TRANSPARENT APPLICATION DEPLOYMENT IN A SECURE, ACCELERATED AND COGNITIVE CLOUD CONTINUUM

Grant Agreement no. 101017168

Deliverable D2.2
SERRANO use cases, platform requirements and KPIs analysis

Programme:	H2020-ICT-2020-2
Project number:	101017168
Project acronym:	SERRANO
Start/End date:	01/01/2021 – 31/12/2023

Deliverable type:	Report
Related WP:	WP2
Responsible Editor:	INNOV
Due date:	30/06/2021
Actual submission date:	30/06/2021

Dissemination level:	Public
Revision:	Final

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017168

Revision History

Date	Editor	Status	Version	Changes
29.01.21	INNOV	Draft	0.1	Initial ToC and brief instructions per section.
01.02.21 - 05.03.21	All contributors	Draft	0.2-0.4	Consolidated version including first contributions regarding the analysis process (section 3), the UC descriptions (section 4) and the requirements analysis (section 5) aspects.
08.03.21 - 22.03.21	All contributors	Draft	0.5-0.8	Consolidated versions including updates in sections 3, 4 and 5.
23.03.21 - 12.05.21	All contributors	Draft	0.9-0.15	Consolidated versions including updates in sections 3, 4, 5 and first contributions regarding the business vision (section 6) aspect.
18.05.21	All contributors		0.16	Consolidated version including updates in sections 3, 4, 5 and 6.
26.05.21	All contributors		0.17	Consolidated version for internal review. Includes updates in sections 3, 4, 5, 6 and first version of the introduction (section 2) and the conclusions (section 7).
08.06.21	IDEKO, UVT, ICCS, INNOV		0.18	Consolidated version including all comments and feedback from internal review.
09.06.21 - 24.06.21	All contributors		0.19-0.20	Consolidated version addressing the internal review feedback, including updates in sections 2, 3, 4, 5, 6 and the first version of the executive summary (section 1).
28.06.21	All contributors		0.21	Consolidated version with final updates in all sections.
29.06.21	INNOV, ICCS		1.0	Final version for submission.

Author List

Organization	Author
AUTH	Kostas Siozios, Argyris Kokkinis
CC	Marton Sipos, Marcell Feher
ICCS	Emmanouel Varvarigos, Aristotelis Kretsis, Panagiotis Kokkinos, Polyzois Soumplis
IDEKO	Javier Martín, Elena Urkia, Amaia Arregui, Aitor Fernández
INB	Javier Castillo, Ferad Zyulkyarov
INNOV	Vassiliki Andronikou, Efstathios Karanastasis, Efthymios Chondrogiannis, Filia Filippou
INTRASOFT	Makis Karadimas, Paraskevas Bourgos
MLNX	Yoray Zack, Juan Jose Vegas Olmos
NBFC	Anastassios Nanos, Babis Chalios
USTUTT/HLRS	Dimitry Khabi
UVT	Gabriel Iuhasz, Silviu Panica

Internal Reviewers

Aitor Fernández, IDEKO

Silviu Panica, UVT

Abstract: This deliverable presents the outcomes of the *Task 2.2 - Concept, Use Case Requirements and Business Prospects* of *Work Package 2 - Requirements and System Design* of the SERRANO project during its first iteration. It includes the full specification of the SERRANO use cases, the definition of the requirements of each building block/key area of the project, an overview of the business vision for each use case and the overall SERRANO project and platform, the main identified success criteria and the definitions and quantifications of the first version of the KPIs to be used for evaluating the innovations developed during the project.

Keywords: Use case definition, requirements elicitation and analysis, business vision, KPIs, SERRANO platform

Disclaimer: *The information, documentation and figures available in this deliverable are written by the SERRANO Consortium partners under EC co-financing (project H2020-ICT-101017168) and do not necessarily reflect the view of the European Commission. The information in this document is provided "as is", and no guarantee or warranty is given that the information is fit for any particular purpose. The reader uses the information at his/her sole risk and liability.*

Table of Contents

List of Figures

List of Tables

Abbreviations

ASIC	Application-Specific Integrated Circuits
AES	Advanced Encryption Standar
AI	Artificial Intelligence
API	Application Programming Interface
ARDIA	A Resource reference model for Data-Intensive Applications
ARIMA	Auto-Regressive Integrated Moving Average
ART	Automatic Reports Tool
BPaaS	Business Processes as a Service
CAGR	Compound Annual Growth Rate
CDSSaaS	Distributed Secure Storage as a Service
CI/CD	Continuous integration/Continuous Deployment
DL	Deep Learning
CPU	Central Processing Unit
CSP	Cloud Service Provider
CSV	Comma Separated Values
DPO	Dynamic Portfolio Optimization
DSP	Digital Signal Processing
ECHAR	Edge, Cloud and HPC Acceleration Requirement
ESAaaS	Extreme Scale Analytics as a Service
EU	European Union
F	Functional
FaaS	Financial (investment management) as a Service
FFT	Fast Fourier Transform
FPGA	Field-Programmable Gate Array
GB	GigaByte
GCC	GNU Compiler Collection
GCM	Galois/Counter Mode
GDPR	General Data Protection Regulation
GPU	Graphics Processing Unit
GR	General Requirement
HDF	Hierarchical Data Format
HPC	High Performance Computing
HPCG	High Performance Conjugate Gradients (benchmark)
HPL	High-Performance Linpack (benchmark)
HW	Hardware

HWW	Höchstleistungsrechner für Wissenschaft und Wirtschaft
IaaS	Infrastructure as a Service
IEC	International Electrotechnical Commission
IoT	Internet of Things
IPDRR	Integration and Platform Development Requirement
IPsec	Internet Protocol Security
ISO	International Organization for Standardization
KPI	Key Performance Indicator
KPP	Key Performance Parameter
LAN	Local Area Network
LSTM	Long short-term memory
MB	MegaByte
ML	Machine Learning
MPSoC	Multiprocessor System on a Chip
MMTR	Mean Time To Repair
MOE	Measure of Effectiveness
MOP	Measure of Performance
MPI	Message Passing Interface
N/A	Not Applicable
NCTFR	Network and Cloud Telemetry Framework Requirement
NF	Non-functional
NIST	National Institute of Standards and Technology
NVMe	Non-Volatile Memory
PaaS	Platform as a Service
PCIe	Peripheral Component Interconnect Express
RASaaAAS	Roboadvisor as a Service
REST	Representational state transfer
ROSAR	Resource Orchestration and Service Assurance Requirement
SaaS	Software as a Service
SDK	Software Development Kit
SIR	Secure Infrastructure Requirement
SME	Small and Medium-sized Enterprises
SoC	System on a Chip
SOR	Service Orchestration Requirement
TBD	To Be Discussed
TCO	Total Cost of Ownership
TG	Technical Group

TLS	Transport Layer Security
UC	Use Case
VPN	Virtual Private Network
WP	Work Package
YAML	Yet Another Mark-up Language

1 Executive Summary

The deliverable is divided in eight main section. Section 1 is this executive summary.

Section 2 serves as an introduction to the document. It provides information about the document's purpose, including references to the relevant SERRANO project Work Packages, Tasks and Deliverables, its structure and the audience to which it is mainly addressed.

Section 3 presents in detail the process followed for the collection and analysis of the requirements, the outcome of which is presented in this deliverable. It introduces the definition of the requirements and an enumeration of their types as well as the stakeholders that were involved in this process. It then discusses in detail the good practices that were followed and the rules that were taken into account during the requirements definition process. It further describes the measures that were taken for preventing common mistakes that often occur in such cases. Subsequently, it presents the organization of work that took place and the detailed methodology that was used for elaborating the use cases as well as for collecting and analysing the requirements. The requirements were grouped in seven key areas and a number of iterations were performed involving both technical and use case consortium partners.

Section 4 presents the overview and analysis of the three SERRANO use cases, namely (i) Secure Storage, (ii) Fintech and (iii) Anomaly detection in manufacturing settings. For each of the use cases, detailed information is provided regarding their motivation followed by a narration, a description of the stakeholders involved and their high-level requirements. Then, the main applications/tools, which are part of each use case, and the services/processes, which are part of these applications, are presented in detail along with their main characteristics and needs. The use case workflow which is further provided, illustrates how these components are used and interact with each other for covering the use case functionality. Additionally, the data involved as well as the infrastructure to be integrated with SERRANO, are presented in more detail. Finally, the use case success criteria are also listed and a first approach is made to introduce a number of relevant KPIs along with their estimated target values.

Section 5 presents in detail the requirements elicited in this first version of the deliverable. The section is divided in eight sections, one concerning the general and non-functional requirements of the SERRANO platform and one for each of the key areas identified. Each of these sections starts with the provision of relevant information and a statement of its purpose, and is followed by the systematic presentation of each requirement, its ID, title and detailed description, its priority, the stakeholders involved, its rationale and goal, its acceptance measures (quantified wherever relevant) and its dependencies with other requirements or specific Work Packages of the SERRANO project.

Section 6 discusses the business vision of the SERRANO platform, including its main goals and aspirations, as well as its business ecosystem. There is also an additional section for each of the three individual use cases, presenting the business vision from their point of view.

Section 7 concludes the deliverable. It discusses the main findings and future work to be undertaken in the SERRANO project.

Section 8 contains information about other sources that have been referred to in the various sections of the document.

2 Introduction

2.1 Purpose of this document

This deliverable presents the outcomes of the *Task 2.2 - Concept, Use Case Requirements and Business Prospects* of *Work Package 2 - Requirements and System Design* of the SERRANO project during its first iteration. This task involves the description of the full specification of the SERRANO use cases (UC/s from now on), as well as the definition of the requirements of each building block/key area of the project.

The process of the requirements collection and analysis, which is based on a variety of modalities and good practices, follows a specific methodology that includes several rounds of collaboration and contributions that is specifically detailed in this document.

The requirements are gathered, defined in detail and prioritised so as to serve as the basis for designing the architecture of the SERRANO platform in *Task 2.3 - Architecture Specification* and its services in *Work Packages 3-5*.

This deliverable also presents the UCs that are associated with the scenarios of the project, which were identified, defined and refined in parallel to the collection of the requirements. It further gives an overview of the business vision for each of the UCs as well as the overall SERRANO project and the resulting platform. These initial findings shall be elaborated in *Work Package 7 - Business Modelling, Dissemination, Exploitation and Standardization* of the project.

The document further presents the main identified success criteria, defines and quantifies a first version of the KPIs used for evaluating the innovations developed in the project, which are to be further elaborated in *Work Package 6 - Platform Integration and Testing, Use Cases Development and Evaluation*.

An updated version of this document entitled as *D2.4 - Final version of SERRANO use cases, platform requirements and KPIs analysis* will be provided in M16 of the project timeline.

2.2 Document structure

The present deliverable is split into eight major sections:

- Executive summary
- Introduction
- The SERRANO requirements analysis process
- Use cases overview and analysis
- Requirements Analysis
- Business Vision
- Conclusions

- References

2.3 Audience

This document is publicly available and should be of use to anyone interested in the detailed specification of the SERRANO project and its UCs, the SERRANO platform requirements and the business vision of the project.

3 The SERRANO requirements analysis process

3.1 Requirements types and definition

Among the key objectives of the SERRANO Work Package 2 is to collect and analyse the SERRANO platform requirements for each of its different building blocks, based on the experience of stakeholders and the detailed description of the UCs associated with the scenarios of the project.

A successful delivery of the expected system can be achieved when based on high-quality requirements. The process of requirements elicitation reduces potential gaps between the technical team (e.g., developers) and its resulting output and the end users. The results of a thorough requirements definition can also serve as a basis for the evaluation of the final system.

Requirements are of two main types:

- *Functional (F)*, which are associated with the capability or application needed to directly support the users' accomplishment of their mission and tasks, and
- *Non-functional (NF)*, which are typically implicit and technical in nature and emerge as system requirements to satisfy the users' functional needs, e.g., availability, quality of service, timeliness, and accuracy.

3.2 Good practices in requirements definition

The requirements definition is an essential step in order to increase the chances of a project being successful. There is no perfect way for defining them, but there are undoubtedly various good practices to consider. The ways to follow in order to gather requirements can be chosen depending on the nature of a project.

Some examples of methods used to identify the requirements are the following:

- Interviews with stakeholders
- Use case descriptions
- Questionnaires
- Brainstorming
- Storyboarding

The following good practices might as well help the process of requirement identification:

- Identify all the stakeholders
- Make sure all the players are involved in the procedure
- Understand the big picture before deep diving
- Prioritize and iterate
- Use a terminology that is employed by the stakeholders

- Drive your solution using the requirements and not inversely

The requirement identification is not a static process but rather an evolving one with several iterations and interactive discussions between the partners of the project.

According to the analysis of MITRE systems engineers [1], among the most important best practices in requirements definition are the following:

- *Baseline and agree*. Developing requirements is usually a *collaborative* activity involving users, developers, maintainers, integrators, etc., to avoid placing the responsibility of requirements analysis solely on one stakeholder. When *all team members* acknowledge that a set of requirements is done, this is called a baseline (this also means recognising that definitions will evolve).
- *Requirements analysis is an iterative process*. At each step, the results must be compared for *traceability* and *consistency* with stakeholders' requirements, and then *verified* with users, or *go back into the process for further analysis*, before being used to drive architecture and design.
- Special attention is to be paid to *interface requirements*. Requirements must clearly capture *all the interactions with external systems* and the *external environment* so that boundaries are clear.
- *Be flexible*. To balance out the rigidness of baselining requirements, a development team should consider what constitutes a "change of requirements" as distinguished from a valid interpretation of requirements. The key is to find a *balance* between adherence to a baseline and sufficient flexibility.
- Use *templates* and *tools* that suit your needs.

Also, based on a study on deficiencies in specifying requirements [2], among the top identified defects are the following: (a) Incorrect information, (b) omissions, (c) ambiguities, (d) poorly written, (e) misplaced, and (f) implementation or operations-related, and not design-free definitions.

Table 1: Preventive measures for requirement specification defects

Requirement specification defects	Preventive measures
Incorrect Information	Complete scope Operational concepts Rationale Include stakeholders
Omissions	Identify necessary reiteration Use a standard outline/checklist
Ambiguities	Use standards Validate
Poorly Written	Use simple format Use editor
Misplaced	Standard outline (template)

Implementation or Operations	Ask "Why?" Ask "What do you want to verify?"

In the SERRANO project in particular, a good understanding of the UCs plays a central role in order to identify the platform requirements. Even though the UCs had been defined in the proposal, it was nonetheless essential to specify them and re-define them as needed, so as to create the context in which the solutions were going to evolve during the project. A common understanding between the UC providers, also representing their end users and the technical partners, was a sine qua non for a successful requirements identification.

Based on the above, the following rules were taken into account for an optimal definition of the requirements:

- A requirement must be *traceable* to an operational need and *attributable* to an authoritative source, e.g., a person or a document. Once defined, it receives a *unique identifier* allowing the software design, code, and test procedures to be precisely traced back to the requirement.
- A requirement definition should be *unambiguous*. A good practice is to test the wording of the requirement from the perspectives of different stakeholders and check whether it can be interpreted in multiple ways. Vague and general statements must be avoided. This will allow testing the requirements and demonstrating that they are satisfied by the end product or service. Descriptions need to be *clear*, *specific* and *singular*.
- Definitions need to be *measurable*, either quantitatively or qualitatively. Typical categories of measures are:
    - Measures of Effectiveness (MOEs), i.e. measures of mission success from stakeholders' point of view
    - Measures of Performance (MOPs), used to determine whether the system meets performance requirements necessary to satisfy the MOE
    - Key Performance Parameters (KPPs) or Indicators (KPIs) defined by stakeholders as measures of minimal and critical system or project performance and level of acceptance
- Requirements must be *uniquely identified*, *consistent* and *compatible* with each other.
- Requirements need to be *feasible*, i.e. they are attainable in terms of technology, cost, and schedule. If a requirement cannot be implemented, it should be removed from the statement.
- The specification of requirements should be *design-free*, reflecting *what* the system needs to accomplish.

3.3 The SERRANO requirements capturing and analysis methodology

In the SERRANO project, there are 3 UC providers and a number of Technical partners. The SERRANO partners, on the whole, identified 7 (seven) key areas for requirements collection and analysis based on the project's objectives and vision, as well as on the individual partners' expertise. In order to ensure high quality contribution in all 7 topics within the predefined time and work plan, one group, which included both technical and UC partners, was formulated under each topic of work (as presented in the Table 2), with a mandate to focus on the collection and analysis of the requirements relevant to the specific topic.

Table 2: Requirements collection and analysis groups

Group ID	Topic	SERRANO Lead partner	SERRANO participating partners	Relevant tasks
G1	General requirements	ICCS	All	All
G2	Cloud and edge acceleration	AUTH	MLNX, USTUTT, UVT, NBFC, ICCS	Tasks 4.1-4.4
G3	Secure infrastructure	MLNX	CC, INTRA, NBFC	Tasks 3.1-3.4
G4	Network and cloud telemetry framework	ICCS	MLNX	Task 5.3
G5	Resource orchestration and service assurance	ICCS	NBFC, USTUTT, INNOV	Tasks 5.2, 5.4, 5.5
G6	Service orchestration	INNOV	UVT	Task 5.1
G7	Integration and platform development	INTRA	All	Task 6.1

The groups formed used a number of modalities in order to elicit the requirements:

1. Directly collaborating with the UC partners
2. Incorporating their own experience
3. Introducing and analysing the experience of past projects and related ongoing ones, where information is available and can be used
4. Analysing official reports and documents, including EU documents
5. Reviewing related international scientific literature
6. Collaborating with each other for commonalities

Use cases partners provided details based on section 4 of the deliverable (motivation, narration, apps/tools/services information, workflow, data, success criteria, among others).

Each Technical Group (TG from now on) with assigned partners performed the following:

- Rounds of **discussions** and/or sharing of **questionnaires** with the UCs (per group)

- Discussions around **"guiding" questions** having been developed beforehand with the collaboration of all the TG members
- **Sharing their findings** and results of **analysis** prior to every biweekly Task 2.2-2.3 telco for discussion and ensuring focus on the project's objectives
- Joint communication of some groups with the UCs for avoiding overlaps and syncing effort
- **Communication of the TG leaders** when required for synchronising efforts and ensuring alignment with SERRANO objectives and developments
- Editing **the respective section(s) in section 5** of the deliverable based on:
    - Use cases analysis (using the UCs descriptions, the discussions and/or the filled in questionnaires)
    - Own experience
    - Scientific Literature
    - Developments in other relevant projects

It should be noted though that the contribution to each topic was not necessarily limited to the partners initially identified, as presented in the table above. During the requirements collection and analysis process, broader discussions were held with the rest of the Consortium members. Their main purpose had been to ensure that a common vision is depicted in each requirement and different views and experiences are incorporated.

In practice, three rounds of discussions have been held between each UC provider and all seven Technical Groups. A specific description of each UC was the starting point for every round of discussions, which was followed by a series of questions posed by the TGs, mainly by the TG leaders, that served as questionnaires. TGs prepared their questions beforehand and the goal was to clarify as much as possible the UC and start identifying the requirements in a more detailed manner.

At the end of each round of discussions, the UC provider had to update the description of their respective UC following what had been discussed and agreed upon, so as to allow the TGs to define the requirements more in depth.

The rounds of discussions were open to the whole consortium and the idea was to make sure that the information was shared evenly to everyone and that possible overlapping was avoided and so the time was spent wisely. The minutes of each round of discussions were systematically provided the day after every discussion on the shared folder of the task. All partners continued exchanging information in an offline manner keeping the whole consortium informed.

Based on the feedback received through the aforementioned interactions two types of tables were designed and gradually fine-tuned (table for Applications/Tools and table for Services/Processes/Tasks) so that the main information regarding the most important parameters of each key area could be captured in a common and structured manner. Each UC filled in the applicable fields of these tables, in a number of iterations, as discussions proceeded and the topics were gradually clarified. However, it may be the case that some of

these parameters will be further detailed, elaborated or clarified in the upcoming works of the SERRANO project and as part of the second iteration.

The results of the above-mentioned interactions were presented during the second plenary meeting which was held on the 12th and 13th of April 2021. Following the plenary, the working mode shifted to *ad-hoc* teleconferences and offline discussions among the TGs and UCs, as required, focusing on the elaboration and finalisation of the requirements. This phase also included several rounds of contributions and updates in the requirements sections of this document, by the TGs.

4 Use cases overview and analysis

This section presents a detailed overview of the three SERRANO UCs.

4.1 Secure Storage (UC1)

In this UC, we demonstrate the envisioned capabilities of the SERRANO platform in the context of secure file sharing and storage. We plan to extend Chocolate Cloud's commercial SkyFlok[1] multi-cloud distributed storage service with new functionality related to edge data storage and accelerated edge processing.

SkyFlok is a secure file storage solution designed to tailor to the needs of both businesses and individuals. It enables users to easily and reliably store their data in the cloud. Compared to competing solutions, it gives customers full control over where data are stored by leveraging a multi-cloud approach. SkyFlok supports the three major cloud providers (Amazon, Google, Microsoft) as well as several smaller local solutions (OVH, Deutsche Telekom, Orange Business Cloud, IONOS, Exoscale, etc.). Through the use of EU-based providers, SkyFlok offers GDPR-compatible storage.

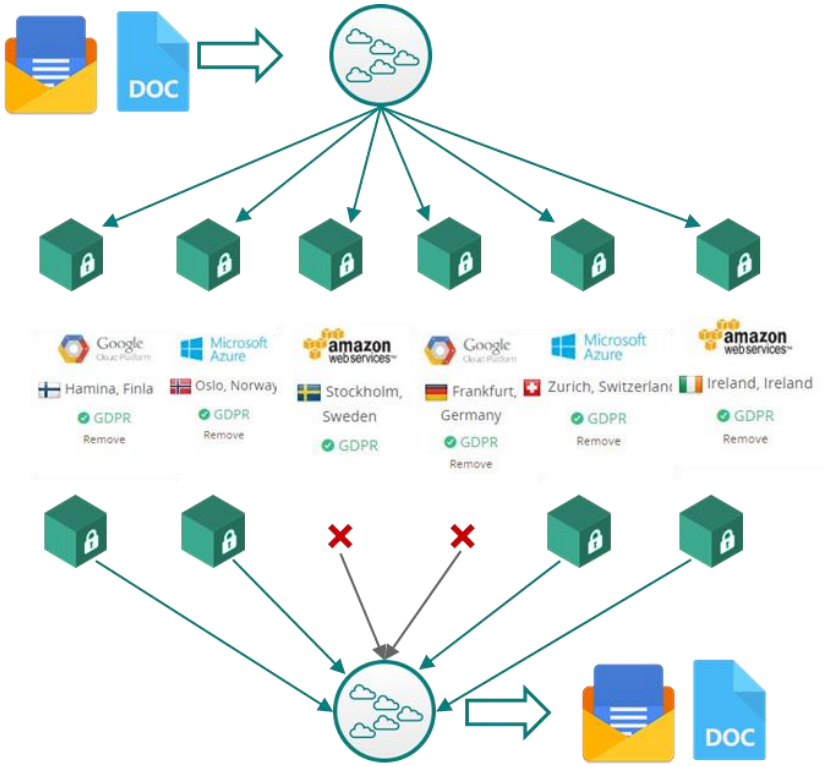


Figure 1. Overview of SkyFlok file storage and retrieval

[1] https://www.SkyFlok.com

Figure 1 shows a high-level view of how files are uploaded and retrieved. During upload, files are encrypted and erasure coded in the user's browser. The erasure coded fragments are then distributed across the selected public cloud locations. The Skyflok.com backend coordinates the process, acting as a sort of conductor. File data are transferred directly between the browser and the cloud location. Retrieving files is quite similar and involves downloading enough erasure coded fragments, then decoding and finally decrypting.

The whole process is coordinated with the use of a storage policy. A storage policy is like a recipe in that it defines what encryption and erasure coding scheme is used (as well as the schemes' parameters) and the chosen locations for the coded fragments. Each SkyFlok customer has their own custom storage policy, defined based on their particular requirements as well as on their physical location. An example is given in Section 4.1.6.

4.1.1 Use case motivation

Medium and large businesses (250+ employees) that are SkyFlok customers would like to extend their use of Chocolate Cloud's SkyFlok service. At the moment, SkyFlok works great for file-based collaboration and file sharing workflows as well as for data archival purposes.

However, given its fully cloud-based architecture, it lags behind in terms of latency compared to an on- premise storage solution. By moving data closer to the edge, download and upload latencies can be greatly improved. While many customers choose SkyFlok over competing solutions thanks to the privacy guarantees it offers, privacy concerns remain a major impediment to more wide-spread adoption of cloud storage in general. Due to legal requirements or internal policies, enterprises want strong guarantees that their data cannot be accessed by third parties, including the storage provider. Conventional cloud storage can only achieve this to a limited degree. Moving file encryption/decryption process on premises, under full control of the enterprise is key to providing these guarantees.

This UC demonstrates how the SERRANO platform enriches SkyFlok by offering edge storage locations to enterprise customers who already own the required hardware, or are willing to invest in on-premise storage infrastructure. These devices will show up in SkyFlok and will be used in combination with the traditional cloud-based storage services.

SkyFlok is currently offered as a web application, running in a browser. File encryption and erasure coding is performed there, and the resulting data fragments are transferred directly between the browser and the user-selected storage locations. While this browser-based interface works well for many small businesses, it does not satisfy the requirements that many larger enterprise customers may have. A REST API-based service that enables programmatic access and simple integration with existing IT infrastructure, is desired.

However, the implementation of this new feature faces a significant challenge: moving file processing, especially encryption to the cloud, would compromise the strong privacy guarantees of the SkyFlok service, since files would have to be transmitted unencrypted to the site that provides the encryption functionalities. Therefore, a more efficient solution that is enabled by the SERRANO platform, will be employed. As part of the project, we will

implement an on-premise storage gateway component that will be deployed on the customer's premises. It will perform encryption and erasure coding and will provide the aforementioned REST interface, while keeping the strong confidentiality guarantees of SkyFlok. The gateway will make use of the benefits offered by the other SERRANO platform services to accelerate file access. Finally, the SERRANO orchestration and monitoring services will play a key role in enabling efficient data storage and retrieval. These will make it possible to match user- initiated storage operations with the best possible storage resources and policies. In SERRANO, we introduce the concept of storage task to denote a group of storage operations that share requirements. This solution avoids the need to match every individual read and write to a policy. Examples for storage tasks are given in Section 4.1.6.

4.1.2 Use case narration

This UC focuses on providing secure and high-performance storage and sharing of files with lower latency than a purely cloud-based approach. We plan to achieve this goal by extending SkyFlok with on-premises edge devices that can act as storage locations. This component will be developed as part of WP3 and feature an industry-standard, S3-compatible API for easy integration with the existing Skyflok.com backend. Enterprise IT administrators will be given the choice in configuring how the storage locations are used through the definition of storage policies. Through this mechanism, they will be able to set how much data are stored in reliable cloud locations and how much on the less reliable but potentially faster edge devices.

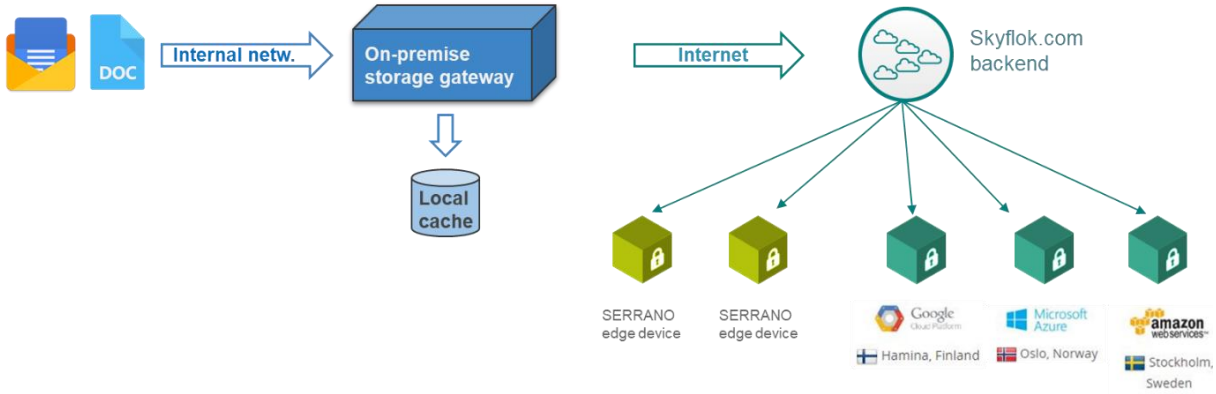


Figure 2: Secure Storage UC components

Furthermore, the UC will require a second edge component for larger enterprises that will be deployed on premise. It will act as a storage gateway, offering a simple public API that exposes SkyFlok's most important features and is crucial to achieve easy integration with existing applications. At its core, it will support the most common S3 operations on buckets and objects. The gateway will also perform encryption and erasure coding tasks, thus giving more control to the users and improving privacy. This has the additional benefit of releasing low-power client devices from the computational burden associated with encryption and encoding. Finally, a simple caching layer integrated inside this component will ensure that frequently used files can be accessed with significantly lower latency compared to data

distributed to SERRANO edge devices or cloud locations. Figure 2 gives an overview of the connections between the UC components.

From a high-level perspective, this UC will also involve the SERRANO platform's orchestrator service in two meaningful ways. First, it will rely on the orchestrator for the deployment of the services associated with the edge storage locations and the on-premises gateway. Second, the orchestrator will provide information on edge storage locations, regarding their status, availability, cost, latency etc. This will help ensure that each storage task is placed on the appropriate storage resources/site.

This second function enables the automatic creation of storage policies based on the formally described requirements of each storage task. Alternatively, an algorithm that selects the appropriate policy from a manually created list, can also be envisioned. There are many possibilities that we may explore as part of SERRANO with respect to this aspect, including the possibility of (automatically) migrating data between storage locations.

Finally, there might be benefits if the storage service is aware of the location of the user's application. If an appropriate erasure coding configuration and edge locations are used, some read tasks may be more efficiently served using nearby locations. This can be accomplished using an optional extension to the standard S3 interface that allows the orchestrator (or the client application with support from the orchestrator) to give hints on the ideal[2] storage locations to access when retrieving a file.

Given the on-premise gateway's role in accelerating file operations, it is a performance-critical component. In order to provide low-latency access to files for a large number of concurrent users, it will use hardware acceleration for encrypting TLS connections by leveraging MLNX Bluefield cards, when available. This will reduce some of the load on the CPU and may increase the number of supported concurrent connections. It might also be possible to accelerate encryption and erasure coding through the use of other SERRANO tools and services. This may lead to further CPU offloading as well as a reduction of processing time.

4.1.3 Stakeholders involved

Chocolate Cloud is behind the development and commercialization of the extended SkyFlok platform presented in this UC. The second important stakeholder group is made up of present and future SkyFlok customers. These customers potentially include cloud services providers moving to the edge, public entities managing personal and critical data, private companies managing critical information, e.g., content providers, telecommunication companies who wish to augment their products with cloud storage, automotive industry.

[2] The ideal set of locations could potentially be considered the set that should provide the lowest latency, lowest transfer cost or other metrics depending on the requirements of the application and the type of information available.

4.1.4 High-level requirements

At a high level, the resulting system in the Secure Storage UC must provide lower-than-cloud latencies when downloading data. This should be achieved through the use of edge devices.

SkyFlok already provides industry-leading privacy thanks to its proprietary use of network coding, distributing data to multiple cloud locations and proven encryption techniques. This is to be further enhanced for enterprises also deploying an on-premise gateway component by performing encryption and erasure coding on site.

SkyFlok offers GDPR-compliant storage. The UC should build on this and maintain compliance when possible.

Programmatic access to the storage service should be provided using a REST API that exposes at least the most important operations with the basic options present in the 'standard' S3 API.

Users must be given the possibility to define how and especially where their data are stored, using a set of storage policies. The creation or selection of storage policies for individual storage tasks should be automated. On the other hand, the system should also be able to operate transparently with regard to the choice of storage locations. User applications should not be required to explicitly state which storage locations to use.

4.1.5 Applications/tools involved

The Skyflok.com backend is an important part of this UC because of the services it offers. However, from a deployment perspective, it is an external component to SERRANO as it is hosted on public cloud infrastructure. It is not managed by the SERRANO platform services and as such its storage/computational/security requirements are not described in detail here.

Table 3: Skyflok.com backend application for UC1

Application / Tool	Skyflok.com backend	
Brief Description	The backend of Chocolate Cloud's multi-cloud storage service. It powers a turnkey service for secure, GDPR compliant, privacy-aware data storage and sharing.	
Storage Needs	**Capacity**	Deployed to public cloud infrastructure - N/A
	Availability	Deployed to public cloud infrastructure - N/A
	Latency	Deployed to public cloud infrastructure - N/A
Computational Needs	**Compute or memory bound?**	Deployed to public cloud infrastructure - N/A
Parallelization	**Is Parallelized (Yes /No /No but can be)**	No
	If yes, Parallelization Paradigm used	N/A

Communication Needs		**Latency**	Deployed to public cloud infrastructure - N/A
		Bandwidth	Deployed to public cloud infrastructure - N/A
		Latency or Bandwidth dominates?	Latency dominates
		Topology	Microservice-based
Acceptable Latency			Deployed to public cloud infrastructure - N/A
Data Information	**Input Data**	**Volume**	N/A
		Source(s)	Parameters passed to non-public REST API
		Description	The service is accessed through a series of non-public REST APIs. Input data consist of parameters related to file storage, user management, sharing, etc. File data do not pass through the Skyflok.com backend.
	Output Data	**Volume**	N/A
		Format (s)	Responses to the non-public REST API requests
		Description	Metadata related to file storage, sharing, etc.
	Intermediate Data	**Volume**	N/A
		Format(s)	N/A
Security Needs		**Are personal data processed? (Yes / No)**	Yes
Mean Runtime Duration [condition (if any) – duration OR formula if any]			Does not apply directly, each exposed endpoint has varying runtime duration, in some cases dependent on the input/output data size.
Energy Needs / Limitations		**% of energy resources consumed**	Deployed to public cloud infrastructure - N/A
Runtime temporal distribution		**IO**	Deployed to public cloud infrastructure - N/A
		Computational Part	Deployed to public cloud infrastructure - N/A
		Communication Part (if parallel)	Deployed to public cloud infrastructure - N/A
		Other (please specify)	N/A
Implementation Information		**Source code shareable**	No
		Programming Language used	Python
		External Libraries used	N/A
		Compilers used	N/A

The components (services/processes) of the SkyFlok backend are external to the project and as such are not described in this document.

The Secure Storage Service is implemented for SERRANO as part of WP3. While its naming might suggest that this is a monolithic service, in reality it is a collection of services that provide storage on cloud and edge locations. It consists of a single on-premises storage gateway and several SERRANO edge storage locations.

Table 4: Edge device storage service for UC1

Service / Process	SERRANO edge storage	**Application / Tool it is part of**	Secure Storage Service
Brief Description	Storage component developed/deployed to provide a storage location on edge devices. An off-the-shelf solution like OpenStack Swift or Ceph, will be most likely used.		
Storage Needs	**Capacity**	SERRANO edge storage location capacity requirements are dependent on concrete storage task requirements. For the demonstration of the UC, these requirements will be specified on a later date in such a way as to represent the usage patterns of users of a medium to large enterprise (estimated at a maximum of 50-100 GB). Since data are distributed across both edge and cloud locations, each will only need to provide a fraction of this volume.	
	Availability	Cloud locations are assumed to have high availability (>99.9%). As such, edge storage requirements depend on the storage policy in place. For example, if cloud locations store enough coded fragments to reconstruct the original data, lower availability is acceptable for SERRANO edge storage.	
	Latency	Latency offered by typical HDDs should be sufficient.	
Computational Needs	**Is service computationally or memory intensive?**	No	
Acceleration Needs	**Services execution % of application's/tool's total execution time**	It cannot be measured at this time, given that this component will be deployed/developed as part of SERRANO.	
	Input dataset range	Circa 10kB - 20MB	
	Quality of Service (QoS) requirements	N/A – does not require acceleration	
	Error tolerations	None	
	Error metric used	N/A	
Parallelization	**Is Parallelized (Yes /No /No but can be)**	No	

	If yes, Parallelization Paradigm used		N/A
Communication Needs	**Latency**		Low-latency (1ms – typical for LANs) access
	Bandwidth		At least 1Gb Ethernet connection is desired.
	Latency of Bandwidth dominates?		For small encoded fragments, a lower latency is usually desirable, for large fragments high bandwidth.
	Topology		Part of a star topology – the non-central node
Acceptable Latency			~1ms as typical of LANs
Data Information	**Input Data**	**Volume**	Dependant on storage task (number and size of coded fragments to write/read) and storage policy (percentage of fragments that are served by an individual edge location). Total volume over time is described in Storage Needs – Capacity row.
		Source(s)	On-premise storage gateway, users' browser
		Description	Encrypted, erasure coded fragments
	Output Data	**Volume**	Dependant on the volume of input data and the storage task (ratio of reads to writes).
		Format (s)	Same as input data
		Description	Same as input data
	Intermediate Data	**Volume**	N/A
		Format(s)	N/A
Security Needs	**Are personal data processed? (Yes / No)**		No. While the source of the input data may originally contain personal data, it reaches this service in an encrypted and erasure coded form.
Mean Runtime Duration [condition (if any) – duration OR formula if any]			Cannot be measured at this time, given that this component will be deployed/developed as part of SERRANO.
Energy Needs / Limitations	**% of energy resources consumed**		N/A - This component will not be run on battery-powered devices.
Algorithms Included			No particular algorithms included. This component simply provides access to data stored in a local file system.
Runtime temporal distribution	**IO**		80-90%
	Computational Part		10-20%
	Communication Part (if parallel)		N/A
	Other (please specify)		N/A

Implementation Information	**Source code shareable**	If developed in-house for SERRANO, then generally yes. If an off-the-shelf solution is deployed, then it will depend on the solution.
	Programming Language used	TBD
	External Libraries used	TBD
	Compilers used	TBD

Table 5: On-premise storage gateway service for UC1

Service / Process	On-premise storage gateway	**Application / Tool it is part of**	Secure Storage Service
Brief Description	On-premise storage component developed to act as a gateway for accessing the Skyflok.com backend as well as edge and cloud locations. It will also provide a cache to improve file access performance.		
Storage Needs	**Capacity**	Service can operate without storage if caching is disabled. If caching is enabled, the capacity is dependent on the number of users and their activity level (number and variety of files being accessed). A larger capacity (e.g. 10 GB) provides a better file access performance.	
	Availability	Low. The component should also function without caching capability but with reduced performance.	
	Latency	As provided ideally by an SSD - will also provide performance benefits in relying on an HDD.	
Computational Needs	**Is service computationally or memory intensive?**	Compute rather than memory bound. However, memory requirements do rise with larger file sizes.	
Acceleration Needs	**Services execution % of application's/tool's total execution time**	This cannot be measured at this time, given that the component will be developed as part of SERRANO. It will likely be dependent on input/output data size as well as on other factors.	
	Input dataset range	Circa 50kB – 1GB	
	Quality of Service (QoS) requirements	N/A	
	Error tolerations	None	
	Error metric used	N/A	
Parallelization	**Is Parallelized (Yes /No /No but can be)**	Erasure coding and potentially encryption could be parallelized.	

	If yes, Parallelization Paradigm used		Encoding with a linear block erasure code is in essence a matrix multiplication over a finite field. As such, several techniques could be used to parallelize it. Both encoding and decoding can be parallelized using multithreading by dividing files into separate chunks (this is performed for large files to limit memory consumption) and processing the chunks in parallel.
Communication Needs	**Latency**		Low-latency access (1ms – typical for LANs) to edge storage locations. Latency to public cloud locations depends on several factors, including physical distance.
	Bandwidth		Dependent on the storage task (number and size of files). As a guideline, bandwidth to edge locations should be equivalent to at least a 1GBit Ethernet connection, while bandwidth towards public cloud locations can be less than that.
	Latency of Bandwidth dominates?		Latency dominates for small files, bandwidth for large files.
	Topology		Star topology with this component being the central node that directly accesses the storage locations (non-central nodes).
Acceptable Latency			The component should provide lower latency than what SkyFlok currently offers through purely cloud locations (100-800ms access time for a coded fragment) when accessing data.
Data Information	**Input Data**	**Volume**	Total volume is dependent on the number of users and their storage habits. Current SkyFlok small enterprises typically use 10-30GB. Volume over time is as described in Table 4.
		Source(s)	User files
		Description	Mostly files created directly by users such as text documents, images, videos, etc.
	Output Data	**Volume**	Dependant on the volume of input data and the storage task (ratio of reads to writes).
		Format (s)	Same as input data
		Description	Same as input data
	Intermediate Data	**Volume**	N/A
		Format(s)	N/A
Security Needs	**Are personal data processed? (Yes / No)**		Yes, if we consider file names and contents as personal data.

Mean Runtime Duration [condition (if any) – duration OR formula if any]		Cannot be measured at this time, given that this component will be deployed/developed as part of SERRANO.
Energy Needs / Limitations	**% of energy resources consumed**	N/A - This component will not be run on battery-powered devices.
Algorithms Included		Erasure coding: encoding and decoding are both O(n^3) (default: Random Linear Network Coding) Encryption and decryption (default: AES-GCM).
Runtime temporal distribution	**IO**	Dominant, we estimate 40-90% when using the cache, negligible otherwise.
	Computational Part	Encoding/decoding should be no more than 10-20%.
	Communication Part (if parallel)	Unless using the cache, we estimate that the majority (70-80%) of time will likely be spent communicating with storage locations.
	Other (please specify)	The runtime temporal distribution detailed above is highly dependent (in addition to whether cache is used) on what storage locations are accessed i.e. fast edge locations mean a larger percentage of time is spent encoding and encrypting and smaller percentage communicating. The opposite is true when using slow cloud locations.
Implementation Information	**Source code shareable**	Yes
	Programming Language used	TBD, probably C++ or Python 3 or a hybrid.
	External Libraries used	TBD
	Compilers used	Clang or GCC for C++

4.1.6 Use case workflow

As part of the UC, we will implement/configure both the component that handles storage on the SERRANO edge devices and the on-premises storage gateway (denoted collectively as the Secure Storage Service). To demonstrate both, all data access will pass through the gateway's REST API and storage will be provided by both cloud and edge locations. Figure 3 shows an overview of which components are deployed on premise and which are deployed outside the company's infrastructure, in the cloud.

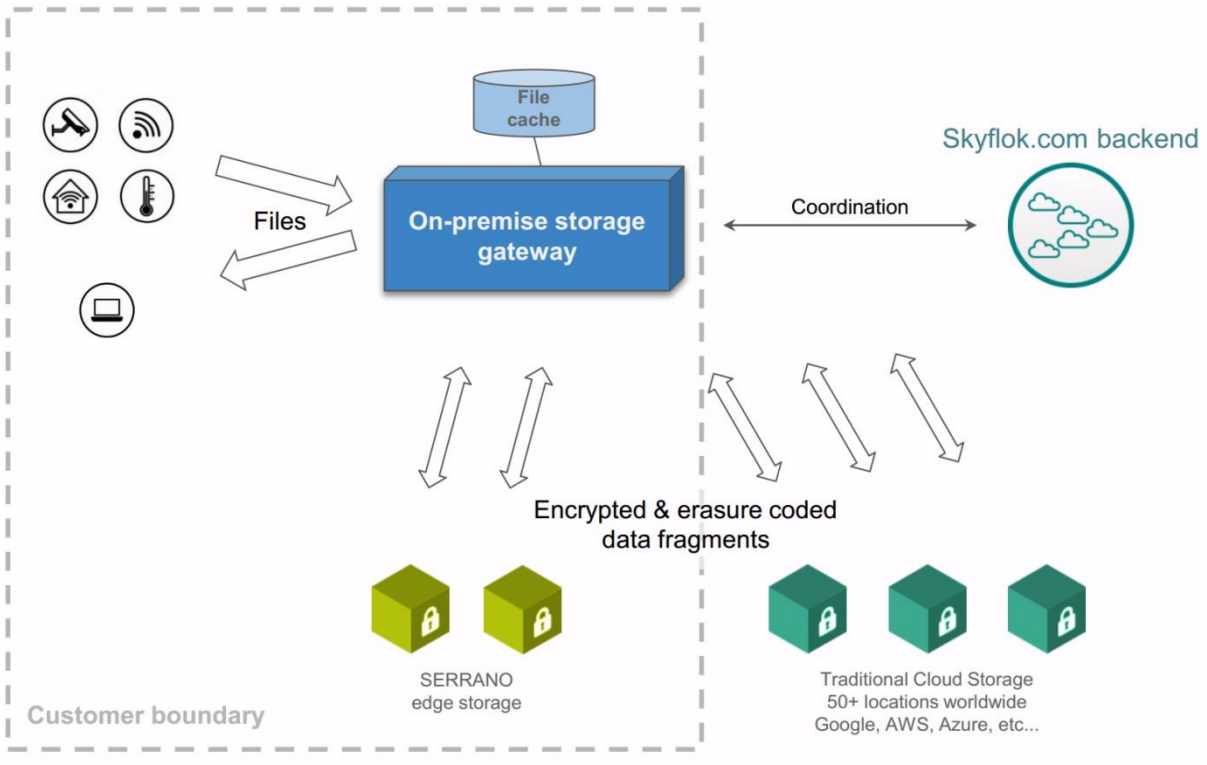


Figure 3: Deployment of Secure Storage UC: separation of cloud and on-premise edge components with actual flow of file data represented separately from coordination tasks

Workflow for storing data:

1. File sent from client device to On-premise storage gateway (Gateway from now on)
2. Gateway saves it to cache
3. Gateway encrypts it
4. Gateway encodes it using an erasure code
5. Gateway distributes coded fragments to cloud (Traditional Cloud Storage locations) and edge locations (SERRANO edge storage)

Workflow for retrieving data:

1. File requested by client device from the Gateway
2. Gateway checks cache and serves the file from cache if possible
3. Gateway retrieves coded fragments from best storage locations (combination of cloud and edge locations)
4. Gateway decodes fragments
5. Gateway decrypts file
6. Gateway saves file data to cache
7. Gateway returns file data to client device

Before users can start to interact with the system, system administrators configure the set of available cloud and edge locations as well as a set of encryption and erasure coding schemas.

This is followed up by the creation of storage policies, recipes on how storage tasks are served. The UC will go beyond this manual configuration step and will make it possible for each incoming storage task to either automatically select a storage policy from an existing list or create a new one.

The following is an example of a YAML configuration file for a storage policy.

```yaml
default_storage_policy:

 locations:
   bucket_ids: [14, 83, 132, 35, 82, 26]
   # Bucket IDs correspond to storage buckets
   # at cloud-based providers (e.g. AWS S3)
   # or edge storage locations

  encryption:
   enabled: True
   type: "AES-GCM-256"


  erasure_coding:
   type: "RLNC"
   symbols: 5
   generation_size_mb: 120
```

Different types of storage tasks have varied requirements in terms of reliability, availability, access performance and cost efficiency. As such, it is important that a wide and good selection of policies is available to suit the heterogeneous requirements. Alternatively, it should be simple and fast to define a new policy. Let us look at two examples on the relationship between storage task requirements and storage policies.

In the first example, let us consider the long-term storage of important documents as a storage task. For such files, it is imperative that storage locations with high reliability and low storage cost are used. Access costs, availability and latency are less of a concern. It might also be prudent to use a future-proof encryption policy and an erasure code that guarantees that data are not lost even in the highly unlikely case when one or more reliable storage locations are lost. Cloud storage locations serve this scenario quite well.

For the second storage task example, let us consider the storage of media files with the purpose of sharing them in a short timespan with the employees of the company (e.g., photos from the Christmas party). In this case, reliability and availability are less important. Conversely to the first example, the storage cost pays a smaller role than access cost as we expect the data to have a short lifespan but be accessed many times. Furthermore, to facilitate a good user experience, the storage locations should provide low latency access. Edge locations serve this scenario better then cloud locations.

It is a natural tendency that the requirements of a storage task change over time. As such, it might make sense in a few weeks' time to move the files described in the second example according to a storage policy that suits the first example's requirements.

4.1.7 Data involved

SkyFlok is first and foremost a secure file storage and sharing service. As such, the UC will cater to common file types (documents, images, video files, etc.) created by end-users. Chocolate Cloud will likely use anonymously gathered metadata from existing SkyFlok users to create a realistic data storage and access model. This would include a distribution of file sizes as well as read and write patterns and could be used in the demonstration and evaluation of the UC. Details are to be worked out as the project progresses.

4.1.8 Infrastructure to be integrated with SERRANO

The Skyflok.com backend is currently running on a public cloud. Given that is has certain customizations that have been designed specifically for this environment, Chocolate Cloud will maintain this deployment separate from the SERRANO platform.

We will strive to limit the number of changes we make to the backend, updating it with new features, when and if necessary, to ensure compatibility with the components developed as part of the SERRANO project.

4.1.9 UC Success criteria

In the frame of this deliverable and given the focus of Task 2.2, a number of criteria associated with the successful outcome of the Secure Storage UC and their relevant KPIs are briefly described in this section. These, along with the detailed UC description and analysis, will serve as introductory information for the more detailed formulation of the SERRANO KPIs and evaluation methodology to be provided as part of deliverable D6.2 and finalized in D6.7. Table 6 presents the technical success criteria of UC1.

Table 6: UC1 technical success criteria

Success criterion	KPI	Estimated target value
Successful integration of edge devices into the SkyFlok and SERRANO ecosystem with the goal of reducing latency.	Read and write latency reduction	Reduction by 10 - 50% with respect to existing cloud locations
Demonstration of client applications storing data in the edge/cloud infrastructure using S3 REST API.	Number of applications to be successfully demonstrated	20-100 instances of client applications using the service simultaneously
Demonstration of encoding and encryption algorithms running on the on-premises storage gateway.	Demonstration of encoding and encryption algorithms	Demonstration successful

Possibility of using hardware acceleration for encryption of TLS connections on the on-premise storage gateway.	Reduction in CPU load associated with encryption for TLS connections	Reduction of at least 10-20% with respect to no hardware acceleration
Transparent operation with regard to the choice of storage locations. User applications should not need to explicitly state storage locations to use.	Storage task execution without intervention from the user	Demonstration successful

4.2 Fintech (UC2)

This UC aims at demonstrating the cloud continuum capabilities of the SERRANO project within the context of investment portfolio management. The UC will be implemented through microservices which are deployed on the cloud as well as the edge and accelerated through FPGA or GPU accelerators.

4.2.1 Use case motivation

InbestMe provides automated and personalized investment management. Investment portfolios are composed of different financial instruments, such as shares, ETFs, funds, bonds, etc. Portfolios are personalized, based on the clients' investment capacity, risk tolerance, objectives and market conditions. Typically, investment portfolios are managed by an investment manager and several analysts. The process consists of continuously monitoring, analysing and adjusting so that the portfolios have optimal return and risk balance.

Specifically, the decisions about how to adjust the portfolios are complex and based on past, current and predicted future market conditions. The markets and the portfolios are simulated for *what if* conditions. The result of these operations are investment profiles. The investment profiles, like a template portfolio, specify what should be the composition of the investment portfolios for a specific time horizon. Subsequently, the actual investment portfolios are aligned to the investment profiles through a process called rebalancing. The rebalancing consists of comparing every investment portfolio with the investment profile that it corresponds to and creating trading orders (buy/sell) to make the actual investment portfolios have the same distribution as the investment profiles.

The entire investment management process is accomplished through in-house developed multiple applications that are executed in an orchestrated manner. The execution of these applications is highly saleable and depends on several factors, such as the number of assets managed, the amount of technical analysis performed, the number of accounts managed, the period for which the analysis is performed, etc.

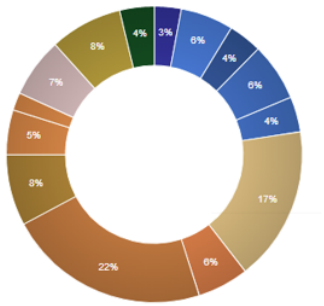
Through the FinTech UC, InbestMe identifies the huge potential for digitalization and innovation in the finance sector. For example, with increased digitalization an investment

manager would be able to manage up to 10.000 accounts instead of 10-100 accounts, as it is now, reduce expensive human made errors, improve client services, such as online account opening, instant approval, fast transactions, report generation and visualization. SERRANO's ability to determine automatically the optimal execution platform enables the intelligent and transparent deployment of computationally and data intensive applications into a diverse set of cloud, edge and HPC platforms. This capability will enable unprecedented innovation in investment management (higher return and lower risk), applying Machine Learning (ML) and Artificial Intelligence (AI), as well as high-performance Big Data processing and analysis.

4.2.2 Use case narration

The investment management is a continuous process of constructing investment portfolios composed of investment instruments such as shares, ETFs, funds, options, etc. Typically, an investment portfolio may be composed of tens to thousands of instruments.

Figure 4 shows an example InbestMe portfolio.



Asset type	Value	%	Ret.
Preffered Shares	431,86 €	2,94%	-3,3%
Aggregate Bonds EU	849,21 €	5,78%	1,6%
Long term Government Bonds USA	538,38 €	3,67%	-3,7%
Emerging Market $ Bonds (H€)	925,70 €	6,30%	10,2%
China Bonds $	564,85 €	3,85%	1,5%
Equity Europe	2.485,00 €	16,92%	23,6%
European Equities Dividend	824,07 €	5,61%	13,7%
US Equities (H€)	3.247,92 €	22,12%	29,4%
Equity Emerging Markets	1.141,24 €	7,77%	34,5%
Equity Japan	717,96 €	4,89%	17,5%
Asia Pacific Equity ex Japan	286,16 €	1,95%	9,7%
Real Estate Equity EU	955,15 €	6,51%	11,9%
Gold and Other Precious Materials	1.162,79 €	7,92%	4,7%
Cash	553,33 €	3,77%	0,0%
Total	*14.683,62 €*	*100 %*	

Figure 4: Example investment portfolio composed of different classes of investment instruments. Each class contains several other investment instruments.

Investment portfolios are evaluated through two important metrics, return and risk. Return is the money made or lost on an investment over some period. Typically return is expressed as a percentage derived from the ratio of profit to investment. At InbestMe the return is typically expressed in percentage for one year. Risk is the probability of gaining and losing money at a specific moment at present or in the future.

Figure 5 shows a chart, which displays the expected return, the risk and the actual passed return of the portfolio. The blue line represents the expected cumulative return in time. The fading blue area up and down from the blue line is the probability distribution of the return in time. The green line is the actual return of the portfolio in the past.

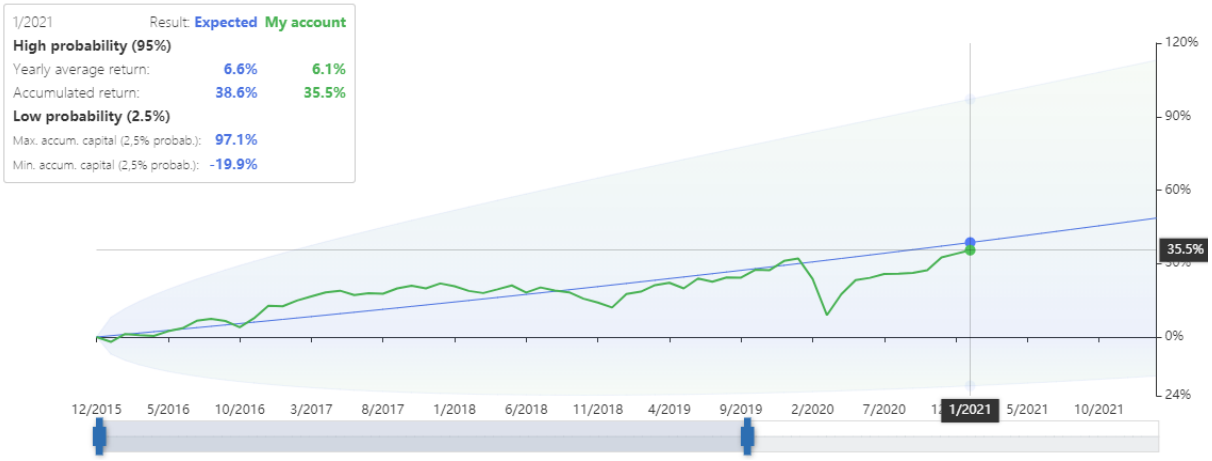


Figure 5: Example return and risk analysis for a specific portfolio compared to the actual return in time.

The entire investment management can be summarized in five steps which are being continuously repeated (Figure 6). The **market analysis** consists of analyzing and classifying the investment instruments. The analysis includes calculating many different technical indicators for one or more instruments, either individually or collectively. Example indicators which are calculated include return, risk, distribution, moving averages, volatility, Treynor[3] index, Sharpe ratio[4], Sortino ratio[5], Jensen's measure[6] , drawdown, underwater, Fast Fourier transform (FFT), and many others. Typically, these indicators are calculated for different time windows, for example 1, 2, 3 ... 60 months. Because of the high number of possible combinations that exist, the market analysis can be a very computationally intensive phase. Once the metrics are calculated, the results are further analysed focusing on what can be the best instrument or set of instruments to use in portfolios.

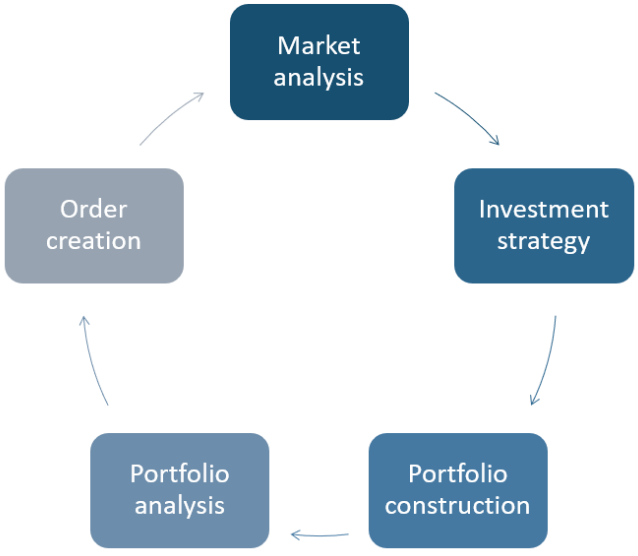


Figure 6: Investment management workflow summarized in five continuously repeating steps

[3] Treynor index. https://www.investopedia.com/terms/t/treynor-index.asp
[4] Sharpe ratio. https://www.investopedia.com/terms/s/sharperatio.asp
[5] Sortino ratio. https://www.investopedia.com/terms/s/sortinoratio.asp
[6] Jensen's measure. https://www.investopedia.com/terms/j/jensensmeasure.asp

During the **investment strategy** phase, the results from the market analysis are analysed with respect to the investor's own profile such as risk tolerance, investment capacity, objectives, horizon and other criteria. Example strategies are speculation for rapid and risky growth, long term but volatile growth or low risk slow growth for savings. During this phase, different investment instruments are classified as suitable or not suitable for different investment strategies.

The **portfolio constructions** combine the results from the market analysis and the investment strategy in building investment profiles. Investment profiles are template portfolios, which are composed of the instruments analysed during market analysis and shortlisted after applying investment strategies. Additionally, and most importantly, besides choosing what to buy or sell, this phase also determines what should be the distribution of each investment instrument in the investment profile.

Figure 7 shows an example of an investment profile with risk/return level of 8 out of 10. Higher profile means higher expected return but also higher risk to lose money due to higher volatility. During this phase, a lot of *what if* portfolio analysis and simulations are made in order to estimate the expected return and the probability distribution in time. The analysis can be very extensive and compute intensive due to the huge number of combinations that exists between investment instruments, their distributions and the investment horizon.

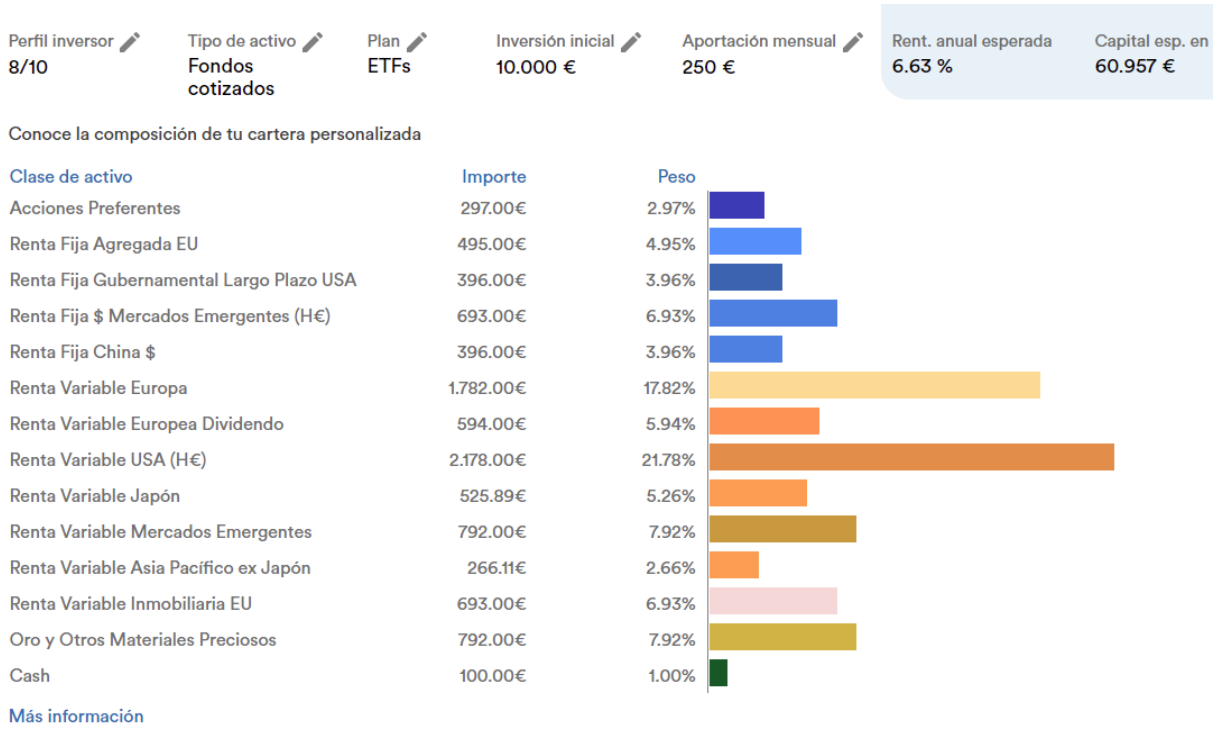


Figure 7: Example investment profile, a template portfolio with specific expected return, risk and probability distribution

After the portfolio construction, follows the **portfolio analysis**. Portfolio construction and analysis can be iterative. During this step, the actual portfolios as well as the investment portfolios are continuously analysed as to whether they are aligned and whether the actual portfolio distribution matches the expected portfolio distribution. Additionally, their

behaviour is monitored and compared to past activity, present and future market conditions. Therefore, this step takes as input the results from the market analysis and pass its results as input to the investment strategy as well as the portfolio construction.

Figure 8: Screenshot from executing BUY and SELL orders

The last step in the investment management is the **order creation** or also what InbestMe refers to as **rebalancing**. During the rebalancing, orders are created to make the actual distribution of the portfolio match the investment profile to which the portfolio is assigned. For example, the investment portfolio from Figure 4 is of a profile 8 and is slightly off when compared to the expected distribution level, since the cash of the portfolio is expected to be 1% but it is 3.77%. As a result, during the rebalancing, orders will be created and executed in order to align the actual portfolio with the expected distribution.

The SERRANO project will contribute to the investment management UC by providing a framework which will simplify the deployment, management, operation and monitoring of the application. Additionally, for the provision of the investment management as a service (SaaS) the UC will benefit from the secure storage extension which will keep the data of third parties secure via encryption, at all times. The UC will also explore the advantages of cloud-based acceleration of various computationally intensive operations.

4.2.3 Stakeholders involved

The stakeholders and actors involved in the FinTech UC are the following:

- INB the primary stakeholder who provides investment management service, develops a system for investment management.
- INB retail clients – the retail clients of InbestMe as the beneficiaries of receiving better investment management service, lower cost, higher returns and lower risk.
- INB business clients – the business clients of InbestMe who use the investment management services provided by INB to offer different or improved FinTech services.

- Brokers – brokers are INB partners through which INB manages the investment portfolios and trades.
- Market data providers – these are market data providers for both live (real-time) or offline historical data at various granularities.
- Payment processors – these are third party services that provide payment processing and transfer and storage of cash.
- Regulators – these are national and international regulators that monitor that INB, its clients and providers are compliant.

4.2.4 High-level requirements

The resulting system in the FinTech UC has the following high-level requirements:

- Container architecture – implementing and deploying the INB platform through containers.
- System deployment and management – deploy, manage, operate, update, migrate, replicate multiple independent INB system instances.
- Serverless cloud functions – on demand and lightweight deployment and execution of system services.
- Scalability – execute and orchestrate many fine-grain serverless tasks.
- Security – secure storage and execution of the INB services, with special focus on the business clients.

4.2.5 Applications/tools involved

Table 7: Dynamic Portfolio Optimization (DPO) application for UC2

Application / Tool	Dynamic Portfolio Optimization (DPO)	
Brief Description	Analyses and optimises the dynamic portfolios.	
Storage Needs	**Capacity**	100MB-1GB
	Availability	N/A (This tool does not require any sophisticated storage. Data are read once and executed in memory)
	Latency	N/A
Computational Needs	**Compute or memory bound?**	Compute bound
Parallelization	**Is Parallelized (Yes /No /No but can be)**	No, but can be
	If yes, Parallelization Paradigm used	N/A
Communication Needs	**Latency**	N/A
	Bandwidth	N/A
	Latency or Bandwidth dominates?	N/A

	Topology		N/A
Acceptable Latency			N/A
Data Information	**Input Data**	**Volume**	1MB-100MB
		Source(s)	Database, file, online feed
		Description	Market data per financial instrument such as historical prices and trade volumes. Account data from database.
	Output Data	**Volume**	2MB-10MB
		Format (s)	Excel, CSV, chart, database update
		Description	Excel report about the composition of the dynamic investment profiles. Back testing and comparison with the past. What-if analysis.
	Intermediate Data	**Volume**	N/A
		Format(s)	N/A
Security Needs	**Are personal data processed? (Yes / No)**		No
Mean Runtime Duration [condition (if any) – duration OR formula if any]			60 seconds to a few hours, depends on the analysed financial instruments, the technical indicators calculated and the number of the analysed portfolios. Usually, tens to hundreds of instruments are analysed.
Energy Needs / Limitations	**% of energy resources consumed**		N/A
Runtime temporal distribution	**IO**		N/A
	Computational Part		N/A
	Communication Part (if parallel)		N/A
	Other (please specify)		N/A
Implementation Information	**Source code shareable**		Can be shared on a need-to-know basis (in principle no)
	Programming Language used		Python, .NET Core
	External Libraries used		Yes
	Compilers used		N/A

Table 8: Automatic Reports Tool (ART) application for UC2

Application / Tool	Automatic Reports Tool (ART)	
Brief Description	This application generates various reports. The types of reports are business intelligence, compliance, informative.	
Storage Needs	**Capacity**	10MB-10GB

	Availability		N/A (This tool does not require any sophisticated storage. Data are read once and executed in memory)
	Latency		N/A
Computational Needs	**Compute or memory bound?**		Compute bound/memory bound depending on the report.
Parallelization	**Is Parallelized (Yes /No /No but can be)**		No but can be
	If yes, Parallelization Paradigm used		N/A
Communication Needs	**Latency**		N/A
	Bandwidth		N/A
	Latency or Bandwidth dominates?		N/A
	Topology		N/A
Acceptable Latency			N/A
Data Information	**Input Data**	**Volume**	1MB-100MB
		Source(s)	Database, file, online feed
		Description	Accounts data
	Output Data	**Volume**	10MB-1GB
		Format (s)	Excel, CSV, chart, database update, PDF
		Description	Various reports for clients, regulators, or business intelligence
	Intermediate Data	**Volume**	N/A
		Format(s)	N/A
Security Needs	**Are personal data processed? (Yes / No)**		Yes, depends on the report
Mean Runtime Duration [condition (if any) – duration OR formula if any]			60 seconds to a few hours, depending on the report
Energy Needs / Limitations	**% of energy resources consumed**		N/A
Runtime temporal distribution	**IO**		N/A
	Computational Part		N/A
	Communication Part (if parallel)		N/A
	Other (please specify)		N/A
Implementation Information	**Source code shareable**		Can be shared on a need-to-know basis (in principle no).
	Programming Language used		Python, .NET Core
	External Libraries used		Yes

	Compilers used	N/A

Table 9: Rebalancing application for UC2

Application / Tool	Rebalancing application		
Brief Description	This application performs rebalancing for the accounts. It creates and executes trading orders in order to ensure matching between the distribution of the investment portfolios and the expected distribution based on the corresponding investment profile associated with the account.		
Storage Needs	**Capacity**		10MB-10GB
	Availability		100% (needs full availability for creating and executing orders)
	Latency		N/A
Computational Needs	**Compute or memory bound?**		Compute bound
Parallelization	**Is Parallelized (Yes /No /No but can be)**		No but can be
	If yes, Parallelization Paradigm used		N/A
Communication Needs	**Latency**		N/A
	Bandwidth		N/A
	Latency or Bandwidth dominates?		N/A
	Topology		N/A
Acceptable Latency			N/A
Data Information	**Input Data**	**Volume**	1MB-100MB
		Source(s)	Database, file, online feed
		Description	Financial instruments market data, accounts data
	Output Data	**Volume**	1MB
		Format (s)	Database updates, Excel
		Description	Orders and the execution of the orders
	Intermediate Data	**Volume**	N/A
		Format(s)	N/A
Security Needs	**Are personal data processed? (Yes / No)**		Yes
Mean Runtime Duration [condition (if any) – duration OR formula if any]			60 seconds to a few hours, depending on the spread of the orders and the number of accounts rebalanced.
Energy Needs / Limitations	**% of energy resources consumed**		N/A

Runtime temporal distribution	**IO**	N/A
	Computational Part	N/A
	Communication Part (if parallel)	N/A
	Other (please specify)	N/A
Implementation Information	**Source code shareable**	No
	Programming Language used	.NET Core, Python
	External Libraries used	Yes
	Compilers used	N/A

Table 10: Market data access service for UC2

Service / Process	**Market data access**		**Application / Tool it is part of**	**DPO, ART**
Brief Description	This service provides access to market data such as the price of investment instruments. This can be obtained through data which is stored locally or through third party providers.			
Storage Needs	**Capacity**	10MB-10GB		
	Availability	100% (requires full availability to up-to-date market data in order to perform the analysis)		
	Latency	N/A		
Computational Needs	**Is service computationally or memory intensive?**	No		
Acceleration Needs	**Services execution % of application's/tool's total execution time**	N/A		
	Input dataset range	N/A		
	Quality of Service (QoS) requirements	N/A		
	Error tolerations	N/A		
	Error metric used	N/A		
Parallelization	**Is Parallelized (Yes /No /No but can be)**	No		
	If yes, Parallelization Paradigm used	N/A		
Communication	**Latency**	N/A		

Needs	**Bandwidth**		N/A
	Latency of Bandwidth dominates?		N/A
	Topology		N/A
Acceptable Latency			N/A
Data Information	**Input Data**	**Volume**	N/A (The input volume is insignificant, only a list of investment instruments)
		Source(s)	N/A
		Description	N/A
	Output Data	**Volume**	10MB-10GB
		Format (s)	CSV, database, Excel
		Description	Financial instrument prices.
	Intermediate Data	**Volume**	N/A
		Format(s)	N/A
Security Needs	**Are personal data processed? (Yes / No)**		No
Mean Runtime Duration [condition (if any) – duration OR formula if any]			60 seconds to a few hours, depending on the input data.
Energy Needs / Limitations	**% of energy resources consumed**		N/A
Algorithms Included			No algorithms are used. It just provides access to market data which might be available locally or through third-party providers.
Runtime temporal distribution	**IO**		99%
	Computational Part		0 %
	Communication Part (if parallel)		0 %
	Other (please specify)		N/A
Implementation Information	**Source code shareable**		Can be shared on a need-to-know basis.
	Programming Language used		.NET Core, python
	External Libraries used		Yes
	Compilers used		N/A

Table 11: Market data analysis service for UC2

Service / Process	**Market data analysis**		**Application / Tool it is part of**	**DPO**
Brief Description	The market data analysis service implements a set of algorithms for technical analysis of financial instruments, such as shares, ETFs, funds, bonds, etc. Each technical analysis is applied/performed independently.			
Storage Needs	**Capacity**	1MB-1GB per financial instrument.		
	Availability	100% (requires access to up-to-data market data)		
	Latency	N/A		
Computational Needs	**Is service computationally or memory intensive?**	No		
Acceleration Needs	**Services execution % of application's/tool's total execution time**	Constitutes a large percentage of the total application execution.		
	Input dataset range	1MB-1GB per-financial instrument, depending on the granularity.		
	Quality of Service (QoS) requirements	N/A		
	Error tolerations	Depending on the analysis performed, error tolerance can be OK. For example, for the denoise algorithm, transformation, etc.		
	Error metric used	N/A		
Parallelization	**Is Parallelized (Yes /No /No but can be)**	No but can be. Some of the algorithms can be parallelized as they perform matrix operations. Additionally, the execution can be distributed.		
	If yes, Parallelization Paradigm used	N/A		
Communication Needs	**Latency**	N/A		
	Bandwidth	N/A		
	Latency of Bandwidth dominates?	N/A		
	Topology	N/A		
Acceptable Latency		N/A		
Data Information	**Input Data**	**Volume**	1MB-1GB per financial instrument	
		Source(s)	Database, CSV, Excel	
		Description	Market data.	

	Output Data	Volume	1MB-1GB per financial instrument, depending on the algorithm.
		Format (s)	CSV, database, Excel
		Description	Financial instrument prices.
	Intermediate Data	Volume	N/A
		Format(s)	N/A
Security Needs	Are personal data processed? (Yes / No)		No
Mean Runtime Duration [condition (if any) – duration OR formula if any]			60 seconds to a few hours, depending on the input data.
Energy Needs / Limitations	% of energy resources consumed		N/A
Algorithms Included			Return, risk (value at risk), moving averages, volatility, Treynor measure, various normalization, Sharpe ratio, Sortino ratio, Jensen measure, Drawdown, Underwater, Fast Fourier, Wavelet, Savitzky-Golay and others.
Runtime temporal distribution	IO		N/A
	Computational Part		N/A
	Communication Part (if parallel)		N/A
	Other (please specify)		N/A
Implementation Information	Source code shareable		Can be shared on a need-to-know basis only if really required.
	Programming Language used		.NET Core, python
	External Libraries used		Yes
	Compilers used		N/A

Table 12: Forecasting service for UC2

Service / Process	Forecasting	Application / Tool it is part of	DPO
Brief Description	This service implements various forecasting algorithms for investment instruments and portfolios. Forecasting includes price prediction, trend prediction, probability distribution etc. It implements different sets of statistical, ML and AI based algorithms.		
Storage Needs	Capacity	1MB-1GB	

	Availability		100% (requires access to up-to-date market data)
	Latency		N/A
Computational Needs	**Is service computationally or memory intensive?**		Computationally intensive
Acceleration Needs	**Services execution % of application's/tool's total execution time**		Constitutes a large percentage of the total application execution.
	Input dataset range		1MB-1GB per-financial instrument depending on the granularity.
	Quality of Service (QoS) requirements		N/A
	Error tolerations		Depends on the analysis performed. In general ML and AI based algorithms are inherently inaccurate.
	Error metric used		N/A
Parallelization	**Is Parallelized (Yes /No /No but can be)**		Some ML/AI algorithms are already parallelized. The application execution can be distributed.
	If yes, Parallelization Paradigm used		N/A
Communication Needs	**Latency**		N/A
	Bandwidth		N/A
	Latency of Bandwidth dominates?		N/A
	Topology		N/A
Acceptable Latency			N/A
Data Information	**Input Data**	**Volume**	1MB-1GB per financial instrument.
		Source(s)	Database, CSV, Excel
		Description	Market data
	Output Data	**Volume**	1MB-1GB per financial instrument, depending on the algorithm.
		Format (s)	CSV, database, Excel
		Description	Financial instrument prices
	Intermediate Data	**Volume**	N/A
		Format(s)	N/A

Security Needs	**Are personal data processed? (Yes / No)**	No
Mean Runtime Duration [condition (if any) – duration OR formula if any]		60 seconds to a few hours, depending on the input data.
Energy Needs / Limitations	**% of energy resources consumed**	N/A
Algorithms Included		ARIMA, LSTM, regression, reinforcement learning, prophet, and many others.
Runtime temporal distribution	**IO**	N/A
	Computational Part	N/A
	Communication Part (if parallel)	N/A
	Other (please specify)	N/A
Implementation Information	**Source code shareable**	The code can be shared on a need-to-know basis.
	Programming Language used	.NET Core, python
	External Libraries used	Yes
	Compilers used	N/A

Table 13: Back testing service for UC2

Service / Process	**Back testing**	**Application / Tool it is part of**	**DPO**
Brief Description	This service performs back testing for portfolios and investment strategies. It also implements portfolio modelling.		
Storage Needs	**Capacity**	1MB-1GB depends on the number of financial instruments and time window for back testing.	
	Availability	100% (requires access to up-to-date market data)	
	Latency	N/A	
Computational Needs	**Is service computationally or memory intensive?**	Computationally intensive	
Acceleration Needs	**Services execution % of application's/tool's total execution time**	Constitutes a large percentage of the total application execution.	

	Input dataset range		1MB-1GB per-financial instrument, depending on the granularity
	Quality of Service (QoS) requirements		N/A
	Error tolerations		It can tolerate a certain error/accuracy margin. For example, it would be OK if return for the past is calculated to be 6.2 instead of 6.22.
	Error metric used		N/A
Parallelization	**Is Parallelized (Yes /No /No but can be)**		No but can be
	If yes, Parallelization Paradigm used		N/A
Communication Needs	**Latency**		N/A
	Bandwidth		N/A
	Latency of Bandwidth dominates?		N/A
	Topology		N/A
Acceptable Latency			N/A
Data Information	**Input Data**	**Volume**	N/A
		Source(s)	N/A
		Description	N/A
	Output Data	**Volume**	1MB-1GB per financial instrument.
		Format (s)	Database, CSV, Excel
		Description	Market data
	Intermediate Data	**Volume**	N/A
		Format(s)	N/A
Security Needs	**Are personal data processed? (Yes / No)**		No
Mean Runtime Duration [condition (if any) – duration OR formula if any]			60 seconds-1 hour, depending on the portfolios
Energy Needs / Limitations	**% of energy resources consumed**		N/A
Algorithms Included			Proprietary algorithms for portfolio modelling
Runtime temporal distribution	**IO**		N/A
	Computational Part		N/A

	Communication Part (if parallel)	N/A
	Other (please specify)	N/A
Implementation Information	**Source code shareable**	Code can be shared on a need-to-know basis.
	Programming Language used	.NET Core, python
	External Libraries used	Yes
	Compilers used	N/A

Table 14: Order creation service for UC2

Service / Process	**Order creation**	**Application / Tool it is part of**	**Rebalancing**
Brief Description	This service creates order in such a way that the investment portfolios match as much as possible the expected distributions. It implements linear and non-linear optimization algorithms, black-hole algorithm, genetic algorithms, etc.		
Storage Needs	**Capacity**	1MB (it requires minimal to no storage)	
	Availability	100% (requires access to up-to-date live market data)	
	Latency	N/A	
Computational Needs	**Is service computationally or memory intensive?**	Computationally intensive	
Acceleration Needs	**Services execution % of application's/tool's total execution time**	Constitutes a large percentage of the total application execution.	
	Input dataset range	Few kilobytes, only the list of portfolios and investment instruments.	
	Quality of Service (QoS) requirements	The order creation is time sensitive and should be performed as fast as possible. For example, 1000 orders in less than 1 second.	
	Error tolerations	No	
	Error metric used	N/A	
Parallelization	**Is Parallelized (Yes /No /No but can be)**	No but can be	
	If yes,	N/A	

	Parallelization Paradigm used		
Communication Needs	**Latency**		N/A
	Bandwidth		N/A
	Latency of Bandwidth dominates?		N/A
	Topology		N/A
Acceptable Latency			N/A
Data Information	**Input Data**	**Volume**	Insignificant
		Source(s)	Database
		Description	Market price, accounts data.
	Output Data	**Volume**	Insignificant
		Format (s)	Database, Excel
		Description	Order information
	Intermediate Data	**Volume**	N/A
		Format(s)	N/A
Security Needs	**Are personal data processed? (Yes / No)**		Yes
Mean Runtime Duration [condition (if any) – duration OR formula if any]			60-600 seconds, depending on how many accounts are rebalanced.
Energy Needs / Limitations	**% of energy resources consumed**		N/A
Algorithms Included			Linear optimization, non-linear optimization, black hole, genetic
Runtime temporal distribution	**IO**		N/A
	Computational Part		N/A
	Communication Part (if parallel)		N/A
	Other (please specify)		N/A
Implementation Information	**Source code shareable**		Can be shared on a need-to-know basis.
	Programming Language used		.NET Core, python
	External Libraries used		Yes
	Compilers used		N/A

4.2.6 Use case workflow

Figure 9 shows a typical workflow for portfolio optimization. Unlike Figure 6 that shows the general processes involved in investment management, portfolio optimization is a particular investment management process focused on optimizing portfolios with respect to specific metrics.

Typically, the key metrics which are optimized are: return, volatility, underwater and drawdown. As a reminder, return is how much in percentage a portfolio will increase in a specific time window; volatility is how often the portfolio varies and is expressed through variance and standard deviation; underwater is how many days the portfolio's value is below the money invested in it; drawdown is peak-to-through decline of a portfolio or, in other words, how much down a portfolio goes after its latest highest value.

Portfolio optimization is a very representative and demanding investment management process that can benefit from SERRANO. Portfolio optimization starts by getting the market data required for the analysis. The market data are analysed by applying sets of technical calculations. Subsequently, forecasting algorithms and various investment strategies are applied, in parallel, in the investment instruments. The output from the forecasting and the investment strategies is used for creating new investment profiles. The investment profiles are again analysed by applying forecasting methods and back testing. Finally, the investment profiles are rebalanced to match the expected distribution of the investment profiles.

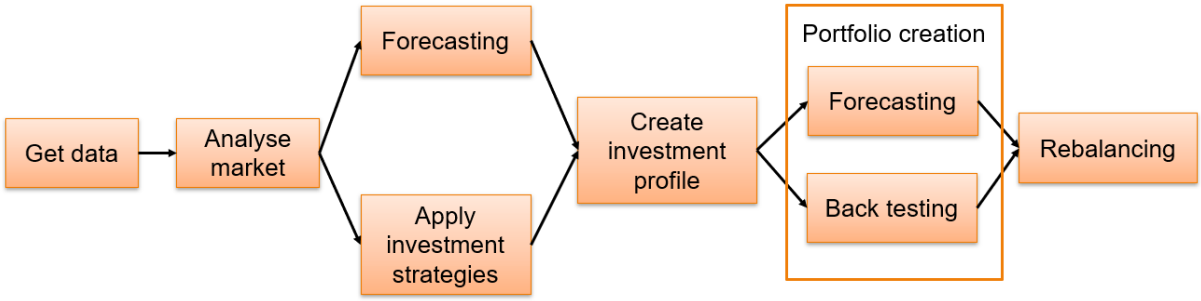


Figure 9: Portfolio optimization workflow

The market analysis is a computationally intensive operation that executes various algorithms for calculating technical indicators for a specific investment instrument. Some indicative technical indicators, which are calculated are moving averages, Sharpe ratio, Sortino ratio, volatility, etc. These technical indicators are applied on different periods of past historical prices of an asset.

Usually, during a market analysis, a big number of assets are analysed. The market analysis has a high degree of parallelism. For example, each specific calculation with configuration for a particular investment instrument (e.g. AMZN), time period and technical indicator can be executed in parallel. The algorithms for calculating technical indicators have a high degree of data parallelism and are good candidates for acceleration. For example, moving averages relate to the calculation of an average price in a particular time window, an operation that involves vector operations. The market analysis operation has a high degree of parallelism.

The analysis of each investment instrument, for a specific period and for a specific technical metric can be performed independently and in parallel. For example, for 100 investment instruments, 9 periods (1 month, 3 months, 6 months, 1 year, 2 years, 5 years, 10 years, 20 years, all) and 20 technical indicators, the parallelism is 100*9*20 = 1800 tasks. Also, the algorithms for calculating the technical indicators can be further accelerated in FPGA. InbestMe has extracted 10 kernels and provided these for further analysis and study on how they can be accelerated.

While market analysis is related with the past and historical values of an asset, forecasting is related with the future. Specifically, forecasting tries to predict the future price of an asset and with what probability that could happen. Forecasting is also a computationally intensive operation. InbestMe implements various algorithms based on statistical approaches as well as ML and AI. Forecasting is also a highly parallelisable operation. It is performed per asset, time window and algorithm (e.g. Sharpe ratio, volatility, Sortino ratio, normalization etc.). Additionally, the forecasting algorithms can be parallelised and accelerated.

The application of investment strategies is not so computationally intensive. It analyses the results obtained from market analysis and, depending on them, identifies candidate assets that can be used for a portfolio construction. For example, highly volatile assets are suitable for risky and long-term investment profiles and low volatility assets are suitable for low-risk profiles.

Many criteria are applied in identifying and grouping the assets in asset classes. A prominent computationally intensive algorithm that is run during this step is the k-means. Besides classifying the investment instruments, the application of the investment strategies also analyses what investment strategy can be suitable for a specific investment instrument or a specific class. For example, there are investment strategies that are suitable for instruments with high volatility that rapidly go up or down, there are investment strategies for instruments that have up/down trends, strategies for instruments that change trends from up to down and vice versa. Also, there are strategies that can be related to speculation (high-risk), perseverance (low-risk), wealth growth (long term non-speculative), etc. These strategies are analysed for the current market trends. The analysis of the strategies has good parallelisation potential where each strategy can be analysed for each instrument or a class of instruments.

The portfolio creation is a very computationally intensive process that uses the selected assets to construct portfolios. The result of this step is to assign weights to the assets or building distributions. During this process, the profiles are analysed for efficient frontier and a lot of simulations and *what if* analyses are performed about return and risk. Forecasting and back testing are part of this process and are applied repeatedly and in parallel for various combinations. The respective algorithms have a high degree of parallelism and are good candidates for acceleration.

Rebalancing is not a computationally intensive operation but has some real-time requirements. Specifically, orders should be created and executed as fast as possible, before

the asset prices change significantly. This process may be very difficult to demonstrate in a UC due to the complexity of placing orders, while it is impossible to simulate this step.

4.2.7 Data involved

The data involved in the FinTech UC are the following:

- Market data – live and historical data for financial instruments. Data include information such as price (open, low, high, close), with various granularity, volume, etc.
- Account data – data related to the account and the investment profile.

4.2.8 Infrastructure to be integrated with SERRANO

- Application for dynamic portfolio optimization
- Application for rebalancing
- Application for portfolio analysis

4.2.9 UC Success criteria

In the frame of this deliverable and given the focus of Task 2.2, a number of criteria associated with the successful outcome of the FinTech UC, and relevant KPIs are briefly described in this section. These, along with the detailed UC description and analysis, will serve as introductory information for the more detailed formulation of the SERRANO KPIs and evaluation methodology to be provided as part of deliverable D6.2 and finalized in D6.7. Table 15 presents the business success criteria and Table 16 the technical ones.

Table 15: UC2 business success criteria

Success criterion	KPI	Estimated target value
Reduce cloud costs by deploying a hybrid cloud infrastructure	Percentage of cloud costs reduction	Reduced by 50% or more
Increase portfolio performance (return/risk)	Percentage of portfolio performance increase	Increased by 10% or more
Improve accuracy of forecasting and prediction	Percentage of improvement	Improved by 10% or more

Table 16: UC2 technical success criteria

Success criterion	KPI	Estimated target value
Convert/adapt the INB applications and system to cloud-based containers	Conversion/adaptation to cloud-based containers	Conversion/adaptation successful

Deploy independent cloud-based instances of the INB system for third parties	Independent instances deployment	Deployment successful
Create real-time orders using live prices	Real-time orders creation	Real-time orders creation successful
Continuous market analysis	Rate of market analysis	100 financial assets per hour or more
Continuous portfolio analysis	Rate of portfolio analysis	100 portfolios per hour or more

4.3 Anomaly detection in manufacturing settings (UC3)

This UC aims at developing a system able to detect machine's ball-screws anomalies, by processing the amount of data generated in real-time by high-frequency sensors.

4.3.1 Use case motivation

Downtimes of failed devices in an industrial plant must be kept to a minimum to achieve high system availability. In the very competitive manufacturing world, getting the most out of the machine may be the difference between being competitive or not. Thus, and mainly after the irruption of the Industry 4.0 paradigm, lots of techniques and methods are being widely applied to meet a simple yet complex goal: keep the machine working most of the time.

Companies that manufacture expensive high added-value parts are very demanding in terms of machine availability and quality assurance. Predictive maintenance, remaining lifetime assessment and diagnosis of critical machine elements are state-of-the-art practices. However, some of the utilized techniques require from the machine to stop before performing the analysis. As a result, the various hardware components are idle most of the time, waiting for the analysis procedures to start, something that the manufacturing industries are keen to avoid.

The machine is usually stopped after some operational time and a diagnostic Computer Numerical Control (CNC) program is run. This CNC program orders the machine under inspection to perform some predefined movements at a controlled speed. The results from previous tests are then compared with the new ones, to compute the current status of a component and decide whether its condition is good enough to keep machining, a maintenance is needed, or the component has entered a critical state and further analysis is needed to decide for its replacement.

However, the high-frequency and high-accuracy sensors used for data acquisition generate high volumes of data which are difficult to process in real-time at the edge due to limited availability of resources. Introducing mechanisms that optimally orchestrate data and computationally demanding tasks in the edge, cloud and any other available device, can overcome this obstacle.

This UC proposes an approach where data analysis is performed continuously, while the hardware equipment keeps running most of the time and the state of the various independent components, along with the overall status is continuously reported. Moreover, the UC will focus on a single critical component, ball screws.

Ball screws are expensive machine components with a long useful lifetime but whose breakage implies stopping the machine for a large period of time. Having stock replacements could shorten that period, however, the high cost (€10000+) of a single ball screw makes this not an option for most companies. Furthermore, this component may have a 6+ week delivery time, while also considering technician personnel costs makes the scenario of a ball screw breakage, a real nightmare for the companies.

In contrast, if the client has early knowledge of a potential failure can request a replacement unit in advance and can even request the substitution to a local company, decreasing both costs and machine unavailability.

4.3.2 Use case narration

This UC aims at developing a system able to detect machine's ball-screws anomalies, by processing the amount of data generated by high-frequency sensors, in real-time.

This UC will leverage SERRANO platform to change the state-of-the-art approach and to perform the ball-screws' status assessment without stopping the machine. The challenges faced by this approach are:

- High data volume to be processed at the edge
- High data velocity
- Performing anomaly detection over a non-controlled machine
- Performing anomaly detection over highly variable equipment, such as CNC machines

The UC will deploy an application to analyze real-time signals coming from the ball-screw sensors and check for anomalies with the following two objectives:

1. Detect abnormal behaviours that may affect part quality
2. Predict imminent failures

4.3.2.1 Ball screws

A ball screw (Figure 10) is an expensive mechanical linear actuator that translates rotational motion to linear motion with little friction.

Figure 10: A ball screw

A machine tool may have 3 or 4 ball screws usually, depending on the model and the number of axis. The condition of ball screws may affect the precision of the machine, whereas a ball screw breaking may affect several other machine components and seriously incapacitate the machine.

4.3.2.2 Data acquisition

IDEKO will make available for the project a ball-screw testbed where real data can be generated. The testbed sensor distribution is depicted in Figure 11.

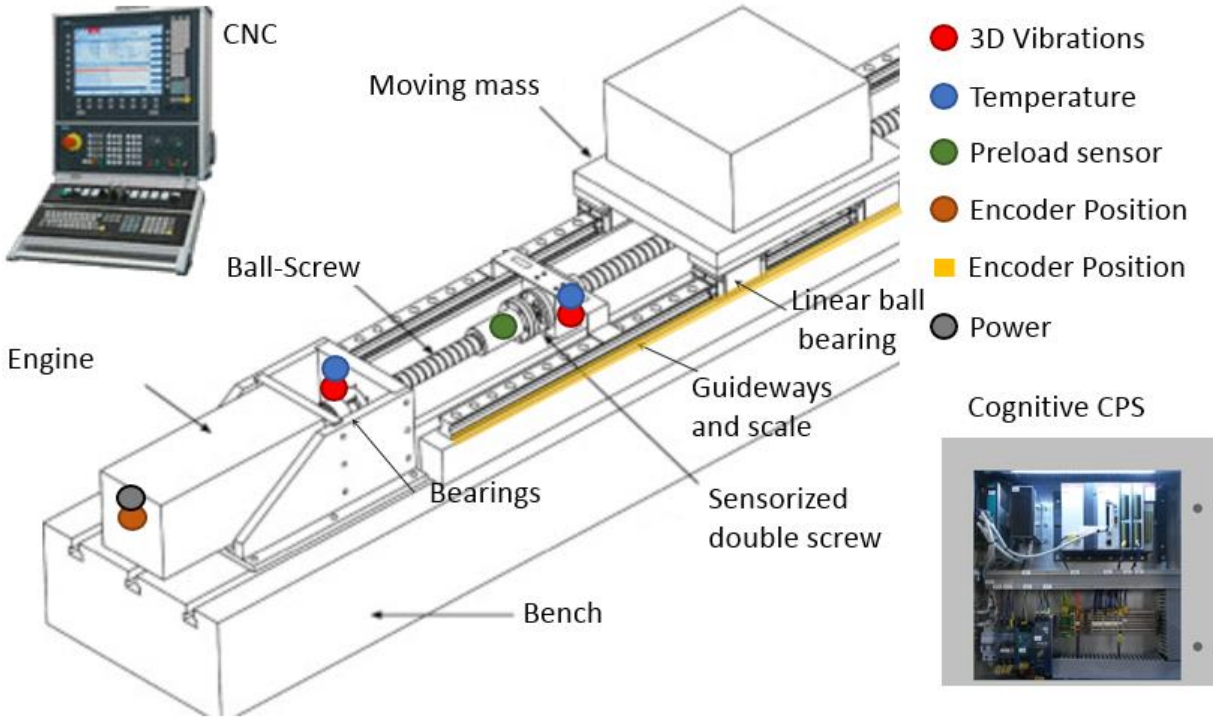


Figure 11: Sensor distribution for a single ball screw

Table 17 describes sensor distribution per ball screw with their corresponding technical details.

Table 17: Sensor distribution per ball screw

Type	Quantity	Data Acq. Freq.	Volume (MB/s)
Acceleration	2	25 kHz	37.2
Temperature	2	25 kHz	12.4
Load	1	500 Hz	0.124
Position	2	500 Hz	0.124
Electric current	1	500 Hz	0.124
		TOTAL	~50 MB/s

NOTE: the above table describes the sensor distribution for a single ball-screw on a single machine. A machine may have between 3 or 4 ball screws and a client has several machines in a shop floor.

4.3.2.3 Use case set-up

The UC will simulate a client with 3 machines, with 9 ball screws in total. Each ball screw will have the previous list of sensors attached and will generate data.

Figure 12: High level set-up

IDEKO owns a ball-screws testbed that generated sensor data. In the context of SERRANO, IDEKO will develop a machine ball screw simulator which, using the test bed data, will simulate a real machine in operating mode.

SERRANO will orchestrate the data flow and the processing performed, so that the client's requirements and UC objectives are met, while detecting anomalies.

4.3.3 Stakeholders involved

The manufacturers of machine tools and ball-screws suppliers present rules and guidelines for the prevention of the ball-screws anomalies, but these guidelines are very general and may not fit the particular way a client is using the machine. In these solutions, conservative conditions are suggested that are far from efficient. Another problem is that these solutions must be understood and executed by the different users involved in the machining processes, who usually do not know all the process data that define the problem.

The communication and involvement of the different stakeholders, including machine manufacturers, machine users and experts in data analysis and exploitation, is necessary in order to be able to achieve optimal results in the project.

4.3.4 High-level requirements

These are the high-level requirements for the UC:

- Perform near-real time analysis to avoid time deviations that may lead to analyse past and outdated data.
- Parallelize data analysis tasks to meet latency requirements.
- Automatically adjust data accuracy to meet latency requirements.
- Prioritize suspicious components' data over those functioning in expected conditions.
- Perform dynamic trading between performance and quality of service to meet application requirements.

4.3.5 Applications/tools involved

Table 18: Data Processing application for UC3

Application / Tool	Data Processing Application		
Brief Description	A layer for analysing data coming from machine sensors. This layer is composed by several Data Processor Applications that analyse the incoming data depending on their type. These applications perform analysis over acceleration data, temperature, load, position, etc.		
Storage Needs	**Capacity**		30GB estimated
	Availability		98% - 99% is acceptable
	Latency		Not known yet
Computational Needs	**Compute or memory bound?**		Compute bound
Parallelization	**Is Parallelized (Yes /No /No but can be)**		No but can be
	If yes, Parallelization Paradigm used		Independent services
Communication Needs	**Latency**		Depends on the data generation ratio
	Bandwidth		Not known yet
	Latency or Bandwidth dominates?		Latency dominates
	Topology		Star topology
Acceptable Latency			Depends on the data generation ratio
Data Information	**Input Data**	**Volume**	~50 MB/s per machine
		Source(s)	Machine sensors
		Description	Different nature: acceleration, temperature, etc.
	Output Data	**Volume**	Not known yet but presumably of a negligible volume.
		Format (s)	CSV

		Description	Some indicators of the component's health computed by the Data Processors of this layer. View workflow section.
	Intermediate Data	**Volume**	N/A
		Format(s)	N/A
Security Needs	**Are personal data processed? (Yes / No)**		No
Mean Runtime Duration [condition (if any) – duration OR formula if any]			N/A
Energy Needs / Limitations	**% of energy resources consumed**		N/A
Runtime temporal distribution	**IO**		Not known yet
	Computational Part		Not known yet
	Communication Part (if parallel)		N/A
	Other (please specify)		N/A
Implementation Information	**Source code shareable**		Yes – ongoing internal project, not available yet
	Programming Language used		Python
	External Libraries used		pandas, numpy, scipy
	Compilers used		N/A

The Data Processing application is composed by several Data Processor services that analyse the incoming data, depending on their type. These services perform analysis over acceleration data, temperature, load, position, etc., with the Acceleration Processor being the most demanding one in terms of data volume. The following table describes this processor service, since the respective UC will focus in the processing of acceleration data that we consider the most challenging scenario.

Table 19: Acceleration Processor service for UC3

Service / Process	**Acceleration Processor**	**Application / Tool it is part of**	Data Processing Layer
Brief Description	Analyses the acceleration data coming from the vibration sensors looking for anomalies. Will perform the FFT of the incoming data and then run several analyses over it.		
Storage Needs	**Capacity**	2-4GB	
	Availability	99%	
	Latency	N/A	
Computational Needs	**Is service computationally or memory intensive?**	Not tested	

Acceleration Needs	**Services execution % of application's/tool's total execution time**		Not tested
	Input dataset range		N/A
	Quality of Service (QoS) requirements		Adapt to the streaming generation speed
	Error tolerations		5% - 10% per single stream window is tolerable. Maybe more due the continuous data flow and the several opportunities to detect any anomaly.
	Error metric used		Not known yet
Parallelization	**Is Parallelized (Yes /No /No but can be)**		No but can be
	If yes, Parallelization Paradigm used		N/A
Communication Needs	**Latency**		Latency should adapt to the data stream window and the amount of parallel services running. Expected data generation speed: 37.2 MB/s
	Bandwidth		Not known yet
	Latency of Bandwidth dominates?		Latency dominates, probably
	Topology		N/A
Acceptable Latency			Subject to data stream generation speed
Data Information	**Input Data**	**Volume**	N/A
		Source(s)	IoT vibrations sensors
		Description	CSV-formatted data with several signals per file, corresponding to a short period of time with the raw data.
	Output Data	**Volume**	<1MB
		Format (s)	To be defined
		Description	Anomaly detection phase results
	Intermediate Data	**Volume**	N/A
		Format(s)	N/A
Security Needs	**Are personal data processed? (Yes / No)**		No
Mean Runtime Duration [condition (if any) – duration OR formula if any]			Not tested yet

Energy Needs / Limitations	**% of energy resources consumed**	N/A
Algorithms Included		FFT and others to be tested during the project.
Runtime temporal distribution	**IO**	Not tested
	Computational Part	Not tested
	Communication Part (if parallel)	Not tested
	Other (please specify)	N/A
Implementation Information	**Source code shareable**	Yes
	Programming Language used	Python
	External Libraries used	Pandas, Numpy
	Compilers used	N/A

4.3.6 Use case workflow

Figure 13 illustrates a high-level workflow of the UC for a single machine.

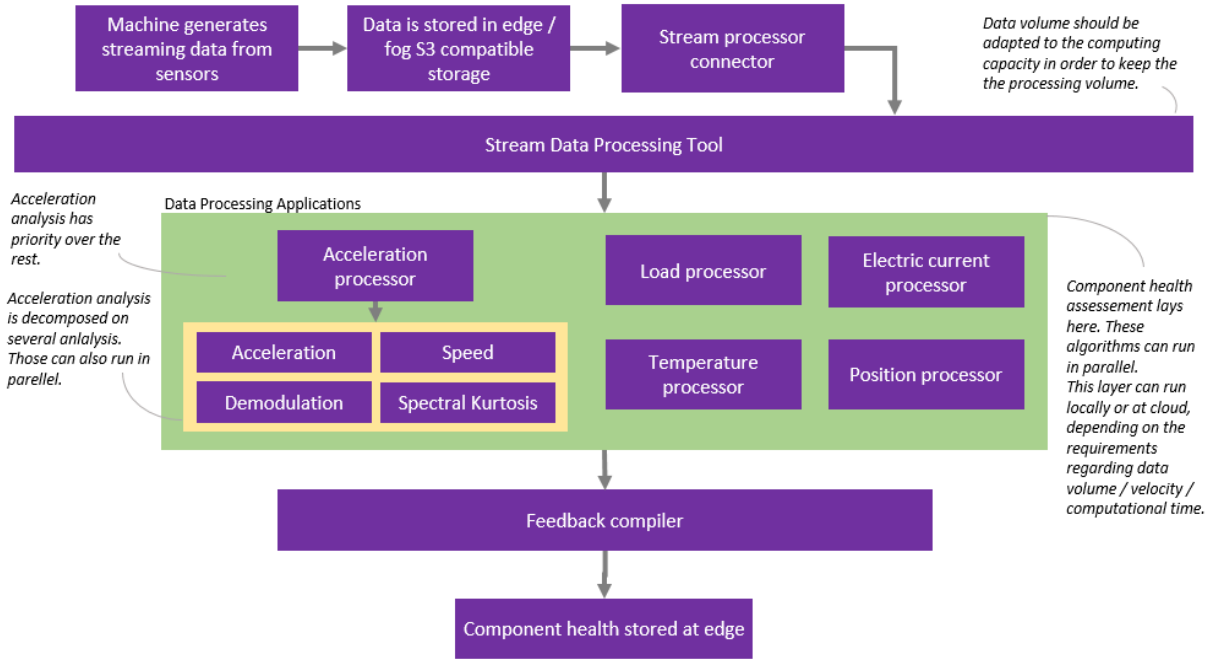


Figure 13: High-level UC workflow

When the machine starts machining, data are continuously/periodically generated. Data are being monitored by a mechanism running on the SmartBox attached to the machine, which is an edge device that monitors every single parameter of the machine (for more information please see section 4.3.8). When new data are found, the Stream Processor Connector sends it to a *Stream Data Processing Tool*. This could be a Kafka-like mechanism or any processing tool provided by SERRANO.

The *Stream Processor Connector* sends the data to the *Data Processing Application*, where the correspondent processor service is triggered depending on the nature of the data. Data coming from accelerometers is sent to the *Acceleration Processor*, while data coming from Temperature sensors will be send to the *Temperature Processor*. These *Processing Services* analyse the incoming data. As an example, the Acceleration Processor may use Fourier Transform to look for abnormal measurements over different frequencies; the *Load Processor* may use some anomaly detection algorithm to look for rapid fluctuations. In short, here is where system intelligence resides.

The following is a brief description of the *Data Processor services:*

- **Acceleration Processor**: This processor will perform data analysis over data coming from the vibration sensors. The first task this application will do, is to compute a Fourier Transform. After that, the application will perform four different types of analyses: speed, acceleration, demodulation and spectral kurtosis.
- **Load Processor**: This processor deals with the analysis of the ball screws under load. It will detect anomalies on the power at which the motor is working with respect to the motor maximum power.
- **Electric Current Processor**: This processor will detect electric current consumption anomalies that may indicate a malfunction of linear motors.
- **Temperature Processor**: This processor will detect unusual temperature variations and operating ranges that may indicate some component malfunction.
- **Position Processor**: This processor will detect errors between commanded and actual positions of the ball screws. It will also detect variations in inversion movements that may indicate, for example, elastic deformation.

Any *Data Processing Application* can run at edge, fog or cloud depending on the client's requirements. This UC may also utilize the secure storage layer of SERRANO, exhibited through UC1.

The results of the analysis performed by the *Processing Services* are sent to a Feedback compiler layer and then stored locally for further use.

This workflow corresponds to a scenario with a single machine. For a more realistic set-up, as stated in 4.3.2, the UC will be demonstrated with 3 machines working at the same time.

4.3.7 Data involved

The data involved in this UC depend on the nature of the sensor that generates it and the features of the data acquisition card on top of the sensor. Thus, some sensors are backed by high-frequency data acquisition cards while some others cannot provide those high frequencies.

Table 20 describes the data type for each type of sensor.

Table 20: Data type for each type of sensor

Sensor type	Format	Acq. Freq.	Additional details
Acceleration	MAT file	25 kHz	x, y, z
Temperature	MAT file	25 kHz	x, y, z
Load	CSV file	500 Hz	Axis load
Position	CSV file	500 Hz	Axis commanded speed, axis real speed, Axis commanded position, Axis real position
Electric current	CSV file	500 Hz	Current

4.3.8 Infrastructure to be integrated with SERRANO

The UC will integrate 3 infrastructure levels into SERRANO (Figure 14):

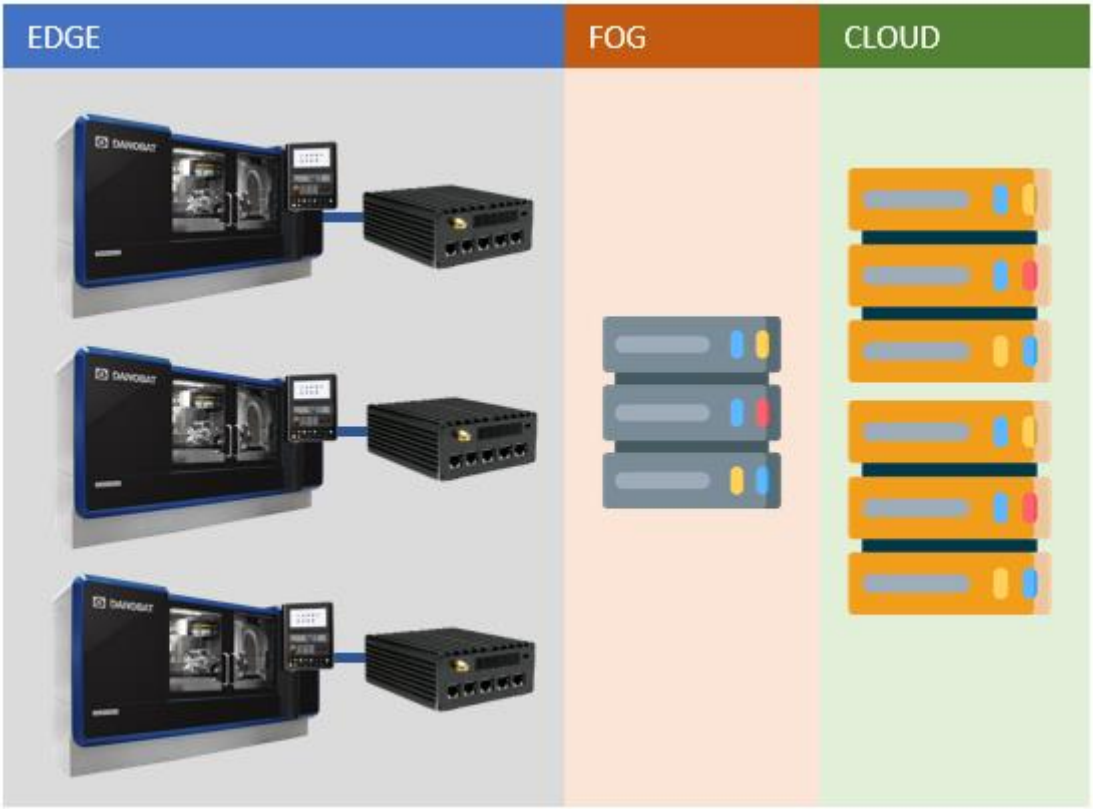


Figure 14: Infrastructure levels

Edge devices: All the machines are connected to an edge device (Smart Box). The Smart Box monitors every single parameter of the machine.

IDEKO works closely with Savvy Data Systems, a technological start-up focused on machine-monitoring and data analytics. In conjunction with them, IDEKO has developed the Smart Box, an industry-ready box for gathering machine data.

The Smart Box is a data gathering and data gateway (Debian Jessie host, 4 - 8GB RAM, 60GB HD, Celeron Quadcore 1.6GHz - 2.09GHz.), an industry PC for gathering machine data that

can connect to the most common CNC models (machines) and other data origins and sensors.

There are different models of boxes, with different price-ranges and computational power. Boxes are attached to machines with 1 to 1 relation. This leads to a scenario where some machines have more computational power available than others, just because the box attached is more powerful.

Fog: Clients may make available some fog servers at their facilities. The UC assumes the client is making available a single fog server. Fog Computing may improve latency when edge devices do not have the required computational capacity.

Cloud: Enables computation for scenarios where cloud fallback is needed.

4.3.9 UC Success criteria

In the frame of this deliverable and given the focus of Task 2.2, a number of criteria associated with the successful outcome of the Anomaly detection UC and their relevant KPIs, are briefly described in this section. These, along with the detailed UC description and analysis, will serve as introductory information for the more detailed formulation of the SERRANO KPIs and evaluation methodology to be provided as part of deliverable D6.2 and finalized in D6.7. presents the business success criteria and the technical ones.

Table 21: UC3 business success criteria

Success criterion	KPI	Estimated target value
Transition from on-demand to real time data analysis for anomaly detection to reduce machine stoppages	Transition to real-time anomaly detection	Transition successful
Anticipate failures comparing to current state-of-the-art techniques	Anticipation of failures	Anticipation successful
Reduce the Mean Time to Repair (MMTR[7])	MMTR reduction	Reduce by 50% or more
Increase of anomaly detection accuracy (avoiding nuisance alerts and false positives/negatives)	Anomaly detection accuracy increase	Increase by 35% or more

Table 22: UC3 technical success criteria

Success criterion	KPI	Estimated target value
Being able to quickly process large amounts of streaming data	Rate of streaming data processing	200MB/s or more

[7] Mean Time To Repair: https://en.wikipedia.org/wiki/Mean_time_to_repair

Increase of machine availability	Increased availability of machine	2% or more, measured in a monthly basis

5 Requirements Analysis

This section presents the SERRANO requirements as extracted, based on the thorough analysis of the UCs. The description of the requirements is presented in a structured manner to be efficiently processed and transformed into architectural decisions. Each requirement is annotated with a descriptive ID, encapsulating information about whether being a Functional (F) or Non-Functional (NF) requirement, the initials of the key area of requirements that it belongs to and a numbering value. Also, given that the project follows an iterative process, the priority of each requirement is captured (core, essential or desired), allowing for its effective mapping in the development cycles.

Further information, such as the stakeholders involved, the description of the requirement, its rationale and/or goal as well as its dependencies with other requirements, are also recorded. Finally, each requirement includes a series of acceptance measures, serving as a basis for evaluating its achievement through the project's developments. Based on the SERRANO goals and technical objectives, the requirements are grouped into 7 (seven) categories, including general requirements, physical resources (incl. edge/fog, cloud and HPC computational, storage and network resources) requirements, infrastructure abstraction requirements, secure infrastructure requirements, resource orchestration and service assurance requirements, service orchestration, integration and platform development requirements.

5.1 General and Non-functional Requirements

5.1.1 Purpose

This section presents the general functional requirements for the SERRANO platform and the overall non-functional requirements. On the one hand, the functional requirements are indispensable parts of the designed solution. Typically, they describe the desired functionality that the designed hardware and software platforms must offer to the end users.

On the other hand, the non-functional requirements express quality requirements that the components of the implemented platform should satisfy. For the non-functional requirements, SERRANO adopts the widely accepted ISO/IEC 25010 [3] standard model of quality characteristics that determines which quality characteristics will be considered when evaluating the properties of a software product. The quality of a system is the degree to which the system satisfies its various stakeholders' stated and implied needs and thus provides value. The ISO/IEC 25010:2017 quality model classifies software quality in a structured set of characteristics and sub-characteristics.

5.1.2 General Functional Requirements Analysis

Table 23: Analysis of general requirements

Requirement ID:	F_GR.1	**Priority:**	Core
Requirement Title:	Provide a unified view of cloud, edge and HPC resources	**Stakeholders Involved:**	All
Description:	SERRANO should combine transparently the heterogeneous resources from multiple individual edge, cloud and HPC infrastructures into a single borderless infrastructure. The platform should intelligently leverage the diverse resources to execute different kinds of dynamic and highly demanding applications.		
Rationale / Goal:	The unification of edge, cloud and HPC resources, based on an automated and self-managed approach, will keep data and processing of extreme low latency services close to where they are produced, while computationally- and data-intensive applications will be intelligently assigned into a diverse set of cloud and HPC platforms.		
Acceptance Measures:	Orchestration mechanisms are aware of the available resources at the individual edge, cloud and HPC infrastructures.		
	Cloud-native application deployment over true computing continuum.		
Dependencies:	Abstraction models for applications and infrastructure resources (F_SOR.2, F_SOR.3).		
	SERRANO service, resource orchestration and telemetry mechanisms (F_SOR.1, F_ROSAR.1, F_NCTFR.1).		

Requirement ID:	F_GR.2	**Priority:**	Core
Requirement Title:	Enable an intent-driven paradigm of federated infrastructures	**Stakeholders Involved:**	All
Description:	The SERRANO platform should receive a high-level description of the application requirements and deployments constraints and produce a valid infrastructure-aware deployment description. This means that the platform should analyse and map the initial infrastructure agnostic descriptions to specific service goals. Hence, applications will experience transparent, adaptive and efficient access to heterogeneous processing and storage resources.		
Rationale / Goal:	By providing end-users to express their preferences and applications' runtime constraints at high-level, SERRANO should be able to intelligently, based on AI/ML techniques, decompose them into infrastructure-specific resource and performance objectives for the cognitive orchestration mechanisms.		
Acceptance Measures:	Provide abstractions for cloud-native application deployment over seamlessly integrated heterogeneous computing resources.		
	Fulfil high-level deployment constraints and application QoS requirements.		
Dependencies:	Abstraction models for applications and infrastructure resources (F_SOR.2,		

	F_SOR.3).
	SERRANO service and resource orchestration mechanisms (F_SOR.1, F_ROSAR.1).

Requirement ID:	F_GR.3	**Priority:**	Core
Requirement Title:	Support transparent application deployment	**Stakeholders Involved:**	All
Description:	SERRANO should provide the necessary mechanisms to support resource interoperability and transparent deployment of applications' workload and data across the entire computing continuum. The appropriate abstractions and common description models along with the support for secure and transparent execution for containerized workloads will be the main pillars for enabling that functionality.		
Rationale / Goal:	The transparent application deployment over highly diverse resources will enable SERRANO to cater for application constraints, while calibrating the configuration of available resources. Moreover, developers will be able to focus solely on business logic for their applications.		
Acceptance Measures:	Deployment of workload across the SERRANO computing continuum.		
	Support develop once, deploy everywhere paradigm.		
Dependencies:	SERRANO-enabled hardware and software resources.		
	Abstraction models for applications and resources (F_SOR.2, F_SOR.3).		

Requirement ID:	F_GR.4	**Priority:**	Core
Requirement Title:	Encompass an autonomous and continuous control loop.	**Stakeholders Involved:**	All
Description:	The SERRANO platform should provide intelligent and autonomous orchestration mechanisms that will provide automatic and continuous adaptations. To this end, SERRANO should include the necessary mechanisms to sense (detect what is happening), discern (interpret senses), infer (understand implications), decide (choose a course of action), and act (take action), over an infinite time horizon control loop.		
Rationale / Goal:	These mechanisms will drive the automated and cognitive application orchestration within the SERRANO platform. The overall goal is to enhance the quality of the provided services by supporting near real-time and zero-touch adaptability.		
Acceptance Measures:	Support monitoring of performance related parameters and events.		
	Detect critical situations and proactively trigger re-optimization actions.		
Dependencies:	SERRANO cloud and network telemetry framework (F_NCTFR.1, F_NCTFR.3, F_NCTFR.8).		

	Resource orchestration and service assurance mechanisms (F_SOR.1, F_ROSAR.1, F_ROSAR.3).

Requirement ID:	F_GR.5	**Priority:**	Core
Requirement Title:	Support safety-critical, latency-sensitive and data-intensive applications	**Stakeholders Involved:**	All
Description:	The SERRANO platform should provide a novel ecosystem of cloud-based technologies, spanning from specialized hardware resources up to software toolsets that will enable application-specific service instantiation and optimal customizations for the UCs in cloud storage services, fintech and manufacturing.		
Rationale / Goal:	Demonstrate the advanced and innovative capabilities of the SERRANO platform in addressing the challenges that SERRANO UCs' safety-critical and high-performance demanding digital services pose.		
Acceptance Measures:	Successful demonstration of improved operation for three UCs by leveraging SERRANO innovations.		
	Provide transparent deployment, cognitive orchestration, service assurance, enhanced security, and hardware acceleration to the three UCs.		
Dependencies:	The SERRANO platform orchestration mechanisms and algorithms (F_ROSAR.1, F_ROSAR.2, F_ROSAR.3, F_ROSAR.4).		
	SERRANO-enabled hardware and software resources.		

Requirement ID:	F_GR.6	**Priority:**	Core
Requirement Title:	Expose well-defined APIs through SERRANO SDK	**Stakeholders Involved:**	All
Description:	The success and wide acceptance of any platform largely depend on the functionality and services offered to end-users and application developers. SERRANO should deliver a holistic Service Development Kit (SDK) that should include a set of well-defined APIs. The SDK will enable a high-level interaction with the core platform components (i.e. Service and Resource Orchestrator, Telemetry, Service Assurance, Secure Storage). Furthermore, it will facilitate end-users to create and deploy their cloud-native applications, without handling the peculiarities stemming from the management and interaction with the individual resources.		
Rationale / Goal:	SERRANO SDK will facilitate the development of UCs for the final demonstrations. The SDK will help developers build, deploy and manage their novel applications over the SERRANO platform. Moreover, it will enable the implementation of future extensions and deployment of diverse cloud-native applications.		
Acceptance	No hidden internal APIs.		

Measures:	Expose methods to interact with core platform components (i.e. Service and Resource Orchestrator, Telemetry, Service Assurance, Secure Storage)
Dependencies:	Abstraction models for applications and resources (F_SOR.2, F_SOR.3).
	Exposed APIs by the integrated platforms and services.

Requirement ID:	F_GR.7	**Priority:**	Desired
Requirement Title:	Support of additional application areas	**Stakeholders Involved:**	All
Description:	SERRANO includes three well-defined UCs that pose very demanding and diverse requirements and characteristics. Their extensive requirement analysis will provide the required functional requirements for the overall architecture. Nonetheless, SERRANO platform orchestration, management and execution mechanisms need to be generic enough to enable the cognitive and transparent deployment of many types of applications and services over the unified heterogeneous resources.		
Rationale / Goal:	The support of additional application areas will highlight the capabilities and advantages of the designed architecture. Moreover, it will ensure the wider exploitation of the SERRANO ecosystem, as well as all innovation features developed within the project.		
Acceptance Measures:	Support applications from various application domains.		
	Minimal required changes for applications that comply with cloud-native principles and standard interfaces.		
Dependencies:	SERRANO Service Development Kit (F_GR.7, F_ROSAR.1).		
	Orchestration platforms and runtime mechanisms at federated infrastructures.		

5.1.3 Overall Non-functional Requirements Analysis

Next, we present the relationship between the non-functional requirement categories of ISO/IEC 25010 and all the functional requirements. Moreover, for each non-functional requirement, a brief description of the SERRANO platform ecosystem's relevant characteristics is also provided.

Table 24: Analysis of overall non-functional requirements

Requirement ID:	NF_GR.1	**Priority:**	Core
Requirement Title:	Performance Efficiency	**Stakeholders Involved:**	All
Description:	This characteristic represents the performance relative to the number of resources used under stated conditions. This characteristic is composed of the following sub-characteristics: • **Time behaviour** - Degree to which the response, processing times and throughput rates of a product or system, when performing its functions,		

	meet the requirements. • **Resource utilization** - Degree to which the amounts and types of resources used by a product or system, when performing its functions, meet the requirements. • **Capacity** - Degree to which the maximum limits of a product or system parameter meet requirements.
Rationale / Goal:	SERRANO platform integrates the project's developed technologies over edge, cloud, and HPC infrastructures, coupling them with intelligent mechanisms to optimize resource utilization. SERRANO will have the ability to cater to application and user constraints while calibrating the configuration of the available resources.
Related Functional Requirements	***F_GR.1*** - Provide a unified view of cloud, edge and HPC resources. ***F_GR.4*** - Encompass an autonomous and continuous control loop. ***F_ECHAR.4*** -Application/System Quality-of-Service requirements. ***F_ECHAR.7*** – Accelerated kernels integration. ***F_ECHAR.9*** – Implement accelerate kernels as vAccel plugins. ***NF_SIR.2*** – Secure storage service should provide low latency access. ***F_SIR.3*** – Automatic creation/selection of storage policies. ***F_NCTFR.3*** – Estimate inter-site and intra-site network characteristics. ***F_NCTFR.4*** – Autonomously monitor cloud-native applications over heterogeneous distributed resources. ***F_NCTFR.8*** – Ability to centralize the monitoring information in a common view or place. ***F_ROSAR.4*** – Support cognitive and multi-object resource allocation algorithms. ***F_ROSAR.10*** - Access to the real time monitoring stream bus. ***F_ROSAR.11*** – Transprecise adaptation ML methods. ***F_ROSAR.12*** – Transprecise hyper-parameter optimization methods. ***F_SOR.1*** – Application/Tool specification. ***F_SOR.4*** – Task orchestration. ***F_SOR.7*** – Orchestration as a Service.

Requirement ID:	NF_GR.2	**Priority:**	Core
Requirement Title:	Functional Suitability	**Stakeholders Involved:**	All
Description:	This characteristic represents the degree to which a product or system provides functions that meet stated and implied needs when used under specified conditions. This characteristic is composed of the following sub-characteristics: • **Functional completeness** - Degree to which the set of functions covers all the specified tasks and user objectives. • **Functional correctness** - Degree to which a product or system provides		

	the correct results with the needed degree of precision. • **Functional appropriateness** - Degree to which the functions facilitate the accomplishment of specified tasks and objectives**.**
Rationale / Goal:	The SERRANO platform will provide an abstraction layer that automates the application deployment by continuously optimize the allocation and configuration of the available heterogeneous resources based on high-level requirements. Moreover, it will support the autonomous adaptation and management of the deployed services and resources to ensure that applications perform as intended.
Related Functional Requirements	***F_GR.2*** - Enable an intent-driven paradigm of federated infrastructures. ***F_GR.4*** – Encompass an autonomous and continuous control loop. ***F_ECHAR.4*** – Application/System Quality-of-Service requirements. ***F_SIR.3*** – Automatic creation/selection of storage policies. ***F_NCTFR.4*** – Autonomously monitor cloud-native applications over heterogeneous distributed resources. ***F_NCTFR.5*** – Detect critical situations that may lead to application reconfigurations or data migration. ***F_ROSAR.1*** – Support cognitive and multi-object resource allocation algorithms. ***F_ROSAR.2*** – Support seamless and declarative orchestration of self-organized distributed orchestration systems. ***F_ROSAR.11*** – Transprecise adaptation ML methods. ***F_ROSAR.12*** – Transprecise hyper-parameter optimization methods. ***F_SOR.1*** – Application/Tool specification. ***F_SOR.2*** – Task description/metadata. ***F_SOR.5*** – Requirements forecasting.

Requirement ID:	NF_GR.3	**Priority:**	Core
Requirement Title:	Compatibility	**Stakeholders Involved:**	All
Description:	Degree to which a product, system or component can exchange information with other products, systems or components, and/or perform its required functions while sharing the same hardware or software environment. This characteristic is composed of the following sub-characteristics: • **Co-existence -** Degree to which a product can perform its required functions efficiently while sharing a common environment and resources with other products, without detrimental impact on any other product. • **Interoperability** - Degree to which two or more systems, products or components can exchange information and use the information that has been exchanged.		
Rationale / Goal:	The SERRANO platform integrates decentralized and heterogeneous edge, cloud and HPC infrastructures through an intent-driven paradigm. To this end, it is important to facilitate interoperability and, most of all, integrability with		

	existing platforms and resource management mechanisms. SERRANO will build the appropriate abstractions and common description models to promote resource interoperability and transparent application deployment. The platform will not be built from scratch but as an integration of available mature solutions, through the implementation of all the required extensions and improvements. Moreover, all the designed APIs will expose all the supported functionalities.
Related Functional Requirements	***F_GR.1*** - Provide a unified view of cloud, edge and HPC resources. ***F_GR.6*** - Expose well-defined APIs through SERRANO SDK. ***F_ECHAR.1*** - Full application source code for profiling & acceleration. ***F_ECHAR.2*** - Partial application source code for acceleration. ***F_ECHAR.6*** - Device selection in design time. ***F_SIR.2*** - Secure storage service should support most common S3 operations. ***F_SIR.4*** - Secure storage service should maintain data access from browsers. ***F_NCTFR.1*** – Discover and monitor heterogeneous resources. ***F_NCTFR.6*** – Exchange events between the components of the hierarchical telemetry infrastructure. ***F_ROSAR.2*** – Support seamless and declarative orchestration of self-organized distributed orchestration systems. ***F_ROSAR.8*** - Persistent storage of monitoring data. ***F_ROSAR.9*** - Data formatting and pre-processing. ***F_SOR.1*** – Application/Tool specification. ***F_SOR.2*** – Task description/metadata. ***F_SOR.3*** – Resources description. ***F_IPDR.7*** – Software abstraction layer for every hardware-dependent platform component.

Requirement ID:	NF_GR.4	**Priority:**	Core
Requirement Title:	Usability	**Stakeholders Involved:**	All
Description:	Degree to which a product or system can be used by specified users to achieve specified goals with effectiveness and efficiency in a specified context of use. This characteristic is composed of the following sub-characteristics: • **Appropriateness recognizability** - Degree to which users can recognize whether a product or system is appropriate to their needs. • **Learnability -** Degree to which a product or system can be used by specified users to achieve specified goals of learning to use the product or system with effectiveness, efficiency, freedom from risk and satisfaction in a specified context of use. • **Operability** - Degree to which a product or system has attributes that make it easy to operate and control. • **User error protection** - Degree to which a system protects users against making errors.		

Rationale / Goal:	The SERRANO platform will support seamless deployment through the adoption of the *develop once, deploy everywhere* approach. Users will develop their applications and will express their high-level deployment requirements in an infrastructure agnostic manner. SERRANO will provide the appropriate SDK and all the necessary functionalities to facilitate the deployment and monitoring of diverse applications.
Related Functional Requirements	***F_GR.2*** - Enable an intent-driven paradigm of federated infrastructures. ***F_GR.6*** – Expose well-defined APIs through SERRANO SDK. ***F_ROSAR.1*** – Support cognitive and multi-object resource allocation algorithms. ***F_ROSAR.2*** - Support seamless and declarative orchestration of self-organized distributed orchestration systems***.*** ***F_NCTFR.2*** - Dynamically monitor short-lived applications (support serverless architecture). ***F_NCTFR.4*** - Autonomously monitor cloud-native applications over heterogeneous distributed resources. ***F_NCTFR.8*** - Ability to centralize the monitoring information in a common view or place. ***F_SOR.1*** – Application/Tool specification. ***F_SOR.2*** – Task description/metadata. ***F_SOR.5*** – Requirements forecasting ***F_IPDR.2*** – Docker image of component to be used for creating a running container. ***F_IPDR.4*** – CI/CD guidelines and DevSecOps. ***F_IPDR.6*** - Code Quality and Security.

Requirement ID:	NF_GR.5	**Priority:**	Core
Requirement Title:	Reliability	**Stakeholders Involved:**	All
Description:	Degree to which a system, product or component performs specified functions under specified conditions for a specified period of time. This characteristic is composed of the following sub-characteristics: • **Maturity** - Degree to which a system, product or component meets needs for reliability under normal operation. • **Availability** - Degree to which a system, product or component is operational and accessible when required for use. • **Fault tolerance** - Degree to which a system, product or component operates as intended despite the presence of hardware or software faults. • **Recoverability** - Degree to which, in the event of an interruption or a failure, a product or system can recover the data directly affected and re-establish the desired state of the system.		
Rationale / Goal:	Within the context of the SERRANO platform, specific service assurance and		

	telemetry mechanisms will be architecturally defined and implemented, which will safeguard the deployed applications through the automatic monitoring and triggering of necessary self-optimization procedures.
Related Functional Requirements	***F_GR.4*** - Encompass an autonomous and continuous control loop. ***F_ECHAR.3*** - Application error tolerations. ***F_NCTFR.5*** - Detect critical situations that may lead to application reconfigurations or data migration. ***F_NCTFR.7*** - Zero-touch and data-driven reconfigurability of telemetry components. ***F_ROSAR.3*** - Ability to coordinate workload migration. ***F_ROSAR.4*** - Support automatic data migration operations. ***F_ROSAR.11*** - Transprecise adaptation ML methods. ***F_ROSAR.12*** - Transprecise hyper-parameter optimization methods.

Requirement ID:	NF_GR.6	**Priority:**	Core
Requirement Title:	Security	**Stakeholders Involved:**	All
Description:	Degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization. This characteristic is composed of the following sub-characteristics: • **Confidentiality** - Degree to which a product or system ensures that data are accessible only to those authorized to have access. • **Integrity** - Degree to which a system, product or component prevents unauthorized access to, or modification of, computer programs or data. • **Non-repudiation** - Degree to which actions or events can be proven to have taken place so that the events or actions cannot be repudiated later. • **Accountability** - Degree to which the actions of an entity can be traced uniquely to the entity. • **Authenticity** - Degree to which the identity of a subject or resource can be proved to be the one claimed.		
Rationale / Goal:	The SERRANO platform will encompass a number of security and trustworthiness mechanisms, operating in multiple layers, to provide by design, privacy and resiliency against security threats.		
Related Functional Requirements	***F_GR.5*** **-** Support safety-critical, latency-sensitive and data-intensive applications. ***F_ECHAR.5*** **-** Application error tolerations. ***F_SIR.1*** - Authentication, encryption, privacy and data integrity of communications. ***NF_SIR.1*** - Secure storage service should provide low latency access. ***NF_SIR.2*** - Secure storage service should enforce end to end encryption. ***F_SIR.5*** - Secure execution of workloads.		

	F_NCTFR.3 - Estimate inter-site and intra-site network characteristics. ***F_NCTFR.5*** – Detect critical situations that may lead to application reconfigurations or data migration. ***F_NCTFR.6*** - Exchange events between the components of the hierarchical telemetry infrastructure. ***F_ROSAR.7*** - Time-based ordering of monitoring data entries. ***F_SOR.6*** - Security and privacy. ***F_IPDR.4*** - CI/CD guidelines and DevSecOps. ***F_IPDR.5*** - Adequate unit, integration, and security tests. ***F_IPDR.6*** - Code Quality and Security.

Requirement ID:	NF_GR.7	**Priority:**	Core
Requirement Title:	Maintainability	**Stakeholders Involved:**	All
Description:	This characteristic represents the degree of effectiveness and efficiency with which a product or system can be modified to improve it, correct it or adapt it to changes in environment, and in requirements. This characteristic is composed of the following sub-characteristics: • **Modularity** - Degree to which a system or computer program is composed of discrete components, such that a change to one component has minimal impact on other components. • **Reusability** - Degree to which an asset can be used in more than one system, or in building other assets. • **Analysability** - Degree of effectiveness and efficiency with which it is possible to assess the impact on a product or system of an intended change to one or more of its parts, or to diagnose a product for deficiencies or causes of failures, or to identify parts to be modified**.** • **Modifiability** - Degree to which a product or system can be effectively and efficiently modified without introducing defects or degrading existing product quality. • **Testability** - Degree of effectiveness and efficiency with which test criteria can be established for a system, product or component and tests can be performed to determine whether those criteria have been met.		
Rationale / Goal:	The SERRANO platform integration will follow DevSecOps best practices, with integration to be carried out on a step-by-step basis under a planned procedure, running local interoperability tests after each step. Through the above procedure, the platform will be tested for its technical conformity and efficiency. Moreover, the selected modular architecture and the well-defined and open interfaces will enable the effortless implementation of future extensions for separate components.		
Related Functional Requirements	***F_GR.4*** - Encompass an autonomous and continuous control loop. ***F_GR.6*** - Expose well-defined APIs through SERRANO SDK. ***F_ECHAR.1*** - Full application source code for profiling & acceleration.		

	F_ECHAR.8 - Dataset to stress test the devices. ***F_SIR.2*** - Secure storage service should support most common S3 operations. ***F_NCTFR.8*** - Ability to centralize the monitoring information in a common view or place. ***F_ROSAR.6*** – Ability to implement custom scheduling policies. ***F_ROSAR.8*** - Persistent storage of monitoring data. ***F_SOR.3*** – Resources description. ***F_IPDR.1*** – Documentation of all integration points. ***F_IPDR.3*** - Gitlab as code repository. ***F_IPDR.4*** - CI/CD guidelines and DevSecOps. ***F_IPDR.5*** - Adequate unit, integration, and security tests. ***F_IPDR.6*** - Code Quality and Security.

Requirement ID:	NF_GR.8	**Priority:**	Core
Requirement Title:	Portability	**Stakeholders Involved:**	All
Description:	Degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another. This characteristic is composed of the following sub-characteristics: • **Adaptability** - Degree to which a product or system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environments. • **Installability** - Degree of effectiveness and efficiency with which a product or system can be successfully installed and/or uninstalled in a specified environment. • **Replaceability** - Degree to which a product can replace another specified software product for the same purpose in the same environment.		
Rationale / Goal:	The SERRANO platform aspires to provide the necessary mechanisms to enable the transparent and efficient deployment of workload over heterogeneous infrastructures. The platform will be built on top of available open-source technologies and state of the art platforms. Components will interact through well-defined interfaces to ensure future extensions and portability to new hardware and software platforms.		
Related Functional Requirements	***F_GR.1*** - Provide a unified view of cloud, edge and HPC resources. ***F_GR.2*** - Enable an intent-driven paradigm of federated infrastructures. ***F_GR.3*** - Support transparent application deployment. ***F_GR.6*** - Expose well-defined APIs through SERRANO SDK. ***F_GR.7*** - Support of additional application areas. ***F_ECHAR.3*** – Application specifications and examples. ***F_ECHAR.6*** - Device selection in design time. ***F_ECHAR.7*** - Accelerated kernels integration.		

	F_ECHAR.9 – Implement accelerate kernels as vAccel plugins. ***F_SIR.2*** - Secure storage service should support most common S3 operations. ***F_SIR.3*** – Automatic creation/selection of storage policies. ***F_SIR.4*** - Secure storage service should maintain data access from browsers. ***F_SIR.5*** - Secure execution of workloads. ***F_NCTFR.1*** - Discover and monitor heterogeneous resources. ***F_NCTFR.4*** – Autonomously monitor cloud-native applications over heterogeneous distributed resources. ***F_NCTFR.7*** - Zero-touch and data-driven reconfigurability of telemetry components. ***F_ROSAR.2*** - Support seamless and declarative orchestration of self-organized distributed orchestration systems. ***F_ROSAR.3*** – Ability to coordinate workload migration. ***F_ROSAR.5*** - Orchestrate deployment of data segments over distributed edge and cloud resources. ***F_ROSAR.6*** - Ability to implement custom scheduling policies. ***F_SOR.1*** – Application/Tool specification. ***F_SOR.2*** – Task description/metadata. ***F_SOR.3*** – Resources description. ***F_IPDR.1*** - Documentation of all integration points. ***F_IPDR.2*** – Docker image of component to be used for creating a running container. ***F_IPDR.7*** - Software abstraction layer for every hardware-dependent platform component.

5.2 Edge, Cloud and HPC Acceleration Requirements

5.2.1 Purpose

Even though edge-cloud architectures expand the computational capacity of the traditional cloud paradigm by introducing an additional huge pool of computing resources, the inefficiency of traditional CPUs to provide fast, near real-time executions has also led to the introduction of hardware accelerators, such as GPUs and FPGAs, to the aforementioned hierarchy. Typical examples of accelerators include power-efficient devices at the edge (e.g., NVIDIA Jetson and Xilinx MPSoC), to high-performance, massively-parallel devices in the cloud (e.g., NVIDIA Ampere and Xilinx Alveo). On top of that, High Performance Computing (HPC) infrastructures provide extra computing capacity and acceleration capabilities through the provision of a massive pool of performance efficient servers, delivering peak performances in Petaflops order of magnitude.

While hardware accelerators and HPC systems provide increased performance gains, these benefits do not come for free, as such devices and platforms typically have greater programming effort and operating power requirements. From the edge point of view, energy efficiency has always been a first-class system - design concern, to provide low-

power embedded systems design. On the cloud and HPC side, the considerable share of electricity expenses over the total cost of ownership (TCO), as well as the shift towards energy-efficient (green) computing at the data-center level, also indicate the need for efficient software and hardware acceleration. Towards building more energy proportional computing systems, approximation techniques have been identified as a promising solution to reduce energy consumption and increase performance while retaining the output quality of applications. *Approximate computing* is based on the observation that many applications feature intrinsic error-resilience properties (i.e. in DSP or machine learning domain). On top of that, automatic compilation flows have been leveraged to apply hardware transformations in FPGA designs in order to achieve faster and more optimal results. SERRANO leverages approximation techniques both in the Edge/Cloud as well as the HPC perspective to provide performance-energy trade-offs of accelerated kernels. By applying hardware and software approximation techniques, SERRANO's orchestration mechanisms are able to deploy QoS-, system- and/or user-driven optimized applications.

5.2.1.1 The need for acceleration using GPUs and FPGAs

GPUs and FPGAs are hardware devices usually attached in a PCIe slot or packaged in a complete System on a Chip (SoC). They are used for heavy processing or low latency applications as they have many parallel processing elements that can accelerate the execution of a program. Under optimal configuration GPUs and FPGAs can tackle the high computational demands of an application and provide substantial performance increase (or low latency and power). This is an important condition for the integration of these devices in the SERRANO project in order to benefit from the hardware acceleration. Table 24 depicts in short, the core differences between FPGA and GPU devices, regarding their flexibility in terms of programmability and performance efficiency.

Table 25: Main differences between FPGAs and GPUs towards acceleration

Feature	FPGA (Xilinx)	GPU (Nvidia)
Programming model	Flexible	Flexible
Design Time	Long	Relatively short
Processing pattern	DSPs, Pipeline	Bulk, SIMD array
Performance benefits	Latency, throughput	Throughput
Programming method	Vivado, Vitis	Cuda, Cuda-X
Hardware adaptable	Yes	Limited

We will select the appropriate hardware accelerator devices depending on the characteristics of the UC applications and provide a library of accelerated kernels with different versions depending on the accuracy, performance, power tradeoffs. In Table 26 we can observe the detailed specs between the PCIe devices that will be used in the SERRANO project.

Table 26: Hardware PCIe Devices lineup - Detailed specs

Device name	Alveo U200	Alveo U50	Nvidia T4
Width	Dual Slot	Single Slot	Single Slot
Thermal Cooling	Passive	Passive	Passive
PCI Express	Gen3x16	Gen3x16, 2x Gen4	Gen3x16, x8
DDR Total Capacity	64 GB	-	16 GB (GDDR6)
DDR Total Bandwidth	77 GB/s	-	320.0 GB/s
HBM Total Capacity	-	8 GB	-
HBM Total Bandwidth	-	460 GB/s	-
Network Interface	2x QSFP28	1x QSFP28	-
DSP Slices	6840	5952	-
Maximum Power	225W	75W	70W

5.2.1.2 SERRANO HPC Resources

The High-Performance Computing Center Stuttgart (HLRS)[8] provides the computational resources that allow scientists and engineers to solve complex problems. It operates various HPC systems for a wide range of HPC applications. The HPE Apollo 9000 Hawk system is the most powerful of them. At the time of installation, it ranked among the fastest high-performance computers in the world and the fastest general-purpose machine for industrial production in Europe. In Top500's November 2020 list, the Hawk ranked 16th for HPL and 18th for HPCG[9]. Table 27 below describes the main component of the Hawk. More details can be found in reference [4].

Table 27: Main parameters of HPE Apollo (Hawk) HPC System at HLRS

Number of cabinets	44	**CPUs per node**	2
Number of compute nodes	5,632	**Cores per CPU**	64
System peak performance	26 Petaflops	**Number of compute cores**	720,896
Interconnect topology	Enh.9D-Hypercube	**CPU frequency**	2.25 GHz
Workspace (lustre)	~25 PB	**DIMMs in system**	90,112
Power consump. (Avrg./HPL)	~3.5 MW/ ~4.1 MW	**Total system memory**	~ 1.44 PB

The computing power of the supercomputers results from the higher number of compute nodes, high-performance interconnect and performance-optimized software. The high cost of the HPC infrastructure requires efficient use of the supercomputers. This is an important condition for integrating the HPC services in the SERRANO platform and imposes certain

[8] GCS, Gauss center for supercomputing. https://www.gauss-centre.eu
[9] TOP500.org, "TOP500", 1993-2020. https://www.top500.org

requirements on the UCs of the project and the SERRANO orchestration mechanisms. Table 28 shows main differences between HPC, edge and cloud from an acceleration point of view (compare to Table 25).

Table 28: Main differences between HPC and edge and cloud towards acceleration

Feature	HPC	Edge & Cloud
Programming model	Limited	Flexible
Design Time	Relatively Long	Relatively short
Processing pattern	Vectorization, distribution	Bulk, SIMD array
Performance benefits	Latency, throughput, parallelization	Throughput
Programming method	Low-Level languages (C/C++, Fortran), MPI[10], MPI-OpenMP[11], …	Low- and High-level languages (C/C++, Python), Cuda, Cuda-X, HLS
Hardware adaptable	Limited	Limited

HPC systems are designed for tasks that require significant computing power and memory, which cannot be provided by personal computers or small clusters. The distributed applications must satisfy the following requirements in order to be run on a supercomputer:

- High-level degree of parallelization: An important condition for scaling is an even distribution of the workload among the processes in a parallel program. For this reason, compute nodes are used exclusively by one parallel program at a time.
- Although the supercomputer network has a high capacity, it is many times slower than the speed of memory and processors. Therefore, a balance between computation and communication is needed. On the other hand, performing many independent small size tasks causes the network to be underused, which reduces the efficiency of the system.
- IO is a shared and limited resource. The compute nodes do not have any local hard disks. The file reading and writing must be done on a parallel file system, like Lustre. For a satisfying performance, one must use the parallel libraries, like e.g., MPI-IO[12], Parallel HDF5[13], TecIO-MPI[14].

Figure 15 shows a schematic diagram of the interface.

[10] Message Passing Interface (MPI) is a standardized and portable message-passing standard designed to function on parallel computing architectures (wikipedia).
[11] OpenMP (Open Multi-Processing) supports multi-platform shared-memory multiprocessing programming (wikipedia).
[12] MPI-IO is the I/O part of the MPI-2 standards. The library lets the programmer share open files across multiple parallel processes in an efficient way.
[13] Hierarchical Data Format (HDF) is designed to organize, store and load large amounts of data. Parallel HDF5 uses the MPI standard for interprocess communication.
[14] The TecIO library supports to writing and reading of the Tecplot binary files, which can be opened with visualization & analysis software tools developed by Tecplot, Inc. Its parallel version, TecIO-MPI, uses MPI-IO interface for parallel writing and reading of data.

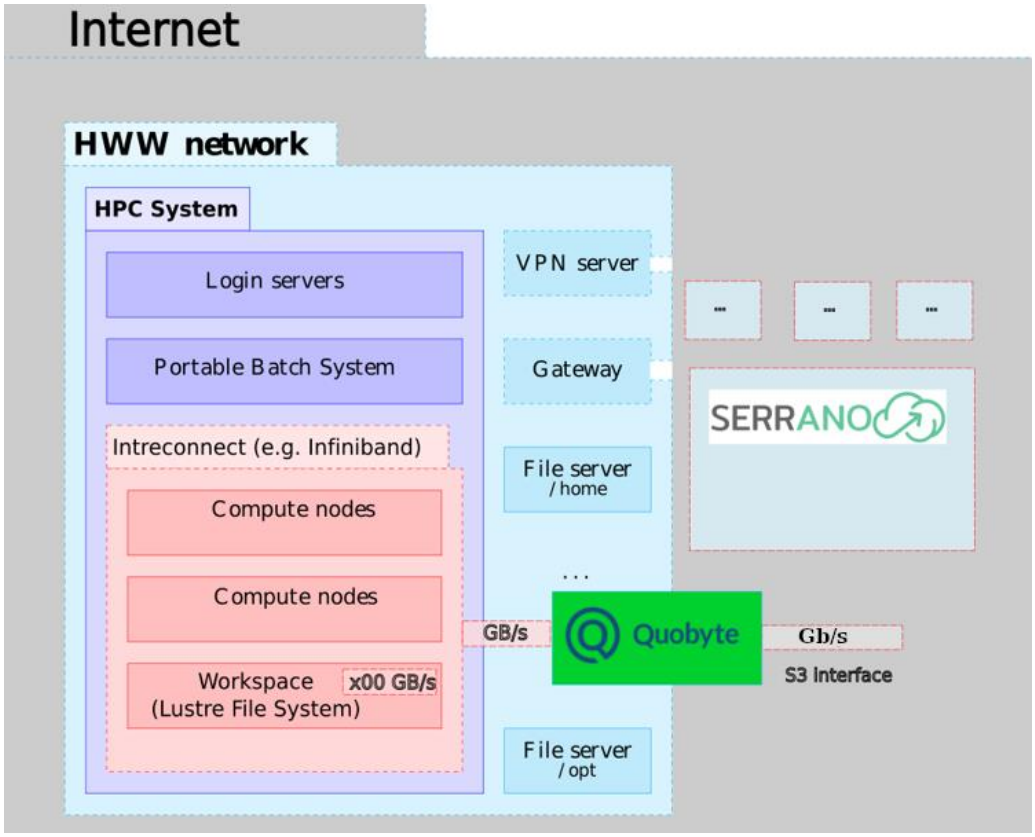


Figure 15: Interface between SERRANO Orchestrator and HPC System

The interface between the orchestrator and the HPC Systems presents another challenge in the SERRANO project. For security reasons, all HLRS's HPC platforms and their infrastructure are interconnected within the internal HWW[15] network. However, there are now platform solutions available in the marketplace that provide easy data exchange between HPC and non HPC environments, such as the "Quobyte" object file system [6]. Although not all features are currently available in the HLRS "Quobyte" file system, it can already be used as a gateway between the SERRANO Orchestrator and Hawk. All data in "Quobyte" is accessible through a native S3 storage interface.

5.2.2 Requirements Analysis

Table 29: Analysis of edge, cloud and HPC acceleration requirements

Requirement ID:	F_ECHAR.1	**Priority [core/ essential/ desired]:**	Essential
Requirement Title:	Full application source code for profiling & acceleration	**Stakeholders Involved:**	UC providers [CC, IDEKO, INB]
Description:	This requirement concerns the provision of the source code from the UC providers' side of the applications that are targeted for acceleration.		

[15] Höchstleistungsrechner für Wissenschaft und Wirtschaft GmbH (HWW) is the joint operating company of the High-Performance Computing Center Stuttgart (HLRS), T-Systems, Porsche AG and the Karlsruhe Institute of Technology (KIT).

Rationale / Goal:	Develop HW/SW accelerated and/or approximate versions of the provided application (WP4).
Acceptance Measures:	Provide the necessary code by M5.
Dependencies:	Source-code confidentiality.

Requirement ID:	F_ECHAR.2	**Priority [core/ essential/ desired]:**	Core
Requirement Title:	Partial application source code for acceleration	**Stakeholders Involved:**	UC providers [CC, IDEKO, INB]
Description:	In case F_ECHAR.1 cannot be fulfilled, this requirement will concern the provision of a computationally intensive part of the full UC application. This part (function) should be derived from a profiling of the application and consume the major portion of the application's execution time.		
Rationale / Goal:	Develop HW/SW accelerated and/or approximate versions of the provided function (WP4).		
Acceptance Measures:	Provide the necessary code by M6.		
Dependencies:	In case full applications source-code cannot be provided due to confidentiality/privacy reasons (F_ECHAR.1)		

Requirement ID:	F_ECHAR.3	**Priority:**	Core
Requirement Title:	Application specifications and examples	**Stakeholders Involved:**	UC providers [CC, IDEKO, INB]
Description:	This requirement concerns the provision of UC applications specifications and how-to guidelines. Regarding specifications, the UC partners should describe the input datasets used as well as any relevant input parameters that are required for the execution of the code. In addition, detailed examples should be provided, explaining the procedure followed to run the application, as well as expected outputs and/or results.		
Rationale / Goal:	In align with F_ECHAR.1 or F_ECHAR.2		
Acceptance Measures:	Given the how-to guidelines, code should be running without any errors.		
	The code should be compatible with Linux systems.		
Dependencies:	F_ECHAR.1 or F_ECHAR.2		
	Examples and guidelines should be accompanied by the respective application code (F_ECHAR.1 or F_ECHAR.2). In case of Linux non-compatible codes, identical Linux-friendly alternatives should be provided by the respective UC partner.		

Requirement ID:	F_ECHAR.4	**Priority:**	Desired
Requirement Title:	Application/System Quality-of-Service requirements	**Stakeholders Involved:**	UC and orchestration providers [CC, IDEKO, INB, ICCS]
Description:	This requirement concerns the specification of any application QoS requirements (e.g., latency, throughput) or system requirements (e.g., power consumption).		
Rationale / Goal:	Develop HW/SW accelerated libraries under QoS/system requirements.		
Acceptance Measures:	The target QoS provided by each UC partner should meet the KPIs defined in the SERRANO description.		
	Any target system requirements provided by the orchestrator should correspond to realistic values, given the underlying HW infrastructure.		
Dependencies:	Regarding applications, QoS should be feasible for any size of input data or, otherwise, be defined explicitly.		
	For target QoS requirements at runtime, the orchestrator should provide feasible objectives, taking into account any overheads related to data transfer, network congestion, interference effects, etc.		

Requirement ID:	F_ECHAR.5	**Priority:**	Desired
Requirement Title:	Application error tolerations	**Stakeholders Involved:**	UC providers [CC, IDEKO, INB]
Description:	This requirement concerns the specification of the applications' output error tolerations. UC providers should specify the error metric (e.g., prediction accuracy, R^2 score or any specific code used to calculate the error) used to measure the accuracy/error. In addition, maximum error thresholds should be specified for applications.		
Rationale / Goal:	Develop approximate HW/SW accelerated kernels according to the error tolerations provided.		
Acceptance Measures:	UC providers should define an error toleration of at least 5% for at least one of the provided candidate acceleration kernels (from F_ECHAR.1/F_ECHAR.2) to reveal the benefits of approximate computing.		
Dependencies:	UC target KPIs in SERRANO.		

Requirement ID:	F_ECHAR.6	**Priority:**	Essential
Requirement Title:	Device selection in design time	**Stakeholders Involved:**	UC providers [CC, IDEKO, INB]
Description:	This requirement concerns the specifications of the applications' characteristics in order to determine the appropriate device for acceleration in design time. Possible characteristics might include the type of processing		

	(compute intensive, memory intensive), the location of the computation (e.g., edge, fog, cloud, HPC), the size of memory required (i.e., whether can it fit in on-chip memory), error tolerations, approximation potential or synthesizability (for FPGA case).
Rationale / Goal:	Determine the feasibility of the application acceleration for a specific device.
Acceptance Measures:	UC providers should define the specifications for each application regarding the memory required, computations involved, error toleration and latency/throughput requirements.
Dependencies:	F_ECHAR.1 or F_ECHAR.2
	The appropriate device for each application should derive from the given application characteristics of each UC input.

Requirement ID:	F_ECHAR.7	**Priority:**	Essential
Requirement Title:	Accelerated kernels integration	**Stakeholders Involved:**	UC providers [CC, IDEKO, INB]
Description:	This requirement concerns the integration of the final accelerated kernels for the end-to-end applications' execution. The hardware accelerated versions of the given functions from the UC providers must be linked seamlessly in the UC framework/application (i.e., the kernels could be given as a dynamic library that can be linked). The API for hardware acceleration must also be included in the application (or imported in case of python code). Also, the hardware kernel must be given as input in the application through an argument (i.e., the Bitstream in case of FPGA acceleration could be provided that way).		
Rationale / Goal:	Enable the accelerated library integration with the UC application.		
Acceptance Measures:	The integration of the acceleration kernels should comply with the target KPIs in the SERRANO platform interface.		
Dependencies:	The final programming model of the UC applications should be aligned with the initial source code provided in the beginning of the project (F_ECHAR.1 or F_ECHAR.2).		
	Hardware accelerators (e.g., GPUs, FPGAs) should be exposed in virtualized environments (e.g., containers, unikernels) to properly run the accelerated code.		

Requirement ID:	F_ECHAR.8	**Priority:**	Desired
Requirement Title:	Dataset to stress test the devices	**Stakeholders Involved:**	UC providers [CC, IDEKO, INB]
Description:	This requirement concerns the ability to stress test the hardware devices (CPU, GPU, FPGA) given a large dataset. The provided datasets can be enlarged in order to utilize the devices in longer periods of time instead of a couple of		

	seconds. The datasets can be synthesized if it is not possible to find real inputs. The aim is to eliminate the device overheads when the application's execution time is very small.
Rationale / Goal:	Test the devices with larger datasets to better evaluate their performance.
Acceptance Measures:	To showcase the benefits of hardware (GPU, FPGA) acceleration, the application's input dataset should be large enough to keep the accelerator busy for a satisfactory period of time (>10sec).
Dependencies:	UC target KPIs in SERRANO.

Requirement ID:	F_ECHAR.9	**Priority:**	Desired
Requirement Title:	Implement accelerate-able kernels as vAccel plugins	**Stakeholders Involved:**	NBFC, AUTH
Description:	Implementing the kernels required by the UC as operations accelerated on FPGA devices will boost performance and energy-efficiency of UC applications. One way of integrating the UC applications with the FPGA implementations is through vAccel. The vAccel runtime will expose the accelerated operations on the front-end. On the back-end, the FPGA implementation of the respective kernels will be shipped in the form of a vAccel plugin. Implementations of the same kernel can be developed for other accelerators, such as GPUs or even the CPU itself, as vAccel plugins. The UC application will link against the vAccel library instead of linking directly with the FPGA implementation, making it easily portable to any platform for which a corresponding vAccel plugin exists.		
Rationale / Goal:	Facilitate development and portability of applications on the SERRANO platform.		
Acceptance Measures:	Develop FPGA accelerated-kernels as vAccel plugins.		
	Implement same kernels on other architectures (GPUs, CPUs, etc.) to showcase the offered portability.		
Dependencies:	N/A		

5.3 Secure Infrastructure Requirements

5.3.1 Purpose

Communication networks rely on control processing units (CPUs) as main engine to support the control and management of all the network stack. For example, if we consider the Open System Interconnection model (OSI model), every single operation that any of the layers conduct, uses CPU cycles. As networks become more complex in terms of layer composition and supported traffic due to diverse applications, the amount of needed CPU cycles grows. Needless to say, we can scale network capacity by increasing the amount of CPU clocks the CPU can conduct per unit of time. However, aspects such as cost, price, power consumption or parallelization trade-offs make the capacity of CPUs plateau in this respect. Hence, it is very relevant to adopt measures to off-load CPU processes and execute them on dedicated

System-on-Chips (SoC), either in the form of Field-Programmable Gate Arrays (FPGAs) or Application-Specific Integrated Circuits (ASICs). For example, in Artificial Intelligence (AI), Machine Learning (ML) or Deep Learning (DL), Graphic Processing Units (GPUs) are found to be much more efficient in conducting processing tasks than CPUs. SERRANO will identify functions that need to be hardware accelerated and, regardless of the nature of the HW accelerator of choice (i.e. GPU, ASIC, FPGA, etc.), develop technologies that enable to have such building blocks integrated into the overall system through network connections and networking processes; in this way, HW accelerators that are embedded into processing units can be reached by any service or application operating on the network.

In the area of networking, one of the most demanding CPU activities is the securitization of communications. Since SERRANO aims at a cloud-edge continuum when it comes to the access of processing power, the underlying networks need to be end-to-end securitized. When a connection is established between two network elements, data transmitted over that link is normally encrypted, a process through which data are encoded so that it remains hidden from or inaccessible to unauthorized users or third parties. The gold standard for encoding is the Advanced Encryption Standard (AES), generated and maintained by the National Institute of Standards and Technology (NIST) in 2001, which is a set of specifications for the encryption of electronic data. Adhering to the same specifications ensures interoperability as well as the comparability of benchmarks made for the performance assessment of systems and subsystems. AES is a symmetric key cipher, which means the same secret key is used for both encryption and decryption, and both the sender and receiver of the data need a copy of the key. By contrast, asymmetric key systems use a different key for each of the two processes. The advantage of symmetric systems like AES is their speed because a symmetric key algorithm requires less computational power than an asymmetric one. AES can be used in three flavours: 128-bit, 192-bit, and 256-bit. Each type uses 128-bit blocks, with the difference lying in the length of the key. As the longest, the 256-bit key provides the strongest level of encryption (2^{256} combinations). The three AES varieties are also distinguished by the number of rounds of encryption. AES 128 uses 10 rounds, AES 192 uses 12 rounds, and AES 256 uses 14 rounds. AES 256 uses 40% more system resources than AES 192.

In the context of SERRANO, the utilization of AES or other encryption methodologies is fundamental to ensure isolation and protection among the different users, applications or services utilizing the platform.

Table 30 shows the summary of technical aspects in relation to confidentiality and integrity of IPsec and TLS.

Table 30: Security protocols required in SERRANO nodes

Feature	IPsec	TLS
Authentication	Yes	Yes
Integrity	Yes	Yes
Confidentiality	Yes	Yes

Configuration	Complex	Straightforward
Interoperability problems	Yes	No
TCP apps support	All	Some
UDP support	Yes	Only Datagram TLS
PKI	No	Yes
Compression	Yes	Only OpenSSL
Client-specific software	Yes	No
Multi-environment support	Sometimes	Yes
Apps filter	No	Yes

IPsec, on the one hand, includes protocols for establishing mutual authentication between agents at the beginning of a session and negotiation of cryptographic keys to use during the session. IPsec can protect data flows between a pair of hosts (host-to-host), between a pair of security gateways (network-to-network), or between a security gateway and a host (network-to-host). IPsec uses cryptographic security services to protect communications over IP networks. It supports network-level peer authentication, data-origin authentication, data integrity, data confidentiality (encryption), and replay protection. On the other hand, TLS are a set of cryptographic protocols designed to provide communications security over network units. TLS in itself runs on top of reliable transport protocols (e.g., TCP), aiming primarily to provide privacy and data integrity.

Hence, IPsec and TLS can work together, and by doing so, we can effectively create point-to-point virtual private network (VPN)-like tunnels between network elements, which is a very powerful paradigm providing confidentiality and integrity of communications even within the shortest or smallest communication links. Both technologies will be deployed on the SERRANO infrastructure to ensure securitization.

5.3.2 Requirements Analysis

The following tables present a set of technologies and requirements for the securitization of the SERRANO platform. Apart from requirement NF_SIR.1, the acceptance measure for the others is of type "Go"/"No go". As such, NF_SIR.1 is the only one with a measurable target.

Table 31: Analysis of secure infrastructure requirements

Requirement ID:	F_SIR.1	**Priority :**	Core
Requirement Title:	Authentication, encryption, privacy and data integrity of communications	**Stakeholders Involved:**	MLNX
Description:	Enabling at Data Processing Unit (DPU) level TLS and IPsec protocols for all communications in order to ensure privacy and sovereignty of data.		
Rationale / Goal:	Once TLS and IPsec are enabled, it can be provided as a service to virtual		

	machines or applications (i.e. Chocolate Cloud for storage UC), which allows CC applications to operate natively on a secured channel.
Acceptance Measures:	Increased throughput by 10%-150% depending on the application and its IO size. compared to the same setup without the TLS/IPSEC offload.
Dependencies:	Ability to have HW offloading for IPsec and TLS by MLNX, currently under investigation.

Requirement ID:	NF_SIR.1	**Priority :**	Essential
Requirement Title:	Secure storage service should provide low latency file access	**Stakeholders Involved:**	CC
Description:	Data access should happen with low latency.		
Rationale / Goal:	Data access latency should be improved when also employing edge storage locations judiciously.		
Acceptance Measures:	Reduction of latency by 10-50% when system is augmented with edge storage locations compared to a purely cloud-based scenario (100-800ms per coded fragment).		
Dependencies:	Reliable data about storage locations: F_SOR.3		
	Sufficient available edge storage locations available on premise.		

Requirement ID:	NF_SIR.2	**Priority :**	Essential
Requirement Title:	Secure storage service should enforce end to end encryption	**Stakeholders Involved:**	CC
Description:	Data should not be accessible in the clear on either the storage locations or on the SkyFlok.com backend.		
Rationale / Goal:	Privacy and security are important goals of the SERRANO platform.		
Acceptance Measures:	No data in the clear in transit.		
	No data in the clear at rest.		
Dependencies:	N/A		

Requirement ID:	F_SIR.2	**Priority :**	Core
Requirement Title:	Secure storage service should support most common S3 operations	**Stakeholders Involved:**	CC
Description:	The storage service should support a large enough subset of the S3 API to enable integration with common applications.		

Rationale / Goal:	The S3 API is the de-facto standard interface for distributed object storage systems.
Acceptance Measures:	Support for most important bucket and object endpoints with basic functionality.
Dependencies:	N/A

Requirement ID:	F_SIR.3	**Priority :**	Essential
Requirement Title:	Automatic creation/ selection of storage policies	**Stakeholders Involved:**	CC
Description:	Whenever a storage task arrives, it should be matched with a storage policy based on its requirements and the characteristics of storage locations.		
Rationale / Goal:	To make best use of resources while limiting costs, storage requirements must be matched with the optimal data distribution, erasure coding and encryption scheme on offer.		
Acceptance Measures:	Automatic creation of a storage policy for an incoming storage task or, alternatively, automatic selection of a storage policy for an incoming storage task from a predefined list.		
	Have a default policy in place for cases when application requirements cannot be captured reliably.		
Dependencies:	Reliable data about storage locations: F_SOR.3		
	Well-defined requirements for storage tasks: F_SOR.2		

Requirement ID:	F_SIR.4	**Priority :**	Desired
Requirement Title:	Secure storage service should maintain data access from browsers	**Stakeholders Involved:**	CC
Description:	If a user is able to access all storage locations (there are no technical obstacles to communication such as firewalls or other security policies) and the storage gateway (potentially not needed), file access from browsers should be maintained.		
Rationale / Goal:	It would be convenient for users to maintain browser-based access.		
Acceptance Measures:	Ability to download and upload of files using the SkyFlok web application, when client has access to said files' storage locations and storage gateway (potentially not needed).		
Dependencies:	N/A		

Requirement ID:	F_SIR.5	**Priority :**	Essential

Requirement Title:	Secure execution of workloads	**Stakeholders Involved:**	NBFC, ICCS
Description:	Enable the verification of workloads running at the Edge.		
Rationale / Goal:	Workloads scheduled to be executed in Cloud environments run in an isolated enclave (as a VM for instance), protected from a number of attacks (including physical access related). When the same workloads are deployed in an Edge environment it is crucial that data (and/or the workload logic) should be protected from leaks/attacks. To this end, the SERRANO platform provides a secure execution framework to validate legitimate execution and prevent unauthorized access to data.		
Acceptance Measures:	Verify that the executing task is one intended.		
Dependencies:	WP3,5 secure execution and hardened provisioning.		

5.4 Network and Cloud Telemetry Framework Requirements

5.4.1 Purpose

In the SERRANO platform, a sense, discern, infer, decide, and act, continuous control loop will run autonomously over the infinite to adjust resources and migrate the tasks, based on feedback regarding the state if applications and resources. This feedback will be provided by an autonomous, scalable, data-driven network and cloud telemetry framework that will collect and analyse telemetry data across the distributed heterogeneous infrastructures (edge/cloud/HPC) and the deployed applications.

The telemetry service will enable full observability and adaptability of the computing and networking conditions, leading to improved overall Quality of Service (QoS). It will provide the collected information in other SERRANO components. Events based on the collected, correlated and analysed telemetry information will trigger automatic re-optimization adjustments, and data and workload movement. These will be performed by the respective services in central and local level. These actions will be performed both reactively (when the state of the resources degrades to an extent that it violates a goal) and proactively (using predictions based on past experience SERRANO will foresee a likely change in the state of the resources and will act in advance to avoid the violation of the goal).

5.4.2 Requirements Analysis

The following tables present a set of functional requirements for the SERRANO network and cloud telemetry framework. These requirements are derived from the three UCs analysis and the overall envisioned functionality of the SERRANO platform.

Table 32: Analysis of network and cloud telemetry framework requirements

Requirement ID:	F_NCTFR.1	**Priority:**	Core
Requirement Title:	Discover and monitor heterogeneous resources	**Stakeholders Involved:**	All
Description:	The SERRANO network and cloud telemetry framework should automatically maintain a catalogue with the available computational and storage resources within the individual edge, cloud and HPC platforms that constitutes the SERRANO platform. Moreover, it should provide the appropriate resource monitoring mechanisms that will be able to keep track of the available and used resources.		
Rationale / Goal:	The service and resource orchestration mechanisms will be able to retrieve the list of available resources along with their current status in order to orchestrate new application deployments over the heterogeneous SERRANO infrastructure. Moreover, the service assurance mechanisms will leverage the monitoring information for executing corrective actions on existing deployments.		
Acceptance Measures:	Dynamic and detailed inventory parameters (amount type, characteristics) of available computational and storage resources from multiple edge, cloud and HPC platforms.		
	Report resource utilization from all SERRANO hardware and software resources.		
Dependencies:	Resource description based on the ARDIA model (F_SOR.3).		
	Capabilities and exposed APIs of individual cloud, edge and HPC platforms.		

Requirement ID:	F_NCTFR.2	**Priority:**	Core
Requirement Title:	Dynamically monitor short-lived applications (support serverless architecture)	**Stakeholders Involved:**	NBFC, ICCS, UVT, AUTH
Description:	The SERRANO platform will provide hardware acceleration abstraction for short-lived workloads. In this context, the telemetry framework is necessary to collect relevant performance metrics on a per-function invocation basis, and make them available to other core components.		
Rationale / Goal:	The collected information will be leveraged by the orchestration and service assurance mechanisms to improve their decisions and enable distributed tracing.		
Acceptance Measures:	Collect performance metrics on a per-function invocation basis.		
	Response time and monitoring granularity should be acceptable.		
Dependencies:	Lower-level software stack for deploying serverless workloads (F_ECHAR.9).		
	Telemetry Probes (F_NCTFR.1).		

Requirement ID:	F_NCTFR.3	**Priority:**	Core
Requirement Title:	Estimate inter-site and intra-site network characteristics	**Stakeholders Involved:**	ICCS, MLNX, UVT
Description:	The SERRANO telemetry framework should dynamically and intelligently monitor the interconnection links between the distributed edge, cloud and HPC infrastructures. By leveraging non-disruptive monitoring probes and advanced AI/ML algorithms will correlate the network related information in time and space, to infer appropriate metrics (e.g., packet flows, jitter, delay), identify general performance problems and predict the future network state.		
Rationale / Goal:	The provision of estimations about the performance of the network resources that interconnect the distributed SERRANO infrastructure resources will enable the SERRANO resource orchestrator mechanisms to improve their allocation decisions, leading to better overall performance for the deployed applications.		
Acceptance Measures:	Provide adequate estimations for key performance indicators (i.e. bandwidth, delay, jitter).		
	Ability to identify rapid load changes or malicious attacks that will affect the performance of the deployed applications.		
	Ability to identify or predict network performance degradations and other shortcomings.		
Dependencies:	Network devices exposed APIs.		
	AI and ML algorithms for information coloration, analysis and performance degradation prediction. (F_ROSAR.1, F_ROSAR.8)		

Requirement ID:	F_NCTFR.4	**Priority:**	Core
Requirement Title:	Autonomously monitor cloud-native applications over heterogeneous distributed resources	**Stakeholders Involved:**	All
Description:	The telemetry framework should continuously feed the service assurance mechanisms with the most up-to-date information for the deployed applications. Moreover, the end users should always retrieve monitoring information for their applications without knowing details about the specific development of their applications.		
Rationale / Goal:	The telemetry framework should be able to collect, log and correlate monitoring information through the entire lifespan of an application. Hence, the mechanisms of envisioned SERRANO continuous control loop will always be able to adjust resources and migrate workload based on feedback regarding the application's state.		
Acceptance Measures:	Provide real-time and historical monitoring data for deployed applications via the exposed Telemetry API.		
	Ability to dynamically adjust the granularity and parameters of the monitoring		

	information.
	Automatic adjustment of monitoring mechanisms in case of reconfigurations at already deployed applications.
Dependencies:	Telemetry probes and software stack at the resource level (F_NCTFR1, F_ROSAR.2).
	Application and resource description based on the ARDIA model (F_SOR.2, F_SOR.3).

Requirement ID:	F_NCTFR.5	**Priority:**	Core
Requirement Title:	Detect critical situations that may lead to application reconfigurations or data migration	**Stakeholders Involved:**	All
Description:	The telemetry framework should support local and central correlation and data-driven analysis algorithms to identify critical events related to available resources, which contribute to the logic of the hierarchical telemetry infrastructure.		
Rationale / Goal:	The service assurance mechanisms at local and central level should be able to receive notifications about any anomalies during the execution of a deployed application. Events by the telemetry framework will trigger the reactive re-optimization adjustments within the SERRANO platform.		
Acceptance Measures:	Response time should be acceptable from the other services.		
	Identify significant load changes and resources' performance degradations and failures.		
Dependencies:	SERRANO service assurance mechanisms (F_GR.4, F_ROSAR.3).		
	Telemetry AI/ML algorithms for information coloration, analysis, and event identification (F_ROSAR.1, F_ROSAR.8).		

Requirement ID:	F_NCTFR.6	**Priority:**	Core
Requirement Title:	Exchange events between the components of the hierarchical telemetry infrastructure.	**Stakeholders Involved:**	All
Description:	The main telemetry functionalities (i.e. collecting and correlating data, events identification) are spread into different layers. The mechanisms at the lower levels have to forward related events and report aggregated monitoring data to their upper layer. This requirement is critical for the overall operation of the orchestration and service assurance mechanisms of the SERRANO platform.		
Rationale / Goal:	The monitoring and analysis functionalities for resources and applications are spread into several layers with the aim of meeting the requirement of scalability. In the SERRANO platform, the service assurance and orchestration mechanisms at local level are the initial entities where the autonomous control loop implements the observability and analysis functionalities.		

Acceptance Measures:	Response time should be acceptable from the other services.
	Ability to deliverer events to multiple endpoints.
Dependencies:	SERRANO service assurance mechanisms (F_GR.4, F_ROSAR.3).
	Telemetry AI/ML algorithms for information coloration, analysis and event identification (F_ROSAR.1, F_ROSAR.8).

Requirement ID:	F_NCTFR.7	**Priority:**	Core
Requirement Title:	Zero-touch and data-driven reconfigurability of telemetry components	**Stakeholders Involved:**	ICCS, MLNX, UVT
Description:	The telemetry framework should be able to automatically and dynamically adjust, based on analysis of collected data, the granularity of the monitoring information and even deploy supplementary active probes to collect additional monitoring data.		
Rationale / Goal:	This capability will enable the SERRANO platform to dynamically collect additional monitoring data for specific parts of the overall infrastructure or deployed services. Thus, the service assurance and orchestration data-driven mechanisms will always have an adequate amount of information to infer the current conditions and execute required reconfigurations.		
Acceptance Measures:	Automatic deployment of new monitoring probes to retrieve additional information.		
	Dynamic adjustment of monitoring parameters based on feedback from service assurance mechanisms.		
Dependencies:	Telemetry probes (F_NCTFR1).		
	AI and ML algorithms for information coloration and analysis (F_ROSAR.1, F_ROSAR.8).		

Requirement ID:	F_NCTFR.8	**Priority:**	Core
Requirement Title:	Ability to centralize the monitoring information in a common view or place	**Stakeholders Involved:**	All
Description:	The SERRANO network and cloud telemetry framework should provide the collected information to other SERRANO components and external services. A hierarchical monitoring architecture will be developed to intelligently and dynamically monitor the infrastructure and services. It will pre-process and correlate the respective information producing key performance indicators/metrics and identifying basic events that are then forwarded to the Central Telemetry Handler (the root in SERRANO's telemetry hierarchy). The monitoring information and the respective events will be exposed to other services through the developed Telemetry API.		
Rationale / Goal:	The optimization and AI/ML algorithms at the orchestration and service		

	assurance mechanisms in central and local level will leverage the available information for their decisions. Moreover, external services (like the SkyFlok Gateway) will be able to retrieve up to date information for the current state of the infrastructure resources. Finally, the SERRANO platform administrator will be able to access real-time and historical monitoring data via a simple web-based graphical user interface.
Acceptance Measures:	SERRANO platform high-level services (i.e. Service and Resource Orchestrator) and certified external services (e.g. SkyFlok Gateway) able to retrieve up to date monitoring information.
	Ability to provide access to real-time and historical monitoring data via the exposed Telemetry API.
Dependencies:	Application and resource description based on the ARDIA model (F_SOR.2, F_SOR.3).

Requirement ID:	F_NCTFR.9	**Priority:**	Essential
Requirement Title:	Low-level metrics availability	**Stakeholders Involved:**	NBFC, ICCS
Description:	Low-level metrics are essential to determine the health of the hardware platform, as well as the health of the executing workload. Traditional OS metrics provide generic granularity for system-wide metrics; however, they lack fine-grained insights on the behaviour of the workload being executed. The SERRANO platform enhances the available information by gathering low-level metrics from the hardware platform as well as custom metrics from the systems stack hosting the workloads.		
Rationale / Goal:	Provide insights on scheduling decisions and/or execution modes (approximation, hardware acceleration etc.).		
Acceptance Measures:	Metrics for custom hardware state and availability for execution.		
	Metrics for application-specific characteristics (e.g. increased cache miss ratio).		
Dependencies:	N/A		

5.5 Resource Orchestration and Service Assurance Requirements

5.5.1 Purpose

The complexity and heterogeneity of distributed computing infrastructures pose significant challenges for the management and deployment of services. Heading to more complex environments, the development of intelligent and self-managed orchestration mechanisms is key to optimizing resource utilization for present and future workloads. The SERRANO platform will automatically determine the most appropriate (computing, storage) resources to be used by an application, and then transparently deploy related workloads and

coordinate data movement. The overall orchestration is performed in a lean, automated, holistic and integrated manner, overcoming the complexity barriers stemming from the heterogeneity of computing units.

5.5.2 Requirements Analysis

The following tables present the functional requirements for the resource orchestration and service assurance mechanisms according to the analysis of the three UCs and the overall envisioned functionality of the SERRANO platform.

Table 33: Analysis of resource orchestration and service assurance requirements

Requirement ID:	F_ROSAR.1	**Priority:**	Core
Requirement Title:	Support cognitive and multi-object resource allocation algorithms	**Stakeholders Involved:**	All
Description:	The SERRANO platform should be able to take near-optimal resource allocation decisions based on applications' preferences and infrastructure constraints by exploiting multi-objective optimization algorithms and AI/ML techniques. Due to the high number of conflicting objectives, the SERRANO resource allocation algorithms must be able to support a wide range of different trade-offs for between optimality and complexity.		
Rationale / Goal:	The developed algorithms will be part of the Resource Optimization Toolkit (ROT) that will enable the execution of a diverse set of algorithms. These information-based/cognitive algorithms will provide the necessary intelligence to the SERRANO orchestration mechanisms to optimize resource utilization for present and future workloads.		
Acceptance Measures:	Decisions take into consideration the current status of the available resources.		
	Application deployment satisfies user's preferences, while ensures high utilization for the available resources.		
Dependencies:	Application and resource description based on the ARDIA model (F_SOR.2, F_SOR.3).		
	Network and cloud telemetry information (F_NCTFR.1, F_NCTFR.3, F_NCTFR.8).		

Requirement ID:	F_ROSAR.2	**Priority:**	Core
Requirement Title:	Support seamless and declarative orchestration of self-organized distributed orchestration systems	**Stakeholders Involved:**	All
Description:	The orchestration system should support distributed control and management divided into local control loops at the disaggregated and heterogeneous edge, cloud and HPC infrastructures. Each local orchestrator will be responsible for orchestrating the resources under its control. The SERRANO central orchestrator should provide a high-level platform-agnostic description for the		

	workload requirements at local orchestration mechanisms.
Rationale / Goal:	SERRANO should enable the transparent application deployment over federated heterogeneous infrastructures. To this end, the platform will support a hierarchical end-to-end orchestration of loosely coupled individual orchestration mechanisms and respective platforms. The overall goal is to overcome the complexity barriers stemming from the heterogeneity of available resources. The declarative approach for describing the requirements will provide several degrees of freedom to the local control loops.
Acceptance Measures:	Transparent deployment of applications in heterogeneous edge, cloud and HPC resources.
	Ability to coordinate different orchestration platforms.
Dependencies:	Resource orchestration algorithms (F_ROSAR.1).
	Orchestration platforms at the individual edge, cloud and HPC infrastructures.

Requirement ID:	F_ROSAR.3	**Priority:**	Core
Requirement Title:	Ability to coordinate workload migration	**Stakeholders Involved:**	All
Description:	The SERRANO platform continuously monitors the deployed applications to safeguard that they perform as intended, otherwise it triggers the required re-optimization actions. To this end, the orchestration mechanisms should be able to either pause, move, and restart an ongoing workload, or should be able to re-deploy the same workload in another resource (in the same or other platform).		
Rationale / Goal:	The SERRANO service assurance mechanisms based on data-driven techniques will be able to automatically trigger the adaptation of the deployed applications to ensure that applications' operational requirements are always satisfied, regardless of the current status of the infrastructure resources.		
Acceptance Measures:	Workload resumes its operation without user intervention.		
	Imposed delay and downtime are acceptable.		
Dependencies:	Service assurance and resource orchestration algorithms (F_GR.4, F_ROSAR.3, F_ROSAR.1).		
	Orchestration platforms at the individual edge, cloud and HPC infrastructures.		

Requirement ID:	F_ROSAR.4	**Priority:**	Core
Requirement Title:	Support automatic data migration operations	**Stakeholders Involved:**	All
Description:	Orchestration algorithms should support the definition of data placement constraints which should be considered into their decisions regarding the application deployment. Furthermore, the orchestration mechanisms at		

	central and local levels should be able to coordinate the respective operations.
Rationale / Goal:	The SERRANO orchestration mechanisms should be able to automatically and transparently (for the end users) migrate data within the distributed infrastructure, whenever the service assurance triggers an application redeployment. The SERRANO platform should ensure that the required input data are always available to workloads. Moreover, the storage locations for the output data must be updated according to applications' requirements and the current status of the infrastructure.
Acceptance Measures:	All required input data are automatically transferred to the new locations.
	Applications' high-level requirements and operation constraints are met.
Dependencies:	Secure storage service APIs (F_GR.7, F_SIR.3).
	Network and cloud telemetry information (F_NCTFR.1, F_NCTFR.3, F_NCTFR.8).

Requirement ID:	F_ROSAR.5	**Priority:**	Core
Requirement Title:	Orchestrate deployment of data segments over distributed edge and cloud resources	**Stakeholders Involved:**	All
Description:	SERRANO should incorporate various storage edge and cloud service providers that form the distributed storage part of the SERRANO platform. The deployed applications will come along with some necessary data and, most of the time, will produce additional that should be stored in the infrastructure. SERRANO should select the storage locations and coordinate the data transfer between applications and distribute storage resources.		
Rationale / Goal:	The SERRANO orchestration and service assurance mechanisms should be able to automatically coordinate the efficient movement of the required data when applications are deployed or adapted due to re-optimizations. Moreover, the SERRANO platform will transparently orchestrate secure storage of the produced data over the distributed resources.		
Acceptance Measures:	Distribute data over multiple edge and cloud resources.		
	Select storage locations (edge, cloud) based on user preferences and current status of SERRANO platform resources.		
Dependencies:	Application and resource description based on the ARDIA model (F_SOR.2, F_SOR.3).		
	Secure storage service APIs (F_GR.7, F_SIR.3).		

Requirement ID:	F_ROSAR.6	**Priority:**	Core
Requirement Title:	Ability to implement custom scheduling policies	**Stakeholders Involved:**	ICCS, INNOV, UVT, NBFC

Description:	SERRANO should be a platform that includes many different heterogeneous sub-infrastructures for deploying applications with diverse requirements. Furthermore, the platform will be implemented incrementally based on a two-phase development plan. Therefore, the orchestration components should be able to be extended with new scheduling policies based on a well-defined procedure without hidden interfaces.
Rationale / Goal:	The SERRANO orchestration mechanisms will be extendable with new and more advanced scheduling algorithms without modifications and implications in other platform components. Moreover, this design will enhance the opportunity for a more extensive future exploitation of the SERRANO platform.
Acceptance Measures:	Well-defined interfaces to describe scheduling policies input parameters and results.
	Extend the library of scheduling policies without breaking the implementation and functionality of other orchestration components.
Dependencies:	Orchestration platforms at the individual edge, cloud and HPC infrastructures.
	Application and resource description based on the ARDIA model (F_SOR.2, F_SOR.3).

Requirement ID:	F_ROSAR.7	**Priority:**	Core
Requirement Title:	Time-based ordering of monitoring data entries	**Stakeholders Involved:**	UVT
Description:	Monitoring data must be ordered using a sequence of events using the time identification when things occurred (for example, a consistent timestamp in a fully distributed system without a central coordination service).		
Rationale / Goal:	While in general data are analysed based on an application-dependent context, for running trace checking or detecting events in the life cycle of an application, it is important that data are time-based ordered. This ordering will assure the specialised monitoring consuming services that the data reassembles the correct flow of the events logged by the application. This will furthermore allow to approximately identify when the event took place and enable the activation of the proper enforcement action.		
Acceptance Measures:	The granularity of the of the event window should not be greater than 10 seconds.		
Dependencies:	The deployment service should enforce an initial time synchronization at bootstrap time or, if not possible, the monitoring system should have the time synchronized across the platform.		

Requirement ID:	F_ROSAR.8	**Priority:**	Core
Requirement Title:	Persistent storage of monitoring data	**Stakeholders Involved:**	UVT, ICCS

Description:	Monitoring data should be stored in persistent storage which can be accessed and queried by other tools.
Rationale / Goal:	Historical monitoring data are a key requirement for creating application profiles and for defining normal operation as well as anomalous events for any application.
Acceptance Measures:	The existence of a distributed persistent store for short- and long-term data preservation.
Dependencies:	The telemetry platform persistent storage.

Requirement ID:	F_ROSAR.9	**Priority:**	Core
Requirement Title:	Data formatting and pre-processing	**Stakeholders Involved:**	UVT, ICCS
Description:	Monitoring tool could provide via a query language pre-processing and/or data formatting capabilities.		
Rationale / Goal:	By delegating some pre-processing and data formatting processes to the monitoring solution, SERRANO tools will have a more flexible and comprehensive direct overview of the monitored application. Also, the support for output formatting will also make data ingestion easier.		
Acceptance Measures:	The existence of a query interface to facilitate the data pre-processing in a programmatic manner.		
Dependencies:	All the components that need to consume monitoring data based on predefined filters.		
	Exchange events between the components of the hierarchical telemetry infrastructure (F_NCTFR.6).		

Requirement ID:	F_ROSAR.10	**Priority:**	Core
Requirement Title:	Access to the real time monitoring stream bus	**Stakeholders Involved:**	UVT, ICCS
Description:	Monitoring data are normally stored in a data warehouse or store from where the specialised application can consume and analyse it. In some cases, the real-time data are needed to properly detect events while the application is running.		
Rationale / Goal:	While the service assurance normally relies on the historical data for application behaviour profiling, real-time monitoring data are used to detect events when they occur or near their occurrence time frame. Therefore, using stream processing will enable a real time detection of events that might impact the application execution.		
Acceptance Measures:	Existence of a stream monitoring bus where data can be consumed on request.		

Dependencies:	Ability to centralize the monitoring information in a common view or place (F_NCTFR.8).
	Exchange events between the components of the hierarchical telemetry infrastructure (F_NCTFR.6).

Requirement ID:	F_ROSAR.11	**Priority:**	Core
Requirement Title:	Transprecise adaptation ML methods	**Stakeholders Involved:**	UVT
Description:	It is crucial to have transprecise adaptation for ML methods. This adaptation can be done via numerical precision adaptation, specialized libraries and model parameters.		
Rationale / Goal:	ML methods are suitable in most stream processing applications, as once models are trained, inference times and complexity is superior to most other methods.		
Acceptance Measures:	Implementation of transprecise mechanisms into the ML-based services to gain better performance.		
Dependencies:	N/A		

Requirement ID:	F_ROSAR.12	**Priority:**	Core
Requirement Title:	Transprecise Hyper-parameter optimization methods	**Stakeholders Involved:**	UVT
Description:	ML methods have a complex relationship when it comes to their parameters and performance. By using guided optimization methods for training parameters, we can create transprecise ML models, tailor-made for a large variety of scenarios not only in the case of precision.		
Rationale / Goal:	Tuning performance is an extremely complex task, requiring a deep knowledge of not only the application domain but also of the ML method itself. By utilizing these methods, relatively novice users or people unfamiliar with the target scenario can create transprecise ML methods.		
Acceptance Measures:	Implementation of transprecise Hyper-parameter optimisation methods for tuning the performance of the ML-based services.		
Dependencies:	N/A		

5.6 Service Orchestration Requirements

5.6.1 Purpose

The description of the three UCs highlighted some parameters of particular importance for the successful orchestration of the applications/services and especially their internal tasks/micro-services. Overall, the service orchestrator should facilitate the execution of each

application/service taking into account the particular requirements of each one of them and the resources that are currently linked with the SERRANO platform. In particular, it would be responsible for the translation of the high-level application/service requirements to intermediate or low-level resource requirements so that they can be effectively used by the resource orchestrator for the execution of the particular tasks/micro-services of each application with respect to the requirements specified. For this purpose, the service orchestrator should foresee potential application/service requirements that could arise during their execution taking into account the description provided about each application/service (including dependencies among internal tasks/micro-services) along with additional information collected from the analysis of the resource infrastructure (including network). All the aforementioned data should be available in machine processable format so that they can be efficiently used by the service orchestrator.

In the following tables we have summarized the requirements detected through the analysis of the three aforementioned UCs.

5.6.2 Requirements Analysis

Table 34: Analysis of service orchestration requirements

Requirement ID:	F_SOR.1	**Priority:**	Core
Requirement Title:	Application/Tool specification	**Stakeholders Involved:**	All
Description:	The decomposition of applications or services into independent well-defined tasks/micro-services could facilitate their execution with respect to the initial application/service requirements. For this purpose, the application profile should be clearly specified and the internal tasks/micro-services of the application/service should be identified and adequately represented in a machine-processable format..		
Rationale / Goal:	Each particular task has its own requirements and hence it could be independently executed taking into account all the internal and external parameters affecting their performance.		
Acceptance Measures:	The detection of the internal tasks of each application should be feasible. Also, the detection of the critical parameters of each task and the conditions they should satisfy is important.		
	The independent execution of each task should not violate the application/service requirements.		
Dependencies:	The rest of the ARDIA model(s) (F_SOR.2, F_SOR.3)		

Requirement ID:	F_SOR.2	**Priority:**	Core
Requirement Title:	Task description/metadata	**Stakeholders Involved:**	All

Description:	Each Task should be accompanied by additional information (i.e., Metadata and Requirements) so that the software agents can directly or indirectly identify the appropriate resources for their executions.
Rationale / Goal:	The adequate description of the critical parameters of each particular task and especially the conditions they should satisfy (i.e., task requirements) could facilitate their execution.
Acceptance Measures:	The description of the task is based on the elements of the ARDIA model.
	It provides adequate information for their proper scheduling.
Dependencies:	The profile of each application / service and their tasks and the user-defined requirements.
	The ARDIA model(s) that would be developed for capturing High-Level Requirements.

Requirement ID:	F_SOR.3	**Priority:**	Core
Requirement Title:	Resources description	**Stakeholders Involved:**	All
Description:	For each available resource, such as Cloud Providers, Fog nodes and Edge devices, all their capabilities should be accurately described in a machine processable format.		
Rationale / Goal:	The parameters directly (e.g., CPU power) or indirectly (e.g., network bandwidth) linked with each resource, affect the execution of each task.		
Acceptance Measures:	The parameters recorded can be efficiently used by the SERRANO platform and in particular the service orchestrator.		
	The execution of each task in the particular resource is in accordance with the parameters measured.		
Dependencies:	The Resource Orchestrator and the parameters affecting its process.		
	The ARDIA model(s) that would be developed for capturing Low-Level Requirements.		

Requirement ID:	F_SOR.4	**Priority:**	Core
Requirement Title:	Tasks orchestration	**Stakeholders Involved:**	All
Description:	The Service Orchestrator should be capable of detecting the appropriate resources and in particular the conditions they should satisfy so that they can efficiently execute the task with respect to the initial application/service requirements.		
Rationale / Goal:	Execution of each task in the appropriate computer environment is essential for satisfying the service/application requirements.		
Acceptance	The detection of the appropriate resources is driven by the description of each		

Measures:	task.
	The initial service/application requirements are satisfied.
Dependencies:	Application/Service/Task description (including dependencies) and Telemetry Data
	Mappings (directly or indirectly specified) among High- and Low-Level Requirements

Requirement ID:	F_SOR.5	**Priority:**	Core
Requirement Title:	Requirements forecasting	**Stakeholders Involved:**	All
Description:	The Service Orchestrator should be equipped with AI-techniques that enable the systems to predict potential service requirements in the future based on the formal description of each task/service/application along with additional information collected by the SERRANO platform (e.g., Telemetry Data) from the previous execution of each task/service/application.		
Rationale / Goal:	Knowing in advance the requirements of each particular task/service/application could result to better decisions made during their execution.		
Acceptance Measures:	There is no significant deviation from the initial requirements of each particular task during their execution.		
	The predicted requirements could lead to better decisions during their scheduling.		
Dependencies:	Application/Service/Task description (including dependencies) and Telemetry Data.		
	The Machine Learning / Data Mining algorithms that will be used.		

Requirement ID:	F_SOR.6	**Priority:**	Core
Requirement Title:	Security and privacy	**Stakeholders Involved:**	All
Description:	Any particular security/privacy requirement regarding the input/output data should be specified in advance.		
Rationale / Goal:	The security and privacy of input/output data could affect the decisions made by the orchestrator.		
Acceptance Measures:	The execution is compatible with the security and privacy data access policy.		
	The overall application/service requirements are satisfied.		
Dependencies:	Application/Service/Task description and especially the interactions among them.		
	Task Description and Application/Service Workflow.		

Requirement ID:	F_SOR.7	**Priority:**	Core
Requirement Title:	Orchestration as a service	**Stakeholders Involved:**	All
Description:	The functionality provided by the service orchestration should be available to the other components of the SERRANO platform through a web interface.		
Rationale / Goal:	The provision of the functionality of the AI-enhanced service orchestration though a well-defined software API could facilitate the integration of this component with the other ones of the SERRANO platform.		
Acceptance Measures:	Results provided should facilitate the execution of the applications/services/tasks.		
	Response Time should be acceptable.		
Dependencies:	Application/Service/Task description based on the ARDIA model.		
	Telemetry Data collection.		

5.7 Integration and Platform Development Requirements

5.7.1 Purpose

This section focuses on the requirements for the Platform Integration and Testing, for unifying the outcomes of the developed components and services to release the integrated SERRANO platform. The requirements defined below will support the development activities of the SERRANO components, as well as the overall integrated performance. More specifically, the requirements will be used as part of the guidelines (more information in the guidelines will be provided in D2.3) for the technological developments necessary for each UC, in terms of adaptation, customizations and further development of UC leaders' existing services and applications, continuously integrate mechanisms, models and frameworks developed in WP3, WP4 and WP5 to create the SERRANO platform, following a DevSecOps approach, and perform testing of the integrated platform.

5.7.2 Requirements Analysis

Table 35: Analysis of integration and platform development requirements

Requirement ID:	F_IPDR.1	**Priority**:	Core
Requirement Title:	Documentation of all integration points	**Stakeholders Involved:**	All
Description:	Each integration point should be well documented. Integration points refer to components or modules that will be part of or use the SERRANO platform. The documentation should cover cases of successful and unsuccessful communication.		

Rationale / Goal:	Minor incompatibilities might impede the integration process significantly. Without detailed documentation, it is usually hard to pinpoint those minor differences between expected and actual request and responses.
Acceptance Measures:	Documentation should cover all relevant endpoints adhering to well-known specification (e.g. OpenAPI v3.0).
	Documentation of integration points should refer to provided sample requests/responses in Postman collections or other similar formats (Insomnia collections, OpenAPI YML, etc.).
	Correctly implemented integration points that work as described in the documentation.
	Documented integration points should be known and agreed with involved members that use it.
Dependencies:	The detailed system design describing all integration points of the system, as described in D2.3 and D2.5 deliverables.

Requirement ID:	F_IPDR.2	**Priority :**	Core
Requirement Title:	Docker image of component to be used for creating a running container	**Stakeholders Involved:**	All
Description:	For every component, there should be at least one Docker image to be used for deploying this component in a container that will integrate with other components. In case multiple containers are required, there should be a docker-compose file.		
Rationale / Goal:	The integration process is greatly simplified by having components running in containers. For example, it allows the CI/CD process to deploy more than one component on the same VM without unexpected conflicts between them.		
Acceptance Measures:	One Dockerfile or one image per component is provided.		
	The container can run without issues. No errors should exist in the container logs, warnings can be present when not affecting the execution.		
	Example docker run command with required parameters (ports, environment variables, etc.).		
Dependencies:	A functional component already developed and documented (F_IPDR.1).		

Requirement ID:	F_IPDR.3	**Priority ::**	Essential
Requirement Title:	Gitlab as code repository	**Stakeholders Involved:**	All
Description:	Gitlab will be the code repository of choice for every component. The code should be placed under the corresponding work package and partner folder.		
Rationale / Goal:	CI/CD pipelines should be able to retrieve available code from a common		

	location. Gitlab provides integration with Jenkins that will perform the build of the components.
Acceptance Measures:	Available code of a component is placed into the corresponding Gitlab folder.
	Readme file or script that uses this code to build the component.
Dependencies:	Gitlab account for every user.

Requirement ID:	F_IPDR.4	**Priority :**	Essential
Requirement Title:	CI/CD guidelines and DevSecOps	**Stakeholders Involved:**	All
Description:	All partners should follow the CI/CD guidelines. Code development must follow the guidelines and methodology imposed by the CI/CD pipeline of processes and DevSecOps approach.		
Rationale / Goal:	To continuously integrate mechanisms, models and frameworks developed in SERRANO, we follow a DevSecOps approach. As part of the CI/CD process, security vulnerabilities can be detected. This process is automated and incorporated into the CI/CD pipeline. Also, code development should follow good practices as described in the DevSecOps approach, such as usage of SonarLint and implementation of security unit tests.		
Acceptance Measures:	Steps of CI/CD pipeline should be defined in a Jenkinsfile.		
Dependencies:	Well defined System platform architecture.		
	Well defined CI/CD guidelines based on the requirements of SERRANO platform integration. (F_IPDR.1, F_IPDR.2, F_IPDR.3, F_IPDR.5, F_IPDR.6, F_IPDR.7)		

Requirement ID:	F_IPDR.5	**Priority :**	Essential
Requirement Title:	Adequate unit, integration, and security tests	**Stakeholders Involved:**	All
Description:	Unit, integration, and security tests will be developed for all target platform components. Documentation and templates will be available to cover test developments.		
Rationale / Goal:	Code should be accompanied by unit tests to ensure that the implemented functionality matches the desired functionality. Security tests are highly desirable to highlight that security has been considered when the code was implemented. Integration tests should exist for all integration points that can be used by other components.		
Acceptance Measures:	Unit, integration, and security tests should be available in the code repository and ready to be added to the CI/CD pipeline steps. At least one integration test per component should be present. Unit tests should cover as much code as possible. At least one test per class (if		

	applicable) should be present.
Dependencies:	Defined SERRANO platform requirements.
	Risk Assessment as covered in deliverables D1.2, D1.3 and D1.7.

Requirement ID:	F_IPDR.6	**Priority :**	Desired
Requirement Title:	Code Quality and Security	**Stakeholders Involved:**	All
Description:	All partners are encouraged to adopt code quality inspection tools such as SonarLint and SonarQube to ensure code quality and security in the platform components as part of the DevSecOps approach. SonarQube will be provided by the CI/CD solution.		
Rationale / Goal:	To analyze source code to ensure code quality and find security vulnerabilities that make SERRANO platform components susceptible to attacks.		
Acceptance Measures:	Code smells and flagged vulnerabilities should be checked to ensure no major impact is expected. No security vulnerabilities should be present.		
Dependencies:	Defined SERRANO platform requirements.		
	Risk Assessment as covered in deliverables D1.2, D1.3 and D1.7.		

Requirement ID:	F_IPDR.7	**Priority :**	Core
Requirement Title:	Software abstraction layer for every hardware-dependent platform component.	**Stakeholders Involved:**	All
Description:	For all platform components, containerized or not and hardware-dependent, the responsible partner will ensure the presence of a software abstraction layer that will allow a seamless integration to the platform.		
Rationale / Goal:	Components that cannot be containerized as well as components that depend on FPGA/GPU/HPC or other hardware for acceleration could add complexity to their integration to the platform.		
Acceptance Measures:	Presence of abstraction layer that allows integration to platform.		
	Abstraction layer with well-documented API (F_IPDR.1).		
Dependencies:	A functional hardware-dependent or non-containerizable component (WP4 deliverables).		

6 Business Vision

6.1 The SERRANO platform goals and aspirations

The SERRANO platform integrates novel hardware and software technologies and methodologies towards an application-optimized and secure service instantiation. SERRANO pursues several research and innovation goals: (i) hardware acceleration for encrypted storage and scalable secure storage systems, (ii) multi-level approximate hardware accelerators for dynamic and input-driven approximations, (iii) trusted execution in multi-tenant environments by enhancing the low-level software stack, (iv) intelligent decision-making algorithms and orchestration mechanisms to move computation and data between the edge, cloud and HPC infrastructures. Achieving the above targets will guarantee SERRANO a strong exploitation potential, creating a plethora of SERRANO-based services. These can be placed in the centre of an innovative market ecosystem, driving business innovation, enterprise transformation and value (Figure 16).

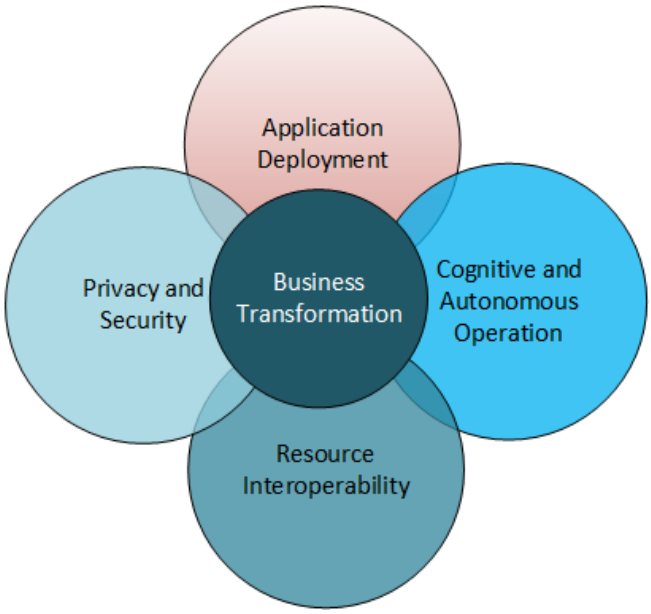


Figure 16: Business innovation in the cloud continuum

The SERRANO-enabled IaaS, PaaS, SaaS and other product variants services include: (i) secure, accelerated, federated infrastructures consisting of edge, cloud and HPC resources that also utilize novel cognitive mechanisms for the automation and optimization of their internal operations (SERRANO IaaS), (ii) domain specific and generic platforms for deploying and executing safety-critical, low-latency, data-intensive applications and other workflows (SERRANO PaaS), (iii) Cognitive Distributed Secure Storage as a Service (CDSSaaS) and Extreme Scale Analytics as a Service (ESAaaS) (SERRANO SaaS) and (iv) business processes (e.g. for fintech and manufacturing) as a service (SERRANO BPaas). Both PaaS and SaaS versions can also lead to the creation of new SaaS and BPaaS products that target generic or domain specific operations.

Cloud Service Providers (CSPs) like Amazon, IBM, Google may purchase SERRANO-based services to optimize the utilization and orchestration of their computing, storage and networking infrastructure, even transforming it to a SERRANO-ready service (SERRANO IaaS). In addition, SMEs/private cloud companies and telecom operators can provide their extra available fog/edge resources (i.e. the resources not utilized for the company's internal processes), in the form of SERRANO IaaS. SERRANO IaaS products can take a share of the cloud storage market, which was valued at $46.12 billion in 2019 and is expected to reach $222.25 billion by 2027, corresponding to a Compound Annual Growth Rate (CAGR) of 21.9% [7]. Other companies may opt for purchasing a SERRANO Platform as a Service (SERRANO PaaS) product variant for application development, so as to fulfil their performance and security requirements. Due to the plethora and diversity of security threats, many platforms for deploying and executing safety-critical workflows gain attraction. SERRANO, empowered by its technology advancements, will be among the major contenders of the cloud security market, which is expected to grow from $4.1 billion in 2017 to $12.7 billion by 2022, with a CAGR of 25.5% [8].

Also, SERRANO's business ecosystem envisages cooperation with other market stakeholders, who can enrich the proposed services, thus building B2B2X relationships and more complex business models in the long run. More specifically, SERRANO plans to interact with the consultancy services' industry for selling SERRANO Cognitive Distributed Secure Storage (CDSSaaS) and Extreme Scale Analysis as a Service (ESAaaS) and Business Processes as a Service (BPaaS) products to IT consulting and business analytics/intelligence companies. These will exploit industry specific expertise to provide sophisticated and tailor-made services to their customers.

6.2 Secure Storage Business Vision

Chocolate Cloud envisions the SERRANO platform as a set of integrated services that encompasses the entire cloud continuum. The Secure Storage Service is a key component in it and will be an important addition to the company's product portfolio. It is perfectly aligned with the company's vision of being a provider of file storage and sharing solutions that can be tailored to the requirements of both small and large enterprises.

Compared to our competitors, SkyFlok offers many benefits to existing customers (mostly small to medium companies) in terms of privacy and control over where the data are stored. SERRANO will augment this in two key ways. First, some of the stored data will be moved on premises, guaranteeing lower latency and greater privacy. Second, a standard S3-compatible API will allow simple integration with existing services and applications. Both benefits are crucial when offering the product to large companies and as such to fulfilling our vision.

The service by Chocolate Cloud is aligned with Mellanox when it comes to the relevance of secure storage. Figure 17 highlights the impact of cyber threats: major cost losses, stolen records, down time in IT infrastructure, malware implantation and other effects that impact not only the profitability of infrastructure and service providers but also generate a negative effect on the adoption of new digitalization technologies by citizenry.

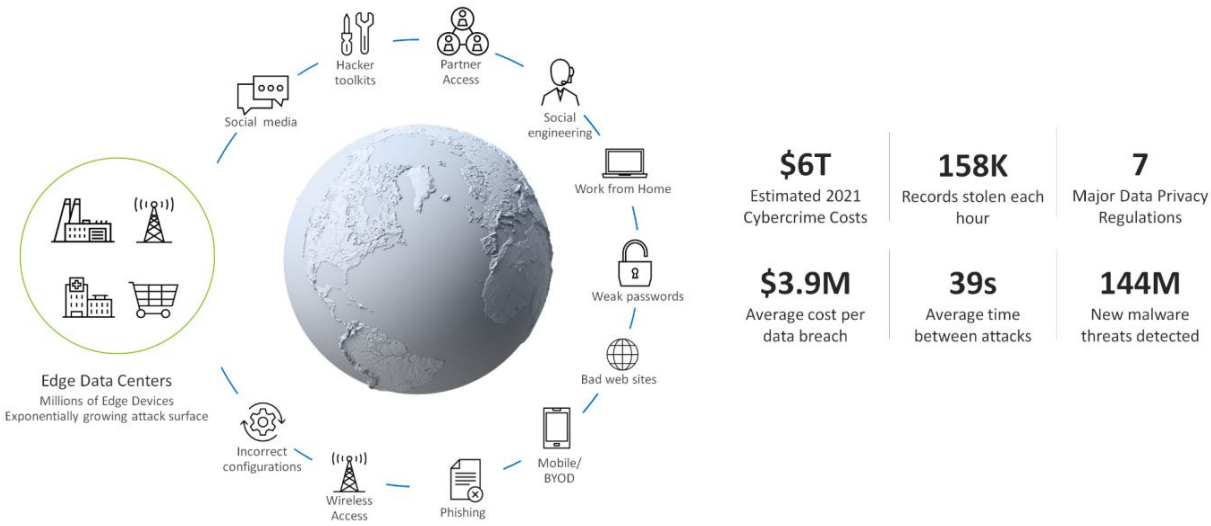


Figure 17: Security is becoming more relevant due to high business impact

When it comes to storage, Mellanox works in different technology fronts: adoption of securitized channels of communication and in-line encryption of data towards final storage. As much as both technologies use common blocks, the first refers to how information gets in and out of the platforms, whereas the second refers to how information is securely stored.

In SERRANO, the main bulk of Mellanox work will be around TLS, IPsec and NVMe accelerators; these are the key enabling technologies for enhanced secure storage. For example, NVMe enables transfer of data between a host computer and a target solid-state storage device or system over a network such as Ethernet, Fibre Channel and Infiniband, at high-throughput and low latency. These developments are not monetized directly through commercialization of specific software packages, but rather incorporated in public libraries for end-users to utilize depending on the UC. This allows Mellanox to market its network interface cards with extra-added features not attainable by competitors; currently Mellanox has a market share of >75% of adapters operating at 25G or above. This market share is sustained because of the open libraries that add value to the hardware.

6.3 Business Vision for Financial Services

Cloud computing is key enabler of innovation in financial services. It provides affordable technological solutions for the implementation, the deployment and the operation of complex financial applications which are integrated with multiple third-party cloud applications. Although the significant advances aimed at simplifying the development of cloud applications, still important barriers across the cloud continuum remain unaddressed. InbestMe envisions that SERRANO will further extend the cloud computing continuum in financial services which will facilitate the better and simpler cloud-to-cloud integration of financial services. Specifically, InbestMe envisions that SERRANO will enable:

1. The provision of investment management through FaaS cloud-functions by making the digitalization of complex business processes simpler.
2. The transparent execution of cloud services across different cloud providers including proprietary cloud solutions by providing unified control and administration of different cloud platforms.
3. The simplification of the lifecycle of investment management cloud applications including deployment and management by providing a turnkey or one-click deployment of applications which scale on demand.

The solution listed above will enable InbestMe to develop a Roboadvisor-as-a-Service (RASaaS) solution which:

1. will be offered to other FinTechs as a pay-per-use of each service; and/or
2. can entirely or partially deployed and integrated on premise; and/or
3. can be possibly extended with new functionalities through plug-ins as well as scale.

All the three features are of significant importance for the FinTech sector. (1) will increase the innovation and the competition through building new services on top of the investment management building blocks. (2) will enable financial institutions which, due to regulations and compliance, cannot share their services to integrate with RASaaS with their on-premise cloud applications. (3) will enable other service providers to further extend the RASaaS with new functionalities like in a marketplace model.

6.4 Business Vision for Anomaly Detection in manufacturing settings

Data processing and applied artificial intelligence are core business fields both for IDEKO and DANOBATGROUP, our strategic client, composed of several machine building companies.

Every day, more and more machine tool clients connect their machines to different data-gateway mechanisms that allow, both machine manufacturers and third-party agents, to develop valuable applications and services to leverage the data generated by their machines. IDEKO is constantly requested to provide added value services in three areas of interest: (i) machine condition, (ii) machining process and (iii) production.

Given this market demand, DANOBATGROUP is consolidating a *servitisation* strategy that is currently starting to yield results and is expected to grow rapidly. This *servitisation* strategy requires novel techniques that enable fast and reliable data processing tasks. In this context, DANOBATGROUP is developing a roadmap for data-based solution applications and services in the fields of predictive maintenance, remaining useful life assessment or critical components health monitoring. For this to succeed, DANOBATGROUP is heavily investing in data intensive applications and architecture research, and SERRANO directly impacts these critical areas of interest.

Moreover, IDEKO, aligned with the core business models of DANOBATGROUP, is currently developing a four-year Strategic Plan where Artificial Intelligence applied to machine data are a core field. Given that, IDEKO will be working side by side with DANOBATGROUP on this discipline, assisting them in getting a successful result on their *servitisation* strategy.

7 Conclusions

This deliverable presented an overview and in-depth analysis of the SERRANO UCs as well as the SERRANO platform requirements, as identified and assessed during the first iteration of the project's developments. The description of the UCs included their motivation, narration, involved stakeholders and a description of their high-level requirements as well as the applications/tools, tasks, and services involved in each UC and the associated workflow. Also, the data involved and the infrastructure to be linked with SERRANO.

The elicitation of the requirements was done according to the methodology and the good practices elaborated in section 3 of the document, following an iterative and collaborative interaction among the technical and UC partners of the project. The outcome of this work was a detailed and rich set of sixty five (65) functional and non-functional platform requirements, grouped in seven (7) main topics also including a general one, which shall be used as guidance for the design of the SERRANO platform architecture in deliverable *D2.3 - SERRANO architecture* and the more precise definition of the relevant components in *Work Packages 3-5*. Each of the requirements was described, prioritised according to its overall importance for the SERRANO platform and assigned a unique ID. Additionally, its acceptance measures and dependencies with other requirements or SERRANO platform components were provided.

Furthermore, in this deliverable, a number of success criteria associated with the outcome of the UCs as well as the relevant KPIs were introduced. The updated and final version of these KPIs, the UCs and the platform requirements shall be provided in *D2.4 Final version of SERRANO use cases, platform requirements and KPIs analysis.* The KPIs shall be further elaborated and used for the formulation of the SERRANO KPIs and evaluation methodology as part of deliverable *D6.2 - KPIs and evaluation methodology* and finalized in *D6.6 - Final version of KPIs and evaluation methodology* of Work Package 6.

Finally, the deliverable presented the first version of each UC's business vision as well as the overall SERRANO platform goals and aspirations, which shall serve as input to deliverable *D7.4 - Business modelling, exploitation and sustainability plans and activities*.

8 References

[1] Analyzing and Defining Requirements, in Systems Engineering Guide. McLean, VA:MITRE. Available online: https://www.mitre.org/publications/systems-engineering-guide/se-lifecycle-building-blocks/requirements-engineering/analyzing-and-defining-requirements

[2] Hooks, I. (2005). Writing Defect-Free Requirements. Boerne, TX: Compliance Automation.

[3] ISO/IEC, "ISO/IEC 25010: 2011 Systems and software engineering--Systems and software Quality Requirements and Evaluation (SQuaRE)--System and software quality models, (2011).

[4] HLRS Systems HPE Apollo (Hawk) (2021). Available online: https://www.hlrs.de/systems

[5] B. Dick, T. Bönisch, B. Krischok, Hawk Interconnect Network, HLRS, (2020), Available online: https://kb.hlrs.de/platforms/upload/Interconnect.pdf

[6] Quobyte, "CASE STUDY EFFICIENT OPERATIONS AND FAST DATA ACCESSENABLE BOTTLENECK-FREE RESARCH AT HLRS", Available online: https://www.quobyte.com/case-studies/hlrs

[7] “Cloud Storage Market by Component (Solution and Services), Deployment Type (Private, Public, and Hybrid), User Type (Large Enterprises and Small and Medium Enterprises), and Industry Vertical (BFSI, Government & Public Sector, Healthcare, IT & Telecom, Retail, Manufacturing, Media & Entertainment, and Others): Global Opportunity Analysis and Industry Forecast, 2020–2027,” Available online: https://www.alliedmarketresearch.com/cloud-storage-market

[8] “Cloud Security Market by Service Type (IAM, DLP, IDS/IPS, SIEM, and Encryption), Security Type, Service Model (IaaS, PaaS, and SaaS), Deployment Type (Public, Private, and Hybrid), Organization Size, Vertical, and Region - Global Forecast to 2022,” Available online: https://www.marketsandmarkets.com/Market-Reports/cloud-security-market-100018098.html