amounts of data, especially for real time video streaming; no stream interruption or quality degradation should occur during transcoding. Horizontal scalability is going to be supported and triggered based on monitoring of the application performance and the detection of need for additional/less resources.

Another important issue for video streaming is high availability. Users would be very annoyed if their video were interrupted (especially for live content), or their quality got worsen. Availability is

therefore another important matter for this use case; to this aim, enough spare resources should always be available to cope variations in the workload or failures of computing or networking equipment.

Both scalability and availability aspects for the considered application require for resources to be reserved and provisioned in advance, due to long times needed to set them up. Workload prediction is therefore necessary to foresee the future computation needs and to pre-provision resources, in order to avoid service degradation and poor Quality of Experience (QoE) for the user. However, running idle resources to meet future demand leads to large energy waste. Hence, orchestration and placement strategies and power management mechanisms that minimize the energy consumption of the underlying infrastructure are needed.

The basic requirements that have to be supported are the following:

- The media service provider defines video transcoding requirements, such as video resolution (e.g. 720p), bit-rate (e.g. 2Kb/s), coding scheme (e.g. H.264, MPEG-4, etc.), audio format (e.g. AAC, MP3 etc.), and the number of video outputs (e.g. 5 different output video files). This can be based on specific profiles, or set up in with a configuration script.
- The video transcoding application registers transcoding tasks, in order to process the current workload with minimal latency and cope with small workload variations in the short-term. Requirements concern the number of Virtual Machines, CPU budget for transcoding tasks, memory, network bandwidth, delay constraints, etc. Many media service providers may request simultaneously video transcoding services from the IaaS provider. That means video transcoding services should scale in and out based on service provider's requests.
- The video distribution services (e.g. CDNs), on the other hand, scale in and out, according to the current workload. Actually, the application can demand more resources than needed, to cope with sudden and unexpected workload variations and failures. Profiling and prediction techniques are used to compute the amount of spare capacity.
- The IaaS provider implements energy-efficiency mechanisms, in order to make energy usage in a more efficient manner. The IaaS provider exposes an interface to drive the SP's orchestration according to its energy-efficiency policies.
- The Smart Controller deploys the service taking into account the QoS/QoE requirements, the policies set by the Service Provider, and the information exposed by each Infrastructure Provider (e.g., cost, energy consumption, available resources).

Summing up, the IaaS provider manages its physical infrastructure with the aim of minimizing energy consumption, but without affecting the SLA with Service Providers; in this use case with the video streaming application provider. That means *it can handle the allocation of VMs and could exploit power saving capability to cut down energy for unused and spare resources, which must anyway remain present and available in few seconds.*

2.1.3 Relation to the ARCADIA framework

The ARCADIA framework enables to implement the use case in an effective way. Figure 2 shows the conceptual implementation of the use case in the ARCADIA framework; both the programming abstraction and the Smart Controller are involved.

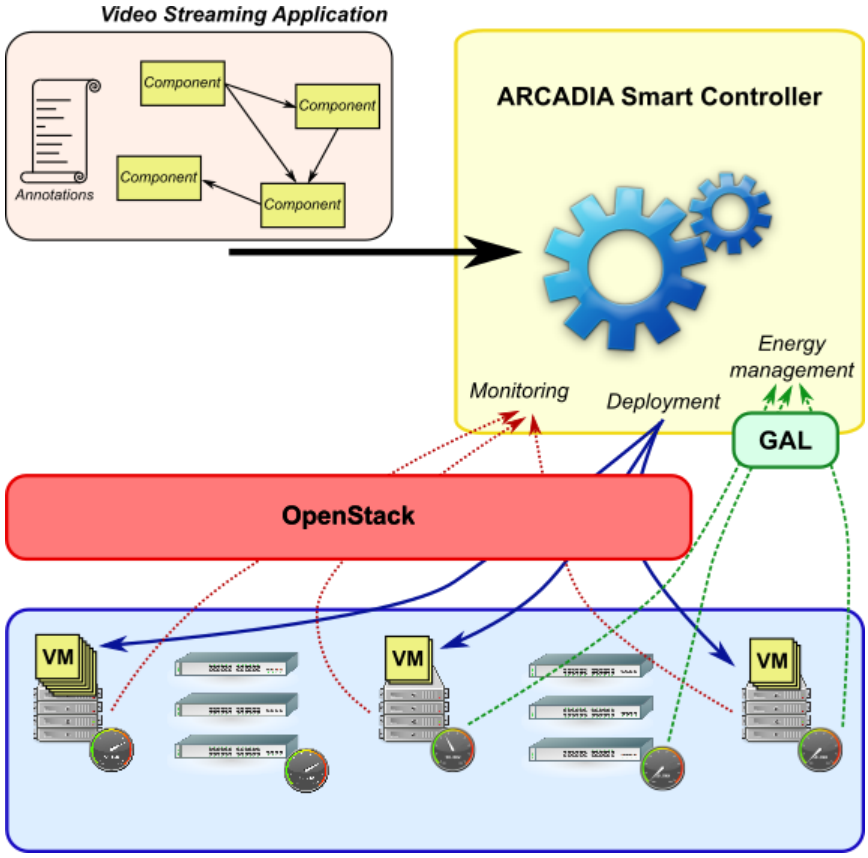


Figure 2. EE vs. QoS Use Case implementation in the ARCADIA framework.

The video streaming application will be developed according to the ARCADIA software development paradigm. The ARCADIA Context Model is going to be used for including annotations in the software, while the ARCADIA development/deployment toolkit is going to be used for preparing the distributed application's service chain. Information with regards to scalability and redundancy aspects of the internal software components (*micro-services* in the ARCADIA dialects), memory and processing power for each container, bandwidth requirements among the different components, and responsiveness of each component (i.e., the maximum delay to process new requests) is going to be included. Details about the service graph and the software components that build the video streaming application are given in Section 2.2.

The Smart Controller (SC) will set up the proper execution environment according to the deployment script, by provisioning software containers for the execution of each software component. Moreover, the Smart Controller will deploy all the probes needed to monitor the proper execution and performance of the application, so to scale it in and out when necessary. Based on continuous monitoring, historical logs and temporal usage patterns, the SC is going also to support predictions for future workload and, whenever necessary, pre-provision additional resources.

The Resource Manager interface (possibly based on Openstack) is going to be used for resource management purposes, augmented with information related to the trade-off between energy-efficiency and QoS of the underlying programmable infrastructure (based on the Green Abstraction Layer – GAL specification). The SC uses this information to choose the best infrastructure to run each software component of the application, and to make optimal placement of components according to energy-efficiency strategies. Details about the SP policies for this use case are given in Section 2.3.

2.2 Service Graph Description

2.2.1 ARCADIA Components

There are two different areas where energy efficiency policies can be applied in order to show case the applicability of the ARCADIA framework. The first one relates to video transcoding. This service needs to scale out in case of unpredicted peaks in order to meet QoE requirements based on SLA between the video service provider and the Service Transcoding provider. The second one relates the video client side. Clients will suffer both from delays and low quality when trying to retrieve a video file during peak times. Again the underlying infrastructure should be able to scale out and minimise these negative effects. Figure 3 provides a conceptual representation of the use case service graph.

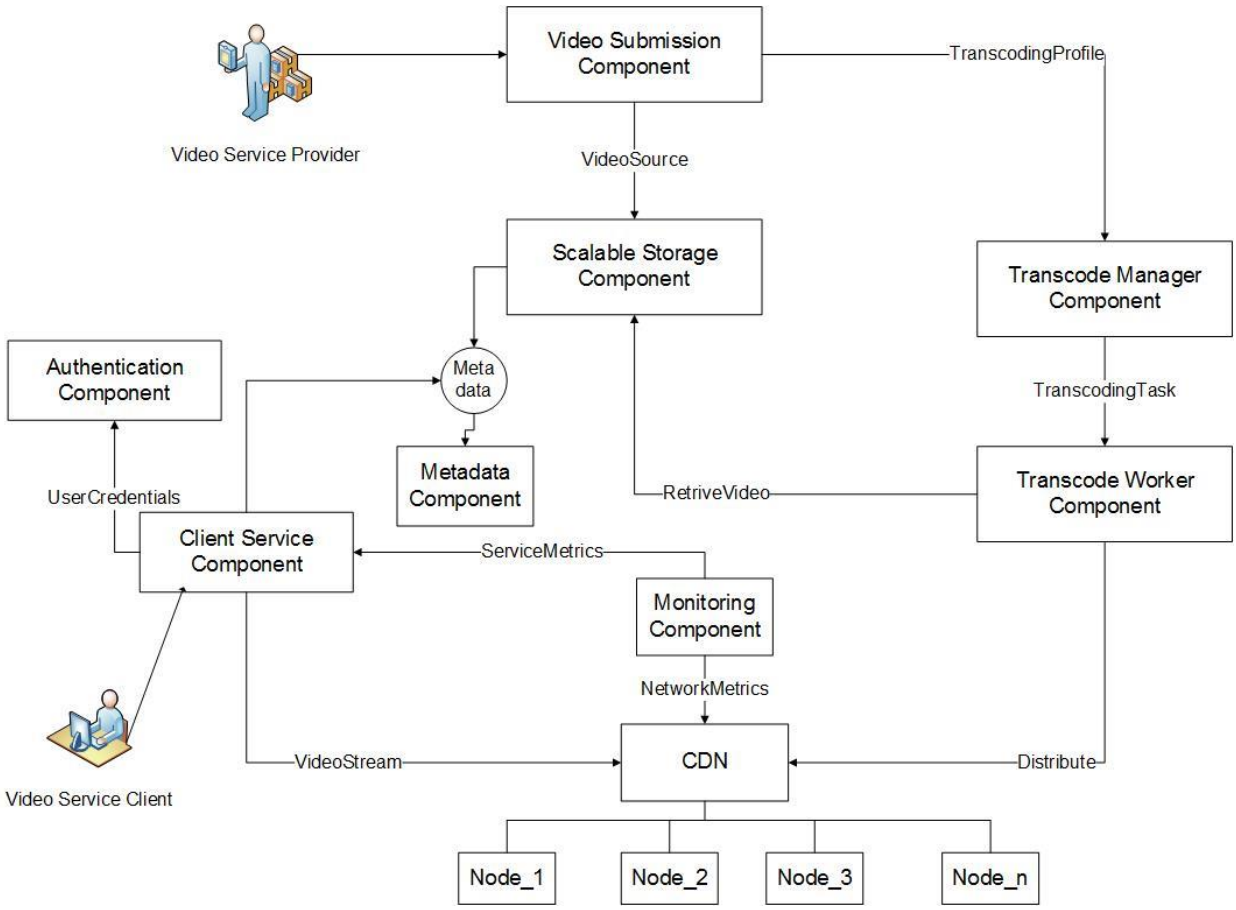


Figure 3. EE vs. QoS Use Case Service Graph.

Following we provide a short description of the various components in the Service Graph.

The *Video Submission Component* will serve as the front-end for video service providers to upload video files and request transcoding services. It will provide a catalogue with the available attributes for a transcoding service such as resolution, quality encoding level (bitrate), available video and audio coders. Requests can be set up dynamically via a configuration file or with pre-existing profiles.

The *Scalable Storage Component* will provide storage services. Storage should be scalable to support all incoming requests from video service providers.

The *Transcode Manager Component* will extract transcoding requirements based on requests set by video service providers, perform transcoding service provisions and assign transcoding tasks to a set of transcoding workers.

The *Transcode Worker Component* is responsible to transcode a video file into multiple copies and different resolution and quality. The output video files are passed to CDN Component for distribution.

The *Metadata Component* keeps the metadata of all video files for easier indexing and retrieval.

The *Client Service Component* is the front-end for video consumers (clients). It provides a web-based interface for video retrieval with all available formats and quality.

The *Monitoring Component* monitors the efficiency of networking functions based on QoS metrics. This component is also responsible for the prediction of future demands based on current load and historical data.

The *CDN Component* provides video streaming services to video service clients. This component scales out dynamically based on client requests.

Finally the *Authentication Component* provides identity and profile management for video consumers.

2.2.2 Binding Interfaces

In this section we provide a short description of the binding interfaces between the various components:

`VideoSource`

A set of parameters (e.g. video_name, size etc.) are passed to Scalable Storage Component from the Video Submission Component.

`TranscodingProfile`

This is the interface for passing video transcoding requests to Transcode Manager Component. A set of parameters (e.g. video_name, video_coder, audio_coder, quality, etc.) is defined for each transcoding request.

`RetrieveVideo`

This is the interface between the Transcode Worker Component and the Scalable Storage Component. It is used to retrieve the video file for further processing.

`TranscodingTask`

This interface is used by the Transcode Manager Component to pass transcoding tasks to a Transcoding worker. It sets the parameters for video transcoding along with QoE criteria.

`Metadata`

This interfaces provides the video metadata to Client Service Component.

`Distribute`

This interface is used by the Transcode Worker Component to forward transcoded video files to CDN Component for distribution to video clients.

`ServiceMetrics`

This interface is used to retrieve data related to current requests sent by Video Service Clients over a time interval.

`NetworkMetrics`

This interface is used by the Monitoring Component to retrieve network related status from the CDN Component.

`VideoStream`

This interface is used by the Client Service Component to initiate video streaming for the users. This may implement HTTP DASH or HTTP Live Streaming (HLS) or any other adaptive streaming solution.

`UserCredentials`

This is the interface for user authentication.

2.2.3 Metrics and Mitigation Plan per Component

In this section we provide a brief statement of actions to serve as an indicative mitigation plan when services are failed or disrupted. We will break the mitigation plan at the component level on only on those that a failure will interrupt the offered services (Figure 3).

Video Submission Component

Event: Unexpected video load

Mitigation: Scale out submission service.

Metric: Response time

Scalable Storage Component

Event: Unexpected video load

Mitigation: Scale out storage component.

Metric: Average throughput

Transcode Manager Component

Event: Unexpected transcoding tasks not able to handle with existing transcode workers.

Mitigation: Scale out the number of transcoder workers

Metric: Service queuing time

Transcode Worker Component

Event: Unavailable infrastructure resources for video transcoding.

Mitigation: Scale out infrastructure resources.

Metric: Transcoding time per video file.

Metadata Component

Event: Unexpected work load

Mitigation: Scale out storage component.

Metric: Average throughput

Monitoring Component

Event: Lack of infrastructure processing resources.

Mitigation: Scale out infrastructure processing resources.

Metric: Average execution time for providing results.

CDN Component

Event: Unexpected video requests

Mitigation: Scale out number of Nodes.

Metric: Response time per video request

Client Service Component

Event: Unexpected workload of video requests.

Mitigation: Scale out CDN Component.

Metric: Average Service time vs. number of clients.

Authentication Component

Event: Unexpected load of authentication requests

Mitigation: Scale out authentication services.

Metric: Response time per request

2.3 Service Provider Policies

In this use case, we consider the policy for energy efficiency. The main objective for this policy is to find the optimal usage of the infrastructure managed by the SP, given the typical non-linear behavior of ICT infrastructures[1].

There are two main aspects involved in this policy. The first is the choice to run software components in the own infrastructure or to rent resources from other external IaaS. For different workload condition, it could be less expensive to get virtual resources from an external provider than powering up local hardware. This aspect also concerns the price of electricity and the commercial agreements with other actors, and will not be showed in this use case. The second aspect considers the usage of consolidation techniques to reduce the amount of active servers, by keeping into account QoS constraints of running applications. We will focus on this policy in this use case.

According to the basic scenario outlined in Section 2.2, the Smart Controller will scale applications according to the predicted workload. However, according to metadata QoS information, there should be some "spare" resources, available to process sudden peaks of workload. The underpinning concept for the energy-efficiency policy is to classify each software component according to its current workload and QoS constraints; for instance, the Smart Controller could mark as *red* the components with heavy workload, *yellow* those which are seldom used, and *green* the replicas for backup[2].

The Smart Controller initially relies on Resource Manager filters to deploy virtual machines, setting suitable criteria based on QoS constraints and affinity; then, it periodically does optimal consolidation of VMs, without violating the QoS constraints and the redundancy asked by the application.

Indicatively, the Smart Controller will try to group heavy-loaded VMs on few servers working at full power, seldom used VMs on few servers working at reduced performance (low CPU frequency/voltage, reduced network link rate, etc.) by aggressive resource over-provisioning, and unused and idle replicas on sleeping servers (which can be resumed in the matter of a couple of seconds or less). The consolidation strategy should account for minimal response latency and negligible service degradation perceived by the final user.

The consolidation algorithm will take into account network equipment as well, so to shut down unused switching devices. Obviously, the consolidation strategy will also be constrained by performance degradation that may occur when moving heavy-loaded or strict-QoS components.

2.4 Key Performance Indicators and Acceptance Criteria

The performance evaluation for this use case will mainly focus on energy consumption and QoS aspects. The KPIs for this Use Case will be:

- **Energy consumption**. We will consider the overall infrastructure energy consumption, and its breakdown in the components due to different installations and components.
- **Latency**. The Smart Controller introduces some delay, in particular for carrying out the following operations: i) selecting the best instantiation on the underlying infrastructure; ii) moving software components around, in case a different deployment is necessary. This may lead to service disruption if the migration takes long time, for example due to the need to move

[1] There is a non-linear behavior between the performance and the power consumption of ICT equipment, due to the large amount of energy consumed even when the devices are idle.

[2] This classification is just an example. The actual mechanism will be defined during the project.

large bulks of data; ii) reacting to unexpected events (this happens, for example, when the workload increases or in case of failure).

- **Service disruption**. Service disruption occurs in this use case when the latency is very large or when packets are lost. Service disruption is likely to happen in case of failures and sudden peaks of workload, due to the time to resume "frozen" resources. We will measure the interruption of video rendering at the terminals as a performance indicator for this use case. We will also measure queuing time for video transcoding tasks.
- **Video Quality.** Peak Signal to Noise Ratio (PSNR) will be used to measure quality degradation of video files ([6], [7]). Related to video quality assessment methods we will use in this use case the *full reference* method as defined by the Video Quality Expert Group (VQEG) [8]. Under this method the same video files will be compared (with and without energy-efficient models) in order to assess any video degradation as a result of the implementation of these models.

The energy consumption of the following scenarios will be considered for comparison:

- legacy deployment, with no energy-efficiency features (*Business-as-Usual*, BAU);
- Smart Controller with Energy Efficiency policies;

We will evaluate the KPIs by comparing quality of service when no energy efficient mechanism is used (*business-as-usual*, BAU) and when the proposed algorithms are in action. The KPIs will be evaluated under different operating conditions, which take into account the different ways the Smart Controller actions could affect the system behaviour. In particular, we will evaluate what happens when the Smart Controller consolidates the workload, moves around the VMs, and changes the power status of devices. We will consider the following situations, which encompass all relevant operations undertaken by the Smart Controller:

a) *steady-state condition*: video flows that have been previously requested by users and are currently been streamed; transcoding tasks requested by video service providers are being processed without queueing delays.
b) *variable workload*: the number of requests for both video flows and transcoding tasks increases and decreases slowly;
c) *peak workload*: many requests for both video flows and transcoding tasks are generated in a short period of time;
d) *failure*: a hardware component (server, switch, link) is suddenly disconnected to emulate a failure.

The successful implementation of this use case must result in substantial energy saving with minimal service disruption. Our target acceptance criteria for the energy efficiency use case are set as follows.

KPI	**Values**	**Remark**
Energy saved	<5%	Unsatisfactory
	>5% and <20%	Good
	>20%	Very Good
Service disruption (unexpected event)	<1s	Excellent
	>1s and <2s	Very Good
	<5s	Good
Video disruption (peak of	<200ms	Excellent

workload)	>200ms and <500ms	Very Good
	<1s	Good
Wait time (before video starts)	<1s	Excellent
	>1s and < 3s	Very Good
	<5s	Good
PSNR (dB)	>37	Excellent
	31-37	Good
	25-30	Fair
	20-24	Bad

2.5 Programmable Infrastructure

The Programmable Infrastructure involved in this use case will be enough to emulate the whole lifecycle of a typical video streaming scenario involving transcoding processes and video distribution to clients, and to have room for energy optimization. The purpose is to demonstrate the correct behaviour of the Smart Controller while pursuing the trade-off between the energy consumption of the physical infrastructure and the Quality of Service/Experience perceived by the user of the envisioned service. In particular, the validation will consider

a) the effectiveness of the consolidation strategies, which will be developed by the project, to improve the energy efficiency of the system;
b) the ability of the optimization strategies to guarantee the required level of QoS/QoE;
c) the interoperability of the extended GAL interface with the cloud management software.

The detailed composition and topology will be defined at a later stage, depending on the architectural features that will be included in the implementation of the Smart Controller (WP3) and the availability of cloud infrastructures at partners' premises.

The preliminary composition of the programmable infrastructure is shown in Figure 4, together with all other components involved in the evaluation.

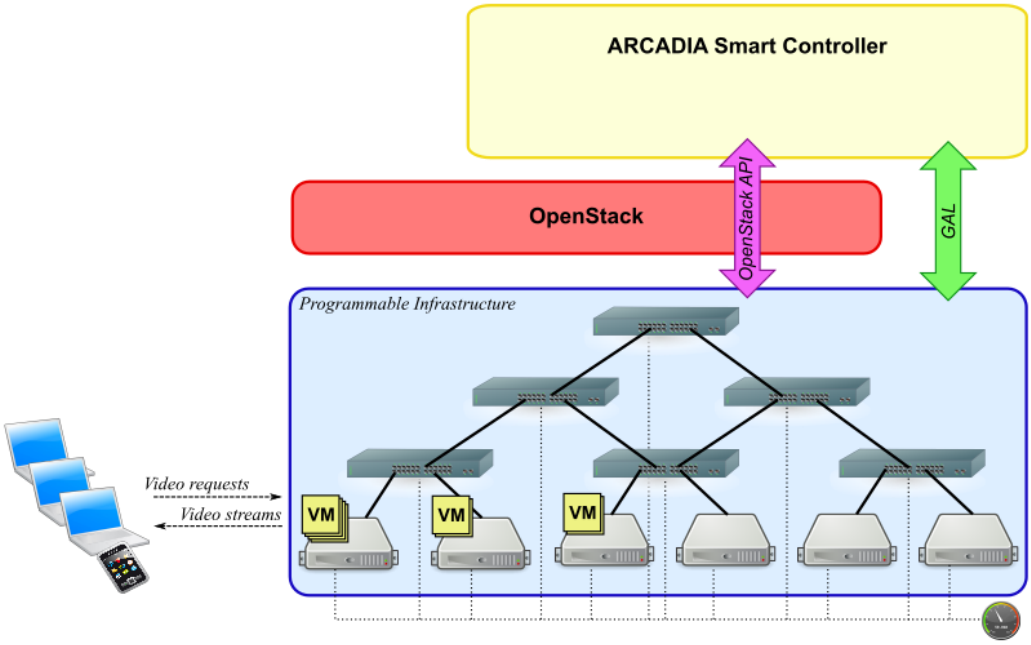


Figure 4. Set-up for validation.

The programmable infrastructure will be managed by OpenStack, and OpenFlow will be used to control the switching equipment (SDN). All compute and network nodes will be plugged into metered power outlets to measure the current power consumption. Failure of hardware components will be emulated to study the reaction of the system and the perceived QoE. The actual size of the infrastructure will be decided according to preliminary evaluation during the development phase.

3 High Performance Survivable Communications in Distributed IoT Deployments

The goal of this use case is to introduce ARCADIA capabilities into an existing heterogeneous communications infrastructure for public protection and disaster relief (PPDR) in order to support survivable and application-aware usage of infrastructure and network capabilities, hence improving availability and survivability of IoT services in emergency situations.

3.1 General description of the use case

WHAT IS 6inACTION: This use case is based on 6inACTION, an advanced distributed system designed to provide public safety agencies with a survivable, scalable and robust data communications and professional IoT-supported intervention management services during day-to-day operation and disaster relief missions. It comprises of:

- 6onCORE, a compact mobile IPv6-powered communications node, designed to equip first responders with survivable data communications during disaster relief missions. This node is deployed on site of the intervention and it provides transparent backhaul connectivity via any available professional, commercial or ad-hoc network (3G/4G, satellite, fixed, WiFi).
- 6onDASHBOARD, a cloud-based professional intervention management application designed for tactical and strategic levels as well as for cross-agency and international public safety operations. This application provides common operational picture as well as various decision

support services for intervention managers (location and tracking of dispatched units, sensor readings from the fields, FR reports, video feeds etc.).It can be accessed from anywhere using a web browser and data connectivity.

- 6onMOBILE, a smartphone/tablet application for field operatives, used to track the location of the operatives and to complete triage reporting, and on site COTS sensor deployments (connected through 6onFIRE, a mobile IoT gateway).

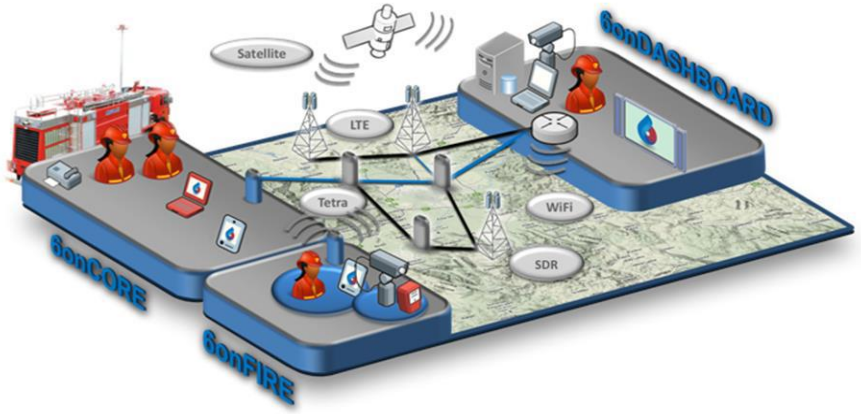


Figure 5: 6inACTION system and components.

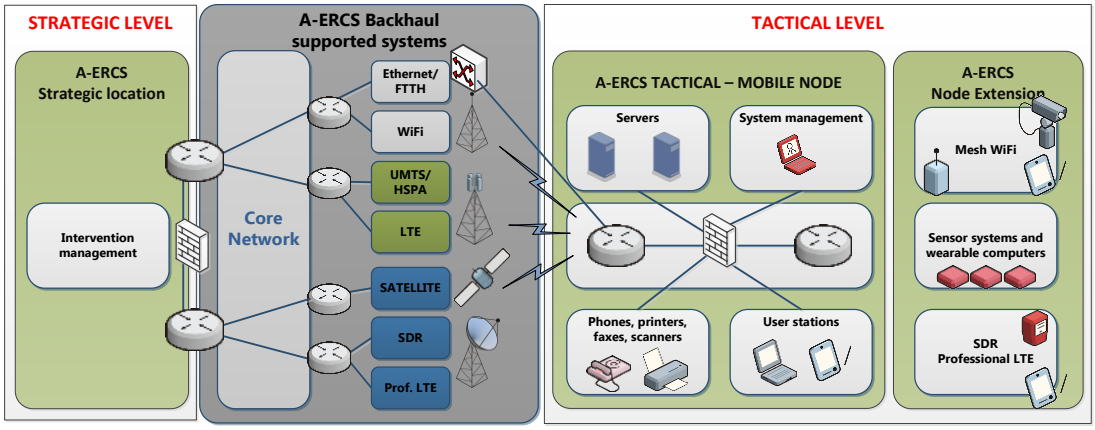


Figure 6: 6inACTION infrastructure architecture.

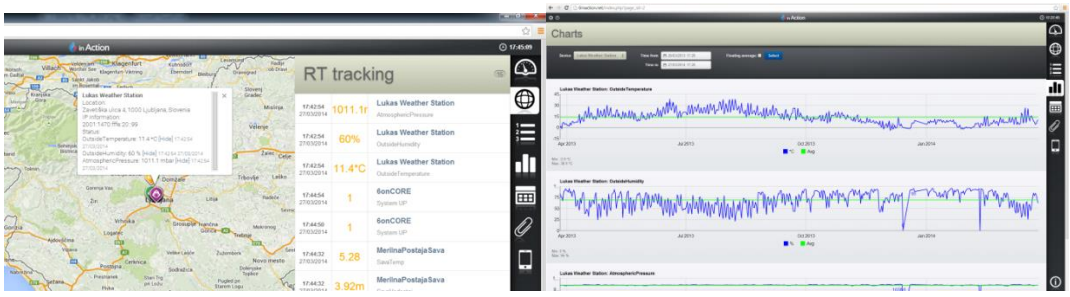


Figure 7: 6onDASHBOARD real time tactical dashboard for situation surveillance and intervention management.

HOW IT WORKS:

Please note that this section explains the operation of the entire 6inACTION solution on both tactical/field and strategic/central locations. For ARCADIA, only the cloud-hosted part of the solution is relevant, whereas the tactical sections are out of scope but might be used to illustrate and demonstrate the operations as necessary.

Deployment phase: 6inACTION system can be used in day-to-day operations as well as to support massive interventions in case of extreme events. Therefore, 6inACTION system is continuously deployed and in operation. One or more 6onDASHBOARD instances, comprising PHP business logic workers, MySQL database and BLOB storage, are deployed in private cloud infrastructure located on strategic command locations, for example a primary one in Ljubljana, Slovenia, and a redundant one in Rimini, Italy. Both instances are in operation and the system equally balances the load between the two.

Both instances are hosted on a cloud-based infrastructure with ARCADIA capabilities (6inACTION cloud), providing the initial deployment of both instances based on the pre-defined deployment plan. Later on, these capabilities ensure both horizontal and vertical scalability in case of increased load or unexpected failures on any of the locations.

Tactical and field personnel is equipped with 6onMOBILE apps on their smart phones and trained how to use it. 6onCORE nodes are pre-installed in the tactical command vehicles and are permanently turned on for continuous operation and automatically connected into the system when the vehicle is in stand-by mode or dispatched to the intervention. Sensor deployments are installed and connected either permanently or ad-hoc during an intervention.

Operations phase: During interventions, 6inACTION services are used to connect intervention sites with tactical and strategic command locations, and to support efficient management of operations. Basic services include:

- Backhaul data connectivity between intervention sites and strategic command centre, via any available backhaul network, transparently provided by 6onCORE;
- Common operational picture in 6onDASHBOARD, used in tactical and strategic command centres, which includes:
    - Tracking of operatives and equipment on the field (using 6onMOBILE app)
    - Graphical visualisations of field operations, instant alerts sent from the field and sensor readings
    - Triage reporting with 6onMOBILE app
    - Communication infrastructure monitoring (QoS/QoE parameter readings from transport networking equipment)
- Field services
    - 6onMOBILE app services:
        - Time-based and distance-based tracking
        - Triage reporting
        - Instant alerting
    - Field sensing using COTS sensor deployments ((e.g., GPS position, temperature and humidity, butane level, etc.) connected to 6onFIRE gateway.

While the operatives use the above 6inACTION services, 6onCORE automatically maintains connectivity between the intervention sites and the strategic command centre where the 6onDASHBOARD cloud is hosted. This is done transparently to the user, the system utilises any of the available backhaul connections (3G/4G, fixed, satellite, WiFi) to ensure survivable connectivity.

Similarly, survivability and availability of services has to be attended also in the cloud-hosted back-end part of the solution. Under extreme conditions, the capabilities of the 6onDASHBOARD system components can be volatile. Therefore the ARCADIA-enabled cloud infrastructure continuously monitors the conditions on the presently active 6onDASHBOARD system components and in case the performance response of the cloud-hosted 6inACTION components, i.e. MySQL database, BLOB storage

or PHP BL workers, deteriorates and is below the acceptable value (or even fails to operate entirely), the ARCADIA smart controller:

- automatically horizontally scales PHP BL workers (e.g. adding new PHP BL virtual servers for serving the user requests) or
- vertically scales the database and BLOB storage capabilities (e.g. adding additional CPU power, RAM or storage capabilities to existing databases or BLOB storage elements).

Disaster phase: During the above operations, unforeseen events can happen, for example a strong earthquake aftershock. This can cause the data centre with infrastructure hosting 6onDASHBOARD system in Slovenia to fail entirely. If this happens, 6onDASHBOARD cloud services on the secondary location in Italy has to be provisioned automatically and all user service requests have to be rerouted to this alternative location. The redundant instance is in this case horizontally and vertically scaled to take over the entire load.

Recovery phase: Following the disaster phase, the failed data centre in Slovenia is eventually rebuilt and the hosted 6inACTION solution has to be restored in a way that allows for restoring of its operation. In this case:

- the 6onDASBHOARD system components have to be re-deployed, scaled and synchronised with the secondary location in Italy, and
- the 6onDASHBOARD service requests must be again balanced between both locations.

3.2 Use case targets and preconditions

The 6inACTION system will not be altered in this use case. Instead, the use case will demonstrate how ARCADIA capabilities can be introduced into public safety environments to improve its capabilities, capacities and survivability. More concretely, it will demonstrate how ARCADIA capabilities can be used in emergency response solutions to ensure scalable and survivable distributed cloud-based hosting infrastructure with sufficient intelligence to respond to unforeseen circumstances (e.g., increased load, unexpected instance failures) and hence provide continuous availability and reliability of the hosted services.

As depicted in Figure 8 the national PPDR network operator will function as an ARCADIA infrastructure provider providing IaaS and network connectivity capabilities to the PPDR Service Providers. Therefore, PPDR Service Provider will enable the function of ARCADIA Service Provider with Smart Controller and its application capabilities for serving public safety users (e.g. Fire Fighters).

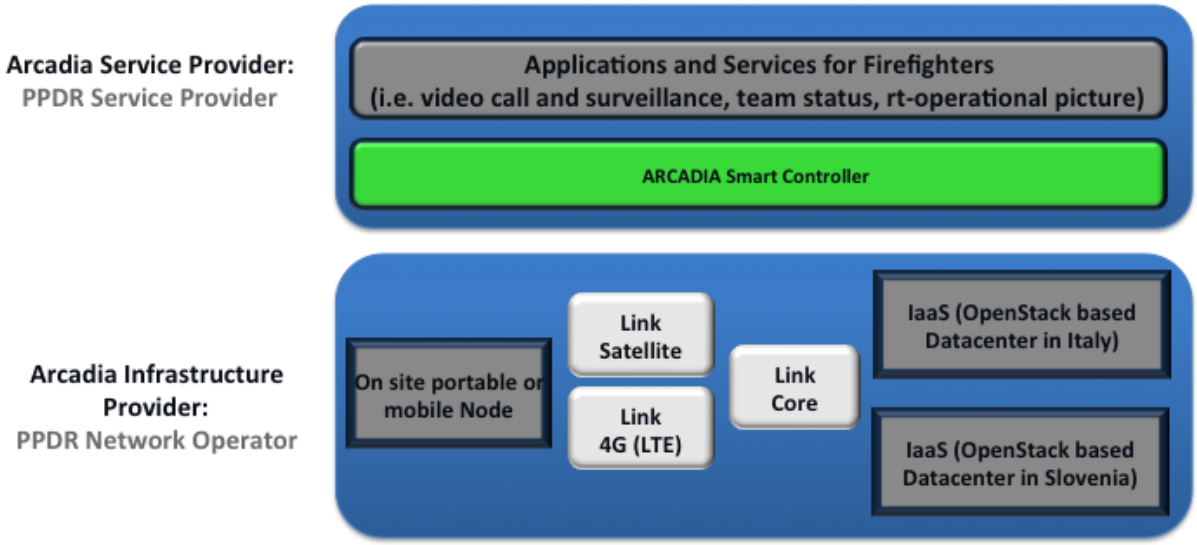
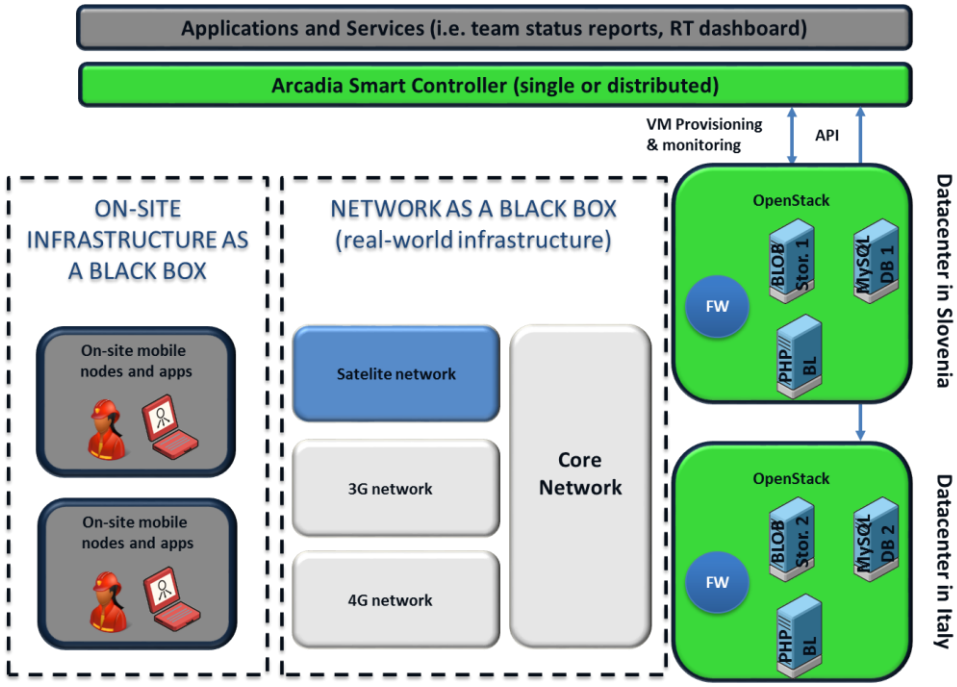


Figure 8: ARCADIA based public safety system architecture

Use Case description	
ID	PPDR UC
Name	High Performance Survivable Communications in Distributed IoT Deployments (short: PPDR Use Case)
Scenario	This scenario covers an IaaS hosted 6inACTION deployment and operation phase (in normal day-to-day operation), and provisioning of 6inACTION services during an extreme event –earthquake in Ljubljana, Slovenia. **TARGET 1:** The current deployment of the 6inACTION system is not done on a NFV-enabled IaaS hosting infrastructure. Instead, primary and redundant cloud-hosted instances are set up manually. The first target of this use case is to demonstrate how an OpenStack-based hosting infrastructure can be provisioned automatically and with optimal configuration of infrastructure resources according to the current application requirements. This will be done by: 1. translating the required application system capabilities and required performance into optimal infrastructure resources configuration by the ARCADIA smart controller, and 2. automatic configuration of the IaaS infrastructure and initiation of the 6onDASHBOARD application's components. **TARGET 2:** The current deployment of the 6inACTION system supports survivable backhaul connectivity through any (currently available on site) professional (TETRA, satellite), commercial (LTE/HSPA/3G, satellite, FTTx, WiMAX etc.) or even ad-hoc (e.g. WiFi) network, by means of 6onCORE nodes that interconnect intervention sites with the tactical and strategic command centres. Survivability of the communication is ensured with transparent fall-back intelligence, which automatically reconnects to the next best available network in case the initial chosen network fails to operate. However, the cloud-based infrastructure hosting 6onDASHBOARD currently does not support intelligent mechanisms for IaaS survivability. The current deployment does not support ARCADIA-based monitoring of the hosting infrastructure and automatic

	reconfiguration of the provisioned hosting resources to dynamically adapt to operational needs. Instead, in case of additional resource needs or failure of any of these, the provisioning is done manually to ensure continuous operation and availability of the hosted services. Therefore, in operations phase, the target of this use case is to ensure continuous monitoring of crucial 6onDASHBOARD KPIs and automatic vertical and horizontal scalability of the provisioned IaaS system components by the ARCADIA smart controller. This will be done by: • automatic horizontal scaling of the IaaS system components (e.g. adding new PHP BL virtual servers for serving the user requests) or • automatic vertical scaling of the IaaS system components (e.g. adding additional CPU power, RAM or storage capabilities to existing databases or BLOB storage elements). Horizontal scaling can be performed either within one location (data centre) or across two distributed data centres, in which case additional redundancy is ensured. **TARGET 3:** In extreme emergency situations (for example a massive earthquake) failure of the hosting infrastructure and the hosted applications is not uncommon. It is therefore necessary to ensure survivability of IaaS capabilities if such hosting is to be used for emergency response services. The current 6inACTION deployment does not support ARCADIA based monitoring of the hosting infrastructure and automatic reconfiguration of the provisioned hosting resources on the redundant location in case the primary hosting facility fails. Therefore, the third target of this use case is to demonstrate how NFV-enabled IaaS-based cloud hosting of 6onDASHBOARD services can be extended with additional mechanisms to continuously monitor availability and performance of the hosting infrastructure as well as of the hosted 6onDASHBOARD services, and in case of a detected downtime/failure of the primary hosting location, automatically provision the failed capacities on the redundant location and rerouting of service requests accordingly. This will be done by: • monitoring the availability/performance/QoS of the deployed IaaS infrastructure and the hosted 6onDASHBOARD instances, and • reconfiguring the provisioned IaaS resources (provision the failed capabilities on secondary location) in case of failure/outage of the primary location, and rerouting (e.g. using fast DNS capabilities) of all requests accordingly. **TARGET 4:** This target is part of the recovery phase and includes recovery of failed hosting IaaS locations and restoration of the hosted 6onDASHBOARD services on the previously failed primary location. The current 6inACTION deployment does not support automatic re-deployment, synchronization and scaling of a 6onDASHBOARD IaaS hosting locations. The target is therefore to demonstrate how an OpenStack-based hosting infrastructure can be re-deployed automatically and with optimal operational configuration of infrastructure resources according to the current application requirements (taking into account current load on secondary location and recovery policies). This will be done by: 1. automatic configuration of the IaaS infrastructure on primary location, its scaling according to recovery policies, and initiation of the 6onDASHBOARD application's components (including synchronization/replication of databases),

	2. rerouting of service requests between primary and secondary locations according to recovery policies, and 3. (optional) down-scaling of the secondary IaaS capabilities following reduced service load. Such mechanisms are necessary in IaaS for hosting of emergency services, and can greatly improve survivability and availability of emergency services under extreme conditions.
Goals	• TARGET 1 in deployment phase: to improve and automate the process of deploying and configuring the cloud hosting capabilities for 6onDASHBOARD application by introducing smart-controller-based automated OpenStack infrastructure provisioning (IaaS) on geographically distributed locations; • TARGET 2 in operations phase: based on monitoring the 6onDASHBOARD applications' KPIs and detection of performance degradation to introduce automatic scaling (horizontally and vertically) of the system components (e.g. PHP BL workers, MySQL database, BLOB storage); • TARGET 3 in disaster phase: to improve survivability and availability of 6onDASHBOARD services and the pertaining cloud infrastructure in case of extreme disaster (causing certain data centre locations to fail to operate) by introducing automated smart-controller-based infrastructure monitoring and dynamic reconfiguration (as part of IaaS provisioning; scaling of secondary location in case of failure of the primary location). • TARGET 4 in recovery phase: to support recovery of failed hosting IaaS locations and restoration of the hosted 6onDASHBOARD services by supporting smart-controller-based automated OpenStack infrastructure re-deployment and scaling.
Actors	• 6inACTION admin – administrator of the current 6inACTION system • 6inACTION DevOps User – prepares deployment script and 6onDASHBOARD images to be deployed in IaaS • 6inACTION users – requesting services • ARCADIA admin – manages Smart Controller, supports the deployment of 6inACTION in IaaS • ARCADIA SW developer – develops required functionalities on the smart controller to support the use case
Figure	**PRELIMINARY PILOT OUTLINE:** A preliminary outline of the pilot deployment is as follows: • ARCADIA IaaS hosted central locations in the function of the strategic command centre o 2x Data centre, one located in Slovenia and one Italy, both connected to the core transport network o 2x 6onDASHBOARD system hosted in data centres in Slovenia and Italy

	• Access and backhaul networks (treated as a BLACK BOX network which is not under the explicit pilot control)
	• On-site mobile nodes as enablers of the mobile and portable LAN networks used by first responders(treated as a BLACK BOX network which is not under the explicit pilot control)
	  ○ 6onCORE industrial mobile router with heterogeneous interfaces (LTE/HSPA, WiFi, Serial and GE) to support connectivity to heterogeneous backhaul transport networks
	  ○ Apps on end-user devices – smart phones/tablets and laptops (6onMOBILE, web browser to access 6onDASHBOARD, other)
	 
	
	Figure 9: Initial outline of the use case structure

Pre-conditions / Initialization	• Preparation of the deployment script and its deployment on the smart controller • Preparation of monitoring and recovery policies • Preparation of the 6onDASHBOARD image to be deployed by the smart controller • Deployment of other 6inACTION components (6onCORE, 6onMOBILE, sensor deployments etc.) – out of scope
Execution start	• Manually triggered deployment of the 6onDASHBOARD instance in OpenStack (via OpenStack Provisioning)
Post-conditions / Results	See section 5.6.

(VALIDATION CRITERIA)	

3.2.1 ARCADIA service modules

- Smart controller
- OpenStrack provisioning module

3.2.2 Use case specific service modules

DNS

Scalability: vertical
Metrics:
CPU load
RAM usage
HDD usage
bytes in/out per second
DNS response time latency
Mitigation action:
increase number of CPU cores
increase amount of RAM
increase amount of HDD space
Programming interfaces
Monitoring streams for acquiring element stats [for each metric]
CPU, RAM, HDD, DNS response time, interface traffic
addDnsEntry(type = A, ...)
getDnsEntry
updateDnsEntry()
removeDnsEntry()
addZone()
getZone()
updateZone()
removeZone()
general PIs
startService, stopService, changeNetworkSettings

FW

Scalability: vertical
Metrics:
CPU load
RAM usage
number of sessions per second
bits in/out per second (interface traffic)
Mitigation actions
increase number of CPU cores
increase amount of RAM
increase amount of HDD space - log space
PIs:
Monitoring streams for acquiring element stats [for each metric]
CPU, RAM, HDD, interface traffic
addRule
deleteRule

Database (MySQL DB)

Scalability: vertical
Metrics:
CPU load
RAM usage
HDD usage
Average query time
number of slow queries
transaction per second
average number of concurrent connections
bytes in/out per second
Mitigation actions
increase number of CPU cores
increase amount of RAM
increase amount of HDD space
PIs
Monitoring streams for acquiring element stats [for each metric]
CPU, RAM, HDD, average query time, ...
getConnection
addUser
removeUser
addPermission
removePermission
createDatabase
dropDatabase
setReplicationParams
startReplication
stopReplication
getBufferSize
setBufferSize
general PIs
startService, stopService, changeNetworkSettings

Upload server (BLOB storage)

Scalability: vertical
Metrics:
CPU load
RAM usage
HDD usage
Mitigation actions:
increaseVolumeSize
PIs:
Monitoring streams for acquiring element stats [for each metric]
CPU, RAM, HDD, DNS response time
uploadFile
deleteFile
getFile
moveFile
createDirectory
deleteDirectory
moveDirectory
general PIs
startService, stopService, changeNetworkSettings

Worker node (PHP codebase)

Scalability: horizontal and vertical
Metrics:
CPU load
RAM usage
HDD usage
Average HTTP response time
bytes in/out per second
number of accesses
number of web server worker threads
Mitigation actions
increase number of CPU cores
increase amount of RAM
increase amount of HDD space
deploy new worker node
PIs:
set worker node DB IP
set worker node BLOB store IP
general PIs
startService, stopService, changeNetworkSettings

3.3 Detailed description of the use case

Deployment phase:

Two 6onDASHBOARD instances are deployed in IaaS cloud infrastructure, one located in the strategic command centre in Ljubljana, Slovenia, and one in Rimini, Italy. Deployment of cloud infrastructure and initialization of the 6onDASHBOARD application is completed by the ARCADIA smart controller and is done dynamically based on 6onDASHBOARD application requirements specified in the deployment script.

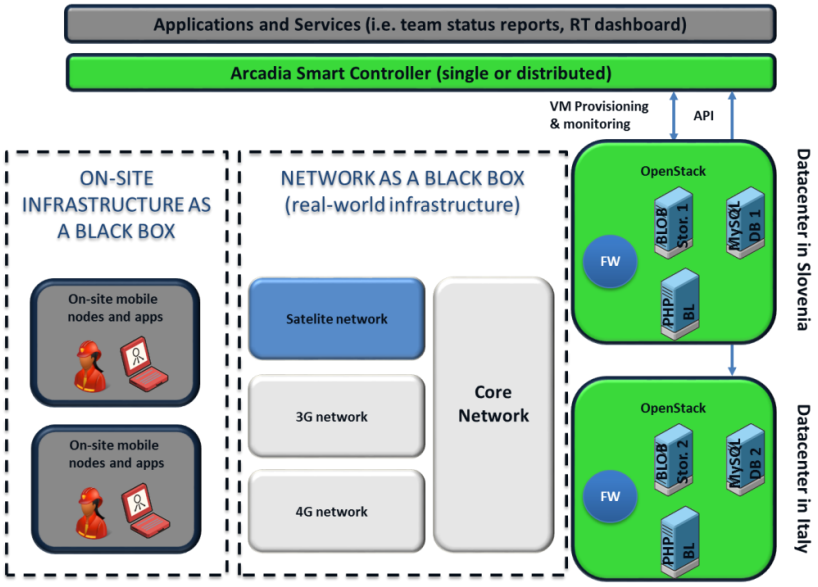
Operational specifics:

- Initial deployment is triggered via the OpenStack Provisioning
- Smart controller performs context matching
  - Application context for 6onDASHBOARD cloud instances from deployment script: CPU requirements, storage requirements, # of processors, RAM requirements, connectivity type IPv4/IPv6
- Smart controller retrieves infrastructure context (information about available infrastructure resources) and checks it against the application context
  - if there is a match, the smart controller sets up the execution environment and initiates the 6onDASHBOARD application
- Smart controller maps in the DNS server the assigned IPv4/IPv6 address and application domain names

Service provider policies:

- When setting up secondary DB, replicate primary DB.

Once the infrastructure is deployed and the 6onDASHBOARD application is running on both locations, the service requests are served in the primary location in Ljubljana and in the secondary location in Italy, with load balancing set to 50:50.
Other parts of the 6inACTION system are deployed manually, which is out of scope for ARCADIA project.



Operations phase:

During normal operation, the 6onDASHBOARD application is running on both locations, and the service requests are equally distributed between the primary location in Ljubljana and the secondary location in Rimini. The MySQL databases and the BLOB storage on both locations continuously sync. The ARCADIA framework continuously monitors availability and performance of the hosting infrastructure as well as of the hosted 6onDASHBOARD services on both locations.
In a certain moment, field activities in Ljubljana increase, which results in an increased load on the hosted 6onDASHBOARD in the data centres. As a consequence the KPI thresholds are met (for example CPU usage on PHP BL reaches 80%, 90% of BLOB storage is used), which causes the smart controller to:

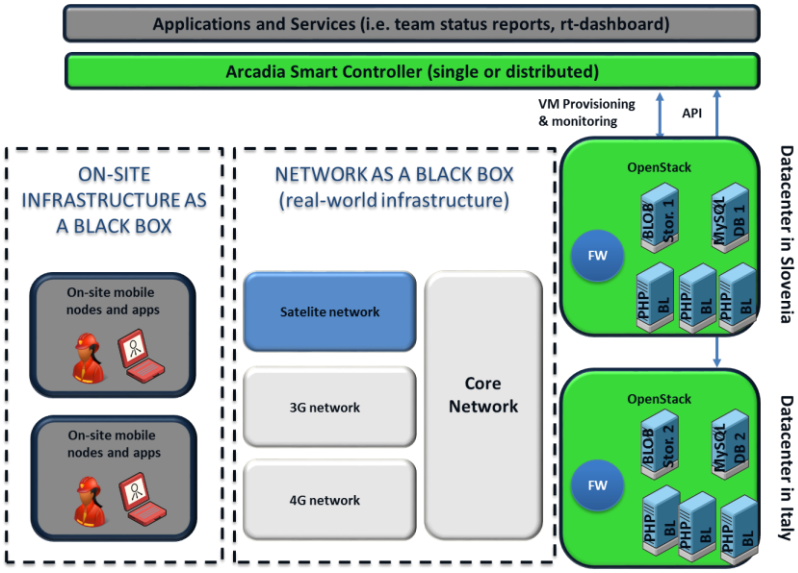
- automatically trigger vertical scaling of the IaaS system components as necessary (e.g. adding storage capabilities to existing databases or BLOB storage elements),
- automatically trigger horizontal scaling (creating new PHP BL instances),
- update the DNS entries to ensure balancing of the requests between all PHP BL instances.

Operational details:

- Smart controller deploys monitoring hooks to continuously monitor performance of the hosted applications and the provided OpenStack IaaS resources: monitoring performance KPIs
- When application performance degradation is detected, smart controller issues commands to
  - add new "PHP BL" nodes–Horizontal scaling
  - add CPU, RAM, Storage –Vertical scaling

Service provider policies (example):

- if worker node CPU > 60%, deploy new worker node and add it to DNS
- if average worker node CPU < 50%, shut down 1 worker node
- if DNS CPU usage > 80%, add 1 core
- if DNS CPU usage < 40%, remove 1 core
- if database number of slow queries increases by 50%, add 1 core and 2GB of RAM
- if database number of slow queries decreases by 75%, remove 1 core and 2GB of RAM
- if BLOB store disk usage > 80%, add additional 100GB of disk storage
- if worker node is unresponsive to HTTP requests, restart worker node



Disaster phase:

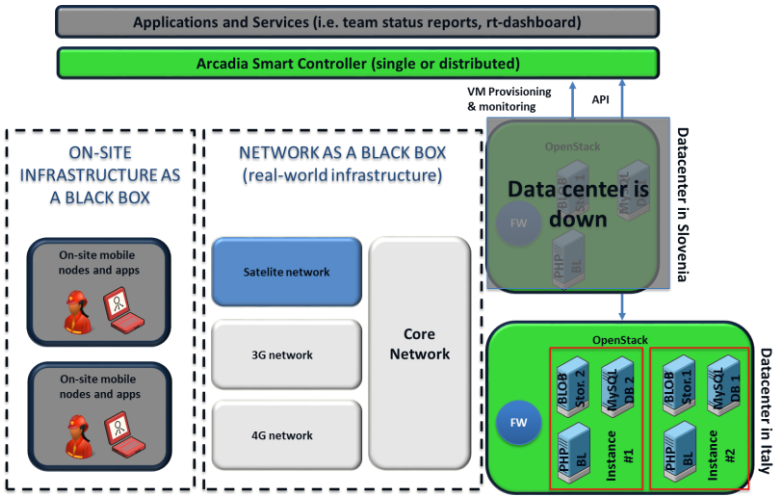
When an earthquake strikes, this results in complete failure of the IaaS infrastructure in the strategic command centre in Ljubljana. All 6onDASHBOARD nodes in that location go down. When this happens, monitoring hooks fail to report the metrics to the smart controller, which learns about the failure. As a consequence, the smart controller automatically initiates rerouting of all user traffic to the redundant instance in Rimini and (if needed) automatic scales the IaaS resources in redundant location to absorb the load.

Operational details:

- Smart controller uses monitoring hooks to continuously monitor availability of the hosted 6onDASHBOARD application and the provided OpenStack IaaS resources: keep-alive messages, interface up, uplink/downlink bandwidth etc.
- When failure of primary IaaS infrastructure is detected, smart controller issues commands to double IaaS resources on the redundant location
- Smart controller updates DNS entries to reroute all service requests to the redundant IaaS location

Service provider policies:

- if worker node is inaccessible on network level (ping IP), remove from DNS



Recovery phase:

Later in the phase of recovery in Ljubljana, the failed data centre is rebuilt along with all ARCADIA IaaS capabilities and the implementation of the 6inACTION. As part of the restoration of the 6inACTION system, the ARCADIA smart controller initiates re-deployment of all necessary 6onDASHBOARD server nodes. This includes dynamic deployment of the cloud infrastructure and initialization of the 6onDASHBOARD application. The difference compared to the initial deployment phase is that the re-deployment is based on 6onDASHBOARD application requirements specified in the deployment script as well as on recovery policies, which take into account load balancing between primary and secondary location and the current service request load.

Operational details:

- Re-deployment is triggered via the OpenStack Provisioning
- Smart controller performs context matching
    - Application context for 6onDASHBOARD cloud instance from deployment script: CPU requirements, storage requirements, # of processors, RAM requirements, connectivity type IPv4/IPv6
- Smart controller retrieves infrastructure context (information about available infrastructure resources) and recovery policies and checks them against the application context
    - if there is a match, the smart controller sets up the execution environment and initiates the 6onDASHBOARD application
- Smart controller maps in the DNS server the assigned IPv4/IPv6 address and application domain names

Service provider policies:

- When setting up secondary DB, replicate primary DB.

3.4 Use case service graph

Deployment phase:

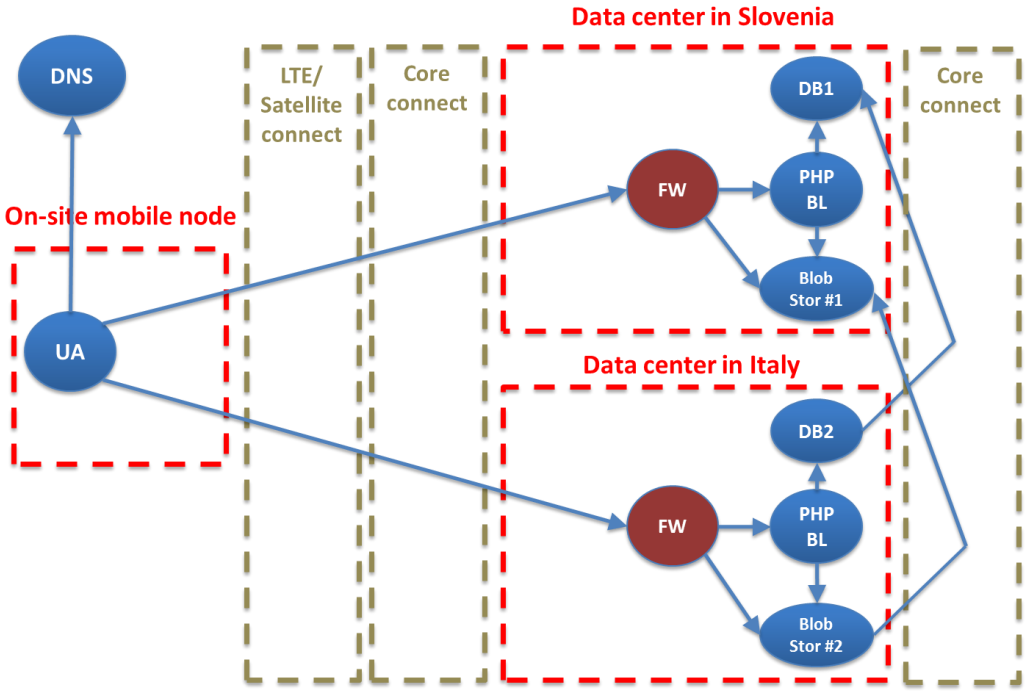


Figure 10: Service chaining graph for the deployment phase – source of the graph represents DB1, therfore it must be deployed first.

Operations phase:

This service graph depicts horizontal and vertical scaling.

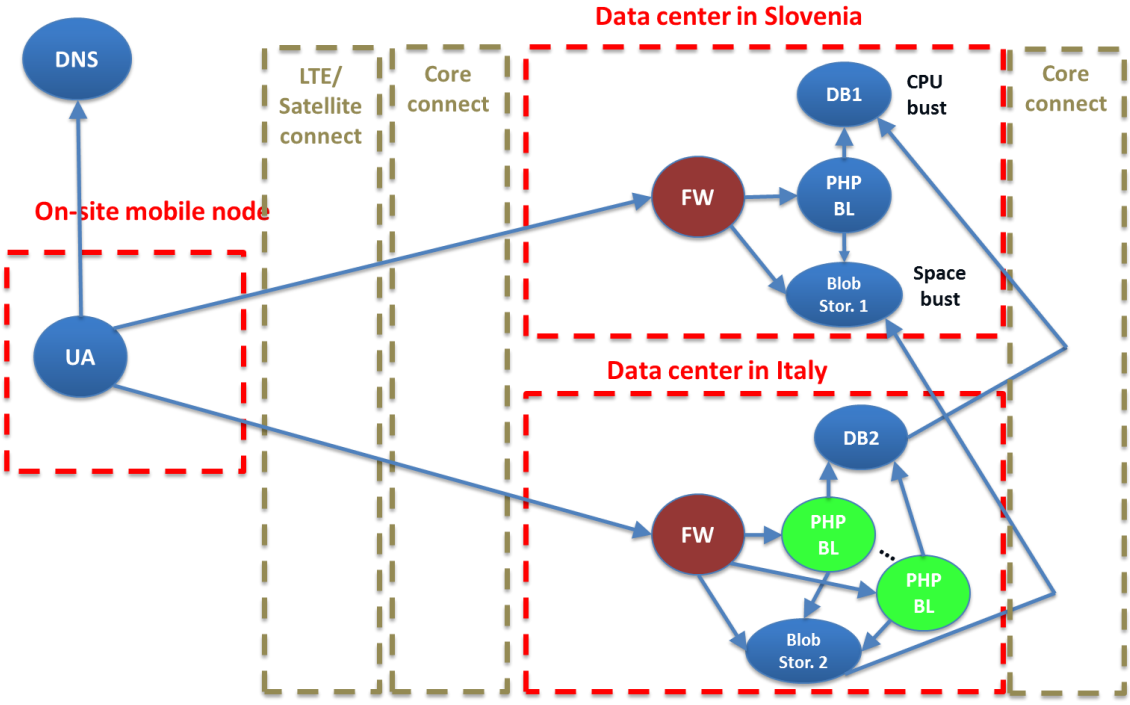


Figure 11: Service chaining graph for the operation phase – vertical and horizontal scaling of the application's components.

Disaster phase:

During disaster phase, primary IaaS location fails. User requests are automatically rerouted to redundant location, which is horizontally or vertically scaled as necessary to sustain the load.

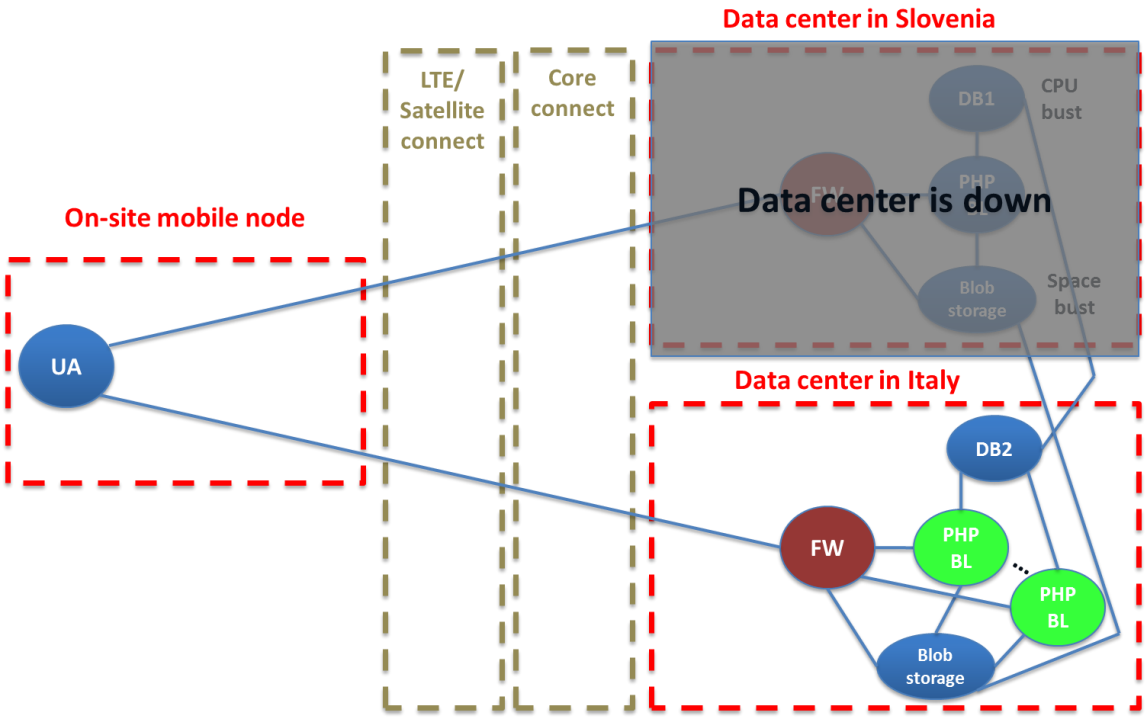


Figure 12: Service chaining graph for the disaster phase – rerouting of all requests to secondary location and scaling of the application's components at the secondary location.

Recovery phase:

The service chaining graph is identical to the deployment phase, with the redundant data center taking the role of primary location.

3.5 Validation

Deployment phase:

CRITERIA	INPUTS/CONDITIONS	OBSERVED OUTPUTS	CRITICALITY
SUCCESS: configured IaaS resources match requirements of the 6onDASHBOARD application defined in the deployment script	• Deployment script comprising application requirements	• Configured IaaS capacities	High
SUCCESS: two 6onDASHBOARD instances deployed, instance in Ljubljana running, instance in Rimini in hot standby mode	• Deployment script comprising application requirements	• Configured and successfully activated IaaS capacities	High
FAIL: not all 6onMOBILE users can use services provided by 6onDASHBOARD	• 6onMOBILE applications deployed (out of scope)	• 6onDASHBOARD service up • DNS entries updated	High

Operations phase:

CRITERIA	INPUTS/CONDITIONS	OBSERVED OUTPUTS	CRITICALITY
SUCCESS: smart controller detects KPI thresholds	• KPI policy (thresholds) • Deployed ARCADIA monitoring hooks	• Monitored KPIs • Smart controller actions based on KPI rules	High
SUCCESS: smart controller triggers vertical scaling in data centre in Ljubljana	• KPI policy (thresholds) • Deployed ARCADIA monitoring hooks	• Monitored KPIs • Configured IaaS capacities	High
SUCCESS: smart controller triggers horizontal scaling in data centre in Ljubljana	• KPI policy (thresholds) • Deployed ARCADIA monitoring hooks	• Monitored KPIs • Reconfigured IaaS capacities	High
SUCCESS: smart controller triggers horizontal scaling in data centre in Ljubljana and in Rimini	• KPI policy (thresholds) • Deployed ARCADIA monitoring hooks	• Monitored KPIs • Configured IaaS capacities	High

Disaster phase:

CRITERIA	INPUTS/CONDITIONS	OBSERVED OUTPUTS	CRITICALITY
FAIL: smart controller does not detect failure of primary IaaS infrastructure	• KPI policy (thresholds) • Deployed ARCADIA monitoring hooks	• Monitored KPIs • Smart controller actions based on KPI rules	High
SUCCESS: IaaS resources on redundant location activated and scaled	• Deployment script comprising application requirements • KPI policy (thresholds) • Deployed ARCADIA monitoring hooks	• Monitored KPIs • Reconfigured IaaS capacities	High
SUCCESS: smart controller triggers horizontal scaling in data centre in Ljubljana	• KPI policy (thresholds) • Deployed ARCADIA monitoring hooks	• Monitored KPIs • Configured IaaS capacities	High
SUCCESS: DNS entries updated by smart controller to reroute service requests to redundant location	• DNS entry updates logic on the smart controller (or app module)	• DNS entries	High
SUCCESS: one instance of 6onDASHBOARD serves all mobile nodes running 6onMOBILE apps	• 6onMOBILE applications used (out of scope)	• 6onDASHBOARD service up • 6onDASHBOARD management console (active users)	High

Recovery phase:

CRITERIA	INPUTS/CONDITIONS	OBSERVED OUTPUTS	CRITICALITY
SUCCESS: re-deployment successful, Ljubljana instance in operation	• Deployment script comprising application requirements • Recovery policy	• Configured IaaS capacities	High
SUCCESS: DNS entries updated by smart controller to reroute service requests to primary location	• DNS entry updates logic on the smart controller (or app	• DNS entries	High

	module)		
FAIL: IaaS resources on primary location not scaled to meet current load	• Deployment script comprising application requirements • Deployed monitoring hooks • Recovery policy	• Configured IaaS capacities	High

3.6 Performance evaluation

KPI	Expected value	Comments
6inACTION service downtime during reconfiguration of the IaaS resources	Under 1 min More than 5 mins	Excellent Unsatisfactory
Balanced load on both data centres	50:50	Excellent
Reroute service requests to redundant location after failure of the primary location	Under 5 min More than 10 min	Excellent Unsatisfactory
DNS record update	Under 1 min More than 5 min	Very good Unsatisfactory

4 Security and Privacy Support in the FIWARE Platform

4.1 Introduction

The recent advances in the cloud-computing technology and in the global deployment cellular networks have become key enablers for a broad range of smart applications and software industry. By 2020, the old paradigm of software developed for a single hardware platform or operating system will be obsolete and most software will run on distributed, heterogeneous and highly parallelized systems. Design software solutions should be **evolvable**, **adaptable** and should guarantee non-functional properties such as **security** and **privacy** [9]. As applications are highly distributed, sensitive data are moving to the cloud, accessible through given APIs, it becomes clear that proper protection solutions should be in place.

4.2 Use case description

This use case relates to **Secure Distributed Healthcare Services** including a **Remote Patient Monitoring (RPM) scenario** and an **Encrypted Communication Service scenario**. The main purpose is to introduce and validate a security and privacy by design approach that helps software developers implement more secure and scalable applications by leveraging emerging software technologies such as **Network Functions Virtualization**. It is worth to mention that, existing microservices from **FIWARE** [10] and its **IoT reference architecture** will be investigated and selected when implementing the use case applications.

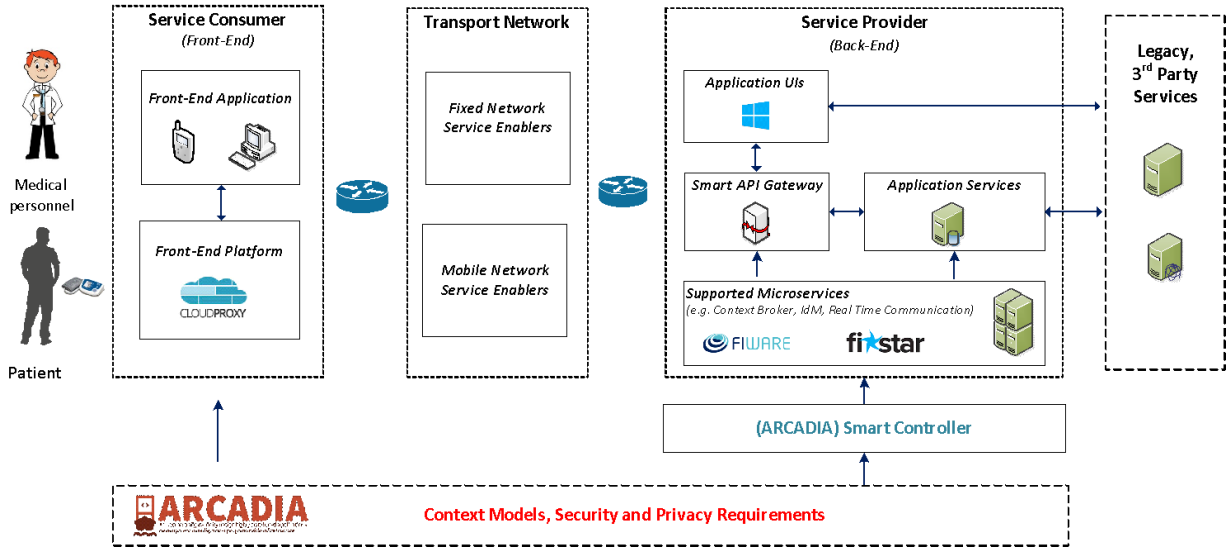


Figure 13: ARCADIA Remote Patient Monitoring application (RPM) architecture

4.2.1 Background

Microservice Pattern Design:

The idea to split application into set of smaller and interconnected services (microservice) is currently getting many interests from application developers and service providers (e.g. Amazon, Netflix, eBay). Such approach brings several advantages as individual services are much faster to develop, and easier to understand and maintain, and each service can be developed, deployed and scaled independently.

Utilisation of FIWARE Enablers as microservices:

The concept of "Enabler" derives from the Future Public Private Partnership program (FI-PPP) by European Commission to accelerate the development and adoption of Future Internet technologies [10]. An enabler is a technological component that provides set of APIs and interoperable interfaces to support a concrete set of functions. There are many research projects and partners involved in the FI-PPP. FIWARE is the core project that specified and implemented a lot of general-purpose enablers (GE) that common to almost usage areas. Such GEs are categorized into several main technical chapters: Cloud Hosting, Data & Context Management, Internet of Thing (IoT), Application, Advanced Middleware and Interface to Network and Device and Security. FIWARE introduces an open and standard **IoT reference architecture** (Figure 14) that offers several benefits for use case application implementation such as simple sensor data integration, device-independent APIs for quick app development & lock-in prevention, modular, scalable, high available. The FIWARE IoT Stack consists of different FIWARE GEs. The platform adopted several security services such as identity management, OAUTH and XACML-based access control. **OAUTH** (Open Authorization framework) is the evolving standard to secure API access. OAuth allows users (resource owner) to grant third-party applications (client) accessing user data (resource server) without sharing credential (e.g. password). The client can be a native mobile application, a desktop application, a server-side or browser-based web application. **XACML** (eXtensible Access Control Markup Language) is developed by OASIS to standardize the authorization decisions in enterprise applications. XACML defines a XML-based policy language, a request / response scheme and access control architecture. The decision based architecture consists of different components such as PEP (Policy Enforcement Point), PDP (Policy Decision Point) or PAP (Policy Administration Point).

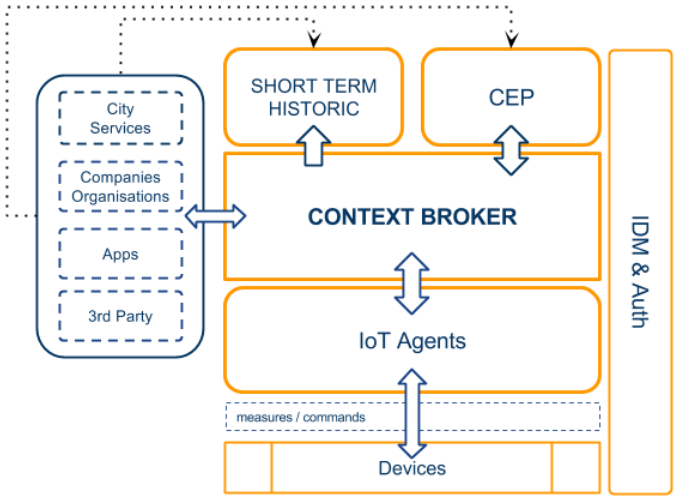


Figure 14: FIWARE IoT Stack

FI-PPP has also supported some use case projects in several important domains such as FI-STAR (healthcare), SAFECITY (smart city), FINESCE (smart energy), FINEST (transport) and FITMAN (manufacturing) [11]. In these projects, besides leveraging the usage of existing FIWARE GEs, they have also specified new specific enablers (SE). Many GEs and SEs have been introduced and available as open source and the number will continuously increase.

4.2.2 Business scenarios

The Remote Patient Monitoring scenario is a typical example of how emerging technologies (IoT, cloud and mobile computing) can support healthcare domain to delivery of high-quality, more accurate diagnosis and treatment. In this application, patients' vital health parameters are securely collected, stored on the cloud and given access through a set of APIs. As healthcare applications have very strict requirements to protect patient data, proper security and privacy solutions must be deployed. Moreover, the need of establishing secure (real-time) communication among involved people (e.g. patient, doctor) is also challenging since it incorporates a lot of technical difficulties especially when two parties want to discuss on a critical issue without the risk of information disclosure. The type of ad-hoc communication (voice, data or both) and the nature of the collaborating people introduce completely different requirements that have to be satisfied. Several challenges are required to be taken into consideration such as the VOIP communication should be established without any static pre-exchange of keys and without huge administrative overhead or the quality of experience should be guaranteed in the frame of a qualitative communication.

Such a scenario requires the usage of multi-layer protection services that undertake (beyond encryption microservices that were mentioned previously) authentication, authorization, audit-logging service, asynchronous messaging etc. One reasonable question is which the cornerstone of the re-used microservices since developing all of them from scratch is a huge overhead. The answer to this question is the usage of the FIWARE 'Enablers'.

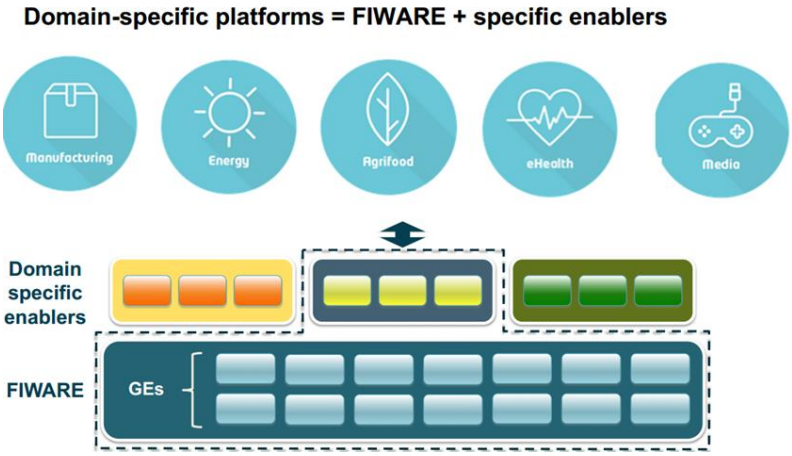


Figure 16: FIWARE Enablers

At this point it should be clarified that the concept of the Enabler and the concept of the ARCADIA Component do not coincide since an ARCADIA component since an orchestratable entity that can adopt to its operational environment. The FI-* enablers have not been created in order to satisfy this principle. This discussion is concretized in the next section.

Highly Distributed Encrypted Communication:

Today the amount of information that is circulated in the healthcare domain through mobile communication is vast. However, the comfort of instant communication compromised the sense of security. This is because people are carried away by the fact that they can communicate at any time and any place ignoring that this communication is weak in terms of security and is highly vulnerable to interception. In fact, today, the ability to intercept and overhear communications (voice and messaging) can be achieved with extremely low-cost equipment by any malicious person. Taking under consideration that overhearing of dialogs is a direct thread for any entrepreneur or governmental personnel; a solution has to be found that combines the comforts of mobile communication (i.e. instant communication at any place) and usable decent security.

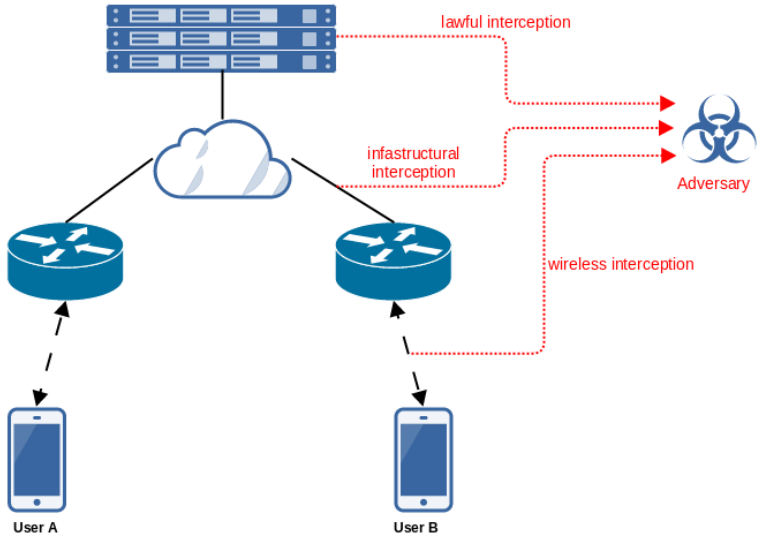


Figure 135: Threats in Mobile Communications

As depicted in figure 15 the threats that relate to the information disclosure during a mobile communication refer either to the communication medium per se (it is depicted as wireless interception) or to potential infrastructural interception that may occur illegally (by a malicious IaaS

administrator) or lawfully. To this end, there are several approaches that can be used in order to provide some run-time guarantees regarding specific types of threats. Most of these approaches incorporate the adoption of several protocols such as the following:

- **Symmetric Data encryption algorithm** such as Advance Encryption Standard 256 bit (AES-256) : The algorithm described by AES is a symmetric-key algorithm, meaning the same key is used for both encrypting and decrypting the data. AES is a winner data encryption-algorithm among many Cryptography Research and Evaluation Committees including CRYPTREC, NESSIE and NSA.
- **Transport Level encryption protocol such as** Secure Sockets Layer (SSL) or TLS: SSL is a cryptographic protocols that provide communication security over the Internet. SSL builds on top of symmetric encryption for confidentiality.
- **Key-agreement protocol** such as Zimmermann Real-time Transport Protocol (ZRTP): ZRTP is a cryptographic key-agreement protocol to negotiate the keys for encryption between two end points in a Voice over Internet Protocol (VoIP) phone telephony call based on the Real-time Transport Protocol.
- **Voice encryption protocol:** Secure Real-time Transport Protocol (SRTP): The Secure Real-time Transport Protocol (or SRTP) defines a profile of RTP (Real-time Transport Protocol), intended to provide encryption, message authentication and integrity, and replay protection to the RTP data in both unicast and multicast applications.

The usage of these protocols is performed by distributed applications that operate on the end-user's terminal device and collaborate with applications that are hosted in diverse IaaS providers. It could be argued that the applications that are hosted in the back-end tend to become stateless reusable Microservices [12].

4.2.3 Relation to ARCADIA framework

As already stated the FIWARE/FI-STAR 'Enablers' will be the cornerstone for the realization of the use case. However, 'Enablers' have to be leveraged in ARCADIA Components. Formally speaking ARCADIA Components entail a strict metamodel (see Figure 17) according to which a component can be used for high-level chaining in order to realize a complex scenario. According to this metamodel specific information regarding the components' configuration layer, governance layer, exposed and required interfaces should be strictly defined.

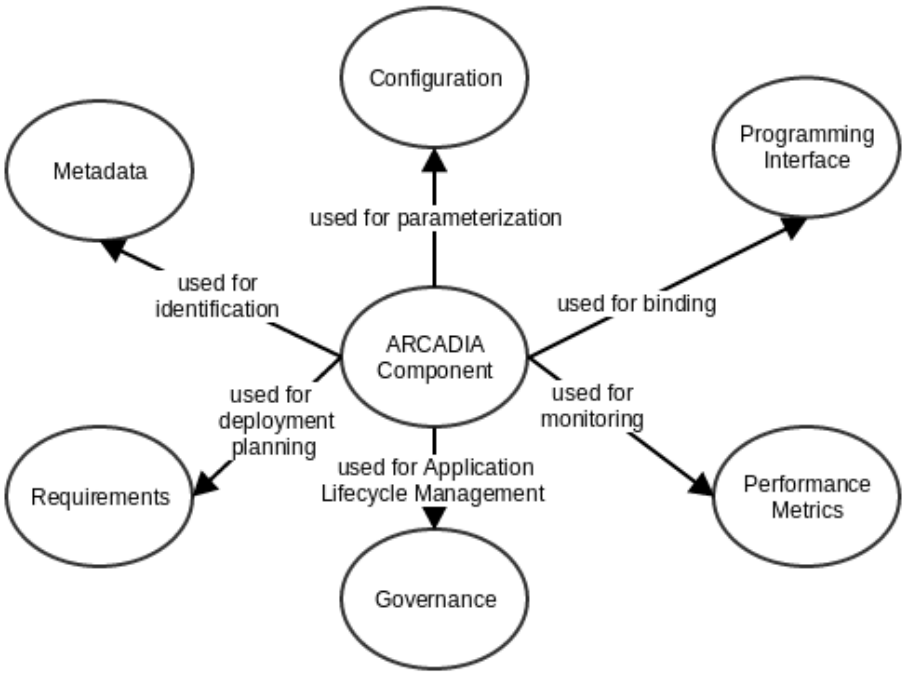


Figure 17: ARCADIA Component Metamodel (source: D2.3)

The realization of the aforementioned scenarios will be achieved through a) the development of the formal ARCADIA Components that 'wrap' the existing specific and generic enablers b) the development of new components that do not correlate with an existing enabler and c) with the creation of the appropriate service graph that actually realizes the scenario. These activities are depicted in the following figure (Figure 18) where the high-level architecture of ARCADIA is used in order to explain the relation between the ARCADIA architectural artefacts and the realized scenario.

As it is depicted, the existing enablers will be leveraged to orchestratable components through the usage of the ARCADIA Component Development Environment. These components will be stored in the Component Store and will be used in order to create the complex graphs of the secure communication scenario. This complex graph will expose several metrics that will be measured during the instantiation. Specific quality restrictions will be formulated as policies using the ARCADIA policy editor. Indicatively, the end-to-end delay in an encrypted call may be subjected to a specific restriction. Moreover, the leveraged FIWARE/FI-STAR enablers will be deployed in multiple IaaS offerings which may be in different locations.

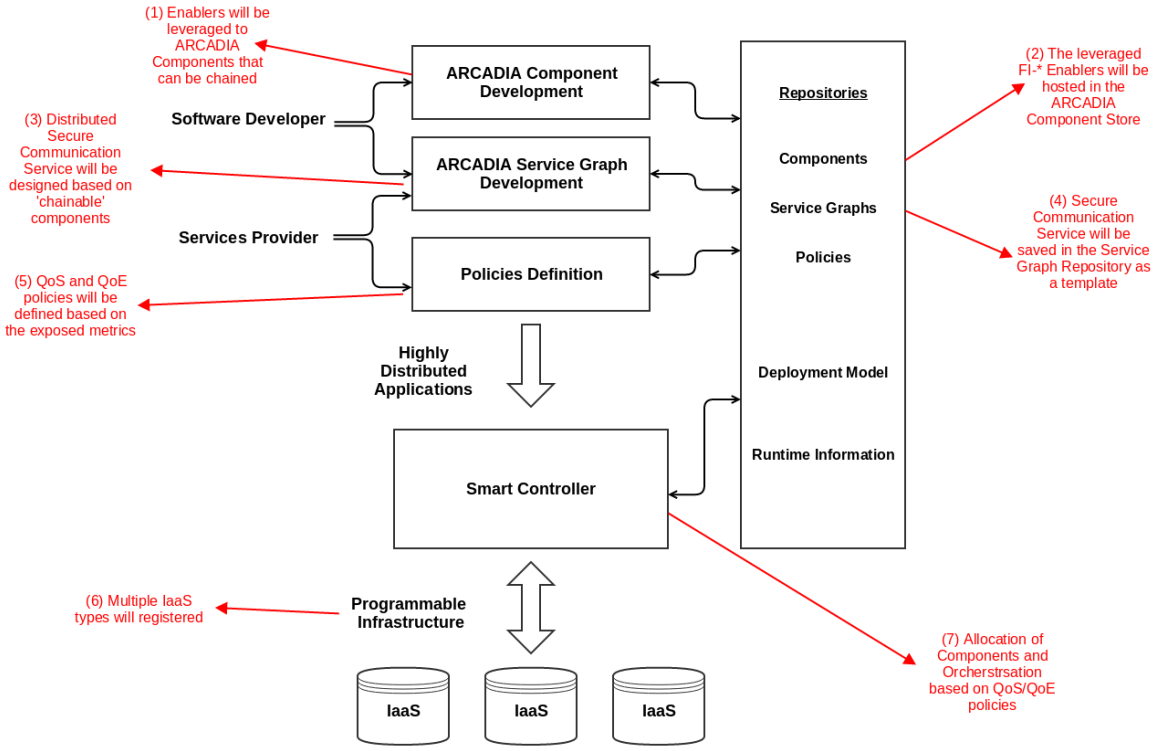


Figure 18: Correlation of high level ARCADIA architecture with FIWARE Enablers

Finally, it should be clarified that the initial orchestration and the continuous optimisation of the secure communication service will be under the supervision of the Smart Controller.

4.3 Use case service graph

Figure 19 presents a representation of the Remote Patient Monitoring Scenario service graph with several tangible microservices along with their concrete binding interfaces.

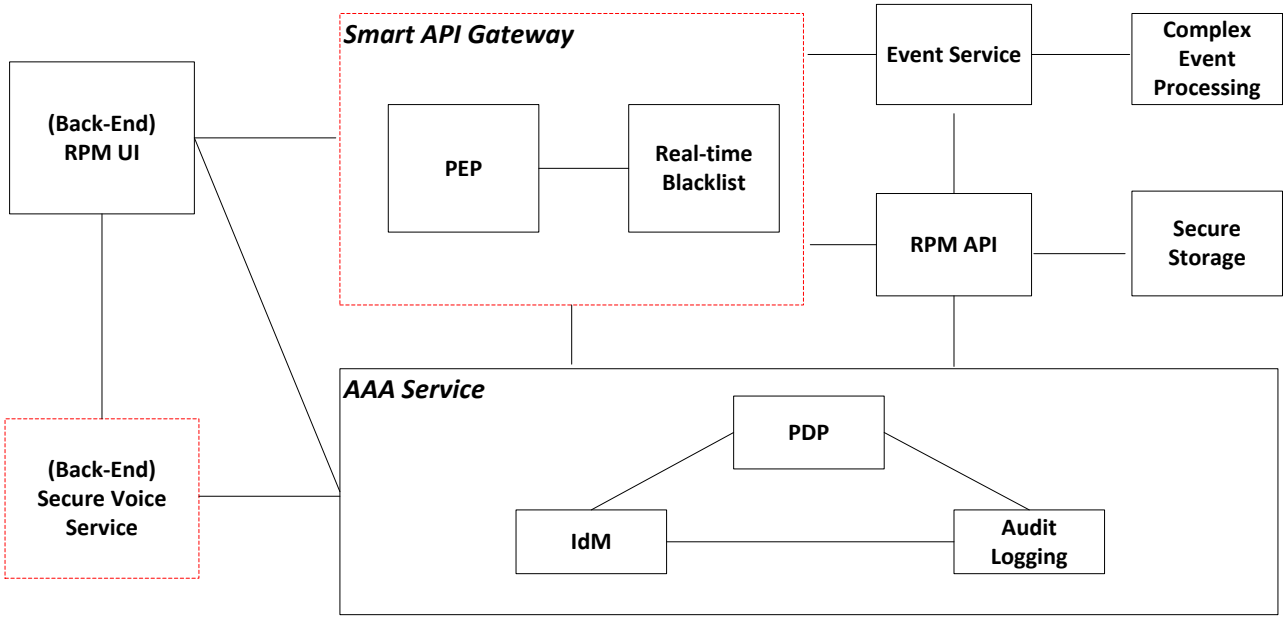


Figure 19: Fine grained service graph of the RPM scenario

This use case requires the orchestration of many components that undertake different tasks. Furthermore, in Figure 20 the Encrypted Communication Service Scenario service graph is depicted. This high level graph includes service registration and activation components, signalling components, call handling components etc.

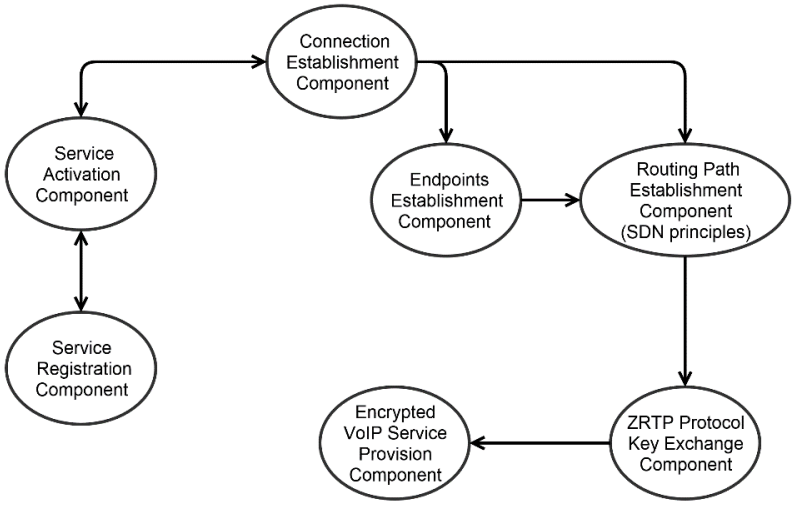


Figure 20: Abstract service graph for the secure voice service

4.3.1 Involved microservices

The aforementioned scenarios require many micro-services. Some of them are briefly discussed as follows:

- Event Service: provides a RESTful interface to be accessed by the event senders and receivers. The published interfaces follow the notation of Pub/Sub service such as the OMA NGSI-9/10 standards [18]
- Complex Event Processing: analyses event data in real-time, generates immediate insight and enables instant response to changing conditions
- IdM: manages information about users, roles and profiles. It also sends and validates tokens (OAuth2), as well as authentication mechanisms.
- PEP: A Policy Enforcement Point (PEP) is a component that protects resources against unauthorized access (which does not comply with the access control policy applicable for these resources). The PEP is the one intercepting each access request to the resource, but relies on the IdM to authenticate the request, and on the PDP to authorize it (deny of permit).
- PDP: provides an API to get authorization decisions based on authorization policies, and authorization requests from PEPs. The API follows the REST architecture style, and complies with XACML.
- Audit Logging: manages and stores access log information which helps maintaining permanent evidence of all authorized and unauthorized access to protected resource.
- Secure Storage: protects collected data through encryption. Depending on the requirements and type of the data, several protection layers can be applied: OS, file, database, application
- Real-time Blacklist: checks whether request come from a blacklist user and takes necessary action. Such response can be silent (e.g. does not alert user but silently logs and alert admin), passive (e.g. notify user, slow down the request but do not prevent) or active (e.g. deny the user access)

4.3.2 Binding Interface and Metric

In this use case, most of binding interfaces are RESTful API via HTTP(S). Depending on the requirements and selected software components, other non-HTTP interface can be applied (e.g. SQL/TCP for the secure storage, XMPP/AMQP for the event service)

In order to support dynamic scaling of cloud application at run-time, we will take into account common monitoring metrics:

- Average Response Time (ms)
- Request Arrival Rate (request/s)
- CPU Usage (%)
- Memory Usage (%)
- I/O Usage (%)

4.4 Service Provider Policy

In this use case we consider two types of service provider policy. The first one is for automatic scaling decision. Such policies can trigger the smart controller to scale up/down application according to exchanged data and predicted workload. For example, a policy could define a maximal response time and an application will be scale up when average response time value is greater than such value. Another policy could take into account resource utilization metrics (e.g. CPU, memory usage). This type of policy is common for all the use cases. Other type of policy can be derived from security and privacy requirements/annotations. For instance, based on the data security requirement, the smart controller will determine whether to use public or private cloud (public for maximum flexibility and efficiency, private for maximum control). To comply with data protection laws, a location-based policy can be specified which allows component placement only in a specific location/country. Finally, specific QoS and QoE characteristics that have to be achieved are interpreted as policy variables since they can be constrained during the service graph instantiation.

4.5 Validation and KPI

The validation of this use case is to demonstrate the correct behaviour of the smart controller to deploy security and privacy protection services to meet the related annotated requirements. It includes:

- The proper deployment of protection services such as back-end smart API gateway service and other supported ones including those at the front-end.
- The capability to mitigate service attacks including those are application-specific
- The capability to scale up and down of such services

The experiment will be performed in our infrastructure through our defined testing use cases. To test the automatic scaling capability, a client emulator will be developed to continuously generate requests towards related software components.

KPI	Expected value/behavior
The proper interpreting security and privacy annotation	Supported annotations are proper interpreted and required protection software components are successful deployed.

Provide security and privacy protection intelligence into application	Detect and mitigate attacks including those are application-specific
The number of FIWARE enabler (GE, SE) is used when implementing applications	As much as possible
Service disruption when scaling up/down	As small as possible

4.6 Programmable Infrastructure

The realization of the pilot requires the usage of many IaaS providers since a secure communication scenario involves multi-Point-of-Presence (multi-PoP) execution environment. To this end, several IaaS offerings will be used hosted by TU-Berlin (OpenStack), Ubitech (OpenStack) Amazon etc. As shown in Figure 21, indicative resources have been already allocated and ready for the use case deployment. Current resources can support up to 10 VMs with total 50GB of RAM.

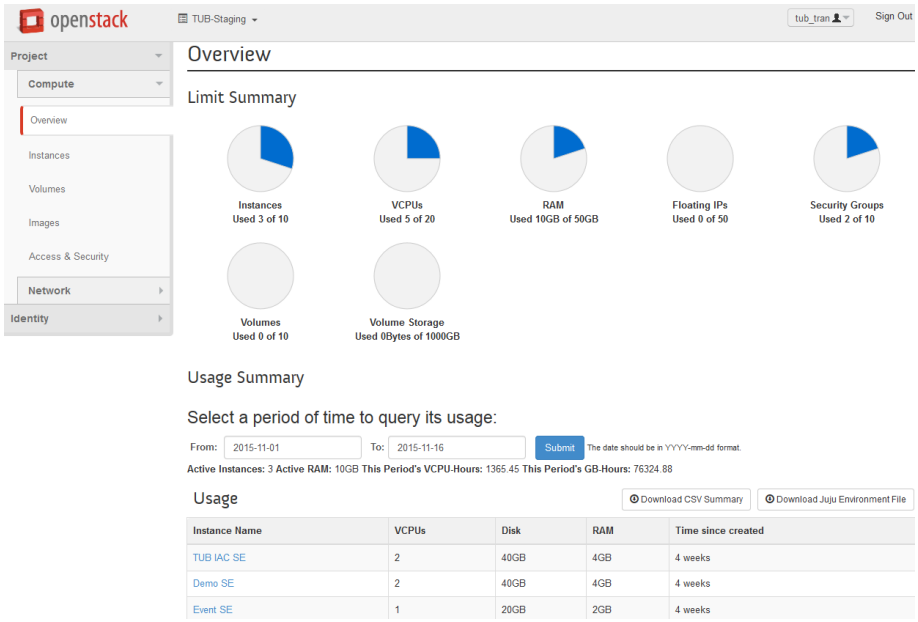


Figure 21: TU-Berlin Cloud Infrastructure

During the realization of the scenario the ARCADIA Orchestrator will 'place' each component that is required in the appropriate execution environment while trying to satisfy the specific QoS and QoE constraints which where mentioned previously.

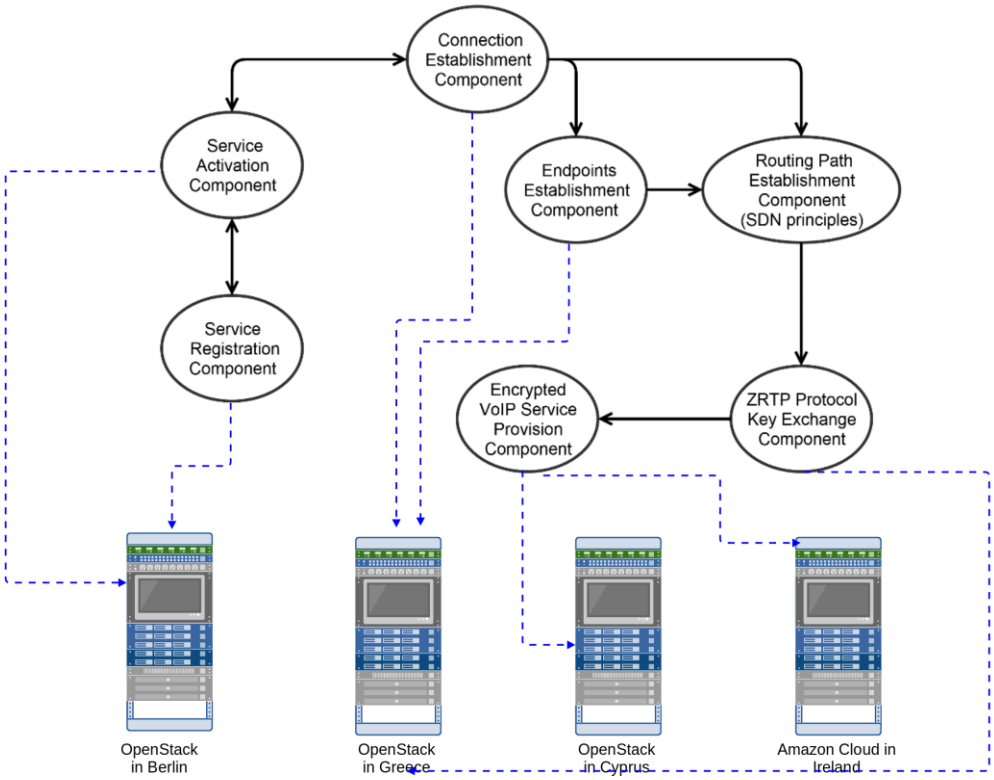


Figure 22: Indicative Component Placement

5 Performance evaluation and validation framework

The performance evaluation and validation framework that is going to be followed in ARCADIA is going to focus at the evaluation and validation of the appropriate and fully functional operation of the ARCADIA architectural components and the evaluation and validation of the successful implementation and deployment of the ARCADIA use cases. To this aim, two sets of performance evaluation criteria are defined. On one hand, criteria related to the guarantee of the provision of the envisaged functionality by the ARCADIA architectural components and, on the other hand, criteria related to the validation of the appropriate operation of the use cases and the tackling of the identified challenges per use case.

The performance evaluation and validation will lead to conclusions, including the coverage of the project requirements, the validation of the software development paradigm and technological choices made, the performance evaluation of the developed components in terms of accuracy, stability, scalability and flexibility and the acceptance factor of the proposed solution on behalf of the involved technical partners (e.g. software developers, network administrators). It should be noted that the specified performance evaluation and validation framework in this deliverable is going to be applied in Task 5.4 of WP5.

5.1 Performance Evaluation Criteria

Following, the technical assessment factors that will be used by the consortium to evaluate the IT performance of the ARCADIA use cases are described, in order to identify problems and shortcomings so as to come out of the project with a fully functional, reliable and stable environment that could

serve the needs of the users and could be exploited by software developers and service and infrastructure providers.

In this context, and following the state-of-the-art in software development, the technical assessment of the ARCADIA architectural components during their deployment and operation in the use cases will be based on an assessment model that includes a set of Key Performance Indicators (KPIs) and criteria extracted from the ISO/IEC 25010:2011 "Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models" standard[3]. Following the main directions of this standard, different elements and criteria will be selected and indicators specific to each element will be defined in order to produce a technical assessment model that can be used for evaluating the technical operation of the ARCADIA architectural components.

The ISO/IEC 25010:2011 standard has replaced the previous standard on software quality, which was the ISO/IEC 9126-1[4], and provides a new view on how software (and thus software platforms) should be assessed. In more detail, the ISO/IEC 25010:2011 defines as stated in its official website:

- *A quality in use model composed of five characteristics (some of which are further subdivided into sub-characteristics) that relate to the outcome of interaction when a product is used in a particular context. This system model is applicable to the complete human-computer system, including both computer systems in use and software products in use.*
- *A product quality model composed of eight characteristics (which are further subdivided into sub-characteristics) that relate to static properties of software and dynamic properties of the computer system. The model is applicable to both computer systems and software products.*

Since the assessment and evaluation of the ARCADIA architectural components covers not only the IT elements that will be delivered, but also the perceived usefulness and appropriateness for use by the end users (e.g. software developers, service providers), the evaluation will be conducted on following the categories set, by both models of the ISO 25010, adapted appropriately to the scope and nature of ARCADIA.

Product Quality Model

The product quality model describes the internal and external measures of software quality. Internal measures describe a set of static internal attributes that can be measured. The external measures focus more on software as a black box and describes external attributes that can be measured.

In general, this model evaluates software quality using a structured set of characteristics (each of them including other sub-characteristic), which are the following:

1. Functional suitability - The degree to which the product provides functions that meet stated and implied needs when the product is used under specified conditions.
2. Performance efficiency - The performance relative to the amount of resources used under stated conditions.
3. Compatibility - The degree to which two or more systems or components can exchange information and/or perform their required functions while sharing the same hardware or software environment.
4. Operability - The degree to which the product has attributes that enable it to be understood, learned, used and attractive to the user, when used under specified conditions.
5. Reliability - The degree to which a system or component performs specified functions under specified conditions for a specified period of time.
6. Security - The degree of protection of information and data so that unauthorised persons or systems cannot read or modify them and authorised persons or systems are not denied access to them.
7. Maintainability - The degree of effectiveness and efficiency with which the product can be modified.

[3] http://www.iso.org/iso/catalogue_detail.htm?csnumber=35733

[4] http://www.iso.org/iso/catalogue_detail.htm?csnumber=22749

8. Portability - The degree to which a system or component can be effectively and efficiently transferred from one hardware, software or other operational or usage environment to another.

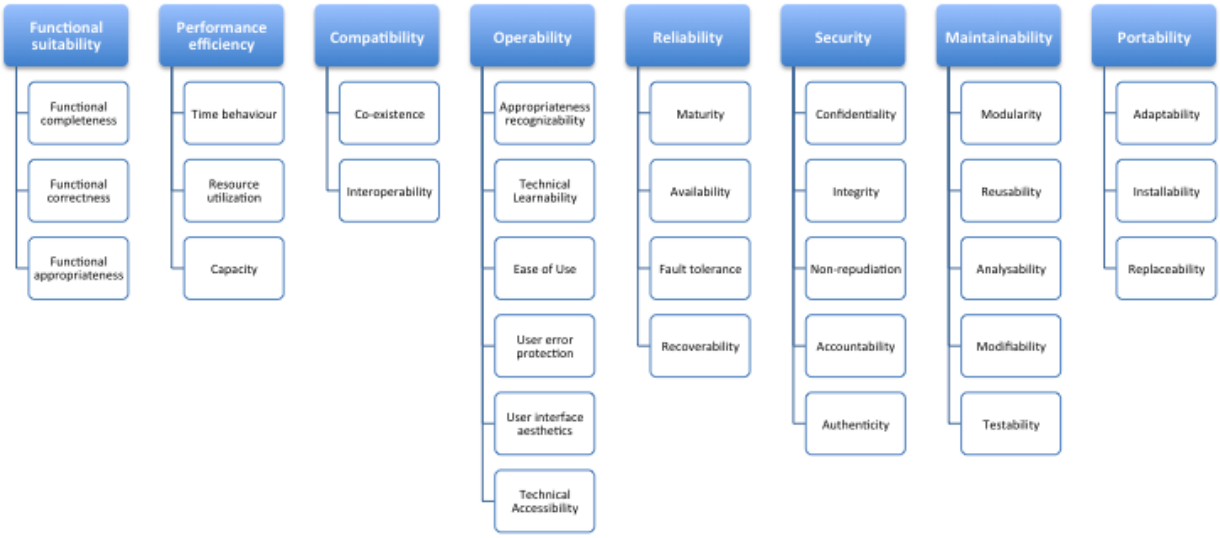


Figure 23: A product quality model view based on the ISO/IEC 25010:2011 standard

The following table (Table 1) showcases the sub-characteristics of each category and indicates their relativity to the ARCADIA Architectural Components.

Table 1: Technical Characteristics, Sub-characteristics and Relevance to ARCADIA

Sub-characteristics	Definition	Relation to ARCADIA	Remarks
Functional suitability			
Functional completeness	Degree to which the set of functions covers all the specified tasks and user objectives.	YES	
Functional correctness	System provides the correct results with the needed degree of precision.	YES	
Functional appropriateness	The functions facilitate the accomplishment of specified tasks and objectives.	YES	
Performance efficiency			
Time behaviour	Response, processing times and throughput rates of a system, when performing its functions, meet requirements.	YES	
Resource utilisation	The amounts and types of resources used by a system, when performing its functions, meet requirements.	YES	
Capacity	The maximum limits of a product or system parameter meet requirements.	YES	
Compatibility			
Co-existence	Product can perform its functions efficiently while sharing environment and resources with other products.	YES	
Interoperability	A system can exchange information with other	YES	

Sub-characteristics	Definition	Relation to ARCADIA	Remarks
	systems and use the information that has been exchanged.		
Operability			
Appropriateness recognisability	Users can recognise whether a system is appropriate for their needs, even before it is implemented.	Partially	Not a core technical assessment issue of ARCADIA
Technical Learnability	The system has functions which enable learning specified operations of it.	YES	
Ease of Use	System has attributes that make it easy to operate and control.	YES	
User error protection	System protects users against making errors.	YES	
User interface aesthetics	User interface enables pleasing and satisfying interaction for the user.	YES	
Technical Accessibility	System can be used by people with the widest range of characteristics and capabilities.	YES	
Reliability			
Maturity	System meets needs for reliability under normal operation.	YES	
Availability	System is operational and accessible when required for use.	YES	
Fault tolerance	System operates as intended despite the presence of hardware or software faults.	YES	
Recoverability	System can recover data affected and re-establish the desired state of the system is case of an interruption or a failure.	YES	
Security			
Confidentiality	System ensures that data are accessible only to those authorised to have access.	YES	
Integrity	System prevents unauthorised access to, or modification of, computer programs or data.	YES	
Non-repudiation	Actions or events can be proven to have taken place, so that the events or actions cannot be repudiated later.	YES	
Accountability	Actions of an entity can be traced uniquely to the entity.	YES	
Authenticity	The identity of a subject or resource can be proved to be the one claimed.	Partially	Not a core technical assessment issue of ARCADIA
Maintainability			
Modularity	System is composed of components such that a change to one component has minimal impact on other components.	YES	
Reusability	An asset can be used in more than one system, or in building other assets.	YES	
Analysability	Effectiveness and efficiency with which it is possible to assess the impact of an intended change.	YES	
Modifiability	System can be effectively and efficiently modified without introducing defects or	YES	

Sub-characteristics	Definition	Relation to ARCADIA	Remarks
	degrading existing product quality.		
Testability	Effectiveness and efficiency with which test criteria can be established for a system.	YES	
Portability			
Adaptability	System can effectively and efficiently be adapted for different or evolving hardware, software or usage environments.	YES	
Installability	Effectiveness and efficiency with which a system can be successfully installed and/or uninstalled.	YES	
Replaceability	Product can be replaced by another specified software product for the same purpose in the same environment.	NO	Not to be tested during the project

Quality in Use Model

Apart from the software quality model, which focuses on core IT requirements and performance, the ISO 25010 introduced the Quality in Use model which describes the perception of the quality of the system from a user's perspective. The different characteristics and sub-characteristics of this model are derived from testing or observing the results of real or simulated use of the system.

As with the previous model, this one assesses software quality (from a user point of view) using the following set of characteristics (each of them including other sub-characteristics):

1. Effectiveness - The accuracy and completeness with which users achieve specified goals;
2. Efficiency - The resources expended in relation to the accuracy and completeness with which users achieve goals;
3. Satisfaction- The degree to which users are satisfied with the experience of using a product in a specified context of use;
4. Safety - The degree to which a product or system does not, under specified conditions, lead to a state in which human life, health, property, or the environment is endangered;
5. Usability - The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use;

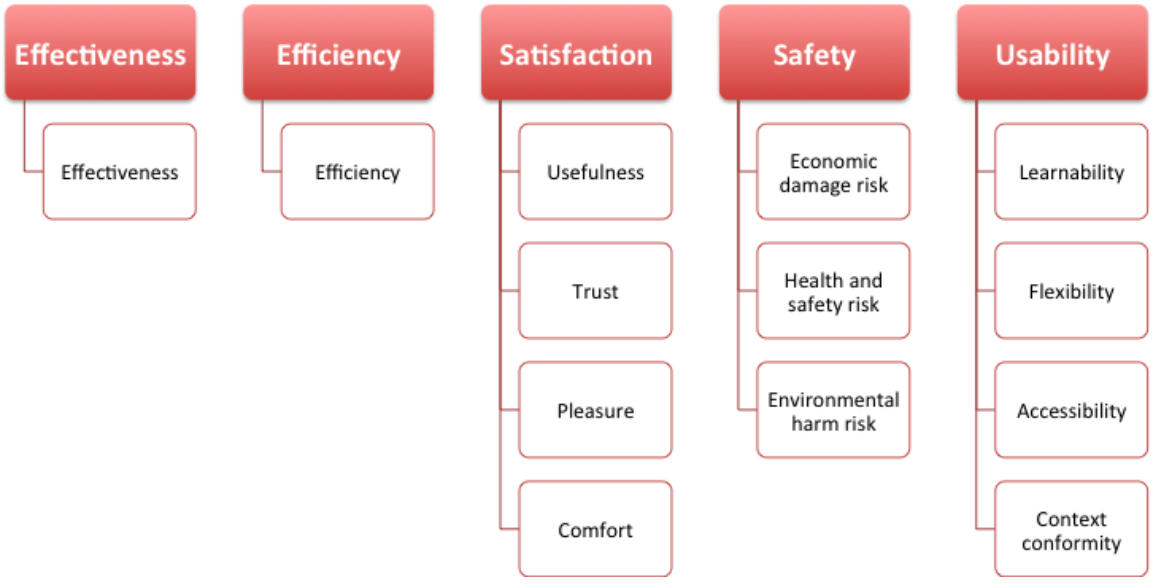


Figure 24: A quality in use model view based on the ISO/IEC 25010:2011 standard

The following table (Table 2) showcases the sub-characteristics of each category and indicates their relativity to ARCADIA.

Table2: Quality in Use Model - Characteristics, Sub-characteristics and Relevance to ARCADIA

Sub-characteristics	Definition	Relation to ARCADIA	Remarks
Effectiveness			
Effectiveness	The degree at which users achieve their goals accurately when using the system	YES	
Efficiency			
Efficiency	The degree to which the users find that the software is efficiently covering its intended purpose	YES	
Satisfaction			
Usefulness	The degree to which users find useful the software and its operations	YES	
Trust	The degree to which users feel that they can trust the system	YES	
Pleasure	The degree to which users find the software's functions a pleasure to use (emotionally)	NO	Not relevant for ARCADIA
Comfort	The degree to which users think that the system provides the comforts needed (physically)	NO	Not relevant for ARCADIA
Safety			
Economic damage risk	Acceptable levels of risk of harm to the operator in the intended contexts of use.	NO	Not relevant for ARCADIA
Health and Safety risk	Acceptable levels of risk of harm to the public in the intended contexts of use.	NO	Not relevant for ARCADIA
Environmental harm risk	Acceptable levels of risk of harm to property or the environment in the intended contexts of use.	NO	Not relevant for ARCADIA
Usability			
Learnability	The extent to which a product can be used by specified users to achieve specified learning goals with effectiveness, efficiency, safety and satisfaction in a specified context of use	YES	
Flexibility	The degree to which usability and safety requirements are met in all the intended contexts of use	YES	
Accessibility	The extent to which a product can be used by users with specified disabilities to achieve specified goals with effectiveness, efficiency, safety and satisfaction in a specified context of use	YES	
Content Conformity	The degree to which usability and safety requirements are met in all the intended contexts of use	NO	Not relevant for ARCADIA

As already stated in Sections 2-4, a set of performance evaluation and acceptance criteria are defined per use case, aiming at evaluating and validation the successful implementation and operation of the

use cases. Such criteria are going to be included in the performance evaluation to be realized per use case, in addition to the aforementioned metrics and criteria that are going to be used for the performance evaluation of the ARCADIA architectural components and the overall ARCADIA framework.

5.2 Performance Evaluation and Validation Process

Within ARCADIA, performance evaluation and validation is going to follow an iterative process within WP5, taking into account the phases of deployment, implementation and evaluation of the defined use cases. This process will be based on the two phases of implementations of the use cases, that are going to lead to the release of deliverables D5.1 in M24 and D5.2 in M30, as well as the release of D5.3 in M36 that is going to include all the performance evaluation and validation results of the project. Primary results in D5.1 will be based on the first version of the ARCADIA Smart Controller and the ARCADIA editing/deployment/development toolkits while the results in D5.2 and D5.3 are going to be based on the final release of the corresponding architectural components.

It should be noted that intense collaboration among all the ARCADIA partners is going to be realized towards the development and the deployment in the use case, while feedback on behalf of the use cases –at their design and draft deployment phase- is going to be provided to WP3 and WP4 towards the release of the final version of the ARCADIA Smart Controller and the ARCADIA editing/deployment/development toolkits.

6 Conclusions

This deliverable has provided the detailed description of the three use cases that are included in the project's technical annex, as well as the methods for their validation and performance evaluation. Each use case is analyzed in micro-services and a corresponding service graph with their interconnections is presented and documented. The work of this deliverable has taken as input the requirements that are included in deliverable D2.1 and the specification of the first version of the ARCADIA context model that is contained in deliverable D2.2. It is also based on the architecture and the functionalities that are expected to exist in the ARCADIA Framework, as it is described in the deliverable D2.3.

More specifically, in each of the sections that are devoted to a use case, namely sections 2, 3 and 4, an introduction with a general description of the use case is provided at the beginning. The business logic, the requirements and any related background information is also included, so that the service graph that follows to be fully comprehensible and clear, in terms of micro-services that are comprised and interconnections among them. Finally, the parameters that are necessary for the validation of the use case, as well as for the evaluation of its performance and the process that is going to be followed to this aim are described.

Annex I: References

[1] Wikipedia, the free encyclopedia, https://en.wikipedia.org, accessed in October - November 2015

[2] C. Bouras, A. Gkamas, G. Kioumourtzis, Performance Evaluation of MPEG-4 Video Transmission with the Adaptive Smooth Multicast Protocol (ASMP), The Fifteenth IEEE Symposium on Computers and Communications (ISCC'10), Riccione, Italy, June 22 - 25 2010,pp. 540 - 545.

[3] C. Bouras, A. Gkamas, G. Kioumourtzis, Adaptive Smooth Multicast Protocol for Multimedia Transmission: Implementation Details and Performance Evaluation, International Journal of Communication Systems, Wiley InterScience, Vol. 23, Issue 3, , 2010,pp. 299 - 333.

[4] International Standard ISO/IEC 23009-, Second edition, 2014-05-15.

[5] Y. Ding, Y. Du, Y. Hu, Z. Liu, L. Wang, K. W. Ross, A. Ghose, "Broadcast Yourself: Understanding YouTube Uploaders," Proc. 2011 ACM Internet Measurement Conference, Berlin, Germany.

[6] H. Sohn, H. Yoo, W. De Neve, C. S. Kim, and Y.-M. Ro., "Full-reference video quality metric for fully scalable and mobile svc content", IEEE Transactions on Broadcasting, 56(3):269{280, Sept 2010.

[7] J. Klaue, B. Rathke and A. Wolisz, "EvalVid – A Framework for Video Transmission and Quality Evaluation", Proceedings of the 13th International Conference on Modeling, Techniques and Tools for Computer Performance Evaluation, Urbana, Illinois, 2003.

[8] Rohaly M., et al., "Video Quality Experts Group: Current Results and Future Directions", In: SPIE Visual Communications and Image Processing, Perth, Australia, June 21-23, 2000, Vol. 4067, p.742-753.

[9] "Toward a Strategic Agenda for Software Technologies in Europe", Information Society Technologies Advisory Group (ISTAG), July 2012, Available Online: http://cordis.europa.eu/fp7/ict/docs/istag-soft-techwgreport2012.pdf , Access: April 2014

[10] FIWARE: Open APIs for Open Minds, http://www.fiware.org

[11] Internet-enabler Innovation in Europe, http://www.fi-ppp.eu/projects/

[12] Microservices Architecture, http://microservices.io/patterns/microservices.html

[13] Network Functions Virtualization, https://portal.etsi.org/nfv/nfv_white_paper.pdf

[14] OAuth 2.0 Authorization Framework, http://tools.ietf.org/html/rfc6749

[15] OASIS eXtensible Access Control Markup Language, https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml

[16] Future Internet Social and Technological Alignment Research - FI-STAR, https://www.fi-star.eu

[17] FIWARE IOT Stack, http://fiware-iot-stack.readthedocs.org/en/latest/index.html

[18] OMA NGSI Context Management, http://technical.openmobilealliance.org/Technical/release_program/docs/NGSI/V1_0-20101207-C/OMA-TS-NGSI_Context_Management-V1_0-20100803-C.pdf