Project Title	High-performance data-centric stack for big data applications and operations
Project Acronym	BigDataStack
Grant Agreement No	779747
Instrument	Research and Innovation action
Call	Information and Communication Technologies Call (H2020-ICT-2016-2017)
Start Date of Project	01/01/2018
Duration of Project	36 months
Project Website	http://bigdatastack.eu/

D2.1 – State of the art and Requirements analysis - I

Work Package	WP2 – Requirements, Architecture & Technical Coordination
Lead Author (Org)	Orlando Avila-García (ATOS)

The work described in this document has been conducted within the project BigDataStack. This project has received funding from the European Union's Horizon 2020 (H2020) research and innovation programme under the Grant Agreement no 779747. This document does not represent the opinion of the European Union, and the European Union is not responsible for any use that might be made of such content.

Contributing Author(s) (Org)	Paula Ta-Shma, Yosef Moatti (IBM), Everton Luís Berz (NEC), Ana Juan Ferrer, Ana Belén González Méndez, Bernat Quesada, Alberto Soler (ATOS), Stathis Plitsos (DAN), Konstantinos Giannakakis, Amaryllis Raouzaiou (ATC), Pavlos Kranas (LXS), Sophia Karagiorgou, Panagiotis Gouvas, Anastasios Zafeiropoulos (UBI), Dimitris Poulopoulos, Giorgos Kousiouris, Stavroula Meimetea, Dimosthenis Kyriazis (UPRC), Valerio Vianello (UPM), Richard McCreadie (GLA), Gal Hammer, Miki Kenneth(RHT), Nikos Drosos (SILO), Maurizio Megliola (GFT)
Reviewer(s) (Org)	Pavlos Kranas (LXS), Dimitris Poulopoulos (UPRC), Yosef Moatti, Eliot Salant (IBM)
Due Date	30.06.2018
Date	12.07.2018
Version	1.0

Dissemination Level

X	PU: Public (*on-line platform)
	PP: Restricted to other programme participants (including the Commission)
	RE: Restricted to a group specified by the consortium (including the Commission)
	CO: Confidential, only for members of the consortium (including the Commission)

Versioning and contribution history

Ver.	Date	Author	Notes
0.1	22.05.2018	Orlando Avila-García (ATOS)	Template for contributions.
0.2	25.05.2018	Orlando Avila-García (ATOS)	Adding contributions to Section 8 from UPM, GLA, LXC, UNIPI; and Section 4 from ATOS.
0.3	29.05.2018	Orlando Avila-García (ATOS)	Adding contributions to Section 8 from UPRC, ATC, NEC, IBM; Section 4 from DAN; and Section 2 - Introduction.
0.4	05.06.2018	Orlando Avila-García (ATOS)	Adding contributions to Section 8 from UBI; Section 6 from ATOS. Adding new Section 3 – Business Goals and Requirements from WP7 partners.
0.5	12.06.2018	Orlando Avila-García (ATOS)	Adding corrections to Section 3 suggested by ATOS, changes on Section 4.2 (Connected Consumer) made by ATOSWLI, new role on Section 5 suggested by UPRC, new system requirements at Section 6.6 from UPRC, new technology requirements at Section 7.12 from UPRC, minor corrections in the Introduction from ATC, new system requirements at Section 6.5 from ATC, new technology requirements at Section 7.13 and 7.15 from ATC.
0.6	19.06.2018	Orlando Avila-García (ATOS)	Adding new system requirements at Section 6.5 from UPRC, software requirements at Sections 7.4, 7.8 and 7.11 from UPRC, software requirements at Section 7.10 from UPM, software requirements at Section 7.6 from LXS, system requirements at Section 6.2 from LXS, software requirements at Section 7.4 and 7.5 from GLA. Correction sin Sections 8.5 and 8.9 from UPRC.
0.7	25.06.2018	Orlando Avila-García (ATOS)	Adding executive summary and conclusions, refining introduction and formatting references. Adding data toolkit system and software requirements at 6.1, 6.4 and 7.14. Adding baseline technology description at 8.1 and 8.5.2 from ATOS.
0.8	28.06.2018	Orlando Avila-García (ATOS)	Adding contributions to system requirements at 8.1, 8.2 and 8.8, and software requirements at 7.1, 7.2 from RHT. Adding CEP metrics in section 8.11. Adding from SILO. Adding one new system requirement at 6.6 from UPRC. Adding minor amendments at section 7.9, 7.16, 8.3, 8.9 and 8.16 from SILO, and software requirements at section 7.3 from SILO. Adding section 4.3 (IMB use cases and scenarios) from GFT. Enhancements for sections 8.10 and 8.13 from IBM.
0.9	29.06.2018	Pavlos Kranas (LXS), Dimitris Poulopoulos (UPRC)	Minor amendments after WP2’s internal review.

1.0	12.07.2018	Yosef Moatti (IBM), Eliot Salant (IBM)	Minor amendments after Project’s internal review.

Table of Contents

List of tables

List of figures

1 Executive Summary

This is the first deliverable of a series of deliverables (i.e. updated versions will follow) that identifies and specifies the technical requirements for BigDataStack environment and tracks them during the project lifecycle. A top-down approach is followed with respect to the user requirements, which was collected through the BigDataStack's **use case providers**. This is complemented with a bottom-up approach aiming to identify, collect, and analyse technical requirements emerging from the technical work packages at both system and software technology levels. However, at this stage of the project, the analysis of technical requirements has focused on the latter as BigDataStack's **technology providers** have started by analysing the role their respective technologies may play in the BigDataStack vision.

The analysis has produced measurable and unambiguous requirements, which will inform and drive architectural design decisions at different levels of the BigDataStack architecture: capabilities, services and technologies. They will also be tracked against the research, architecture and implementation work during the project lifetime to ensure that the BigDataStack environment complexity is fully addressed and properly considered. To contextualize this analysis, this deliverable also introduces the state-of-the-art (baseline) technologies that may play a role in BigDataStack. In fact, such descriptions are a refinement of the internal joint report (with the architecture work) completed at M3. Moreover, it does not simply state some state-of-the-art technologies but rather links them under the context of BigDataStack and what BigDataStack can get from them as a baseline.

During the project, the BigDataStack technologies and requirements are expected to continue being investigated to ensure that the objectives and innovations of the project are valid, and the architecture and development work is alighted with those goals and requirements. Two more iterations of the requirements and state-of-the-art analysis will be carried out and their results will be documented in two new versions of this deliverable, at M11 and M22.

2 Introduction

The main purpose of this deliverable is to describe the initial set of measurable and unambiguous **business and technical requirements for the BigDataStack environment**. This set will be further tracked and validate the architecture development and implementation during the project.

This report represents the first official deliverable of BigDataStack project's Task 2.1, whose main goal is to collect the user and technical requirements and track them during the project. The outcomes of this task are a **key input for the architecture and the research activities** of the project. An internal report was completed in M3 collecting some information that has been further refined later and compiled in this deliverable (M6). Task 2.1 started at M1 and is expected to continue working on requirements and state-of-the-art analysis until M22. In fact, two more deliverables refining this deliverable (D2.1) are expected to be produced in upcoming months:

- D2.2: Requirements & State of the Art Analysis II (M11)
- D2.3: Requirements & State of the Art Analysis III (M22)

Elicitation and analysis of the requirements have been performed considering the needs and concerns coming from the current communities and end users related to BigDataStack's use case and technology providers. Therefore, the analysis specifies not only ***use case requirements*** (called "stakeholder requirements" by ISO/IEC/IEEE 29148:2011 [41]) but also ***technical requirements*** (called either "system requirements" or "software requirements" by the same norm). Due to the early stage of the project in which this deliverable is produced (M6), the complexity of the envisioned architecture and the wide variety of technologies involved, this first iteration of the state-of-the-art and requirements analysis focuses on use case requirements and software technology requirements. System requirements, which sit in the middle of use case and software requirements and are directly related to the BigDataStack platform's main component and services, will be the focus of future iterations.

To contextualize the analysis of requirements, Section 3 (Business Stakeholders and Goals) describes the BigDataStack platform **stakeholders, business model, expected business outcomes and business goals**. To better understand the software technology requirements, this deliverable also includes an introduction to the state-of-the-art (baseline) technologies that are expected to be relevant for BigDataStack (Section 0). In fact, Sections 3 and 0 are a refinement of the sections dedicated respectively to business goals and baseline technologies in the internal joint (with architecture) report completed at M3.

This document is organized as follows: Section 2.1 explains the requirements engineering method that has been followed; Section 0 specifies business scenarios and use case requirements related to three use cases: ***real-time ship management***, ***connected consumer*** and ***intelligent multi-channel banking***; Section 0 introduces the most relevant roles that the actors (stakeholders) interacting with the platform may take; Section 0 and 0 specify technical requirements, that is, system and software technology requirements, respectively; Section 0 describes baseline technologies relevant for BigDataStack; and finally, Section 0 draws some conclusions.

2.1 Method

The requirements engineering method will follow ISO/IEC/IEEE 29148:2011[1] which describes two main processes or practices to be executed in an iterative and recursive fashion:

Process	Purpose	Output
Stakeholder Requirements Definition Process	To define the requirements for a system that can provide the services needed by users and other stakeholders in a defined environment.	Stakeholder Requirements Specification *(StRS)*
Requirements Analysis Process	To transform the stakeholder, requirement-driven view of desired services into a technical view of a required product that could deliver those services.	*System Requirements Specification (SyRS)*
		Software Requirements Specification (SRS)

Table 1 – Requirements engineering processes

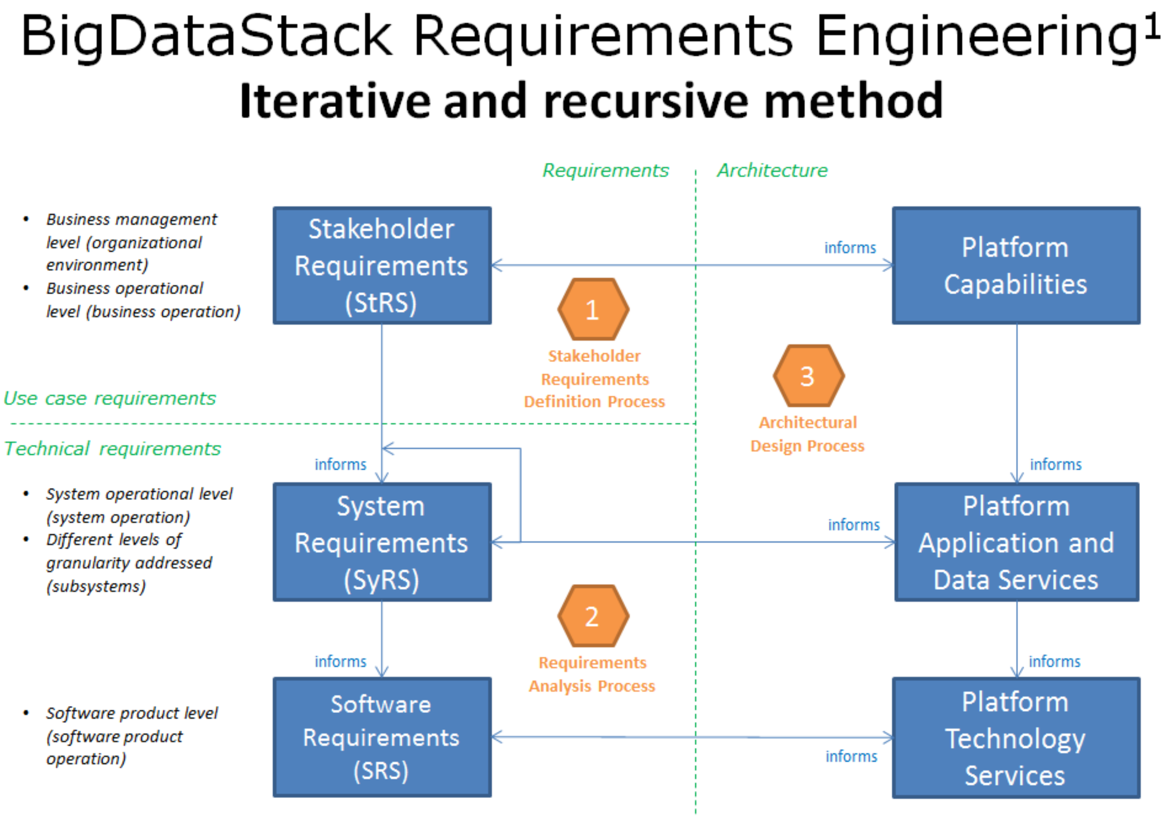

BigDataStack Requirements Engineering1
Iterative and recursive method
Requirements
Architecture
Business management level (organizational environment)
Business operational level (business operation)
Stakeholder Requirements (StRS)
informs
Platform Capabilities
1
Stakeholder Requirements Definition Process
3
Architectural Design Process
Use case requirements
Technical requirements
informs
informs
System operational level (system operation)
Different levels of granularity addressed (subsystems)
System Requirements (SyRS)
informs
Platform Application and Data Services
2
Requirements Analysis Process
informs
informs
Software product level (software product operation)
Software Requirements (SRS)
informs
Platform Technology Services

Figure 1. Requirements engineering method

The work products are requirements specifications at three levels of detail, which serve as input to different practices or stages in the architectural design process. The following table describes each of those levels (from ISO/IEC/IEEE 29148:2011 [41]), including the architecture domain whose decisions are informed by them.

[1] International Organization for Standardization, “ISO/IEC/IEEE 29148:2011 – Systems and software engineering — Life cycle processes — Requirements engineering,” ISO/IEC/IEEE, Nov. 2011.

Work product	Acronym	Description	Informed architecture domain
Stakeholder Requirements Specification	StRS	It identifies stakeholders, or stakeholder classes, involved with the system throughout its life cycle, and their needs, expectations, and desires. It analyses and transforms these into a common set of stakeholder requirements that express the intended interaction the system will have with its operational environment and that are the reference against which each resulting operational service is validated. It specifies: - The required system characteristics and context of use of the product (platform) business functions and services, and operational concepts are specified. - The constraints on a system solution are defined. - Traceability of stakeholder requirements to stakeholders and their needs is achieved. - The stakeholder requirements are defined from the stakeholder's perspective. - Stakeholder requirements for validation are identified.	Platform Capabilities (business architecture)
System Requirements Specification	SyRS	Technical specifications for the selected system of-interest and usability for the envisaged human-system interaction. It characterises system requirements because: - It represents a system (including interfaces of functions and services) that will meet stakeholder requirements. - Allows lower levels of granularity (recursion), i.e., subsystems. - Does not imply any specific implementation. It specifies: - The future system requirements from the domain perspective, background information about the overall objectives for the system, and its target environment. - A statement of the constraints, assumptions and non-functional requirements. - Measurable system requirements specifying, from the supplier's perspective, what characteristics and with what magnitude it is to possess to satisfy stakeholder requirements.	Platform Applications and Data Services Architecture
Software Requirements Specification	SRS	A specification for a software product, program, or set of programs) that performs certain functions in a specific environment.	Platform Technology Architecture

		The SRS may be written by one or more representatives of the supplier, one or more representatives of the acquirer, or by both. Typically, - there will be a requirement specification that will state the interfaces between the system and a software portion; - it will place external performance as well as functionality requirements upon the software portion; - it defines all the required features (e.g., functions) of the specified software product to which it applies; and - it documents the conditions and constraints under which the software portion must perform, and the intended verification approaches for the requirements.	

Table 2 – Levels of requirement specification

Finally, to identify requirements from all stakeholders' point of view, we have taken inspiration from the *TOGAF® Series Guide*[2]*: Business Scenarios* method to shed light on the key business requirements and indicate the implied technical requirements for IT architecture of BigDataStack. This is a technique to validate, elaborate, and/or change the premise behind an architecture effort by understanding and documenting the key elements of a Business Scenario in successive iterations.

Finally, to better formalize the requirements, we use the following attributes:

- **Level of detail**: Following the use of ISO/IEC/IEEE 29148:2011 (see Section 2.1 Methodology), we use the following levels: Stakeholder, System and Software (i.e., technology details).
- **Type**: Types of requirements are functional: FUNC (function), DATA (data); and non-functional: L&F (Look and Feel Requirements), USE (Usability Requirements), PERF (Performance Requirements), ENV (Operational/Environment Requirements), and SUP (Maintainability and Support Requirements).
- **Priority**: Requirements can have different priorities: MAN (mandatory requirement), DES (desirable requirement), OPT (optional requirement), ENH (possible future enhancement).

[2] https://publications.opengroup.org/g176

3 Business Stakeholders and Goals

This section aims to identify the business goals to address in the elicitation of requirements and architecture specification of BigDataStack results. The identification of business needs and requirements will help to implement a solution that meets stakeholders´ expectations and allows a better market positioning for a future exploitation.

It is worth noting that a preliminary wide-reaching Market Analysis will be delivered in M18[3] of the project, which will be used to confirm that the business goals envisioned in this phase of the project, as well as the business model and stakeholders, are valid.

3.1 Stakeholder Categories

As defined in ***BDVA SRIA Agenda***[4], the following key stakeholders are the main categories of actors along the Big Data Value chain: *User Enterprises*, *Data Generators and Providers*, *Technology Providers* and *Service Providers.* These categories are described in the following table, including the BigDataStack platform "side" they will play in: supply versus demand, or solution provider versus consumer.

Id	Name	Side	Description
STA-01	User Enterprises	Demand side	These are, for example, enterprises in all domains and of all size that want to improve their portfolio using Big Data technology.
STA-02	Data Generators and Providers	Supply side	Create, collect, aggregate, transform and model raw data from heterogeneous sources and offer it to customers.
STA-03	Technology Providers	Supply side	Provide tools and/or platforms that offer data management and analytics tools to extract knowledge from data, curate and visualize it.
STA-04	Service Providers	Supply side	Develop Big Data applications on top of the tools and platforms to provide services to user enterprises.

Table 3 – Stakeholders

In the BDVA SRIA Agenda and in many digital media, workshops, congresses, etc., the big data stakeholder ecosystem has concerns about problems like those of the main BigDataStack stakeholders, which may cause a slower uptake of big data applications and solutions.

Stakeholder Categories	Concern	Description
STA-01 STA-02 STA-03 STA-04	Privacy and Security	Potential data users are worried about privacy and security of their data. Due to velocity and volume, different data locations and different type of data (including not only personal data, but also sensitive business data), a robust data protection mechanism is needed. For businesses this is a key point, since 89% of companies avoid doing business

[3] The result of this preliminary wide-reaching Market Analysis will be included in the deliverable **D7.2 - *Exploitation plan and business potential*** [ATOS, Report, Public, M18].

[4] http://www.bdva.eu/sites/default/files/EuropeanBigDataValuePartnership_SRIA__v3.pdf

		with companies that they believe do not protect their privacy[5].
STA-02	Cost	The high data volume and quick scalability of big data projects make difficult to foresee of cost management for enterprises. New provider monetization models are emerging to create innovative cost-effective solutions for big data users to control costs as far as possible.
STA-02	Integration with existing systems	The integration of big data technologies with existing systems is a main question for companies planning to implement big data solutions. Companies know that changing operational/process company systems is a major issue since it leads additional costs, new personal training, etc. But the challenge is not only before the big data implementation, since companies must be prepared to make necessary changes to derive business value from big data, which probably will lead to changes in existing systems.
STA-03	Scalability and performance	Nowadays, companies are increasingly using big data in their business and must deal with a large amount of data. Service providers have to offer attractive cost-effective services to their clients to address this problem.
STA-02 STA-03	Heterogeneity of data and data sources	The emergence of IoT has incorporated a new type of data with different existing ones: data-in-flight from sensors, mobiles, etc. which needs a new management data model and capabilities
STA-03 STA-04	Different analytics capabilities	A key challenge for business is to identify clear business objectives, and this will not be the same for the different sectors, so application service providers need to develop different analytics capabilities to address clients in all domains
STA-03 STA-04	Lack of talent	There are not enough skilled people and new training requires time and money, so providers need big data tools ease to use, to deliver new services in a short time to market.

Table 4 – Stakeholder concerns

3.2 Business Model

To meet the needs of the stakeholder ecosystem, the BigDataStack platform should support whole Big Data management and analytics products and services, addressing needs of data operations and data applications in a Data as a Service (DaaS) model. The table below depicts the envisioned BigDataStack platform value proposition for customers at each side of it (demand and simply sides) as well as the revenue model proposed for them[6]:

[5] https://www2.deloitte.com/content/dam/Deloitte/ca/Documents/Analytics/ca-en-analytics-ipc-big-data.pdf

[6] These assumptions will be deeply explored in the Market Study (***D7.2. Exploitation plan and business potential***) and a best-suited business model will be deployed based of the market analysis result.

Products and Services	Revenue model	Customer	
Turn-Key Big Data management and analytics solutions	Pay-as-you-go	Demand side	**Enterprises of all sizes and all sectors** that want to increase the knowledge or operational efficiency of their business and/or enhance their business offering by using Big Data Analytics and wish a whole outsourcing solution for the management of the data path operation.
Development of different Big Data management and analytics solutions	Pay-as-you-go	Supply side	**Solution Providers** who want to make use of BigDataStack tools to enhance their Big Data products and services, including *technology*, *applications* and *data* offerings.

Table 5 – Preliminary business model

3.3 Business Outcomes

In the proposal stage of the project, a deeper study of the main actors and stakeholders related to BigDataStack solutions was carried out. That study has been enhanced and is summarized in the following table, where the benefits for each stakeholder of using BigDataStack results are included:

Side	Stakeholder Category	Stakeholder	Description
Supply side	**STA-03**	Infrastructure providers	Offer infrastructure solutions to big data needs through efficient and performant management of all resources.
	STA-02	Data providers	Offer cleaned, modelled, stored and analysed data.
	STA-04	Application providers	Provide data-intensive applications with guarantees.
	STA-03	Data practitioners	Develop enhanced algorithms and offer them.
Supply side	**STA-03**	Infrastructure brokers	Act as second-level entities that take advantage of the BigDataStack data-driven infrastructure management solutions from infrastructure providers.
	STA-02	Data aggregators and data resellers	Act as second-level entities (following data providers) that take advantage of the monetization model of Data as a Service according to their business models and goals.
	STA-04	Marketplace owners	Act as second-step entities that take advantage of data-intensive application provisioning by application providers.
Demand side	**STA-01**	Citizens	Use applications, services and products with guaranteed levels of quality.

	STA-01	SMEs and big industries	Satisfy their internal data needs to develop new offering and/or streamline operations by utilizing BigDataStack services offered by application and data providers.
	STA-01	Public organizations	Using BigDataStack for handling data.
	STA-01	Entrepreneurs	Developing, deploying and using data-intensive and/or data-driven applications to power their products or services by utilizing BigDataStack services offered by technology and data providers.
	STA-01	Decision makers	Driving business decisions based on accurate, timely, meaningful data and analytic insights.

Table 6 – Stakeholder requirements

3.4 Business Goals

Business goals are often called “vision requirements.” These are top-level requirements appear first, and to which all the other requirements must be subordinated, to successful market uptake. In this stage of the project, the following business goals have been identified:

Field	Description
Id	BG1
Short Name	**Privacy and Security**
Description	BigDataStack will propose an architecture that enables security and privacy aspects, and will be oriented toward the compliance with data protection regulations.
Rationale	Ensure the protection of personal data and business data.
Involved Stakeholders	All stakeholders

Table 7 – Privacy and security (business goal)

Field	Description
Id	BG2
Short Name	**Attractive revenue and business model**
Description	BigDataStack envisions a **Pay-as-you-go** as revenue model, delivering a cost-effective service for different costumers and looking for strong marketplace positioning.
Rationale	Deliver cost-effective solutions for the whole stakeholder ecosystem as Data as a Service solution.
Involved Stakeholders	All stakeholders

Table 8 – Attractive revenue and business model (business goal)

Field	Description

Id	BG3
Short Name	**High performance, scalability and sharing**
Description	BigDataStack will introduce an architecture that enables real-time data-driven management decisions and will provide a performant, scalable, flexible and dependable environment for the efficient delivery of distributed data operations, data- and storage- intensive applications. The performance and optimization will be achieved by basing all infrastructure management decisions on the data aspects.
Rationale	Data management of different data from several sources, including data at rest and in flight.
Involved Stakeholders	*Infrastructure providers, Data providers, Application providers, Data practitioners, Citizens, SMEs and Large industries, Public Organisations, Entrepreneurs, Decision makers.*

Table 9 – High performance, scalability and sharing (business goal)

Field	Description
Id	BG4
Short Name	**Product integration**
Description	BigDataStack offers a solution catalogue for providers, which can be used to manage the complete data path or only to address parts of a provider's whole solution. For end users, BigDataStack-based turn-key solutions will facilitate the integration of analytics in their businesses.
Rationale	Integration with other systems in end user companies and with other analytic tools for providers.
Involved Stakeholders	All stakeholders

Table 10 – Product integration (business goal)

Field	Description
Id	BG5
Short Name	**Different analytic capabilities**
Description	BigDataStack will validate its solutions in three commercial cases in the maritime, market and financing domains; this will provide a key expertise to BigDataStack to offer guaranteed **turn-key big data solutions** in other domains.
Rationale	Deliver successful solutions for major challenges in main sectors.
Involved Stakeholders	*Citizens, SMEs and Large Industries, Public Organisations, Entrepreneurs, Decision makers.*

Table 11 – Different analytic capabilities (business goal)

Field	Description
Id	BG6
Short Name	**Ease of use**

Description	BigDataStack will put emphasis on **usability** through data toolkits and visualization environments. Its solutions will include mechanisms for deployed data path operations to become faster.
Rationale	Reduce time to market and cost for new data applications.
Involved Stakeholders	*Infrastructure providers, Data providers, Application providers, Data practitioners.*

Table 12 – Ease of use (business goal)

4 Use Case Requirements and Scenarios

This section presents the business usage scenarios and initial requirements elicited from each of the three business use cases of the BigDataStack project. These requirements should be considered as ***Stakeholder Requirements*** focused on specific solutions as required by specific ***User Enterprises*** (see Table 3 – Stakeholders). The business scenarios are representative of a significant business need or problem, and enables *data, technology and service providers* to understand the value to the customer organization of a developed Big Data solution.

Each scenario describes the different usage from a use case perspective at a high-level description. It is not the intention to define the complete and detailed scenarios needed for the development of the solution, rather that the descriptions are more related with defining the behaviour and the scope to identify the necessities and align the architecture definition with the uses case from the beginning of the project. Moreover, the scenarios are by no means complete, as the project has two additional iterations to upgrade and refine them, however, they provide an overview on the main behavioural patterns involving the different actors and aims to define and align the initial design of the architecture (D2.4). Scenario descriptions are complemented with UML Use Case Diagrams to identify the different actors, prerequisites and the description of the behaviour.

Each use case can identify one or more scenarios depending on the complexity or the scope of the definition. For instance, on one side, the necessity for the analysis of the data services and data-intensiveness of the provision (at the dimensioning phase), and on the other side, the scenario for the operational phase where the defined Quality of Service (QoS) and rules should be applied. Thus, this can be described only in one scenario (more complex) or can be split into two scenarios differentiating clearly the objectives, the behaviour and the actors. It should be the decision of each use case provider to take the approach that best suits their purpose.

4.1 Real-time Ship Management

4.1.1 Introduction

This section refers to the use case of Real-time Ship Management (RSM): Maintenance and spare parts inventory planning & dynamic routing. The usage scenarios, that is, a higher-level representation of functional requirements (Section 3.1.2), along with a detailed description of use cases (Section 3.1.3) will be presented.

4.1.2 Scenarios

This case addresses two key challenges in the ship management domain: (i) predictive maintenance combined with spare parts inventory planning, and (ii) dynamic routing. In recent years, increasing fuel prices, depressed market conditions and environmental issues such as emissions from ships, have brought a new perspective to ship routing. Besides being cost-efficient, a ship also must be environmentally friendly with regards to its emissions.

One of the project partners faces similar challenges: DANAOS - a leading international maritime player with more than 60 containerships, transports millions of containers, sails millions of miles to thousands of ports, and consumes millions of tons of fuel oil. Each year, DANAOS senior management, investors, and customers evaluate performance after each voyage and demand the highest level of operational quality. Ship engines and other relevant machinery need to achieve high availability not only to deliver transport services (and thus ensure availability of resources) but also for operational safety, occupational health and environmental impact purposes. High availability of ship engines and machines can only be achieved if they are kept under proper conditions using applicable maintenance strategies, thus the monitoring of machinery has become even more critical to meet the maintenance requirements and achieve predictive maintenance. The latter is based on data that are exploited to estimate the type of failure and time to failure.

An additional problem is the limited availability of spare parts, resulting in expediting inbound replenishment shipments. If the spare parts planning and inventory management processes cannot cope with the unpredictability of the need for parts, then the operation may be starved of critical parts, yet may be flooded with other parts which are not frequently required, resulting in lower productivity due to additional downtime and higher holding costs due to excess inventory, respectively. Spare parts inventory management in relation to maintenance is a complex process because it involves hundreds of parts for a single engine, some of which may have a high level of demand per month whereas some may have a demand of few units per year.

We discuss two different but complementary scenarios: (i) ***monitoring and predictive maintenance*** and (ii) ***requisition of a spare part and dynamic routing to the closest port where this part is available***. The following tables describe in more detail these two scenarios.

Section	Description
Id	**SCE-RSM-01**
Title	Monitoring and predictive maintenance
Description	A vessel must complete its route within a time-frame. When a part of the main engine fails unexpectedly, the ship risks staying off-hire. This can be very damaging to a shipping company, as chartering revenues decrease, while replacing a spare part immediately increases cost. Thus, identification of potential failure allows timely ordering, or even replacement of spare parts before failure. The main engine, posing the highest risk, consists of various spare parts depending on many parameters. Thus, it is difficult to accurately predict failures. If false alarms occur, the operating costs increase, as ordering of unnecessary parts is not optimal.
Actors	Coordinator, Fleet manager
Objectives	- Monitoring the main engine of a vessel. - Notification for an upcoming malfunction. - Minimization of machinery failures that cause the ship to go off-hire.
Pre-conditions	Monitoring and predictive maintenance of the main engine takes for granted a full-scale dataset with measurements from the main engine, along with other factors that may influence the performance of the main engine, such as weather conditions, hull condition, power consumption, fuel quality etc. Furthermore, a recorded history of malfunctions is required.
Process Description	Shipping companies nowadays work preventively against malfunctions via a planned maintenance scheme and a condition-based maintenance scheme. Planned maintenance is performed with the help of a planned maintenance system (PMS) that informs the engineers for actions to be taken for

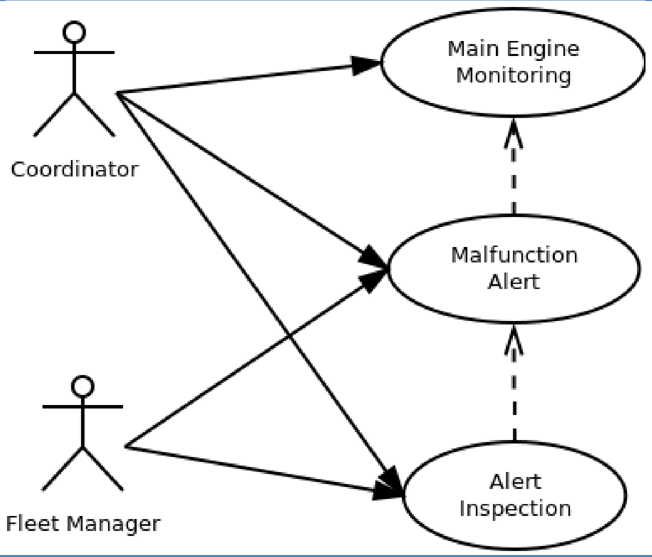
	maintenance from a main-engine component down to a spare-part-level. For example, the change of lube oils or a piston component after a defined period. Condition-based maintenance is performed either separately from planned maintenance via a pure human decision and interaction scheme, or can be included in the PMS. For example, if the tubes of the air-cooler have been cleaned, the air-filter should be replaced.
Variations	In this case, preventive maintenance is addressed as the remaining cases of maintenance that are not included in a condition-based or preventive maintenance scheme. If these two categories are excluded, we discuss about malfunctions that occur unexpectedly, thus should be handled in a different manner.
Post-condition	If a malfunction pattern is identified, the actor is informed via an alert.
Diagrams	 Coordinator Fleet Manager Main Engine Monitoring Malfunction Alert Alert Inspection

Table 13 – Monitoring and predictive maintenance scenario description (scenario)

Section	Description
Id	**SCE-RSM-02**
Title	Requisition and dynamic routing
Description	Once a malfunction is identified and the technical department is informed (Fleet manager, coordinator), spare parts or actions to be taken for maintenance should be clarified from the technical department to the supplies department. The supply department should order the required spare part and proceed with the requisition and delivery process of the part to the vessel. The cost of the spare part depends on the location of the vessel, on the distance where the closest port is, and on the supplier, while some qualitative criteria must be taken also into account. Usually, each shipping company has a list of suppliers who are trusted. Thus, the supply department wishes to minimize the cost of the ordered spare part without compromising the quality of the part itself and replace it on time without letting the damage on the main engine put the vessel off-hire.
Actors	Coordinator, Fleet manager, Supplies coordinator

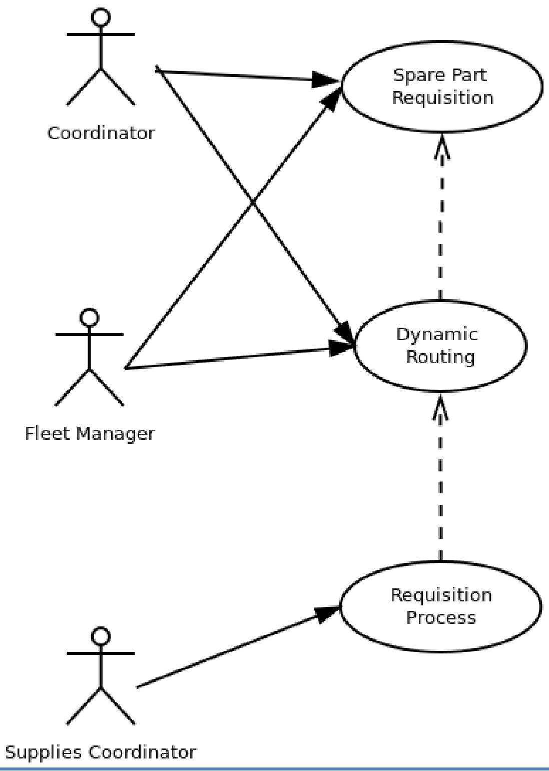
Objectives	- Timely alerting of the supply department for a new order. - Timely ordering of spare parts. - Dynamic routing of the vessel to the closest port with available spare part. - Optimization of the requisition and delivery process.
Pre-conditions	A malfunction has been identified, the technical department is alerted
Process Description	The requisition process of a spare part goes as follows; First a requisition is made by the vessel. This request is processed and pre-checked by the supply department. Next, a Request for Quotation (RFQ) is made via the ***DANAOS platform***, which broadcasts the RFQ to the company's listed suppliers. Given the offers by the suppliers, the supply department performs a comparative table analysis and decides which supplier will place the order. Once the order is placed, it is invoiced and delivered to the vessel.
Variations	An order may be delivered but not invoiced on time.
Post-condition	The required spare part is ordered and delivery is expected to the closest port where the vessel is dynamically routed.
Diagrams	 Coordinator Spare Part Requisition Fleet Manager Dynamic Routing Requisition Process Supplies Coordinator

Table 14 – Order suggestion and dynamic routing (scenario)

4.1.3 Requirements

In the following tables, we show the description of the use requirements defined in each scenario, as described in the previous section.

Section	Description
Id	**REQ-RSM-01**
Level of detail	Stakeholder

Type	FUNC
Short name	Main Engine Monitoring
Description	As data flow, out of sensors installed at the main engine of a vessel, the user wishes to have a look on the current or past values of the provided metrics in a chart, select a set of metrics which are to be drawn on a chart
Additional Information	Data requirements (indicative): - Air Cooler Cooling Water Inlet Pressure (Pa) - Air Cooler Cooling Water Inlet Temperature (°C) - Cooling Fresh Water Inlet Pressure (Pa) - Control Air Pressure (Pa) - Cylinder Lube Oil Temperature (°C) - Exhaust Valve Spring Air Inlet Pressure (Pa) - Fuel Oil Flowrate (lt) - Fuel Oil Inlet Pressure (Pa) - Fuel Oil Inlet Temperature (°C) - Heavy Fuel Oil Viscosity High Low (mm2/s) - HPS Bearing Temperature (°C) - Jacket Cooling Fresh Water Inlet Temperature Low (°C) - Order RPM (Bridge Leverer) - Scavenge Air Inlet Pressure (Pa) - Scavenge Air Receiver Temperature (°C) - Starting Air Pressure (Pa) - Thrust Pad Temperature (°C) - Main Lube Oil Inlet Pressure (Pa) - Main Lube Oil Inlet Temperature (°C) - Fuel Oil Temperature (°C) - Fuel Oil Total Volume (lt) - Power (kW) - Scavenge Air Pressure (Pa) - Torque (N/m) - Fuel Oil Consumption (lt/min) - Fuel Oil Consumption (MT)
Actor	Coordinator
Priority	MAN

Table 15 – Main Engine Monitoring (stakeholder requirement)

Section	Description
Id	**REQ-RSM-02**
Level of detail	Stakeholder
Type	FUNC
Short name	Malfunction Alert

Description	Once a malfunction pattern is identified the user is alerted via a message with minimum and concise information about the upcoming malfunction
Additional Information	Data requirements: - Malfunction name - Estimated time before break-down
Actor	Coordinator, Fleet manager
Priority	MAN

Table 16 – Malfunction alert (stakeholder requirement)

Section	Description
Id	**REQ-RSM-03**
Level of detail	Stakeholder
Type	FUNC
Short name	Alert Inspection
Description	Once the user is informed about an alert, he/she can investigate the malfunction pattern, the metrics and the history of this malfunction.
Additional Information	Data requirements: - Malfunction name - Estimated time before break-down - Previous occurrence of this malfunction - Actions taken in previous occurrence (e.g. ordered spare part) - Chart with values and anomalies on a minute basis
Actor	Coordinator, Fleet manager
Priority	MAN

Table 17 – Alert Inspection (stakeholder requirement)

Section	Description
Id	**REQ-RSM-04**
Level of detail	Stakeholder
Type	FUNC
Short name	Spare Part Requisition
Description	Once the Coordinator or the Fleet manager have inspected an alert for an upcoming malfunction, given the history of actions taken in the past, he/she makes a requisition for the same or another spare part of the main engine.
Additional Information	Data requirements: - Spare part name - Spare part id - Reason for requisition - Date of requisition

	- Description
Actor	Coordinator, Fleet manager
Priority	ENH

Table 18 – Spare Part Requisition (stakeholder requirement)

Section	Description
Id	REQ-RSM-05
Level of detail	Stakeholder
Type	FUNC
Short name	Requisition Process
Description	Once a requisition is made by the technical department, it is processed and pre-checked by the supply department. Next, a Request for Quotation (RFQ) is made via the *DANAOS* platform, which broadcasts the RFQ to the company's listed suppliers. Given the offers by the suppliers, the supply department performs a comparative table analysis and decides to which supplier will place the order. Once the order is placed, it is invoiced and delivered to the vessel.
Additional Information	Data requirements: - Spare part id - Spare part name - Spare part description - List of suppliers - List of offers - List of available ports - List of estimated time of deliveries
Actor	Supplies Coordinator
Priority	ENH

Table 19 – Requisition Process (stakeholder requirement)

Section	Description
Id	REQ-RSM-05
Level of detail	Stakeholder
Type	FUNC
Short name	Dynamic Routing
Description	This use case is a plug-in feature for the requisition process use case, since it can update the data on the comparative table analysis where costs of routing to the port where the spare part is available are included. Furthermore, it can highlight the row in the comparative table the cost-

	efficient solution, to suggest to the Supplies coordinator the best possible choice.
Additional Information	Data requirements: - Spare part id - Spare part name - Spare part description - List of suppliers - List of offers - List of available ports - List of estimated time of deliveries - Voyage estimations to closest ports
Actor	Coordinator, Fleet manager, Supplies Coordinator
Priority	ENH

Table 20 – Dynamic Routing (stakeholder requirement)

4.2 Connected Consumer

4.2.1 Introduction

This section refers to the use case of Connected Consumer (CC): Multi-sided market ecosystem. Here, we discuss the usage scenarios by giving a detailed description of the use cases (Section 4.2.2) along with a description of use requirements (Section 4.2.3).

In a world with instant access to information, where competition is just one click away, attracting and keeping customers is crucial for survival. Predictive analysis is the challenge. It can help predict which consumers are the most loyal or which potential buyers are more likely to purchase a certain product or service, opening new opportunities for retailers, providing new business prospects to customers, with improved shopping experience for consumers and new business opportunities for traders.

In this business domain, *Eroski*[7], one of the largest distribution companies in Spain with more than 35.000 workers, is collaborating with ATOS in the definition and test of a use-case related to the grocery business. It is also contributing with real data for the development of the project. The goal of this scenario is to provide data insights to EROSKI to better understand how to create and offer added-value services to their consumers. In this context, the use case objective is to predict both which products and which promotions are more likely to be interesting for the customers at the right time. In this way, EROSKI can adapt the most appropriate message (i.e. product and/or promotion) for each customer and send it at the right time and through the most appropriate channel, thus increasing the ROI of their marketing activities.

From the analysis of different data sources provided by *Eroski*, the goal is first to predict the list of products that customers with recurrent purchases will need in the current purchase period (trend). Afterwards, add to this prediction those products that can be interesting for the user based on other similar user’s behaviour (cross-selling). Finally, thanks to a deep

[7] https://www.eroski.es/

knowledge of the customer profile, the goal is also to incorporate those promotions that can be interesting for each customer.

Additionally, a scenario that describes a demonstrator that will help users to display and test recommendations made by the user has also been included.

4.2.2 Scenarios

Section	Description
Id	**SCE-CC-01**
Title	Retail Recommender
Description	This scenario is distributed in three steps: - data collection, - calculate recommendations, and - show predictions. **The first step**, data collection, provides services to update those entities needed for the recommender with data coming from external systems: - Product Service (products, categories, products x category) - Sales Service (orders) - Client service - Events service (used by the external systems to provide feedback about the visualization of the recommendations) **The second step**, calculate recommendations, calculates the products and the promotions that would recommend to every user. The input data is being processed, i.e., cleaned (“denoised”) and modelled. In the cleaning process, any unwanted effects in the data are removed (such as missing values or outliers) while maximizing its information. We define noise as any unwanted artefact introduced in the data collection phase that might affect the result of our data analysis and interpretation. In the modelling process, the data is being modelled into some pre-defined model. The pre-defined model is going to be the input for the main process. Therefore, we can split the recommendation process in two phases: 1) Calculate the products that a user would be interested to buy, based on: - Product sales frequency by user - Product sales frequency by product - Product seasonality - Product cross-selling - User feedback (registered with events) 2) Calculate the promotions that will be recommended to the final users. The promotions are calculated in base to: - Representative products that the user could buy (calculated in “calculate products” use case) - Possible promotions, with a priority ranking - User feedback (registered with events)

	Finally, **the third step**, show predictions, provides the needed services for getting the recommendations calculated. Consumers of these services will be the client applications that need to show recommended products or promotions to its users.
Actors	External System, Trigger recommender
Objectives	The main objective of this scenario is to calculate the most interesting products and promotions to recommend. To achieve this main objective, the first thing we need is to collect data and refresh the database with fresh data, including: users, products, promotions and sales data. When we have all the new fresh data collected and stored, we need to process data and prepare it for the main process, denoising and modelling. Then, with the modelled data, we calculate the most interesting products and promotions for every user. Finally, when it's requested, we provide data to client applications with the recommendations requested.
Pre-conditions	The system needs, at least, 2 years of data history to do good predictive recommendations. This data includes: - Sales - Clients - Products - Categories Additionally, we can also have some other data that can be included in the recommendation process, such as user events, user feedback, etc.
Process Description	1. Data collection - External system invokes service - System stores the data provided by the external system 2. Calculate recommendations - Some input data arrives to the system - The system prepares the data pro process it - The system calculates products to recommend - The system calculates promotions to recommend - Process the results from points 3 and 4 and store the final products and promotions to recommend on DB 3. Data preparation - Some input data arrives to the system - The system initiates a denoising process - The system initiates a modelling process 4. Denoising data - The process for denoising is called with some data - The data noise is removed - Denoised data is returned to the caller 5. Data modelling - The process for modelling is called with some data - The data is modelled - Modelled data is returned to the caller 6. Calculate products

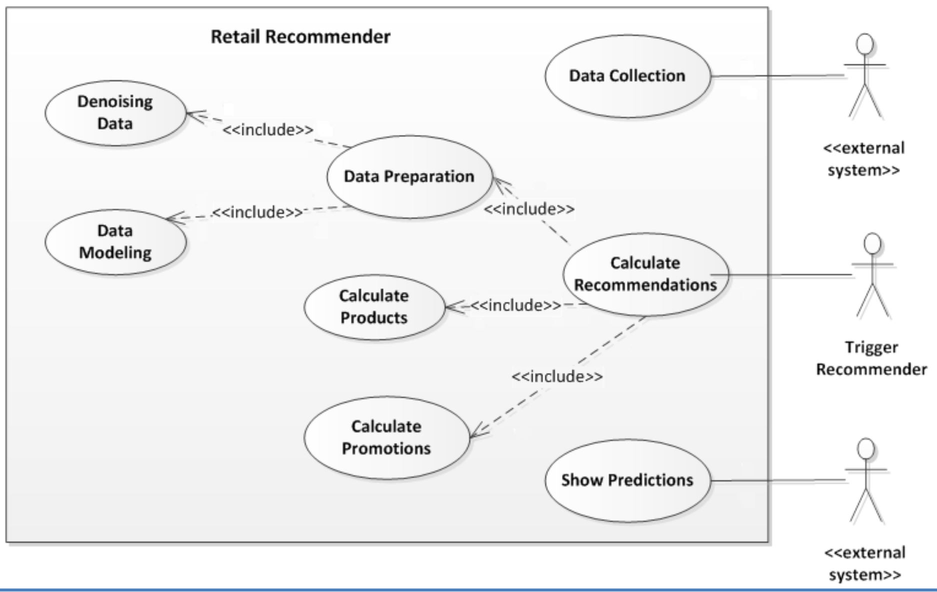
	- The process of calculate products is called with some data - The system calculates recommended products for every user 7. Calculate promotions - The process of calculate promotions is called with some data. - If the data does not include the suggested products calculated, the system calculates recommended products for every user. - The system calculates recommended promotions. 8. Show Predictions - External system invokes service - System provides the predictions requested by the external system
Variations	N/A
Post-condition	The suggested products and promotions shouldn't be null. The system must store in the DB: - Products and promotions being recommended for every user. - Suggested order priority for the recommended products and promotions for every user. - Trace tokens to register possible feedback events for every recommended item.
Diagrams	 Retail Recommender Data Collection Denoising Data <<include>> Data Preparation <<include>> <<include>> Data Modeling Calculate Recommendations Calculate Products <<include>> <<include>> Calculate Promotions Show Predictions <<external system>> Trigger Recommender <<external system>>

Table 21 – Retail Recommender (scenario)

Section	Description
Id	**SCE-CC-02**
Title	Retail Demonstrator
Description	This scenario is distributed in three modules: login, View Predicted Products and View Predicted Promotions. - The login module logs into the demonstrator application for a given customer. - The View Predicted Products module displays the products that

	the system would suggest to the customer. - The View Predicted Promotions module displays the personalized promotions that the system would suggest to the customer. In both modules, the View Predicted Products and View Predicted Promotions, the demonstrator is also giving feedback to the retail recommender about in which products/promotions the customer has shown interest so that the recommender can adapt its recommendations in real time.
Actors	Customer, Products Recommender
Objectives	- Provide a way to switch from one user to another in the app, thus, allowing display of the predicted products and promotions for different users. - Provide an example on how end users could display the products calculated by the recommender. - Provide an example on how end users could display the promotions calculated by the recommender. - Provide a way to show that the recommender is considering the feedback given by the client applications.
Pre-conditions	User is in the list of available users
Process Description	1. Display predicted products - User logs into the system - System verify username exists - User selects *My predicted products* - System retrieves the prediction for the current user from the recommender system - Application displays Suggested Products list - Application gives feedback to the recommender about both which products has been shown and which products the user has shown interest 2. Display predicted promotions - User is logged in the application - User selects 'My predicted promotions' - System retrieves the prediction for the current user from the recommender system - Application displays a suggested promotions list which takes into consideration the list of products predicted for the user - Application give feedback to the recommender about both which promotions has been shown and which promotions the user has shown interest
Variations	N/A
Post-condition	The user is logged in the demonstrator, and the user can view the following in the application: - List of products predicted for the user - List of promotions predicted for the user
Diagrams	

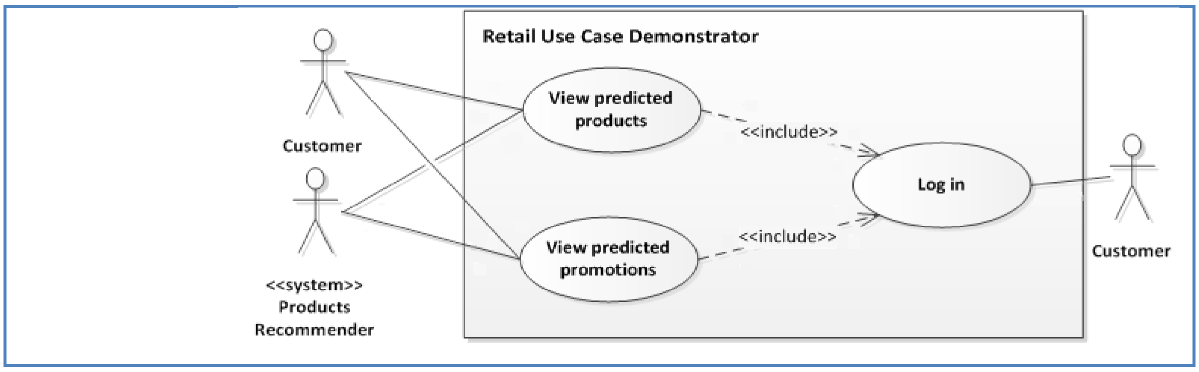

Retail Use Case Demonstrator
Customer
<<system>> Products Recommender
View predicted products
View predicted promotions
<<include>>
<<include>>
Log in
Customer

Table 22 – Retail Demonstrator (scenario)

4.2.3 Requirements

In the following tables, we show the description of the use requirements defined in each scenario, as described in the previous section.

Section	Description
Id	**REQ-CC-01**
Level of detail	Stakeholder
Type	FUNC
Short name	Predict products required by a recurrent user
Description	For a user with previous orders (recurrent), the system should be able to predict a list of items that the user is likely to need in the coming days. The calculation should consider: - History of orders made by the user - Seasonality of the products - Similarity with items that the customer bought and is bound to need - What other customers bought - User segment - Receptivity of the user to the items recommended by the system
Additional Information	The list of items should return for each item a rank that helped the external system to prioritize the display of items. The rank should be assigned to the products considering the probability that the user needs them, that is, buys them.
Priority	MAN

Table 23 – Predict products required by a recurrent user (stakeholder requirement)

Section	Description
Id	**REQ-CC-02**
Level of detail	Stakeholder
Type	FUNC
Short name	Predict products to a new user (user without previous orders)

Description	For a given new user, the system should be able to predict a list of items that the user is likely to buy. The calculation should consider: - Seasonality of the products - What other customers bought - User segment
Additional Information	The list of items should return for each item a rank that helps the external system prioritize the display of items. The rank should be assigned to the products considering the probability that the user buys them.
Priority	MAN

Table 24 – Predict products to a new user (stakeholder requirement)

Section	Description
Id	**REQ-CC-03**
Level of detail	Stakeholder
Type	FUNC
Short name	Recommend personalized discounts
Description	Be able to recommend personalized discounts that users are likely to use in the coming days. The calculation should take into account the following factors: - List of predicted items towards the user (see req-1 for further info) - Category (commercial structure) of the items predicted to the user. The list of discounts proposed by the system will contain promotions that apply on products that belong to the same category than the products predicted for the user - User receptivity to the items recommended by the system
Additional Information	The calculation should consider: - List of items predicted to the user, - Seasonality of the products, - Similarity with items that the customer bought/is bound to need, and - What other customers in the same segment bought, and rank the products according to the probability he will buy them
Priority	MAN

Table 25 – Recommend personalized discounts (stakeholder requirement)

Section	Description
Id	**REQ-CC-04**
Level of detail	Stakeholder
Type	DATA
Short name	Orders requirements

Description	New orders placed by the users should be loaded at least once per day. Orders should have at least the following information: - Client Id - Order Date - Items ○ productId ○ price ○ promotionId
Additional Information	N/A
Priority	MAN

Table 26 – Data Requirement (stakeholder requirement)

Section	Description
Id	**REQ-CC-05**
Level of detail	Stakeholder
Type	DATA
Short name	Clients requirements
Description	New *Eroski* customers should be loaded at least once per day. Customers should have at least the following information: - Client Id - Client segment
Additional Information	N/A
Priority	MAN

Table 27 – Clients requirements (stakeholder requirement)

Section	Description
Id	**REQ-CC-06**
Level of detail	Stakeholder
Type	DATA
Short name	Products requirements
Description	New *Eroski* products should be loaded at least once per day. Products information should have at least the following information: - Product reference - Product category
Additional Information	N/A
Priority	MAN

Table 28 – Products requirements (stakeholder requirement)

Section	Description
Id	**REQ-CC-07**

Level of detail	Stakeholder
Type	L&F
Short name	Multi-device
Description	The demonstrator application UI should adapt to different devices and displays, including mobile, to provide a proper operation of the solution and a good user experience.
Additional Information	Give support to both Android and iOS mobile platforms.
Priority	MAN

Table 29 – Multi-device (stakeholder requirement)

Section	Description
Id	**REQ-CC-08**
Level of detail	Stakeholder
Type	USE
Short name	Easy-to-use
Description	The solution should be easy to use for people of different ages. It should follow the best practices in terms of usability.
Additional Information	N/A
Priority	MAN

Table 30 – Easy-to-use (stakeholder requirement)

Section	Description
Id	**REQ-CC-09**
Level of detail	Stakeholder
Type	ENV
Short name	Multi-user
Description	The solution should be portable and reusable for different users.
Additional Information	N/A
Priority	MAN

Table 31 – Multi-user (stakeholder requirement)

Section	Description
Id	**REQ-CC-10**
Level of detail	Stakeholder
Type	SUP
Short name	Data security

Description	Database must be securely accessible and its data must not be breached.
Additional Information	N/A
Priority	MAN

Table 32 – Data security (stakeholder requirement)

Section	Description
Id	**REQ-CC-11**
Level of detail	Stakeholder
Type	SUP
Short name	Services security
Description	Datasets contain personal information, so security is very important in services.
Additional Information	N/A
Priority	MAN

Table 33 – Services security (stakeholder requirement)

4.3 Intelligent Multi-Channel Baking

4.3.1 Introduction

This section refers to the use case of Intelligent Multi-Channel Banking (IMB): Business-to-customer (B2C), person-to-person (P2P), person-to-business (P2B). Here, we discuss the usage scenarios by giving a detailed description of the use cases (Section 4.3.2) along with a high-level representation of functional requirements (Section 4.3.3).

GFT focuses on the development of complete IT solutions for retail and investment banks, developing software and systems addressing the needs of such institutions. The financial services industry is moving to a data-centric paradigm which is key for the provisioning of services following customer “tailored” requirements. The main goal is to create a 360° view of the customer and provide personalized services. In the context of BigDataStack, a multi-channel scenario will be realized, facilitating intelligent banking powered by data analytics, including online banking and wearable devices and social media.

The latter is key and of increasing importance in EU tablet owners who bank via their tablets will increase from 35% to 68% in 2018.

4.3.2 Scenarios

Section	Description
Id	**SCE-IMB-01**
Title	B2C
Description	The B2C (business to customer) scenario focuses on the provision of a holistic set of financial services through the engagement of a variety of

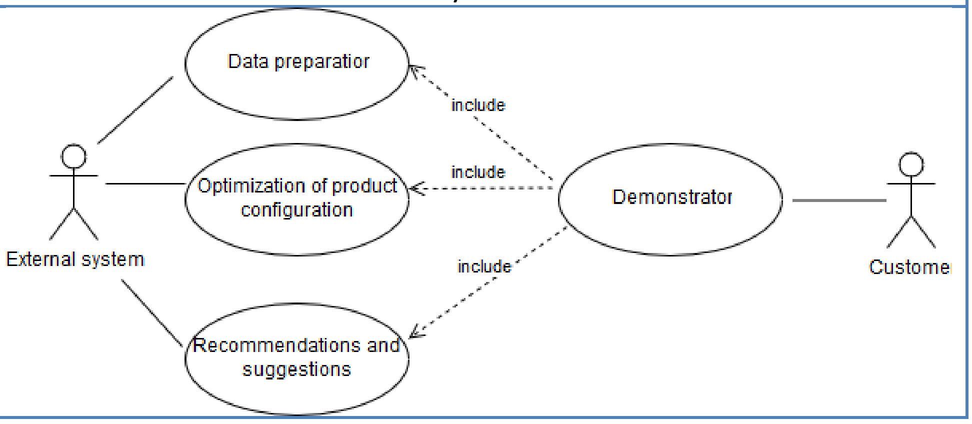
	devices (e.g. tablets, smart phones, wearables, etc.) for different purposes in an adaptive way. Optimization of the product configuration will include alerts for risks, products and maturities, as well as campaigns and cross-selling activities. This scenario is distributed in three steps: - Data collection. - Optimizing the product configuration, and recommendations. - Show recommended items.
Actors	Customer, Service Provider
Objectives	- Provide a holistic set of financial services adapted to the customer's device. - Optimize product configuration including alerts for risks, products and maturities, or suggestions for campaigns and cross selling.
Pre-conditions	N/A
Process Description	The key aspects in this case are related to data analytics in order to predict behaviour models to classify different customers (e.g., most profitable, least satisfied, etc.), analyse present and future profitability, identify target customers, and increase their loyalty through tailored offerings and solutions by predicting which customers are not satisfied to provide them with optimised and personalised offers, and – most importantly – conclude in real-time how customers should be addressed at any point of contact (i.e., email, call centre, branch, online banking).
Variations	N/A
Post-condition	- The suggested recommendations and optimizations should not be null. - The system must store in the DB the optimizations and recommendations for every customer.
Diagrams	 Data preparatior Optimization of product configuration Recommendations and suggestions External system include include include Demonstrator Custome

Table 34 - B2C (scenario)

Section	Description
Id	**SCE-IMB-02**
Title	P2P/P2B

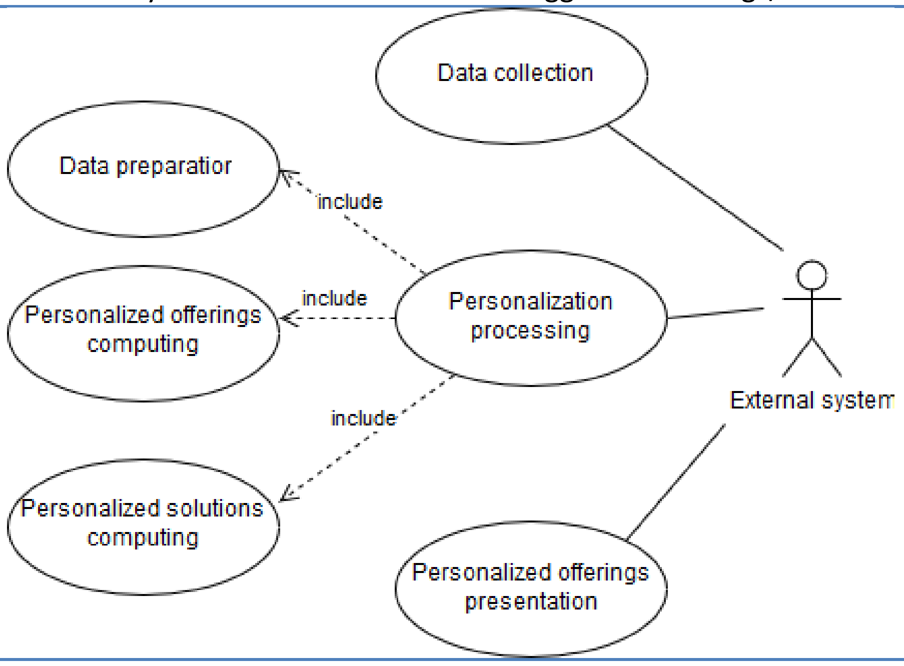
Description	The P2P (Person to Person) and P2B (Person to Business) aspects of the scenario will be based on the GFT W@llet solution. W@llet is an integrated platform providing several functionalities: a single point of access via mobile devices, P2P transfer of money (also through NFC) and P2B functionalities such as purchase and payment services to the business-merchant world. The main goal is to exploit the W@llet solution as the means to provide real-time personalized offerings and solutions to the customers considering both their transactions through W@llet and additional data, increasing their loyalty through tailored offerings and solutions.
Actors	Customer, Service Provider
Objectives	- Provide a set of real-time personalised offerings and solutions to the customers. - Optimize personalised offerings/solutions following social data.
Pre-conditions	The system needs at least 1 year of historical data for efficient predictive recommendations.
Process Description	1. Data collection: the system collects information related to customers' transactions made through the W@llet application. In addition, data coming from social data related to the customers will be collected and stored. 2. Data processing: the intelligent system module analyses the collected data and elaborates in real-time personalized offerings and solution for the different customers. 3. Data visualization: the results from the previous phase are optimized and presented to the customer through a W@llet companion application.
Variations	N/A
Post-condition	- The suggested offerings and solutions should not be null. - The system must store on DB the suggested offerings/solutions.
Use case diagrams	 Data collection Data preparatior include Personalized offerings computing include Personalization processing External systerr include Personalized solutions computing Personalized offerings presentation

Table 35 - P2P/P2B (scenario)

4.3.3 Requirements

The following template introduces the structure of requirements for IMB use case.

Section	Description
Id	**REQ-IMB-01**
Type	FUNC
Short name	Provide personalized offerings required by customer
Description	For any customer, the system should be able to predict a set of personalized offerings for the customer. The calculation should consider the historic of offerings purchases made by the customer, as well as social data related to the customer.
Additional Information	N/A
Priority	MAN

Table 36 – Provide personalized offerings required by customer (stakeholder requirement)

Section	Description
Id	**REQ-IMB-02**
Type	FUNC
Short name	Provide personalized solutions required by customer
Description	For any customer, the system should be able to predict a set of personalized solutions for the customer. The calculation should consider the historic of solutions purchases made by the customer, as well as social data related to the customer.
Additional Information	N/A
Priority	MAN

Table 37 – Provide personalized solutions required by customer (stakeholder requirement)

Section	Description
Id	**REQ-IMB-03**
Type	DATA
Short name	Customers
Description	Customers should have at least the following information: - Client Id
Additional Information	N/A
Priority	MAN

Table 38 – Customers (stakeholder requirement)

Section	Description
Id	**REQ-IMB-04**
Type	DATA
Short name	Offerings
Description	Products information should have at least the following information: - Offerings reference - Offerings category
Additional Information	N/A
Priority	MAN

Table 39 – Offerings (stakeholder requirement)

Section	Description
Id	**REQ-IMB-05**
Type	DATA
Short name	Solutions
Description	Products information should have at least the following information: - Solutions reference - Solutions category
Additional Information	N/A
Priority	MAN

Table 40 – Solutions (stakeholder requirement)

Section	Description
Id	**REQ-IMB-06**
Type	Stakeholder
Short name	USE
Description	Easy-to-use
Additional Information	The application should be easy to use and to understand by people of different ages.
Priority	MAN

Table 41 – Easy-to-use (stakeholder requirement)

Section	Description
Id	**REQ-IMB-07**

Type	Stakeholder
Short name	SUP
Description	Data security
Additional Information	Database must be reached securely.
Priority	MAN

Table 42 – Data security (stakeholder requirement)

5 Platform Roles

The following table shows a description of what BigDataStack offers to different roles related to the development, deployment and operation of Big Data Analytics solutions.

Id	Name	Description
ROL-01	Data Owner	BigDataStack offers a unified Gateway to obtain both streaming and stored data from data owners and store them in its underlying storage infrastructure that supports SQL and NoSQL data stores.
ROL-02	Data Scientist	BigDataStack offers the Data Toolkit to enable data scientists both to easily ingest their analytics tasks by utilizing a declarative paradigm, and to specify their preferences and constraints to be exploited during the dimensioning phase regarding the data services that will be used (for example preferences for the data cleaning service)
ROL-03	Business Analysts	BigDataStack offers the Process Modelling Framework allowing business users to define their functionality-based business processes (through declaratively-defined models) and optimize them based on the outcomes of process analytics that will be triggered by BigDataStack.
ROL-04	Application Engineers and Application Service Owners	BigDataStack offers the Application Dimensioning Workbench to enable application owners and engineers to experiment with their applications and dimension it in terms of its data needs and data-related properties
ROL-05	Data Engineers and Data Service Owners	BigDataStack offers the possibility to Data Service owners (such as the roles implemented by IBM and LXS in this project) to bring in (adapt) their data services to the platform.

Table 43 – BigDataStack Platform roles

6 System Requirements

System requirements represent technical specifications for the BigDataStack platform at the system and subsystem levels, including the usability for the envisaged human-system interaction. They specify the system (including interfaces of functions and services) that will meet measurable stakeholder requirements. They answer to what characteristics the system needs to possess and to what degree to satisfy stakeholder requirements.

In this section, we organize the system requirements in terms of the envisioned Big Data Stack platform capabilities. These are depicted through a full stack that aims not to only facilitate the needs of data operations and applications (all of which tend to be data-intensive) but also to facilitate these needs in an optimum way. As depicted in Figure 2, BigDataStack will provide a complete infrastructure management system that will base the management and deployment decisions on data aspects.

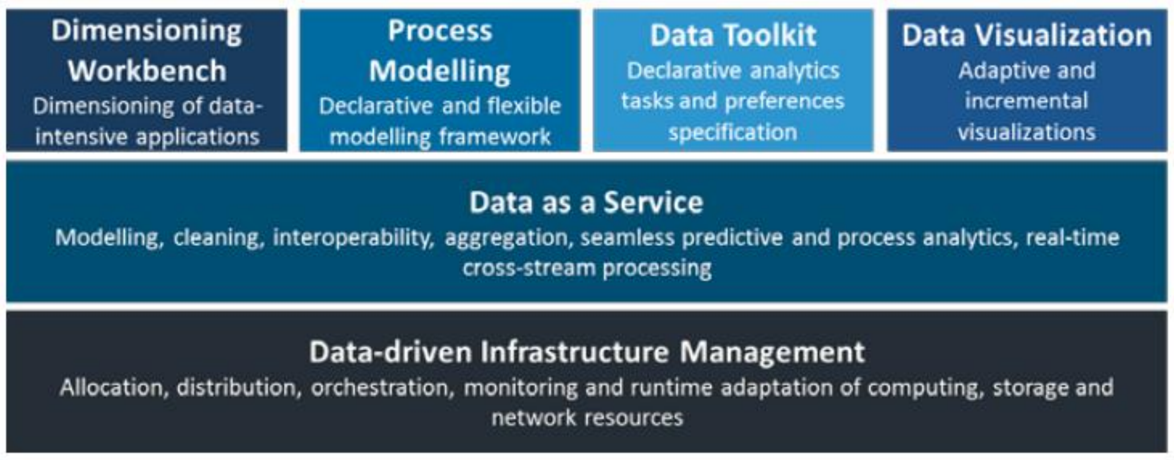

Dimensioning Workbench
Dimensioning of data-intensive applications
Process Modelling
Declarative and flexible modelling framework
Data Toolkit
Declarative analytics tasks and preferences specification
Data Visualization
Adaptive and incremental visualizations
Data as a Service
Modelling, cleaning, interoperability, aggregation, seamless predictive and process analytics, real-time cross-stream processing
Data-driven Infrastructure Management
Allocation, distribution, orchestration, monitoring and runtime adaptation of computing, storage and network resources

Figure 2. BigDataStack core platform capabilities

These six BigDataStack core **platform capabilities** are envisioned to achieve the business goals or expectations from the different stakeholders:

- **Data-driven Infrastructure Management.** The platform capability to provide means for efficient and optimized infrastructure, incorporating all aspects of data-driven management for the computing, storage and networking resources.
- **Data as a Service.** The platform capability to exploit the underlying data-driven infrastructure management system to offer data as a service in a performant, efficient and scalable way. It includes access to a set of technologies addressing the complete data path: modelling and representation, cleaning, aggregation, and data processing and analytics.
- **Data Visualization** capability goes beyond adaptable visualization and presentation of data and analytics outcomes, to performance aspects such as computing, storage and networking infrastructure data, data sources information, and data operations outcomes.
- **Data Toolkit** capability aims at openness, extensibility and wide adoption. The toolkit will allow the ingestion of data analytics functions and the definition of analytics in a declarative way; moreover, it will allow data scientists and administrators to specify requirements and preferences both for the data and infrastructure management.
- **Process Modelling** capability will allow for declarative and flexible modelling of

process analytics. Functionality-based process modelling will then be concretized to technical-level process mining analytics, while a feedback loop will be implemented towards overall process optimization and adaptation.

- **Dimensioning Workbench** capability enables the self-dimensioning of applications in terms of predicting the required data services, their interdependencies with the application micro-services and the required underlying resources.

Due to the early stage of the architecture work, the overall architecture has not yet matured/finalized, and not all system requirements have been identified yet. However, new requirements and constraints over those components will appear in future iterations of the requirements engineering at M11 and M22 (see Introduction).

6.1 Data-Driven Infrastructure Management

Section	Description	
Id	**REQ-SY-DDIM-01**	
Level of detail	Software	
Type	FUNC	
Short name	Quality enforcement through SLA management	
Description	The platform should monitor the quality of service (QoS) of data analytics operations through an SLA (service level agreement) model that is compliant with the *WS-Agreement* (Web Services Agreement Specification) [42] specification from the Open Grid Forum (OGF)[8]. The SLA framework should implement the XML structure for the representation of templates agreements specified by the standard, as well as an "agreement layer" to allow for a layered service model. *WS-Agreement Negotiation* [43] will not need to be supported.	
Additional Information	The structure of a WS-Agreement-based SLA contract is as follows:	*Context*: information about agreement parties, the agreement's lifetime, and template reference.
		Service Description Terms: functional description of the service to provide (domain-specific description).
		Service References: refer to existing services (domain-specific description).
		Service Properties: define variables in the context of an agreement.
		Guarantee Terms: define how guarantees are assessed and compensated

8 https://www.ogf.org/ogf/doku.php

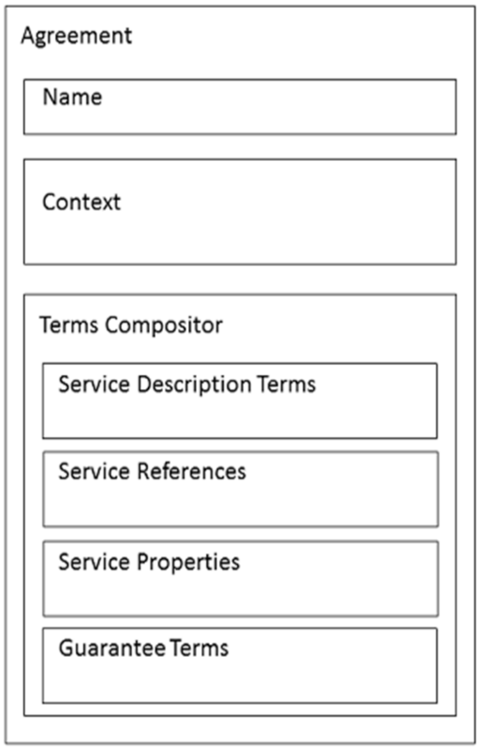
	 Agreement Name Context Terms Compositor Service Description Terms Service References Service Properties Guarantee Terms	in case of meeting or violating the Service Level Objectives (SLO) or QoS.
Priority	DES	
Role		
Source	Technology provider (ATOS)	
Refines		

Table 44 – Quality enforcement through SLA management (system requirement)

Section	Description
Id	**REQ-SY-DDIM-02**
Level of detail	Software
Type	FUNC
Short name	Application, data and infrastructure metrics
Description	The platform must monitor infrastructure resources, application components, and data operations.
Additional Information	Typical application elements to be monitored include: • **Infrastructure resources:** utilization (e.g. CPU, RAM, bandwidths, etc), availability of the hosts, data sources generation rates and windows. • **Application component:** application performance metrics, data flows across application components, availability of the applications etc. • **Data functions/operations:** data analytics, queries progress, queries performance, transactions performance, storage distribution, etc.
Priority	DES

Role	
Source	Technology provider (ATOS)
Refines	

Table 45 – Application, data and infrastructure metrics (system requirement)

6.2 Data as a Service

Section	Description
Id	**REQ-SY-DAS-01**
Level of detail	System
Type	FUNC
Short name	Provide a seamless way for accessing the data
Description	Data consuming services must use a unique API to retrieve data from the data stores offered by BigDataStack environment, either if the data are stored in the relational or object storage of the platform. The services must neither need to know the implementation details for the connectivity to each data store, nor must know beforehand where the data are located.
Additional Information	N/A
Priority	MAN
Role	All
Source	Technology provider (LXS/IBM)
Refines	All

Table 46 – Provide a seamless way for accessing the data (system requirement)

Section	Description
Id	**REQ-SY- DAS-02**
Level of detail	System
Type	PERF
Short name	Scalability of the resources
Description	The distributed storage of the platform must scale according to the usage, either in terms of throughput (increased requests per unit time) or of data volume (increased amount of data).
Additional Information	N/A
Priority	MAN
Role	All
Source	Technology provider (LXS/IBM)
Refines	All

Table 47 – Scalability of the resources (system requirement)

6.3 Data Visualization

Section	Description
Id	**REQ-SY-DV-01**
Level of detail	System
Type	L&F
Short name	Look & feel adapted to different devices
Description	The UI of the visualization environment should adapt to different devices and displays to provide a proper operation of the solution and a good user experience.
Additional Information	In case the solution is web based, and is responsive, the solution should be responsive. In case the solution is application based, it should be available for Android and for iOS users. Furthermore, the solution should be adapted to the original application look & feel requirements.
Priority	MAN
Role	ROL-04
Source	Use case provider (ATOS)
Refines	REQ-CC-07

Table 48 – Look & feel adapted to different devices (system requirement)

6.4 Data Toolkit

Section	Description
Id	**REQ-SY-DT-01**
Level of detail	Software
Type	FUNC
Short name	Playbooks
Description	Support for the description of data mining and analysis processes, interconnected to each other in terms of input/output data streams/objects. This is represented in the form of an analysis application graph that includes the set of individual processes as nodes of the graph along with their binding/dependencies in the form of virtual links.
Additional Information	The playbook must be represented in the form of a descriptor that can be incorporated into the Dimensioning Workbench as well as the Dynamic Orchestrator.
Priority	MAN
Role	ROL-02
Source	Tool provider (UBI)
Refines	REQ-RSM-05

Table 49 – Playbooks (system requirement)

6.5 Process Modelling

Section	Description

Id	**REQ-SY-PM-01**
Level of detail	System
Type	USE
Short name	Process modelling toolkit easy-to-use
Description	The Process Modelling Toolkit should provide a high level of user experience (UX) to the users.
Additional Information	The solution should guide the users to complete the business diagram with easy steps. It should clearly indicate what connections (interactions) are possible and provide comprehensive error messages.
Priority	MAN
Role	ROL-03
Source	Use case provider (ATC)
Refines	REQ-CC-08

Table 50 – Process modelling toolkit easy-to-use (system requirement)

Section	Description
Id	**REQ-SY-PM-02**
Level of detail	System
Type	FUNC
Short name	Process modelling toolkit multi-user
Description	The Process Modelling Toolkit should be available to multiple users at the same time.
Additional Information	Multiple users should be able to use the Process Modelling Toolkit and create diagrams at the same time.
Priority	MAN
Role	ROL-03
Source	Use case provider (ATC)
Refines	REQ-CC-09

Table 51 – Process modelling toolkit multi-user (system requirement)

6.6 Dimensioning Workbench

Section	Description
Id	**REQ-SY-DW-01**
Level of detail	System
Type	PERF
Short name	Response Time
Description	The service provided by the data applications (e.g. recommendation system) must be performant (in terms of response time) so visitors / potential consumers will not notice the time taken by the request.
Additional Information	N/A

Priority	MAN
Role	ROL-02
Source	Use case provider (ATOS)
Refines	REQ-CC-01, REQ-CC-02, REQ-CC-03

Table 52 – Response time (system requirement)

Section	Description
Id	**REQ-SY-DW-02**
Level of detail	System
Type	PERF
Short name	Workload
Description	The service provided by the data applications (e.g. recommendation system) must support a high level of requests per hour, including support for peaks.
Additional Information	N/A
Priority	MAN
Role	ROL-02
Source	Use case provider (ATOS)
Refines	REQ-CC-01, REQ-CC-02, REQ-CC-03

Table 53 – Workload (system requirement)

Section	Description
Id	**REQ-SY-DW-03**
Level of detail	System
Type	PERF
Short name	Scalability of stress tests for load injection
Description	The system to launch the stress tests needs to be able to easily scale to support the client sizes needed. Therefore, a system setup like Docker Swarm mode should exist for supporting the process.
Additional Information	N/A
Priority	MAN
Role	ROL-04
Source	Tool provider (UPRC)
Refines	REQ-CC-01, REQ-CC-02, REQ-CC-09, REQ-SO-DW-02

Table 54 – Scalability of stress tests for load injection (system requirement)

Section	Description
Id	**REQ-SY-DW-04**
Level of detail	System

Type	FUNC
Short name	Dimensioning output
Description	The dimensioning workbench should provide a list of candidate dimensioning suggestions along with the expected QoS levels towards the application deployment patterns component, for the former to filter them based on an extra set of criteria and the latter to perform the final selection.
Additional Information	N/A
Priority	MAN
Role	ROL-04
Source	Tool provider (UPRC), Tool Consumer (GLA, ADS Deploy)
Refines	REQ-CC-01, REQ-CC-02

Table 55 – Dimensioning output (system requirement)

Section	Description
Id	**REQ-SY-DW-05**
Level of detail	System
Type	FUNC
Short name	Monitoring requirements for dimensioning
Description	The dimensioning workbench should have a means to obtain monitoring information from the deployed data services and application components for a given deployment to extract training data for the performance models. The rationale of the requirement is that for every needed metric (workload oriented e.g. number of current users, requests etc. or QoS oriented e.g. response time, throughput) in the model the respective endpoint should exist from which the monitoring component would extract metrics values.
Additional Information	Relevant Tools affected: Data services, application components, monitoring.
Priority	MAN
Role	ROL-04
Source	Tool provider (UPRC)
Refines	REQ-CC-01, REQ-CC-02

Table 56 – Monitoring requirements for dimensioning (system requirement)

7 Software Requirements

Software requirements specify software portions (product, program, or set of programs) that perform certain functions or provide certain capabilities in a specific environment. These are technology or implementation specific characteristics that may be written by one or more representatives of the supplier, one or more representatives of the acquirer, or by both. They may include:

a) the interfaces between the system and the software portion;
b) **external** performance and functionality requirements upon the software portion;
c) required features (e.g. functions) of the specified software portion;
d) conditions and constraints under which the software portion must perform; and
e) approaches to verification intended for the requirements.

In this section, we organize the software (or technology) requirements in terms of the technologies or technological components that have been already identified as suitable candidates to realize the envisioned capabilities of the platform.

Due to the early stage of the architecture work, the overall architecture is not yet matured/finalized, and not all software technology requirements have been identified yet. However, new requirements and constraints over those components will appear in future iterations of the requirements engineering at M11 and M22 (see Introduction).

7.1 Computing Resources Management

Section	Description
Id	**REQ-SO-CRM-01**
Level of detail	Software
Type	FUNC
Short name	Container scheduling
Description	There must be a container-based (cloud native) execution scheduler for the running applications. It would be responsible for the optimal distribution of the workloads across the cluster various resources. The scheduler algorithm expects to receive workload meta data parameters which defines the required resources for the container) and optional extra parameters like special devices needed.
Additional Information	N/A
Priority	MAN
Role	ROL-04
Source	Kubernetes (RHT)
Refines	N/A

Table 57 – Container scheduling (software requirement)

7.2 Storage Resources Management

Section	Description
Id	**REQ-SO-SRM-01**

Level of detail	Software
Type	FUNC
Short name	Storage plug-in interface
Description	The interface to external storage devices (like S3) needs to be align with the defined standard interface defined by CSI, that is, *Container Storage Interface.*
Additional Information	N/A
Priority	MAN
Role	ROL-04
Source	Kubernetes (RHT)
Refines	N/A

Table 58 – Storage plug-in interface (software requirement)

Section	Description
Id	**REQ-SO-SRM-02**
Level of detail	Software
Type	FUNC
Short name	Storage Performance
Description	Storage performance must not degrade when using a container within a virtual machine, to comply with strict security environments.
Additional Information	N/A
Priority	MAN
Role	ROL-04
Source	Qemu Block Layer (RHT)
Refines	N/A

Table 59 – Storage performance (software requirement)

7.3 Data-driven Network Management

Section	Description
Id	**REQ-SO-DNM-01**
Level of detail	Software
Type	FUNC
Short name	Network policies based on type of data
Description	A network policy is a specification (a set of rules) of how two or more components can communicate (send/receive data) with each other.
Additional Information	A different policy must be enforced based on different incoming data requirements, following the type of processing (stream, micro-batch, batch) and the type of data.
Priority	MAN

Role	ROL-02
Source	Tool provider (UBI)
Refines	N/A

Table 60 – Network Policies based on type of data (software requirement)

Section	Description
Id	**REQ-SO-DNM-02**
Level of detail	Software
Type	FUNC
Short name	Network policies based on application
Description	Real-time data processing and/or data analytics may impose time-critical constraints on the handling, operation and transformation of the data. To support decision making in time-critical conditions specific network policies need to be applied to deliver the output within seconds or predefined time limits.
Additional Information	Data scientist must set a policy to prioritize the set of data that need to be handled first.
Priority	MAN
Role	ROL-02
Source	Tool provider (UBI)
Refines	N/A

Table 61 – Network policies based on application (software requirement)

7.4 Dynamic Orchestrator

No software requirement has been identified so far for this technology.

7.5 Triple Monitoring

Section	Description
Id	**REQ-SO-TM-01**
Level of detail	Software
Type	FUNC
Short name	Metrics pusher
Description	This component of the triple monitoring system must receive metrics from different sources, clean them conformably to the technology that is going to be used (e.g., *Netdata* or *Prometheus*) and then ingest them into the monitoring collector.
Additional Information	Metrics pusher is used when probe approach is impossible to apply.
Priority	MAN
Role	N/A
Source	Tool provider(UPRC)
Refines	N/A

Table 62 – Metrics pusher (software requirement)

Section	Description
Id	**REQ-SO-TM-02**
Level of detail	Software
Type	FUNC
Short name	Monitoring manager
Description	The manager handles consumer's subscription, queries *Prometheus* or other similar tools to be used to respond to consumer's request through the REST API.
Additional Information	N/A
Priority	MAN
Role	N/A
Source	Tool provider(UPRC)
Refines	N/A

Table 63 – Monitoring manager (software requirement)

Section	Description
Id	**REQ-SO-TM-03**
Level of detail	Software
Type	FUNC
Short name	Monitoring metrics getter
Description	The monitoring getter is used as the output of the monitoring engine. It accepts a request from consumer, sends this to the manager, and then returns the response. It also provides an oriented connection gate (publisher/subscriber)
Additional Information	The communication between this component and the REQ-SO-TM-02 is performed by a queueing system.
Priority	MAN
Role	N/A
Source	Tool provider(UPRC)
Refines	N/A

Table 64 – Monitoring metrics getter (software requirement)

Section	Description
Id	**REQ-SO-TM-04**
Level of detail	Software
Type	FUNC
Short name	Spark compatible

Description	It is important to be Spark compatible. Spark currently uses *Dropwizard Metrics*[9] for metrics collection. In addition, Spark also allows monitoring bytes scanned for object stores (e.g., *COS* or *S3*)[10].
Additional Information	N/A
Priority	MAN
Role	N/A
Source	Tool provider(IBM)
Refines	N/A

Table 65 – Spark compatible (software requirement)

Section	Description
Id	**REQ-SO-TM-01**
Level of detail	Software
Type	FUNC
Short name	Regular recording of deployment QoS information
Description	When a user's application is deployed, the Triple Monitoring Framework will monitor that application, tracking statistical information about its operation and associated quality of service data. This data is needed to support the learning of ranking models by ADS-Ranking service (part of Application and Service Deployment; see REQ-SO-ADS-03). This data needs to be regularly saved in a centralised data store for later access.
Additional Information	Input: - Candidate Deployment Pattern (application identifier from this is the primary key for saving monitoring data for an application) Output: - Deployment QoS Snapshot (monitoring/QoS data, every few mins) Service Dependencies: - Centralised Data Store (storage service)
Priority	MAN
Role	ROL-04
Source	Tool provider (GLA)
Refines	REQ-CC-01, REQ-CC-02, REQ-CC-09, REQ-SO-DW-02

Table 66 – Regular recording of deployment QoS information (software requirement)

7.6 Applications & Data Services Deployment

Section	Description
Id	**REQ-SO-ADS-01**
Level of detail	Software
Type	FUNC

[9] Dropwizard Metrics. https://github.com/dropwizard/metrics
[10] This patch is described at https://issues.apache.org/jira/browse/SPARK-10912?attachmentSortBy=fileName

Short name	Scalable pattern generation for application playbooks
Description	When the user selects a configuration for their application via the Data Toolkit, a Playbook (Docker compose file) is generated, detailing the specifics for application deployment. However, this Playbook is incomplete, and hence needs to be expanded to form a complete (deployable) configuration. Therefore, there needs to be a service that converts a Playbook into such a configuration. More precisely, as there will be multiple ways that the Playbook might be expanded, multiple possible configurations will need to be created, referred to as Candidate Deployment Patterns. This service is referred to as the Pattern Generator. This service needs to be able to process multiple Playbooks simultaneously.
Additional Information	Input: Playbook (Docker Compose File) Output: List<Candidate Deployment Patterns> (list of Docker Compose Files) Service dependencies: - Data Toolkit (Provides Input) - Application Dimensioning Workbench (Consumes Output) Additional data dependencies: - Cloud Infrastructure Availability Information (Cloud Machine Types and Costs)
Priority	MAN
Role	ROL-04
Source	Tool provider (GLA)
Refines	REQ-CC-01, REQ-CC-02, REQ-CC-09, REQ-SO-DW-02

Table 67 – Scalable pattern generation for application playbooks (software requirement)

Section	Description
Id	**REQ-SO-ADS-02**
Level of detail	Software
Type	FUNC
Short name	On-demand ranking of deployment patterns for first time deployment
Description	The Pattern Generator service (see REQ-SO-ADS-01) produces a series of candidate deployment patterns (ways that the user's application can be deployed) the first time the requests deployment of their application. Each of these deployment patterns will have predicted quality of service features provided by the Dimensioning Workbench (referred to as Benchmarking). To deploy the user's application, one of the candidate deployment patterns need to be selected, based on the user requirements and quality of service estimates. Hence, there is a need for a service to provide this functionality. This service is referred to as ADS-Ranking (Application and Data Services Ranking).
Additional Information	Input: - List<Candidate Deployment Patterns> (list of Docker Compose

	Files). - Each Candidate Deployment Pattern needs to have attached Quality of Service predictions. Output: - Selected Candidate Deployment Pattern (Docker Compose File) Service dependencies: - Application Dimensioning Workbench (Provides Input) - ADS-Deploy (Consumes Output) Additional data dependencies - Supervised Ranking Model (also produced by ADS-Ranking, see REQ-SO-ADS-03)
Priority	MAN
Role	ROL-04
Source	Tool provider (GLA)
Refines	REQ-CC-01, REQ-CC-02, REQ-CC-09, REQ-SO-DW-02

Table 68 – Ranking of deployment patterns for first time deployment (software requirement)

Section	Description
Id	**REQ-SO-ADS-03**
Level of detail	Software
Type	FUNC
Short name	Supervised training of models for ranking deployment patterns
Description	The ADS-Ranking service is responsible for selecting a candidate deployment pattern for a user's application. This will be achieved via ranking the candidate deployment patterns using a supervised learning to rank model. This model needs to be trained on past examples of application deployments and the degree to which they met the user quality of service requirements. Hence, ADS-Ranking is also responsible for training this model and evaluating its effectiveness. Model re-training will happen at periodic intervals (e.g. every hour).
Additional Information	Input: - List<<Candidate Deployment Pattern, Deployment Quality of Service Snapshot> Output: - Candidate Deployment Pattern Ranking Model Service Dependencies: - Centralized Data Store (Provides Input)
	MAN
Role	ROL-04
Source	Tool provider (GLA)
Refines	REQ-CC-01, REQ-CC-02, REQ-CC-09, REQ-SO-DW-02

Table 69 – Supervised training of models for ranking deployment patterns (software requirement)

Section	Description

Id	**REQ-SO-ADS-04**
Level of detail	Software
Type	FUNC
Short name	On-demand re-ranking of deployment patterns
Description	The ADS-Ranking service is responsible for selecting a candidate deployment pattern for a user's application. It is anticipated that either due to poor quality of service estimations, changes in the network infrastructure or application data load, a user's deployed application and associated deployment pattern will be deemed insufficient, e.g. because throughput dropped below a pre-defined threshold. In this case, a new deployment pattern needs to be quickly selected and implemented bring the user's application back up to its quality of service requirements. This will be achieved re-ranking the candidate deployment patterns originally generated for the user's application, accounting for updated quality of service information and application failure mode (what triggered re-ranking). ADS-Ranking is also responsible for this re-ranking.
Additional Information	Input: - List<Candidate Deployment Pattern> (past deployment candidates) - List<<Candidate Deployment Pattern, Deployment Quality of Service Snapshot> (history for the current application) - Failure Mode Output: - Candidate Deployment Pattern (new pattern to deploy) - Candidate Deployment Pattern (old pattern for comparison) Service dependencies: - Centralized Data Store (Provides previous candidates and application deployment history) - Dynamic Orchestrator (Provides failure mode) - ADS-Deploy (Consumes Output)
	MAN
Role	ROL-04
Source	Tool provider (GLA)
Refines	REQ-CC-01, REQ-CC-02, REQ-CC-09, REQ-SO-DW-02

Table 70 – On-demand re-ranking of deployment patterns (software requirement)

Section	**Description**
Id	**REQ-SO-ADS-05**
Level of detail	Software
Type	FUNC
Short name	Deployment of a user's application using a candidate pattern
Description	The ADS-Ranking service is responsible for selecting a candidate deployment pattern for a user's application. Once a pattern has been

	selected, the associated user's application needs to be physically deployed or scheduled for deployment onto the cloud. Hence, there is a need for a service to operationalise application deployment.
Additional Information	Input: - Candidate Deployment Pattern (Configuration for the user's application) Output: - Application Deployment - Notification to Dynamic Orchestrator Service Dependencies: - ADS-Ranking (Provides Input – Candidate Deployment Pattern) - Cloud API (Deployment Methods) - Dynamic Orchestrator (Notification)
	MAN
Role	ROL-04
Source	Tool provider (GLA)
Refines	REQ-CC-01, REQ-CC-02, REQ-CC-09, REQ-SO-DW-02

Table 71 – Deployment of a user's application using a candidate pattern (software requirement)

Section	Description
Id	**REQ-SO-ADS-06**
Level of detail	Software
Type	FUNC
Short name	Re-deployment of a user's application using a candidate pattern
Description	If a user's application is deemed to no longer be meeting quality of service requirements, the ADS-Ranking component will identify an alternative deployment pattern to apply (see REQ-SO-ADS-04). This may require individual containers within a deployment to be moved to different machine types, more replications of a container to be added, or a halt and re-deployment of the entire user's application. The instantiation of the redeployment (reconfiguration) is the responsibility of the ADS-Deployment service. This involves identifying differences between the old and new candidate deployment patterns and calling appropriate cloud API methods to transition to the new configuration.
Additional Information	Input: - Candidate Deployment Pattern (new pattern to deploy) - Candidate Deployment Pattern (old pattern for comparison) Output: - Application Re-Deployment - Notification to Dynamic Orchestrator Service Dependencies: - ADS-Ranking (Provides Input – Candidate Deployment Patterns) - Cloud API (Deployment Methods) - Dynamic Orchestrator (Notification)

	MAN
Role	ROL-04
Source	Tool provider (ATOS)
Refines	REQ-CC-01, REQ-CC-02, REQ-CC-09, REQ-SO-DW-02

Table 72 – Re-deployment of a user's application using a candidate pattern (software requirement)

7.7 Distributed Storage & Analytics

The distributed storage & analytics component is an underlying pillar of BigDataStack that enables access to the data stored in the platform, in a seamless manner, so that the end-user should not care about its internal mechanisms. It is also integrated with other components that provide seamless access to the stored data. Due to this, it affects all defined roles of the platform.

Section	Description
Id	**REQ-SO-DSA-01**
Level of detail	Software
Type	DATA
Short name	Dynamic Load Balancing
Description	The distributed storage should be able to identify "hot-spots," meaning data nodes that consumes resources above a threshold (computational, memory, network, etc.), and re-distribute the deployed data regions accordingly to stabilize the system. This should be performed on the run-time, without downtimes or any performance effect on current analytical workloads. This way the redistribution would be performed seamlessly from the application point of view.
Additional Information	The dynamic load balancing can be achieved if some split points have been already defined, for the system to break down a dataset to specific data regions. The latter could be moved across data nodes. There should be a way for the data provider to define these split points.
Priority	MAN
Role	ROL-02, ROL-03, ROL-04, ROL-05
Source	Data storage technology provider (LXS)
Refines	REQ-SY-DAS-02

Table 73 – Dynamic Load Balancing (software requirement)

Section	Description
Id	**REQ-SO-DSA-02**
Level of detail	Software
Type	DATA
Short name	Elastic reconfiguration of the resources

Description	When data nodes are exhausted and the dynamic load balancing has no effect, we should be able to identify this situation, and following some elasticity rules, request new resources to be deployed from the storage and computing resources management. As soon as the resources are available, the system should be able to scale out to the new node, and perform the dynamic load balancing, to stabilize the system.
Additional Information	The elastic management of the resources for the distributed storage will be done internally. However, an API must exist to request resource acquisition and disposal from the resource management.
Priority	MAN
Role	ROL-02, ROL-03, ROL-04, ROL-05
Source	Data storage technology provider (LXS)
Refines	REQ-SY-DAS-02

Table 74 – Elastic Reconfiguration of the resources (software requirement)

Section	Description
Id	**REQ-SO-DSA-03**
Level of detail	Software
Type	FUNC
Short name	Enable access via the seamless analytical framework
Description	The distributed storage and analytics provides a JDBC and a key value driver for accessing the underlying data, using either its SQL Query Engine or directing accessing its internal storage. However, access to the underlying data must be made also via the seamless analytical framework, so that the user can let the latter decide where the data are stored. This requires that the distributed storage must allow the seamless framework to perform queries on the user's behalf.
Additional Information	N/A
Priority	MAN
Role	ROL-02, ROL-03, ROL-04, ROL-05
Source	Data storage technology provider (LXS)
Refines	REQ-SY-DAS-01

Table 75 – Enable access via the seamless analytical framework (software requirement)

Section	Description
Id	**REQ-SO-DSA-04**
Level of detail	Software
Type	DATA
Short name	Remove historical data and send them to the object store
Description	The distributed storage & analytics component should identify historical data (data that have been added before a given period), and in case these

	data do not concern operational entities, they should be removed from the component and sent to the object store.
Additional Information	N/A
Priority	MAN
Role	ROL-02, ROL-03, ROL-04, ROL-05
Source	Data storage technology provider (LXS)
Refines	REQ-SO-SAF-01

Table 76 – Remove historical data and send them to the object store (software requirement)

Section	Description
Id	**REQ-SO-DSA-05**
Level of detail	Software
Type	FUNC
Short name	Provide storage specific monitoring data
Description	This component must expose an API for the triple-monitoring component to be able to retrieve useful information regarding the DB operational status. This information should be number of current active transactions, number of successful commits, abort ratio, etc.
Additional Information	N/A
Priority	MAN
Role	ROL-02, ROL-03, ROL-04, ROL-05
Source	Data storage technology provider (LXS)
Refines	REQ-SO-TM-01

Table 77 – Provide storage specific monitoring data (software requirement)

7.8 Data Cleaning

Section	Description
Id	**REQ-SO-DC-01**
Level of detail	Software
Type	FUNC
Short name	Deep learning frameworks for building data cleaning models
Description	The framework needs to facilitate the implementation of complex machine learning and deep learning models, that will be able to learn the intricate relationships between data points, to discover and flag possibly corrupted data, in a domain agnostic way.
Additional Information	N/A
Priority	MAN
Role	ROL-03, ROL-04, ROL-05
Source	Tool provider (UPRC)

Refines	N/A

Table 78 – Deep learning frameworks for building data cleaning models (software requirement)

Section	Description
Id	**REQ-SO-DC-02**
Level of detail	Software
Type	FUNC
Short name	Implement deep neural networks on the Spark
Description	As mentioned in REQ-SO-DC-01, a fully-fledged machine learning and deep learning framework should be used to implement the data cleaning models. To enable such a specific implementation to run on the Spark platform, suitable libraries should be exploited. In particular, *Yahoo!*'s open source implementation "TensorFlowOnSpark" is a strong candidate for this task.
Additional Information	N/A
Priority	DES
Role	ROL-03, ROL-04, ROL-05
Source	Tool provider (UPRC)
Refines	REQ-SO-DC-01

Table 79 – Implement deep neural networks on spark (software requirement)

Section	Description
Id	**REQ-SO-DC-03**
Level of detail	Software
Type	FUNC
Short name	Containerization of Machine Learning models
Description	Models that cannot use Apache Spark's platform, and should be used as a Data Cleaning model, will be containerized and used as any other data operation. Thus, a specific Docker image for each algorithm should be provided.
Additional Information	N/A
Priority	DES
Role	ROL-03, ROL-04, ROL-05
Source	Tool provider (UPRC)
Refines	N/A

Table 80 – Containerization of Machine Learning models (software requirement)

7.9 Big Data Layout

Section	Description
Id	**REQ-SO-BDL-01**

Level of detail	Software
Type	FUNC
Short name	Big data layout monitoring
Description	IBM's data layout work will require monitoring query workload as well as measuring number of bytes scanned and number of REST requests per query.
Additional Information	Other data services may want other parameters to be monitored.
Priority	MAN
Role	N/A
Source	Tool provider (IBM)
Refines	N/A

Table 81 – Big data layout monitoring (software requirement)

7.10 Real-time CEP

Section	Description
Id	**REQ-SO-CEP-01**
Level of detail	Software
Type	FUNC
Short name	CEP Queries
Description	The Complex Event Processor streaming can make several queries at a time to provide a range of analytical capabilities so that all continuous queries can run in the system providing correct results with good performance.
Additional Information	N/A
Priority	MAN
Role	N/A
Source	Tool provider (UPM)
Refines	N/A

Table 82 – CEP Queries (software requirement)

Section	Description
Id	**REQ-SO-CEP-02**
Level of detail	Software
Type	FUNC
Short name	Correlation with data at rest
Description	The Complex Event Processor can correlate and aggregate real time events with data at *LeanXcale* data store without compromising performance.

Additional Information	N/A
Priority	MAN
Role	N/A
Source	Tool provider (UPM)
Refines	N/A

Table 83 – Correlation with data at rest (software requirement)

Section	Description
Id	**REQ-SO-CEP-03**
Level of detail	Software
Type	FUNC
Short name	CEP REST interface
Description	The Complex Event Processor should provide RESTful API to enable components developed using front-end technologies to deploy / un-deploy queries and to send events. By deploying a continuous query, the user should be able to validate that it exists in the system and that the event she sends are processed.
Additional Information	N/A
Priority	MAN
Role	N/A
Source	Tool provider (UPM)
Refines	N/A

Table 84 – CEP REST interface (software requirement)

Section	Description
Id	**REQ-SO-CEP-04**
Level of detail	Software
Type	FUNC
Short name	CEP JAVA interface
Description	The Complex Event Processor should provide a JAVA drover to enable components with an API to deploy / un-deploy queries and to send / receive events. By deploying a continuous query, the user should be able to validate that it exists in the system and that the event she sends are processed.
Success criteria	N/A
Priority	MAN
Role	N/A
Source	Tool provider (UPM)
Refines	N/A

Table 85 – CEP JAVA interface (software requirement)

Section	Description
Id	**REQ-SO-CEP-05**
Level of detail	Software
Type	FUNC
Short name	Metric Reporter
Description	The Complex Event Processor report statistics regarding the load being processed and the consumed resources in real time. User should be able to check the current workload statistics and consumed resources through a real-time dashboard.
Additional Information	N/A
Priority	MAN
Role	N/A
Source	Tool provider (UPM)
Refines	N/A

Table 86 – Metric Reporter (software requirement)

Section	Description
Id	**REQ-SO-CEP-06**
Level of detail	Software
Type	PERF
Short name	Edge Computation
Description	The Complex Event Processor provides mechanism to process event at the edge so that the percentage of events processed at the core is reduced.
Additional Information	N/A
Priority	MAN
Role	N/A
Source	Tool provider (UPM)
Refines	N/A

Table 87 – Edge Computation (software requirement)

7.11 Predictive and Process Analytics

Section	Description
Id	**REQ-SO-PPA-01**
Level of detail	Software
Type	ENV
Short name	Analytics Repository

Description	There must exist a NoSQL database support for data set features for meta-learning.
Additional Information	N/A
Priority	MAN
Role	ROL-02
Source	Tool provider (UPRC)
Refines	N/A

Table 88 – Analytics Repository (software requirement)

Section	Description
Id	**REQ-SO-PPA-02**
Level of detail	Software
Type	ENV
Short name	Data Descriptive Model Generator
Description	Process a data set with a given input form and collect attributes of the data from the below metric groups: • Simple (e.g., number of objects, number of features, etc.) • Statistical (e.g., mean, standard deviation, skew, etc.) • Information theoretic (e.g., entropy, mutual information, etc.)
Additional Information	N/A
Priority	MAN
Role	ROL-02
Source	Tool provider (UPRC)
Refines	

Table 89 – Data Descriptive Model Generator (software requirement)

Section	Description
Id	**REQ-SO-PPA-03**
Level of detail	Software
Type	ENV
Short name	Process Model Repository
Description	There must exist a NoSQL database for process models enabling all CRUD operations on the respective models.
Additional Information	N/A
Priority	MAN
Role	ROL-02
Source	Tool provider (UPRC)
Refines	N/A

Table 90 – Process Model Repository (software requirement)

Section	Description
Id	**REQ-SO-PPA-04**
Level of detail	Software
Type	ENV
Short name	ProM tools
Description	There must be an extensible, platform independent, framework supporting a wide variety of process mining techniques in the form of plug-ins.
Additional Information	N/A
Priority	MAN
Role	ROL-02
Source	Tool provider (UPRC)
Refines	N/A

Table 91 – ProM tools (software requirement)

7.12 Seamless Analytics Framework

Section	Description
Id	**REQ-SO-SAF-01**
Level of detail	Software
Type	FUNC
Short name	Seamless analytics framework monitoring
Description	For seamless component, we must monitor the rate of data flow across the pipeline and number of bytes scanned (sent to Spark) by each component.
Additional Information	Other data services may need other parameters to be monitored.
Priority	MAN
Role	N/A
Source	Tool provider (IBM)
Refines	N/A

Table 92 – Seamless analytics framework monitoring (software requirement)

7.13 Application Dimensioning Workbench

Section	Description
Id	**REQ-SO-ADW-01**
Level of detail	Software
Type	FUNC
Short name	Generic and scalable load injection process

Description	The framework needs to support a variety of load injection tools to cater for diverse data and application services dimensioning. Thus, a unified load input and launch process needs to be defined, that is able to adapt to different load generation tools and is able to parallelize the load generation for scalability purposes.
Additional Information	N/A
Priority	MAN
Role	ROL-04+ Data Service Owners
Source	Tool provider (UPRC)
Refines	REQ-CC-01, REQ-CC-02, REQ-CC-09

Table 93 – Generic and scalable load injection process (software requirement)

Section	Description
Id	**REQ-SO- ADW-02**
Level of detail	Software
Type	FUNC
Short name	Load injector dockerization
Description	To support a generic load injection process as indicated by REQ-SO-ADW-01, "dockerization" of the respective load generators per type of service needs to be performed. Thus, a specific Docker container image per needed load generator tool needs to be provided, along with a unified process for feeding the per case load description file based on the Docker API and configuration process.
Additional Information	N/A
Priority	MAN
Role	ROL-04, ROL-05
Source	Tool provider (UPRC)
Refines	REQ-CC-01, REQ-CC-02, REQ-CC-09, REQ-SO-DW-01

Table 94 – Load injector dockerization (software requirement)

Section	Description
Id	**REQ-SO- ADW-03**
Level of detail	Software
Type	FUNC
Short name	Service structure specification
Description	The service graph specification coming as input from the Process Modelling and Data Toolkit should follow the Docker Compose specification, to be understandable by the Dimensioning workbench. Following, the Dimensioning phase should add the respective candidate resource deployment options as additional custom metadata in the file to be used by the Deployment selection.

Additional Information	N/A
Priority	MAN
Role	ROL-04
Source	Tool provider (UPRC)
Refines	N/A

Table 95 – Service structure specification (software requirement)

7.14 Process Modelling Framework

Section	Description
Id	**REQ-SO-PMF-01**
Level of detail	Software
Type	FUNC
Short name	Process Modelling GUI
Description	The Process modelling framework should provide a graphical user-interface for creating business process diagrams. It must provide a palette with all available process steps and support “drag-and-drop.”
Additional Information	N/A
Priority	MAN
Role	ROL-04, ROL-05
Source	Tool provider (ATC)
Refines	N/A

Table 96 – Process Modelling GUI (software requirement)

Section	Description
Id	**REQ-SO- PMF-02**
Level of detail	Software
Type	FUNC
Short name	Process Model Export
Description	The export of the framework should be in an appropriate form to be used by the subsequent layers of the architecture. This form should be flexible enough to accommodate changes in the architecture and the addition/deletion of new process steps.
Additional Information	N/A
Priority	MAN
Role	ROL-04, ROL-05
Source	Tool provider (ATC)
Refines	N/A

Table 97 – Process Model Export (software requirement)

7.15 Data Toolkit

Section	Description
Id	**REQ-SO-DT-01**
Level of detail	Software
Type	USE
Short name	Analytics programming languages
Description	A strict requirement about to the capacity to support various technologies/programming languages for the development of analytics processes, given the existence and dominance of set of them: R, Python, Java, Scala, etc. The developed analytics processes have also to be deployable over big data computing frameworks (e.g., *Apache Spark* or *Apache Flink*).
Additional Information	N/A
Priority	MAN
Role	ROL-02
Source	Tool provider (UBI)
Refines	REQ-SO-DT-01

Table 98 – Analytics programming languages (software requirement)

Section	Description
Id	**REQ-SO-DT-02**
Level of detail	Software
Type	SUP
Short name	Analytic function reuse
Description	The data toolkit must provide open APIs to data scientists to define not only the overall data analytic workflow but also the associated analysis scripts and execution endpoints per workflow step. For each step/node of the graph, the data scientist must be able to implement and register in the catalogue a chainable analytics primitive/software that concerns the analysis process to be executed. Characteristics related to input data parameters (type of data sources without any binding), output data parameters, analysis configuration parameters, execution substrate requirements and the software package must be denoted.
Additional Information	N/A
Priority	MAN
Role	ROL-02
Source	Tool provider (UBI)
Refines	REQ-SO-DT-01

Table 99 – Analytic function reuse (software requirement)

7.16 Adaptable Visualizations

Section	Description
Id	**REQ-SO-AV-01**
Level of detail	Software
Type	FUNC
Short name	Interactive Graphs
Description	The adaptable visualisations should provide a wide range of interactive graphs. The user should be able to select the type of the graph for every usage scenario. Basic and advanced filtering capabilities should be made available.
Additional Information	N/A
Priority	MAN
Role	ROL-03
Source	Tool provider (ATC)
Refines	N/A

Table 100 – Interactive Graphs (software requirement)

Section	Description
Id	**REQ-SO-AV-02**
Level of detail	Software
Type	FUNC
Short name	Real-time Monitoring Data Visualization
Description	The adaptable visualizations should support monitoring of real-time data using a proper data streaming architecture.
Additional Information	N/A
Priority	MAN
Role	ROL-03
Source	Tool provider (ATC)
Refines	N/A

Table 101 – Real-time Monitoring Data Visualization (software requirement)

8 Baseline Technologies

This chapter shortly presents the baseline technologies to be used to introduce the ground for the proposed work and to ensure that non-conflicting (in terms of functional properties and integration with other components) technologies will be used. It does not simply state some state-of-the-art technologies but rather links them under the context of BigDataStack and what BigDataStack can get from them.

8.1 Computing Resources Management

Public clouds (such as ***GCP***[11] or ***IBM Cloud***[12]) offer infrastructure-as-a-service (IaaS). IaaS is a pay-per-use service model to consume virtual computational (e.g., virtual machines), storage and networking resources. This service provides high levels of QoS and wide variety of resource types (e.g., machines with different memory, CPU and acceleration chipset features).

OpenStack[13] is an open source software that aims to create private and public clouds. It lets companies to create an IaaS in their data centre. OpenStack lets you add servers, network and storage components easily and efficiently to your cloud. It controls large pools of compute, storage, and networking resources throughout a data centre, managed through a dashboard or via the OpenStack API.

Container management/orchestration platforms are becoming popular solutions to make it easier provisioning and managing cloud applications. Some examples are ***Kubernetes***[14], ***Docker Swarm***[15], ***Amazon ECS***[16], or ***Mesosphere Marathon***[17]. These typically provide an API for developers to upload, organize, run, scale, manage and stop containers. They also normally provide a command line interface (CLI) and a web console to configure and monitor the performance of the services and resources of the platform, such as:

- container deployment and configuration,
- performance monitoring,
- cluster management and scaling,
- logging, and
- container lifecycle management.

Currently, Kubernetes (K8S) is the most popular container orchestration solution. K8S schedule container deployment and run in a single machine or a cluster of machines, while they manage the availability, capacity, and maintenance of the underlying infrastructure on top of which the cluster run. Everything in the K8S platform is treated as an API object representing a concrete instance of a resource type in the cluster, like a pod or namespace:

- Kubernetes Pod: It represents the smallest and simplest object to be managed by Kubernetes. A pod is a group of related containers collocated on a Kubernetes cluster's

[11] GCP (Google Cloud Platform). https://cloud.google.com
[12] IBM Cloud. https://www.ibm.com/cloud/
[13] Open Stack. https://www.openstack.org/
[14] Kubernetes. https://kubernetes.io/
[15] Docker Swarm. https://docs.docker.com/engine/swarm/
[16] Amazon ECS (Elastic Container Service). https://aws.amazon.com/en/ecs/
[17] Mesosphere Marathon. https://mesosphere.github.io/marathon/

node, sharing storage/network and a specification for how to run the containers[18]. A pod's containers are always co-located and co-scheduled, and run in a shared context. Replica sets let developers specify the number of pods replicas to be running at one time. This is the simplest way to scale up and down the number of instances of a certain component providing a certain service within your application.

- Kubernetes Namespace: It supports multiple virtual clusters on the same physical cluster. Namespaces are used to organize objects in a cluster and provide a way to divide cluster resources. Names of resources need to be unique within a namespace, but not across namespaces. It is possible to specify constraints that limit aggregate resource consumption per Namespace.
- Kubernetes Service: It describes how to access applications, such as a set of pods, and can describe ports and load-balancers. The access point can be internal or external to the cluster.

Kubernetes scheduler is built around the concept of managing CPU and memory resources at a container level. Every Kubernetes cluster' node (instances to schedule containers to) is assigned an amount of schedulable memory and CPU. At deployment time, every container specifies how much memory and CPU it will request. And the scheduler finds the best fit given the allocated CPU and memory on the nodes.

Kubernetes defines the CPU and MEMORY resources using basic constructs:

- The **request** value specifies the min value you will be guaranteed. This corresponds to CPU shares with *CGroups* and is used to determine which containers should get killed first when a system is running out of memory. The **request** value is also used by the scheduler to assign pods to nodes. Therefore, a node is considered available if it can satisfy the requests of all the containers in a pod.
- The limit value specifies the max value you can consume. This corresponds to a *CGroup* CPU quota and memory limit in bytes. Limit is the value applications should be tuned to use.

Kubernetes defines several different quality of service (QoS) tiers based on how request and limit are specified:

- Best-Effort: A request value of 0 (unlimited) with no limit set is classified as best effort. Best-Effort containers are the first to get killed when resources are limited.
- Guaranteed: A container with a request value equal to its limit. These containers will never get killed based on resource constraints.
- Burstable: A container with a request value less than its limit. It will be killed after Best-Effort containers when resources are limited if their usage exceeds their request value.

While those are the basic constructs, different applications have different needs and data driven applications should consume these resources in an "intelligent" way.

[18] https://kubernetes.io/docs/concepts/workloads/pods/pod/

8.2 Storage Resources Management

As we base our infrastructure on Kubernetes (see previous section), we need to tackle usage of storage by the application through the Containers Orchestrators perspective.

The current model for persistent storage for containers is settling out on attaching volumes (LUNs) to a container and formatting that volume with a file system. Through the container engine, the file system is exposed as a mount point within the container. Storage could be taken from local disks (e.g. a volume created using an LVM) or from external storage presented to the container server/node.

To make it easier to plug in storage interfaces into the Container Platform, K8S created the CSI. The aim of the Container Storage Interface (CSI) is to provide a common standard to connect container orchestration platforms (COs) like Kubernetes, Docker Swarm and Mesos to a plugin and ultimately to persistent storage (see previous section).

Theoretically, with a standardised communication protocol, storage vendors will only need to write a plugin to a single specification. CSI sets out the definition of how to talk to the plugin; exactly how the plugin is managed or operates is up to the storage provider. CSI provides the following capabilities.

- Dynamic provisioning and decommissioning of volumes.
- Attachment and detachment of volumes from a host node.
- Mounting and unmounting of a volume from a host node.

This technology enables us in BigDataStack to create flexible and dynamic management for the storage resources needed by the different workloads of the applications.

8.3 Data-driven Network Management

Data-driven network management regards a set of functions related to optimal network setup to cover the operational requirements of analytic processes in terms of efficient data exchange. In most of the cases, analytic process deployments are realised within a data centre (DC) environment and do not impose strict requirements in terms of layer-3 routing mechanisms. Optimal network flows' setup and management must be realised within a DC environment. Towards this direction, Software Defined Networking Mechanisms (SDN) can be applied along with network management mechanisms (e.g. enforcement of network policies) supported by the Orchestration Engine.

For instance, in case of Kubernetes, a network policy is a specification of how groups of pods can communicate with each other and other network endpoints. Network Policy resources use labels to select pods and define rules which specify what traffic should be allowed. In case of application of SDN-based mechanisms, ***OpenFlow***[19] is considered as the dominant communications protocol that gives access to the forwarding plane of a network switch or router over the network. OpenFlow enables network controllers to determine the path of network packets across a network of switches.

The adopted set of solutions for Data-driven Network Management is going to be well bound to the orchestration solution to be designed and implemented within BigDataStack,

[19] https://www.opennetworking.org/technical-communities/areas/specification/open-datapath/

guaranteeing the support of novel and intelligent network management mechanisms, applicable to data-intensive and network-intensive processes.

8.4 Dynamic Orchestrator

Orchestration (as an infrastructure management service) refers to the enablement and the coordinated handling of various optimizations inside the platform. Examples of such optimizations are the placement (or allocation) of tasks to computing resources, decisions regarding parallelization degrees of parallelizable tasks/services, load balancing, algorithm selection, and more. In the state of the art, such optimizations are handled in a way that we call "service-driven". This means that the optimization functions, the criteria, and the system setup are built around basic features such as CPU power, network bandwidth, and task/service requirements to optimize certain metrics (e.g. latency) with regard to the examined service.

Related works [46][47] exploit obvious known synergies such as the fact that running tasks on low-CPU nodes can increase processing time or the fact that overloading certain links can create bottlenecks. Apart from the fact that such concrete optimizations have rarely been handled homogeneously or investigated in a common context, there are also gaps towards making them data-driven rather than service-driven.

To support data-driven overall orchestration and data-driven solutions of specific optimization problems (e.g. placement), we propose techniques to identify the synergies between characteristics of data analytics and system KPIs, e.g. functions that represent how data I/O volumes affect the CPU-intensity of certain tasks etc. We will provide the basis for using such data-related synergies for all system orchestration aspects by defining machine-readable profile specifications (hereinafter referred to as Synergy Profile) for profiling homogenously all entities (e.g. nodes, algorithms, network links, application tasks) that are involved in data-driven orchestration. A rule-based approach for triggering runtime optimizations based on the monitored data will be delivered. We will also define interfaces and procedures where the Dynamic Orchestrator communicates to other controllers to create and maintain the Synergy Profiles. The baseline technologies that will support the ***Sinergy Profiler*** implementation and integration are preliminary defined as the following:

- ***Drools*** as the rule engine for triggering orchestration actions that must be enforced over the operational application based on real-time monitoring data.
- ***Kubernetes*** and ***Mesosphere DC/OS*** for application placement and container orchestration.
- ***LeanXcale*** will provide an interface for data services deployment.
- ***OpenFlow*** for network functions recompilation.
- ***OpenStack*** will support live migration procedures.

8.5 Triple Monitoring

To be able to maintain a good quality and perform best adaptation based on the change that could happen in a system, metrics need to be taken continuously and exposed to the component involved in the evaluation of quality and adaptation. In the context of big data stack, tracking information will be performed by the Triple monitoring engine. Three different groups of metrics need to be tracked: infrastructure information, data operation (data produced by applications running on the platform) and all data involved in database transactions.

Since these metrics are produced by applications with different purposes, specifications, functionalities and technologies, two approaches will be used, the first is to use probe to directly ingest metrics into the monitoring collector. The second approach is to provide a sanitizer to prepare metrics conforming with the specification of the collector and ingest them. This sanitizer will act as a unified API.

The triple monitoring engine has an input REST API which is an entry point of the system and an output REST API for exposing data to all applications data consumer. The monitoring should provide an efficient and fast way of transferring metrics from the input to the manager that handle all the logics of the engine. The big number of metrics from different sources must be organized chronologically and presented to a correct format for their visualization. We've been interested by two main technologies:

Prometheus is a technology for monitoring management, which includes metrics collection facilities. This technology will be very convenient for the following reasons[20]:

- Powerful queries: A flexible query language (NoSQL based) allows slicing and dicing of collected time series data.
- Efficient storage: Prometheus stores times series in memory and on local storage in an efficient custom format. Scaling is achieved by function sharing and federation.
- Extensive integration: Many existing exporters allow bringing data from third-party application to its collector.
- Push gateway: In case it's impossible to scrape metrics (using probe), metrics can be exposed to the Prometheus collector by this mechanism[21].

The manager needs a persistent connection with the output REST API, a connection oriented based technology will be used. **RabbitMQ** will be very convenient because of the following[22]:

- Availability in many languages and platform.
- Asynchronous Messaging: Supports multiple messaging protocols, message queuing, delivery acknowledgement, flexible routing to queues, multiple exchange type. Those features allow for publish/subscribe communication and high-speed asynchronous I/O engines, in a tiny library.
- Distributed Deployment: Deploy as clusters for high availability and throughput; federate across multiple availability zones and regions.

Persistent data need to be stored for later use, since all REST API within triple monitoring engine use JSON format and metrics don't have the same structure because of their respective origin, a convenient technology for saving these data will be using a database that handle JSON format to facilitate data transfer within the triple monitoring engine and to allow polymorphism. Based on the amount of data arriving per second and the huge quantity of operation that need to be perform MongoDB will be very efficient [48].

As said before, the triple monitoring engine provides two REST interfaces.

1. The first has the goal of receiving data from different sources and sending them to the Netdata collector (plugin). This interface will be the input of the monitoring engine.

[20] Prometheus. https://prometheus.io/
[21] Prometheus https://prometheus.io/
[22] RabbitMQ https://www.rabbitmq.com/

The API keeps data in memory until it is consumed by the plugin. Applications (data producers) will have access to this API for sending their measurements.

2. The second interface provides the output of the monitoring engine to applications (consumers). This interface has two kinds of connection to serve results: a REST API and a Publish/Subscribe mechanism that is connection-oriented service.

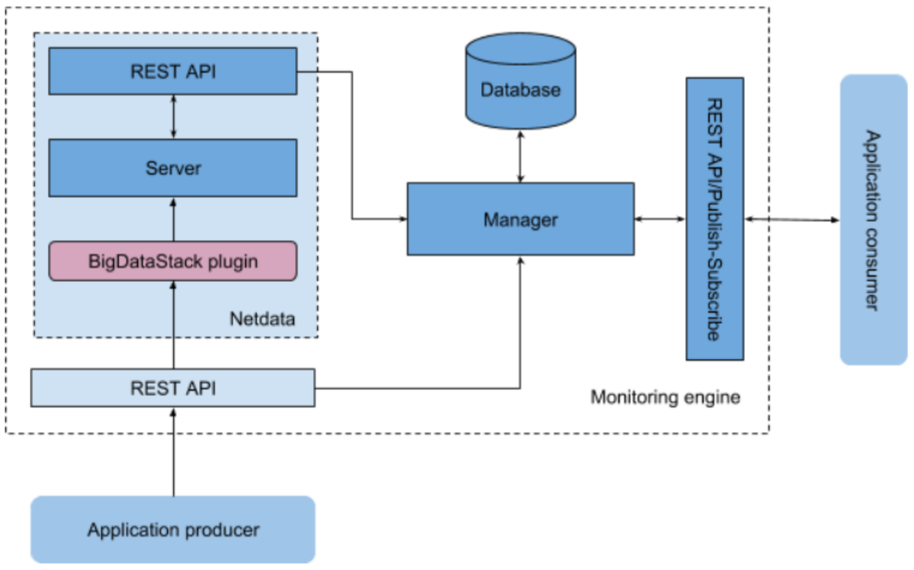


Figure 3. Netdata role in triple monitoring

Netdata is a system for health and performance monitoring of distributed real-time systems. It provides real-time insights of everything happening on the system it runs (including applications such as web and database servers), using interactive web dashboards [7]. Netdata main capabilities are gathering data from different sources and exposing them through a REST API. Netdata architecture is extensible through plugins to read measurements (metrics) from different sources. In Figure 3 , the component named “BigDataStack plugin” is an adapter that needs to be deployed to ingest data into Netdata.

8.6 Applications & Data Services Deployment

Application and Data Services Deployment is concerned with how to deploy the user’s application onto the cloud infrastructure, as well as subsequent re-deployment in cases where the initial deployment did not meet the user’s quality of service requirements. It belongs within the realization engine of the overall BigDataStack platform. More precisely, it is comprised of two main components:

I. **ADS-Ranking**: A component that ranks different possible deployment configurations of the user’s application on the cloud hardware (referred to as candidate deployment patterns) such that we can select the most suitable one given the user’s requirements. In a redeployment scenario, this component re-ranks and selects a new candidate deployment pattern that solves issues identified with the previous deployment.

II. **ADS-Deployment**: A component that takes the best candidate deployment pattern and physically deploys it on the cloud infrastructure. Meanwhile, in a re-deployment scenario, this service facilitates the transition of one-or-more already deployed user services to a new configuration based on an updated candidate deployment pattern.

8.6.1 ADS-Ranking

What is ADS-Ranking? ADS-Ranking is a component that is concerned with how best to deploy the user's application onto the cloud based on information about the application and cluster characteristics. From a practical perspective, its role is to identify which - of a range of potential deployment options - is the best for the current user, given their stated (hard) requirements and other desirable (soft) characteristics (e.g. low cost or high throughput). ADS-Ranking is strongly coupled to the Application & Data Services Dimensioning (ADS-Dimensioning) component of BigDataStack that sits above it in the overall architecture stack. The main output of ADS-Dimensioning is a series of candidate deployment patterns (ways that the user's application might be deployed). It is these deployment patterns that ADS-Deploy takes as input, and subsequently selects the best one to deploy. In practice, this is accomplished using state-of-the-art supervised machine learning techniques to rank the candidate deployment patterns, which model the relationships between the features of each candidate deployment pattern and the user requirements/preferences. ADS-Ranking is also used for application re-deployment in cases where a previously selected deployment was deemed unsuitable, for example, because the provided quality of service was too low or the cost too high. In this case, ADS-Ranking updates the underlying model with evidence from the failing deployment and re-ranks the candidate deployment patterns with the aim of finding a new one that will provide superior performance.

Literature Review Machine Learning: Machine learning refers to the field of approaches that automatically learn solutions to problems using prior data [22]. Machine learning has become closely linked with information retrieval, as many tasks in information retrieval can be formulated in a manner that can be tackled by machine learning approaches, e.g. categorising documents [23] or learning how to rank documents [24]. Moreover, machine learned approaches have shown to be effective for many of these tasks [25]. Indeed, commercial Web search engines like Google and Bing use machine learned models to drive their search rankings. An important concept within machine learning is that of a *feature*. A feature is some property about the subject of the learning. For example, for information retrieval ranking problems, the features might be about the documents or user queries. An example of a query feature is query length, while a document feature might be its PageRank [26] score. For the purposes of ADS-Ranking that we are concerned with here, our features are derived from the 'predicted performance' information within each candidate deployment pattern, i.e. estimations about how effective a candidate deployment pattern is likely to be.

Literature Review Learning-to-Rank: Learning to rank (LTR) approaches use machine learning to tackle item ranking problems [25]. In an information retrieval setting, this typically involves ranking documents with respect to relevancy, although other ranking criteria are possible. The aim of learning to rank approaches is to improve a given item ranking with respect to some property. This is achieved by re-ranking an initial ranking such that items with the desired property are promoted into the top ranks.

In their simplest form, learning to rank techniques use initial item rankings for a set of topics, features about the individual items within those rankings, and relevance assessments about the individual items for each topic, to form a ranking model. This model can then be applied to unseen item rankings, re-ranking them to increase some desired ranking property. When building (or training) a model, an initial item ranking is created, referred to as the *sample*. A sample should have high recall in terms of items with the desired ranking property, e.g. for relevancy-based rankings, the sample should contain many relevant documents [27]. However, these documents do not need to appear within the top ranks; indeed, it is the aim of LTR to achieve this through re-ranking. Next, features about each of the items are extracted. An effective feature should aid in distinguishing the items that have the desired property, e.g. relevance to the query. In effect, the LTR approach aims to find a combination of these features that leads to effective ranking. Indeed, given the sample and its features, a learning to rank approach will try different combinations of those features to find those that lead to increased effectiveness when ranking the sample. LTR approaches repeat this process for many document samples to find the feature combination that leads to improved effectiveness across all those samples. This feature combination is referred to as the ranking model. The idea is that the resultant ranking model will generalise to unseen sample rankings, if the training samples are representative of the types of rankings encountered. The ranking model can be updated with new samples over time, which is referred to as Active/Reinforcement Learning.

Learning to rank approaches can be categorised into three different types. Each type of approach uses a different strategy to evaluate the sample ranking. These types are point wise, pair wise and list wise. Point wise techniques learn on a per-item basis, i.e. each item is considered independently. Pair wise techniques optimise the number of pairs of items correctly ranked. List wise techniques optimise the entire ranking list at one time. Prior work has indicated that list wise techniques learn more effective Models [25].

Within ADS-Ranking we use learning-to-rank techniques to produce models to rank different candidate deployment patterns for the user. In a re-deployment scenario, Active Learning and Reinforcement Learning are used to adapt the ranking model on-the-fly to enable the re-ranking of deployment patterns.

Selected Technologies: For BigDataStack, the ADS-Ranking component will be built on top of the open source Apache Spark framework and will be written in Java. ADS-Ranking will be deployed as a real-time stream processing service using Spark Streaming, which assesses candidate deployment patterns for different user applications and emits a single selected pattern for each. In this way, the component will be scalable to high-demand periods, as well as extensible. The component will be operationalized as a series of transformers within the Spark service, divided into: transformers for converting candidate deployment patterns into features; the learning of ranking models based on those features; and the application of those models for ranking during service operation. Model training will be implemented using Spark MLib.

8.6.2 ADS-Deployment

Cloud application service orchestration frameworks manage all dependencies and relationships between components of the application, facilitating infrastructure-centric orchestration in the cloud. They provide a solution for the provisioning and management (from deployment to adaptation) of complex applications in the cloud. We use the concept of “service template” or “application chart” to refer to the specification of the service structure and “orchestration” to the management of the behaviour of IT infrastructure services where the application is deployed and operated.

OpenStack Heat[23] and ***AWS CloudFormation***[24] are based on specific language descriptors. TOSCA[25] specification provides a language to describe application service components and their relationships using a service topology. This specification also provides mechanisms to describe management procedures that create or modify services using orchestration processes. Popular cloud tools such as OpenStack Heat conform to this specification.

jClouds[26] is a toolkit that facilitates interoperability with different cloud tools. It gives developers the freedom to create applications that are portable across clouds while giving you full control to use cloud-specific features. It is an open source multi-cloud library for the Java platform.

Docker[27] is a technology for operating-system-level virtualization and a popular alternative to package and manage applications through its lifecycle. Docker uses the resource isolation features of the Linux kernel to allow independent "containers" to run within a single Linux instance and therefore avoiding the overhead of managing multiple virtual machines (VMs). To achieve this, Docker let developers package and deploy libraries and any other configuration and dependency at operating system level with the application code.

Container orchestration solutions are based on the concept of Docker containers. ***Docker Swarm***[28] is the native container orchestrator of Docker. Swarm uses the same Docker interface, which enables transparent scalability from Docker use to Swarm use. Swarm manager is responsible for the management of a cluster of so-called Docker hosts over which it distributed the containers for execution.

Kubernetes[29] is the most popular container orchestration solution. Kubernetes scheduler runs as a process alongside the other master components such as the API server. A Kubernetes pod models an application-specific "logical host" and contains one or more application containers which are relatively tightly coupled. Pods can also be exposed as Kubernetes services, which father facilitates the management of containerized application services. Kubernetes charts offer a way to model, deploy and manage the life-cycle of complex containerized applications (i.e. consisting of several inter-related containers, pods and services). ***Helm***[30] is the open source library implementing this capability. It offers an API to deploy applications on Kubernetes and manage their life-cycle. By means of Helm, developers can install, delete and upgrade whole applications/services with one command.

Other solutions for application management in the cloud are ***Alien4Cloud***[31], supporting application SLA (service level agreement) specification, and ***Serf***[32], offering a decentralized

[23] Openstack Heat. https://wiki.openstack.org/wiki/Heat,
[24] AWS CloudFOrmatio. https://aws.amazon.com/cloudformation/,
[25] OASIS Topology and Orchestration Specification for Cloud Applications (TOSCA) , https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=tosca
[26] Apache jClouds. https://jclouds.apache.org/
[27] Docker. https://www.docker.com/
[28] Docker Swarm. https://docs.docker.com/engine/swarm/
[29] Kubernetes. http://kubernetes.io/
[30] Helm. https://helm.sh/
[31] Alien4Cloud. http://alien4cloud.github.io/
[32] Serf. https://www.serf.io/

solution for cluster management, failure detection, and orchestration; ***Cloudify***[33], ***Chef***[34] and ***Puppet***[35], on the other hand, are general purpose infrastructure configuration solutions which provide very flexible automated application lifecycle management.

8.7 Distributed Storage & Analytics

ACID database systems providing transactional semantics had been to the foreground for many decades. However, after the evolve of the cloud computing ecosystem, application components could scale in numerous nodes and serve highly intense workloads, thus requiring from the persistent storage systems to be able to handle these intensive loads. Traditional database systems however rely on the two-phase-commit [28, 29] as the de facto atomic commit protocol. The latter inevitably introduces increased latency, which makes it slow, as it requires two rounds of messages between the coordinator and the participants of the transaction. Additionally, the existence of a central coordinator to manage the transaction lifecycle during the commit phase makes the protocol very difficult to scale adequately to be able to handle numerous nodes. This limited scalability of transactions offered by traditional database systems lead to the emergence of new data management technologies, frequently known as NoSQL, that trade the lack of support for ACID transactions for scalability, thus delegating consistency checks and transactional management on the application level.

To overcome these limitations, during the last decade, quite a few solutions that try to combine scalable transactional support without sacrificing consistency, such as Percolator [30] or Spanner [31] have been proposed. However, they continue to suffer from the limitations as most of them still rely on variations of the two-phase-commit protocol, which inherently introduces increased latency by design or having a centralized transactional manager like Omid [32] and Apache Tephra [33], which prevents them from being able to scale linearly when large deployments are needed. Finally, others make use of expensive hardware for requesting time events, like Spanner, Deuteronomy and LEAP.

Moreover, traditional database systems providing transactional capabilities, thus serving OLTP workloads ensuring ACID properties, rely on locking mechanism to provide the required isolation level. This means that heavy analytical queries will prevent write operations on the database until the formers finish, and vice versa, intensive operational workloads prevent analytical queries to be executed, as each of these types of queries block each other. To overcome this limitation, enterprises used to make use of ETLs to take a snapshot of the data, duplicate it in a data warehouse and perform the analytical operations on the latter. However, the past few years have witnessed a rise in demand for real-time Big Data Analytics for real-time business intelligence with a large range of research terms being adopted [34]. The goal is to develop tools that enable analytical processing over data that should be most up to date, thus processing data as soon as they arrive into the system [35]. Even if most systems claim to have real-time capabilities [36], they should be considered "near-real-time" as they still rely on an update process to acquire the latest data snapshot. As organizations increasingly require analytics on fresh operational data to derive timely insights, the notion of hybrid OLTP and OLAP databases have emerged, currently most known as HTAP (Hybrid Transaction and Analytical Processing) that does not involve the use of some kind of ETLs which are cost-

[33] Cloudify. http://docs.getcloudify.org/3.3.1/intro/what-is-cloudify/
[34] Chef. https://www.chef.io/chef/
[35] Puppet. https://puppet.com/

expensive and introduce data duplication while they do not provide analytical processing over real-time data at the very end. ***Hyper*** [37] and ***SAP HANA*** [38] are typical HTAP database systems. The limitation of Hyper however is that it cannot be scaled horizontally as it must be deployed on a single machine, while SAP HANA suffers under intensive OLTP workloads and performs worse than a single-node typical database system.

To overcome all these challenges, ***LeanXcale***[36], which is a relational data base on top of a key-value store, is capable of handling write-intensive workloads, exploiting its ultra-scalable transactional management engine with its ability to linearly scale to 100s of nodes, while ensuring ACID properties and data coherence. Moreover, it provides tools for analytical processing as an integral part of the data store, which can exploit both inter- and intra-query parallelism, to provide real-time Big Data Analytics, thus truly implement the "just-in-time warehousing" model. LeanXcale internal OLAP engine is also distributed and each instance can be co-located with a corresponding data node, thus exploiting data affinity for improved performance. Moreover, its internal key-value storage engine can distribute its data load, by splitting, and merging data regions, whilst moving them across its data nodes on the runtime. During this process, the data store is fully operational, and no performance overhead is being noticed. Thus, the online re-distribution of the data load is performed seamlessly from the application point of view. Finally, for the application developers to use the distributed engine, an Elastic JDBC driver is provided, which implements all functionalities that can be found in traditional relational database management systems. Moreover, an additional driver can be used for directly accessing the internal key-value data store, which skips the overhead introduced by the Query Engine of the platform for improved performance.

8.8 Live Migration

Live migration is a pervasive technique in the realm of virtual machines, allowing transparent movement of virtual machine instances (VMs) from one physical machine to another with negligible service disruption (hence the term live). Live migration describes a mean to transfer a running VM from one physical host to another host without interrupting the VM execution and transparent to the VM's users. The required information is transferred over the network (e.g., Ethernet) and includes an option to use an encrypted connection.

With recent trends toward scalability and distributed microservices, containerization has quickly become a lightweight and widely adopted alternative to VMs or physical nodes as the unit of deployment. The biggest benefit of containers, is that they can be quickly created and destroyed in large numbers, and therefore relocating a container/pod in Kubernetes is only possible by disposing of the source pod, then recreating a new pod of the same type/template from scratch (see Section 8.1). Application developers need to design around this fact by not relying on longevity of pod-local state and by storing any necessary data into pod-independent persistent storage, such as an external database. Along with the fact that there are many scenarios that would benefit from the ability to relocate active pods, the work on developing live migration for containers has started, but it's still in the early stages.

8.9 Data Cleaning

Data quality and verification is of major importance given that it affects the complete data path: storage, processing, analytics results, decision-making, etc. It poses many challenges in

[36] https://www.leanxcale.com/

several phases of data management. In contrast to the much more researched modelling and analysis phases, the quality analysis and verification step is often seen as a sore point, even though without it the modelling and analysis phases could be of limited value. If the data are not verified and of acceptable quality (e.g. missing values, outliers etc.), the conclusions might be associated with a high level of uncertainty or even reduced to garbage.

In this project, we will develop mechanisms to evaluate the data quality, in terms of completeness and accuracy. These approaches will exploit artificial neural networks (ANN) and Deep Learning (DL) techniques, taking advantage of the work done on Natural Language Processing (NLP), to extract latent features that correlate different fields, and identify possible defects in the content. Furthermore, by harnessing the power of Machine Learning (ML) and DL, the proposed approach will automate the process of *data harmonization*, an approach to data quality that improves the condition and utilization of the data. Data Harmonization is about creating a single source of truth. It works by absorbing diverse data from various sources, brushing away any inconsistencies, purifying it and presenting it as an integrated whole. Artificial Intelligence and ML can simplify and automate this operation, thereby speeding up the process of data modelling.

Over the past few years, both industry and academia have shown great interest on researching different aspects of data cleaning and applying new methods, including but not limited to new abstractions [1]–[4], interfaces [5], [6], approaches for scalability [7]–[10], or even crowdsourcing techniques [11]–[13]. The main differentiator comes from the error definition itself; on one hand, quantitative techniques are used for anomaly detection and outlier exposure, using statistical methods (e.g. a value that is more than three standard deviations from the mean should be an error), on the other hand qualitative techniques use a rule-based approach, to detect errors (e.g. A man can never give birth to a child). Once errors are detected, they can be corrected using a script, a human crowd or human experts. There are even situations that a hybrid of two or more approaches yields better results.

As it follows, a data cleaning process consists of two phases: i) the error detection, and ii) the error repairing. Concerning error detection, the techniques used can be classified based on three main questions [14]:

A. what to detect,
B. how to detect it, and
C. where to detect.

What to detect refers on error type, namely integrity constraints, missing or duplicate values etc. The how to detect question indicates the level of automation in the system. While most methods can be fully automated, like detecting violations of functional dependencies [15], in some others the human element is necessary [16]. Where to detect covers the business logic layer, where errors can be detected in the original source (i.e. the original database) or the target (i.e. the data warehouse) [17], where business logic is defined (e.g. an error on the total budget assumes that some aggregates must be in place).

On the second phase, while repairing errors, the main questions are:

A. what to repair,
B. how to repair, and
C. where to repair.

The first question considers what the learning algorithm assumes. If it has complete confidence in the business rules, then everything diverging from the rules is flagged as an error [4]. If the algorithm trusts the data, then it can relax the constraints to "fit" the data [18]. Finally, there's the option to explore both relaxing business rules and conforming to them [19]. The how to repair question refers once again to the automation level of the process, where automated techniques can be deployed, or the human element shows up, even if they are training learning models to carry out the job [20]. Conclusively, the where to repair question aims to answer if the repairs will be made in place, effectively destructing the original database, or a model is defined to describe possible repairs.

Where Artificial Intelligence (AI) and *pattern recognition* is concerned, machine learning is often used to improve the efficiency and accuracy of the data. For instance, ***ActiveClean*** [21] utilizes appropriate methods to select the most valuable data, while iteratively updates ML models given newly clean data. This way, data models can be correctly produced from a small subset of the data and be as accurate as they would be if the full dataset was employed.

To facilitate the implementation of such intricate deep learning algorithms, a suitable framework will be used. The prevailing option is that of Google's ***Tensorflow***[37]. Several implementations like Yahoo!'s ***TensorFlowOnSpark***[38] or ***SparkNet***[39] allows this framework to run in a distributed way over a Spark cluster, thus, making it totally compatible with the BigDataStack's platform. Yahoo!'s implementation seems superior because it requires minimal change in the original *TensorFlow* code.

Finally, some of the challenges that need to be addressed are those that pose scalability issues, user engagement, processing of semi-structured or unstructured data, streaming data and new privacy regulations or security concerns [15].

8.10 Big Data Layout

Today's best practices to deploy and manage cloud compute and storage services independently leaves us with a problem: it means that potentially huge datasets need to be shipped from the storage service to the micro service to analyse data. If this data needs to be sent across the WAN then this is even more critical. Therefore, it becomes of ultimate importance to minimize the amount of data sent across the network, since this is the key factor affecting cost and performance in this context. Many cloud-based SQL services, for example Amazon Athena[40], bill users according to the amount of data scanned in object storage, outlining the importance of this metric. There are currently three main approaches to limit the number of bytes sent from the storage to Spark. (Note we focus on object storage although this can also be applied more broadly).

The first is to use specialized column based formats such as Parquet[41] and ORC[42]. These formats provide column wise compression, which significantly reduces the number of bytes to be sent. They also support column pruning, so that only columns requested by a query

[37] https://www.tensorflow.org/
[38] https://github.com/yahoo/TensorFlowOnSpark
[39] https://github.com/amplab/SparkNet
[40] https://aws.amazon.com/athena/pricing/
[41] https://parquet.apache.org/
[42] https://orc.apache.org/

need to be sent to Spark. Finally, they sometimes support specialized metadata which can be used to filter columns following query predicates. Parquet can provide more than an order magnitude performance improvement over other formats such as csv[43].

The second approach is to use Hive style partitioning to name the objects using a certain convention. In this case, information about object contents is encoded into object names. For example, if we partition per a date column then each object will contain data records with the same date, and the date is encoded in the object name. Spark SQL understands this convention and can filter the set of objects retrieved when query predicates apply to partitioning columns. This can significantly reduce the number of objects sent to Spark and the number of bytes shipped (for more information see IBM's recent blog post[44]).

The third approach is called Data Skipping and it can complement the first two approaches. This approach utilizes metadata to track information about objects and their dataset columns which can then be used for data skipping i.e. to show that an object is not relevant to a query and therefore does not need to be accessed from storage or sent on the network from object storage to Spark. IBM Research implemented Data Skipping technology in the context of the IOStack H2020 project[45]. To make it efficient, we indexed the metadata, so that during query execution, can quickly filter out irrelevant objects from the list of objects to be accessed by the query. Note that this technique applies to all data formats, and avoids touching irrelevant objects altogether (see our presentation[46] at the Spark Summit, where Databricks announced Data Skipping support in their platform.

To get good Data Skipping one typically needs to pay attention to Data Layout. Data layout refers to all details regarding the storage of the data including object size, format, Hive style partitioning, and data partitioning, i.e. the assignment of data records to objects. We focus now on data partitioning. For any given query, we would like the records which satisfy the query to be grouped together in a small set of objects, so that the remaining objects can be skipped. In general, we need to partition the data so that it gives as much as possible data skipping for an incoming stream of queries (i.e. a workload), not just a single query. Note that the various queries may have conflicting requirements. Moreover, the workload changes over time, as does the data.

This multi-dimensional partitioning and indexing problem has been addressed in the past with space-filling curves. Techniques based on Space-filling curves[47] such as Z-order curves (or Morton curves[48]) map a multi-dimensional space into a single indexing dimension represented by an encoding string (the metadata). These techniques can handle varying data density by issuing a geohash code of varying length. Possible usage of these techniques is to convert a given query bounding box into a one-dimensional code range and to use it against the indexed data. However, the main drawback of these techniques in the fact that the chosen space filling curve and the dataset points completely determines the partitioning. In addition, the query history is not considered. Also, one cannot dynamically change the way partitioning

[43] http://blog.cloudera.com/blog/2016/04/benchmarking-apache-parquet-the-allstate-experience/
[44] https://www.ibm.com/blogs/bluemix/2018/06/big-data-layout/
[45] http://iostack.eu/
[46] https://databricks.com/session/using-pluggable-apache-spark-sql-filters-to-help-gridpocket-users-keep-up-with-the-jones-and-save-the-planet
[47] http://wwwmayr.informatik.tu-muenchen.de/konferenzen/Jass05/courses/2/Valgaerts/Valgaerts_paper.pdf
[48] "Z-order curves." https://en.wikipedia.org/wiki/Z-order_curve

is done, and Space-filling curves treat all dimensions in a symmetric way so no way to "prefer" one dimension over the other (e.g., to achieve metadata compactness and to fit the representation to the query distribution).

Recently data partitioning for the specific purpose of data skipping has become an active research area, and there has been significant work to define the problem and various approaches to solve it. A fundamental paper shows that the general problem is NP-hard, but devises a heuristic algorithm which performs well in practical use cases [49, 50]. A follow-on paper shows how improve the layout further for handle column based formats[49].

In 2017, as part of ***IOStack***, IBM Research independently developed the notion of k-d tree partitioning which uses query history to choose the partitioning columns and dataset medians as cutting points for partitioning[50]. In parallel, a paper was published by an MIT team which used a similar approach and similarly applied it to ***Apache Spark*** for data skipping [51]. The work in this paper went beyond previous work by providing an adaptive approach to repartition datasets on the fly according to a cost model. A recent companion paper covers how their technique can be applied to join processing [52].

This is a cutting-edge research area which is also promising in terms of its applicability to analytics on real world big datasets. We plan to undertake further research in this area as well as apply it to a commercial setting.

8.11 Real-time CEP

Streaming engines are used for real-time analysis of data collected from heterogeneous data sources with very high rates. Given the amount of data to be process in real-time (from thousands to millions of events per second), scalability is a fundamental feature for data streaming technologies. In the last decade, several data streaming systems have been released. ***StreamCloud*** [39] was the first system addressing the scalability problem allowing a parallel distributed processing of massive amount of collected data. ***Apache Storm***[51] and later ***Apache Flink***[52] followed the same path providing commercial solutions able to distribute and parallelize the data processing over several machines to increase the system throughput in terms of number of events processed per second. Apache Spark added streaming capability onto their product later[53]. ***Spark*** approach is not purely streamed, if fact it divides the data stream into a set of micro batches and repeat the processing of these batches in loop.

The streaming engine for the BigDataStack platform will be a scalable complex event processing (CEP) able to run in federated environments and to aggregate and correlate real-time events with structured and non-structured information stored in BigDataStack stores.

BigDataStack CEP we will be built upon the streaming engine owned by UPM that will be extended to run in federated environments and perform correlation on the edge closer to the data sources. Furthermore, techniques will be developed to reduce the access latency to the BigDataStack data stores increasing the efficiency of the correlation among real time data and data at rest.

[49] Skipping-oriented partitioning for columnar layouts VLDB 2016 https://dl.acm.org/citation.cfm?id=3025123
[50] IOStack. http://iostack.eu/deliverables/send/3-deliverables/31-d4-3-summary-and-demonstration-of-results
[51] http://storm.apache.org/
[52] Apache Flink. https://flink.apache.org/
[53] Apache Spark. https://spark.apache.org/streaming

The metrics exported by UPM's CEP technology are the following:

- CPU_LOAD: percentage of CPU used by an Instance Manager (CEP Worker). One value per IM
- Subquery # Tuple Received: Number of tuple received by a sub-query. One value per sub-query instance.
- Operator # Tuple Received: Number of tuple received by a streaming operator. One value per operator instance.
- Operator Latency: Time taken by a streaming operator to process a tuple. One value per operator instance.

8.12 Predictive and Process Analytics

In the predictive and process analytics component of the project there are two main goals to be achieved: Predictive Analytics and Process Analytics. Following is a brief introduction to the tools and technologies which will be used for the above.

8.12.1 Predictive Analytics

In Predictive Analytics, the main goal is the selection of a correct algorithm from a set of available algorithms and model hyper-parameter tuning. To this end Predictive Analytics will utilize the resources of the Spark libraries.

To begin, tools from ***Spark SQL***[54], ***Dataframes*** and ***Datasets Guides*** will be used such as sampling a feature of ***Hive***. When data volume is large, the need to find a subset of data to speed up data analysis becomes apparent. Here it comes to a technique used to select and analyse a subset of data to identify patterns and trends. In Hive, there are three ways of sampling data: random sampling, bucket table sampling, and block sampling.

Finally, tools from the ***Spark MLib*** library will be used such as Cross-Validation, Train-Validation Split, and Approximate Nearest Neighbour Search etc. better described in the Spark documentation under: Model selection and tuning and Extracting, transforming and selecting features.

8.12.2 Process Analytics (Process Mining)

In Process Analytics, the main goal is to enhance, optimize process models derived from the process modelling framework and to discover process models from raw event log files. For the enhancement and optimization phase to take place an event log for the process itself will be required from the global tracker consisting of the steps taken for the application in each component of the architecture.

For these tasks a tool such as ProM (which is short for Process Mining framework) [40] will be a good candidate. ProM is an Open Source framework for process mining algorithms.

The ProM framework integrates the functionality of several existing process mining tools and provides many additional process-mining plug-ins. The ProM framework supports multiple formats and multiple languages, e.g., Petri nets, EPCs, Social Networks, etc. The plug-ins can be used in several ways and combined to be applied in real-life situations.

[54] Spark SQL. https://spark.apache.org/docs/2.2.0/sql-programming-guide.html#supported-hive-features

8.13 Seamless Analytics Framework

Typically, logical data sets of IoT data will become too big to be kept in a single “storage entity” (e.g., a database for ***Cloudant***, or a bucket/container for ***Object Storage***). Therefore, accessing a single logical data set may require targeting a continually changing and possibly large set of storage entities, thus rendering difficult the access to the data. To hide this complexity, the seamless storage driver analyses queries and maps them to exactly the storage entities that contain relevant data.

Figure shows an example where data is ingested with the ***Watson IoT Platform*** within multiple ***Cloudant*** databases with a daily bucketing interval. Using the seamless storage driver, which implements the data sources API[55] (a pluggable mechanism for accessing structured data though Spark SQL), a user can write simple and intuitive queries against the logical dataset without needing to refer to the various underlying databases. Moreover, the driver analyses the queries and accesses only the relevant databases are accessed.

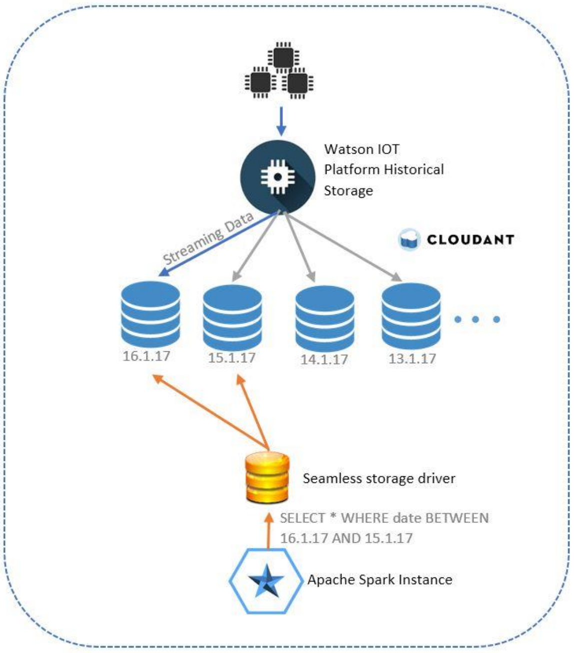


Figure 4. Data ingestion in Watson IoT Platform

8.14 Application Dimensioning Workbench

For the Load injector case, we will utilize ***Docker Swarm*** as an easy mean to scale stress tests towards the target application/data service. For each case/category that needs to be tested, the respective tool from a wide set of available ones can be used to understand and implement a load, to cater for the different range of application types. Thus, baseline ***Docker*** images will be used, prepared in advance for the specific baseline tool case, in which the load file will be injected based on the test scenario. As an example, targeting at the project data stores, YCSB will be one candidate towards the LXS case while ***Jmeter*** jmx files may be used to target application level components such as web servers or for example the case of HTTP requests against the IBM Object store. One of the main benefits of this approach is that we separate the Load Injector framework from the baseline load generation tool, thus being able to reuse the former in different cases of the latter. Another benefit is the easy scale up of the

55 https://databricks.com/blog/2015/01/09/spark-sql-data-sources-api-unified-data-access-for-the-spark-platform.html

Docker Swarm service approach to cater for extreme test conditions and avoidance of client-side bottleneck phenomena.

With relation to other baseline technology aspects, we anticipate inputs coming in and out of the dimensioning tool to be based on the Docker service manifest template (Docker Compose), to indicate service structure, component interconnection and naming. To extend the level of information in the file, we will also abide by the Docker specification (e.g. to include size of the resource per service element), but also inclusion of further metadata based on the custom metadata structure of the specification.

For the case of the dimensioning workbench front end, one candidate tool refers to ***Node-RED*** that can be used to easily create custom dashboards and integrate between different services, while performing the various transformation and adaptation tasks needed (e.g. to understand the input file, start the testing process and generate the output file). This tool is expected to be used mainly for coordinating the various actions and presenting results to the user and not for the core modelling work which would be performed in the backend. For the case of the Modelling Engine backend, dimensioning models are anticipated to be based on artificial neural networks. To this end, candidate technologies include tools such as ***GNU Octave***, an open source equivalent of ***Matlab***, while other candidates include the ***Tensorflow*** library. Selection will be performed during the project. Potential integration of ML within ***Node-RED*** (e.g. through the usage of node-red-contrib-machine-learning node) will also be investigated. However, given that the latter primarily deal with classification aspects (and not regression ones as expected to be used by Dimensioning), this approach would need to be based on a concept such as QoS (Quality of Service) class categories.

8.15 Process modelling framework

KIE (Knowledge Is Everything) is an umbrella project. KIE contains the following different but related projects offering a complete portfolio of solutions for business automation and management:

- ***Drools*** is a business rule management system with a forward-chaining and backward-chaining inference-based rules engine, allowing fast and reliable evaluation of business rules and complex event processing.
- ***jBPM*** is a flexible Business Process Management suite allowing you to model business goals by describing the steps that need to be executed to achieve those goals.
- ***OptaPlanner*** is a constraint solver that optimizes use cases such as employee rostering, vehicle routing, task assignment and cloud optimization.
- ***Drools Workbench*** is a full featured web application for the visual composition of custom business rules and processes.
- ***UberFire*** is a web-based workbench framework inspired by Eclipse Rich Client Platform.

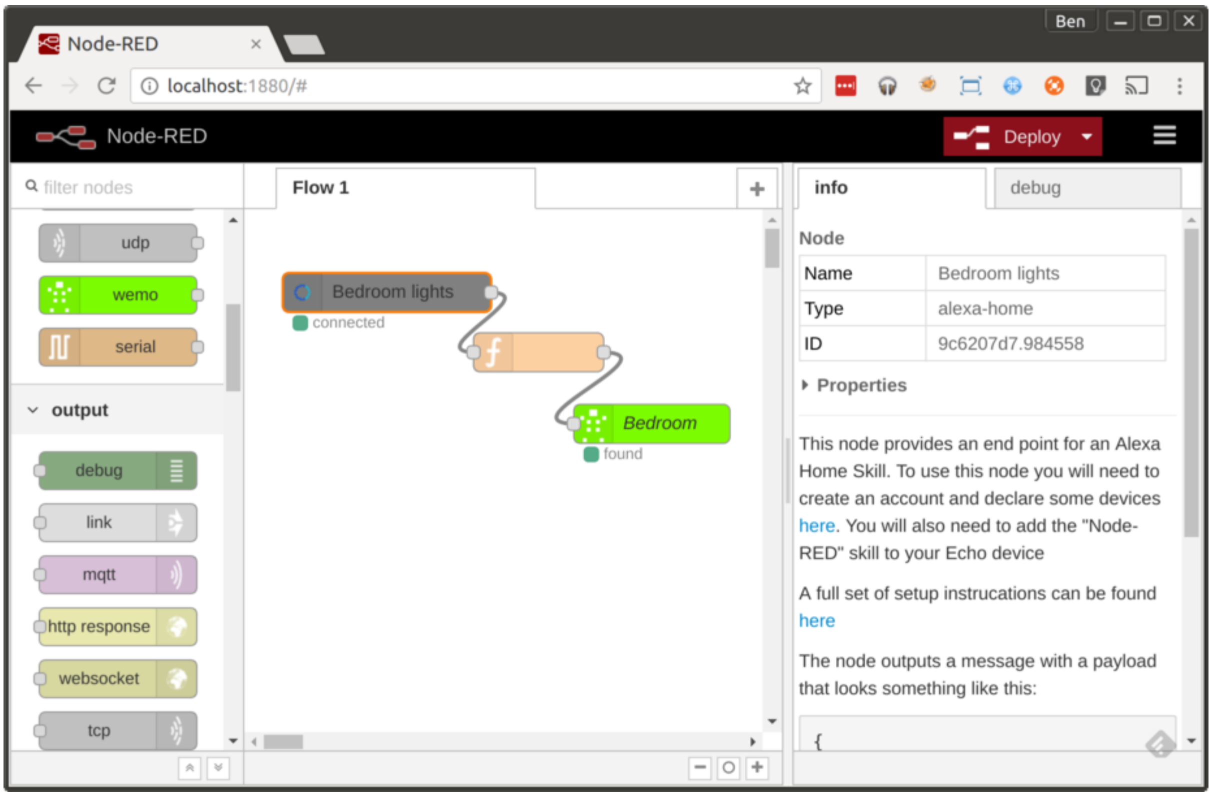
The core of jBPM is a light-weight, extensible workflow engine written in pure Java that allows executing business processes using the BPMN 2.0 specification. It can run in any Java environment, embedded in an application or as a service. jBPM is also not just an isolated process engine. Complex business logic can be modelled as a combination of business processes with business rules and complex event processing. It can be combined with the Drools project to support one unified environment that integrates these paradigms where one can model business logic as a combination of processes, rules and events. It supports adaptive and dynamic processes that require flexibility to model complex, real-life situations that cannot easily be described using a rigid process.

KIE is a full suite and regarding our needs it includes a rules engine, a modelling & execution environment, and several adapters (out of the box REST service task). Although jBPM is a BPMN 2.0 tool, it seems that declarative modelling is possible when combined with Drools. This enables to select a specific set of rules to execute at a specific point in the workflow using the native features of Drools.

At this early phase, it seems as an appropriate base framework to use towards building the processing modelling framework. A more thorough testing and analysis is ongoing while other platforms are under testing as well (i.e., node-red, dpil). Other technologies needed to support all the functions required are under research and dependent on the selections of other components of the architecture.

Node-RED[56] is a programming tool for wiring together hardware devices, APIs and online services in new and interesting ways. It is a browser-based tool that provides a drag-and-drop interface for selecting nodes from a palette and wiring them together.

Node-RED is open source software released under the Apache 2.0 license. Although initially targeted for Internet of Things environments, it is highly extensible and has been successfully been employed in a wide range of applications.



Node

Name	Bedroom lights
Type	alexa-home
ID	9c6207d7.984558

Properties

This node provides an end point for an Alexa Home Skill. To use this node you will need to create an account and declare some devices here. You will also need to add the "Node-RED" skill to your Echo device

A full set of setup instrucations can be found here

The node outputs a message with a payload that looks something like this:

```
{
```

Figure 5. Node-red programming example

56 https://nodered.org/

Node-RED will provide a visually appealing and effective User Interface fulfilling all key requirements of the Process Model Framework. This will allow us to only focus on BigDataStack functionality and avoid having to build an advanced tool from scratch.

8.15.1 Building Blocks Palette

The Process Model Framework will make all building blocks in a suitable palette. The building blocks will be classified in comprehensive categories and it will be possible to apply filters on them. Users will drag-and-drop a building block from the palette into the editor. Visual hints will make apparent what connections between the blocks are allowed. The Process Model Framework will constantly validate the edited model and provide clear error messages to the user.

It is to be expected that during the project the list of building blocks will undergo many changes. The Process Model Framework based on Node-RED will be designed in such a way, so that it can easily and quickly adapt to the changes.

8.15.2 Output Format

The Process Model Framework will export the designed model into an output format suitable for all subsequent layers in the architecture. The following notations could be considered:

- ***Drools***[57] (business rule management system language)
- ***BPMN*** (Business Process Model Notation)
- ***DPIL*** (Declarative Process Intermediate Language)

The exported model will be used by subsequent layers of the architecture. The notation must be so selected that it imposes as few requirements to these layers as possible. Also, the notation must be flexible enough to support requirement changes. For these reasons, it is suggested that a simple Drools file is used. Drools is a widely-used system and allows the creation of both simple and advanced rules notations. It supports adding metadata to rules, which will allow adding BigDataStack properties to model steps. It will finally be easy to read and process the exported model from all subsystems of the architecture.

8.16 Data Toolkit

The main objective of the Data Toolkit is to design and support data analysis workflows. Such analysis workflows are designed based on the business requirements that will drive the process modelling. The set of high level business process analysis steps already identified, along with the indications for the data analysis algorithms that must be used per step, must be detailed in a scientific basis leading to the production of an end-to-end analysis workflow that can be realised over an application's orchestrator. Such an end-to-end analysis workflow is defined as the analysis playbook within BigDataStack.

Playbooks consist of a set of data mining and analysis processes, interconnected among each other in terms of input/output data streams/objects. It is represented in the form of an analysis application graph (following concepts from cloud applications deployment and orchestration) that includes the set of individual processes as nodes of the graph along with their binding/dependencies in the form of virtual links.

Playbooks should let data scientists design and develop complex analytic processes by

[57] https://www.drools.org/

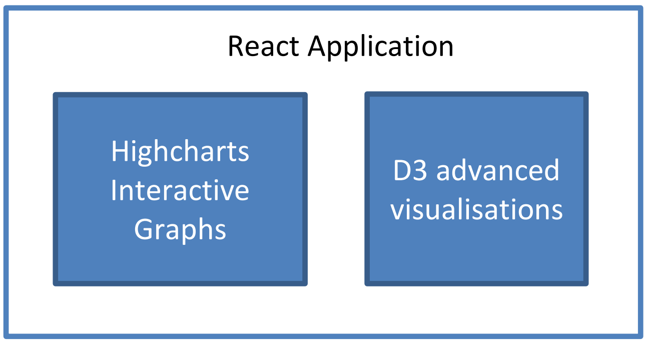
combining set of available or under development analytic functions/primitives. This should include characteristics related to input data parameters (type of data sources without any binding), output data parameters, analysis configuration parameters, execution substrate requirements and the software packages. Following, interconnection of such nodes leads to the production of the playbook (overall graph).

A strict requirement regards the capacity to support various technologies/programming languages for development of analytics processes, given the existence and dominance of set of them (e.g., *R*, *Python*, *Java, Scala*). The developed analytics processes have also to be deployable over big data computing frameworks (e.g. ***Apache Spark***, ***Apache Flink***).

The data toolkit can be implemented based on existing open source solutions along with their appropriate extension and customisation. Such solutions include -among others- tools like ***Conductor***[58] that supports orchestration of micro-services-based process flows, ***OpenCPU***[59] that is a system for embedded scientific computing and reproducible research. The exact tool to be adopted must be decided according the specification of the overall architectural solution of BigDataStack.

8.17 Adaptable Visualizations

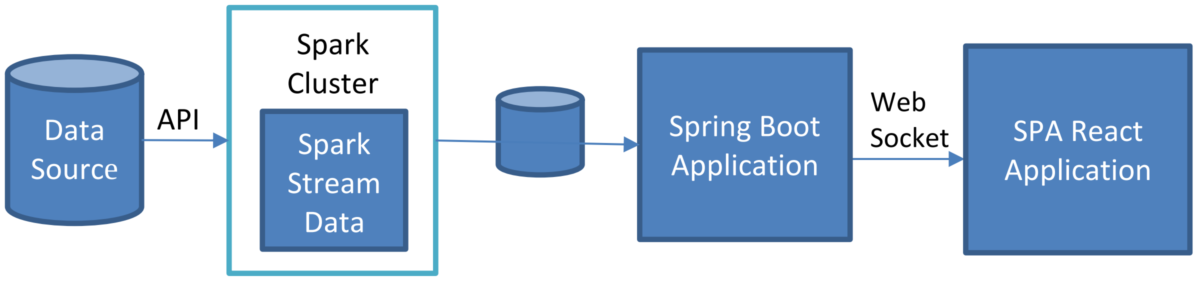
The Adaptive Visualisations will be implemented as a web Single Page Application (SPA). JavaScript libraries will be used for a fast loading, interactive and adaptable user interface (e.g. React or AngularJS) and for data visualization (e.g., Highcharts or D3). The following diagram depicts the main technologies to implement the application.



The application will be implemented with *React*[60] *javascript* library. React is a modern javascript library that encourages good architectural design and follows a component-based approach. It makes it easy to design, debug and test fast interactive applications. For the graphs, *Highcharts*[61] library will be used. Highcharts is a widely used, royalty-free commercial javascript library for creating impressive interactive web diagrams. It supports numerous diagram types that we expect to cover all BigDataStack needs. Should a more advanced or custom diagram is needed, the D3.js[62] library will be employed. For both Highcharts and D3.js ready-made components for React are available.

58 https://netflix.github.io/conductor/
59 https://www.opencpu.org/
60 https://reactjs.org/
61 https://www.highcharts.com/
62 https://d3js.org/

For certain use-cases visualization of real-time data is required. The following diagram depicts the components that will implement the real-time visualizations.



A Spark Cluster will be set-up that will read data from a data source through an appropriate API. The Spark Cluster will provide the Spark Streams that will process the data and produce the appropriate aggregations to be pushed in an intermediate database. A Spring Boot application will consume the aggregated data and will provide a Web Socket interface to the React Application. This will allow the real-time update of the visualization without the user having to refresh the page.

9 Conclusions

In BigDataStack project's Task 2.1, requirements engineering was carried out considering the needs and the concerns from the current BigDataStack stakeholders. The analysis specifies not only ***use case requirements*** (called "stakeholder requirements" by ISO/IEC/IEEE 29148:2011[63]) but also on ***technical requirements*** (called "system requirements" and "software requirements" by the same norm).

To contextualize the analysis of requirements, BigDataStack platform stakeholders, business model, expected business outcomes and business goals were identified. Furthermore, the report also describes the state-of-the-art (baseline) technologies that are expected to be relevant for the BigDataStack platform.

This deliverable is the first version of the state-of-the-art and requirements analysis to drive the architecture and research effort in BigDataStack. A top-down approach was followed with respect to the user requirements, which will be collected through the BigDataStack's **use case providers**. This was complemented with a bottom-up one aiming at technical requirements at both system and software technology level. Due to the early stage of the architecture work, the overall architecture is not yet matured/finalized, and not all software technology requirements could be identified yet. System requirements sit in the middle of use case and software requirements and are related to the services; future iterations of requirements analysis will focus on that requirement level. In other words, in this report, BigDataStack's **technology providers** have been analysing the role their respective technologies may play in the BigDataStack vision. However, some important system (and subsystem) requirements have been identified and specified.

System requirements have been organized following the main capabilities that were envisioned for the BigDataStack platform to achieve the business goals or expectations from the different stakeholders: *Data-driven Infrastructure Management, Data as a Service, Data Visualization, Data Toolkit, Process Modelling, and Dimensioning Workbench.* These organize in a full "stack" that aims at not only facilitating the needs of data operations and applications (all of which tend to be data-intensive) but also facilitating these needs in an optimum way. As already mentioned, at this stage of the project we don't have requirements for some of the platform capabilities yet. However, new requirements and constraints over those capabilities will be the focus of future iterations of the requirements engineering at M11 and M22 (see Introduction).

Software technology requirements have been organized in terms of the technologies or technological components that have been already linked to BigDataStack. For some software (technology) components, no requirements for the BigDataStack platform were identified. However, new requirements coming from those components might appear in future iterations of the requirements engineering at M11 and M22 (see Introduction).

[63] International Organization for Standardization, "ISO/IEC/IEEE 29148:2011 – Systems and software engineering — Life cycle processes — Requirements engineering," ISO/IEC/IEEE, Nov. 2011.

10 Bibliography

[1] G. Beskales, I. F. Ilyas, and L. Golab, “Sampling the repairs of functional dependency violations under hard constraints,” *Proc. VLDB Endow.*, vol. 3, no. 1–2, pp. 197–207, 2010.

[2] W. Fan, J. Li, S. Ma, N. Tang, and W. Yu, “Towards certain fixes with editing rules and master data,” *Proc. VLDB Endow.*, vol. 3, no. 1–2, pp. 173–184, 2010.

[3] J. Wang and N. Tang, “Towards dependable data repairing with fixing rules,” in *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, 2014, pp. 457–468.

[4] X. Chu, I. F. Ilyas, and P. Papotti, “Holistic data cleaning: Putting violations into context,” in *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*, 2013, pp. 458–469.

[5] M. Heinsman, “Trifacta,” *Trifacta*. [Online]. Available: https://www.trifacta.com/. [Accessed: 23-May-2018].

[6] M. Dallachiesa *et al.*, “NADEEF: a commodity data cleaning system,” in *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, 2013, pp. 541–552.

[7] J. Wang, S. Krishnan, M. J. Franklin, K. Goldberg, T. Kraska, and T. Milo, “A sample-and-clean framework for fast and accurate query processing on dirty data,” in *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, 2014, pp. 469–480.

[8] Z. Khayyat *et al.*, “Bigdansing: A system for big data cleansing,” in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, 2015, pp. 1215–1230.

[9] Y. Altowim, D. V. Kalashnikov, and S. Mehrotra, “Progressive approach to relational entity resolution,” *Proc. VLDB Endow.*, vol. 7, no. 11, pp. 999–1010, 2014.

[10] Z. Li, S. Shang, Q. Xie, and X. Zhang, “Cost reduction for web-based data imputation,” in *International Conference on Database Systems for Advanced Applications*, 2014, pp. 438–452.

[11] D. Haas, J. Wang, E. Wu, and M. J. Franklin, “Clamshell: Speeding up crowds for low-latency data labeling,” *Proc. VLDB Endow.*, vol. 9, no. 4, pp. 372–383, 2015.

[12] C. Gokhale *et al.*, “Corleone: hands-off crowdsourcing for entity matching,” in *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, 2014, pp. 601–612.

[13] B. Mozafari, P. Sarkar, M. Franklin, M. Jordan, and S. Madden, “Scaling up crowd-sourcing to very large datasets: a case for active learning,” *Proc. VLDB Endow.*, vol. 8, no. 2, pp. 125–136, 2014.

[14] X. Chu, I. F. Ilyas, S. Krishnan, and J. Wang, “Data Cleaning: Overview and Emerging Challenges,” 2016, pp. 2201–2206.

[15] P. Bohannon, W. Fan, M. Flaster, and R. Rastogi, “A cost-based model and effective heuristic for repairing constraints by value modification,” in *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, 2005, pp. 143–154.

[16] J. Wang, T. Kraska, M. J. Franklin, and J. Feng, “Crowder: Crowdsourcing entity resolution,” *Proc. VLDB Endow.*, vol. 5, no. 11, pp. 1483–1494, 2012.

[17] A. Chalamalla, I. F. Ilyas, M. Ouzzani, and P. Papotti, “Descriptive and prescriptive data cleaning,” in *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, 2014, pp. 445–456.

[18] L. Golab, H. Karloff, F. Korn, D. Srivastava, and B. Yu, “On generating near-optimal tableaux for conditional functional dependencies,” *Proc. VLDB Endow.*, vol. 1, no. 1, pp. 376–390, 2008.

[19] G. Beskales, I. F. Ilyas, L. Golab, and A. Galiullin, “On the relative trust between inconsistent data and inaccurate constraints,” in *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*, 2013, pp. 541–552.

[20] M. Yakout, A. K. Elmagarmid, J. Neville, M. Ouzzani, and I. F. Ilyas, “Guided data repair,” *Proc. VLDB Endow.*, vol. 4, no. 5, pp. 279–289, 2011.

[21] S. Krishnan, J. Wang, E. Wu, M. J. Franklin, and K. Goldberg, “Activeclean: Interactive data cleaning while learning convex loss models,” *ArXiv Prepr. ArXiv160103797*, 2016.

[22] Carbonell, J. (1990). Machine learning: paradigms and methods. Elsevier North-Holland, Inc.

[23] Yu, H., Han, J. & Chang, K. C.-C., “PEBL: Positive example based learning for Web page classification using SVM.” In ‘Proceedings of ACM SIGKDD 2002 International Conference on Knowledge Discovery and Data Mining’.

[24] Agichtein, E., Brill, E. & Dumais, S. T.,“Improving Web search ranking by incorporating user behavior information.” In ‘Proceedings of the 29th International ACM SIGIR Conference on Research and Development in Information Retrieval’.

[25] Liu, T.-Y., “Learning to rank for information retrieval.” Foundations Trends Information Retrieval. 3, 225–331.

[26] Page, L., Brin, S., Motwani, R. & Winograd, T.,“The PageRank Citation Ranking: Bringing Order to the Web.” Technical report. Stanford InfoLab. 1999

[27] Macdonald, C., Santos, R. & Ounis, “The whens and hows of learning to rank.” Information Retrieval. 2012

[28] J. N. Gray, "Notes on data base operating systems," Lecture Notes in Computer Science, vol. 60, pp. 393-481, 1978.

[29] H. Sturgis and B. Lampson, “Crash recovery in a distributed data storage system,” Computer Science Laboratory, Xerox, Palo Alto, 1976.

[30] D. Peng and F. Dabek, "Large-scale incremental processing using distributed transactions and notifications," in Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation (OSDI’10), 2010.

[31] J. C. Corbett, J. Dean, M. Epstein, A. Fikes, C. Frost, J. J. Furman, S. Ghemawat, A. Gubarev, C. Heiser, P. Hochschild, W. Hsieh, S. Kanthak, E. Kogan, H. Li, A. Lloyd and S. Melnik, "Spanner: Google’s globally-distributed database," in Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation (OSDI ’12), 2012.

[32] D. G. Ferro, F. Junqueira, I. Kelly, B. Reed and M. Yabandeh, "Omid: Lock-free transactional support for distributed data stores," in IEEE 30th International Conference on Data Engineering (ICDE), Chicago, 2014.

[33] Apache, "Apache Tephra," [Online]. Available: http://tephra.incubator.apache.org. [Accessed May 2018].

[34] Amr Osman, Mohamed El-Refaey, Ayman Elnaggar, Towards Real-Time Analytics in the Cloud, In Proceedings of IEEE SERVICES, 2013

[35] Mike Barlow, Real-Time Big Data Analytics: Emerging Architecture, O'Reilly Media, Inc.,2013

[36] T. Özsu, P. Valduriez. Principles of Distributed Database Systems. Springer, 2011

[37] Alfons Kemper and Thomas Neumann. HyPer: A hybrid OLTP&OLAP main memory database system based on virtual memory snapshots. In Proceedings of ICDE, 2011

[38] Franz Färber, Sang Kyun Cha, Jürgen Primsch, Christof Bornhövd, Stefan Sigg, and Wolfgang Lehner. SAP HANA database: data management for modern business applications. In Proceedings of SIGMOD, 2012.

[39] V. Gulisano, R. Jiménez-Peris, M. Patiño-Martínez, C. Soriente, P. Valduriez (2012) StreamCloud: An Elastic and Scalable Data Streaming System. IEEE Trans. Parallel Distrib. Syst. 23(12): 2351-2365.

[40] B. F. van Dongen, A. K. A. de Medeiros, H. M. W. Verbeek, A. J. M. M. Weijters, and W. M. P. van der Aalst, "The ProM Framework: A New Era in Process Mining Tool Support," in Applications and Theory of Petri Nets 2005, vol. 3536, G. Ciardo and P. Darondeau, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 444–454.

[41] International Organization for Standardization, "ISO/IEC/IEEE 29148:2011 – Systems and software engineering — Life cycle processes — Requirements engineering," ISO/IEC/IEEE, Nov. 2011.

[42] Open Grid Forum, "Web Services Agreement Specification (WS-Agreement)," Oct. 10, 2011. http://ogf.org/documents/GFD.192.pdf

[43] Open Grid Forum, "WS-Agreement Negotiation Version 1.0," Jan. 31, 2011. https://www.ogf.org/Public_Comment_Docs/Documents/2011-03/WS-Agreement-Negotiation+v1.0.pdf

[44] P. Pietzuch, J. Ledlie, J. Shneidman, M. Roussopoulos, M. Welsh, and M. Seltzer, "Network-Aware Operator Placement for Stream-Processing Systems", 22nd International Conference on Data Engineering (ICDE '06), pp. 49–53, IEEE Computer Society, 2006.

[45] V. Cardellini, V. Grassi, F. Lo Presti, and M. Nardelli, "Distributed QoS-aware Scheduling in Storm", 9th ACM International Conference on Distributed Event-Based Systems, pp. 344-347, ACM, 2015.

[46] Y. Xing, S. Zdonik, and J.-H. Hwang, "Dynamic Load Distribution in the Borealis Stream Processor", 21st International Conference on Data Engineering (ICDE '05), pp. 791–802, IEEE Computer Society, 2005.

[47] M. Hirzel, R. Soule, S. Schneider, B. Gedik, and R. Grimm, "A Catalog of Stream Processing Optimizations", ACM Computing Surveys, vol. 46, Mar. 2014, pp 1–34.

[48] MongoDB MongoDB and MySQL Compare. [Accessed: 27/05/2018] https://www.mongodb.com/compare/mongodb-mysql.

[49] L. Sun, M. J. Franklin, S. Krishnan, and R. S. Xin, "Fine-grained partitioning for aggressive data skipping," SIGMOD, 2014.

[50] L. Sun, S. Krishnan, R. S. Xin, and M. J. Franklin, "A partitioning framework for aggressive data skipping," VLDB, 2014.

[51] A. Shanbhag, A. Jindal, S. Madden, J. Quiane, and A. J. Elmore, "A robust partitioning scheme for ad-hoc query workloads," SoCC, 2017.

[52] Y. Lu, A. Shanbhag, A. Jindal, and S. Madden, "Adaptdb: Adaptive partitioning for distributed joins," VLDB, 2017.