

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/

D2.2 – Requirements & State of the Art Analysis – II

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

Contributing Author(s) (Org)	<p>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 Raouzaïou (ATC), Pavlos Kranas (LXS), Sophia Karagiorgou, Panagiotis Gouvas, Anastasios Zafeiropoulos (UBI), Dimitris Pouloupoulos, Timoleon Labrinos, Stavroula Meimetea, Dimosthenis Kyriazis (UPRC), Valerio Vianello (UPM), Richard McCreadie (GLA), Gal Hammer, Miki Kenneth, Luis Tomas (RHT), Nikos Drosos (SILO), Maurizio Megliola (GFT)</p>
Reviewer(s) (Org)	<p>Yosef Moatti (IBM), Amaryllis Raouzaïou (ATC), Dimosthenis Kyriazis (UPRC)</p>
Due Date	30.11.2018
Date	18.12.2018
Version	1.0

Dissemination Level

- 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	10.12.2018	Orlando Avila-García (ATOS)	Incorporation of a new version of use case requirements and scenarios by GFT.
0.2	11.12.2018	Orlando Avila-García (ATOS)	Incorporation of requirements specified during the elaboration of D3.1, D4.1 and D5.1 by IBM, NEC, ATOS, DAN, ATC, LXS, UBI, UPRC, UPM, GLA, RHT, SILO and GFT.
0.3	12.12.2018	Orlando Avila-García (ATOS)	Amendments pointed out by IBM and ATC internal reviews.
1.0	18.12.2018	Orlando Avila-García (ATOS)	Final amendments pointed out by UPRC. Finalization of the deliverable.

Disclaimer

This document contains information that is proprietary to the BigDataStack Consortium. Neither this document nor the information contained herein shall be used, duplicated or communicated by any means to a third party, in whole or parts, except with the prior consent of BigDataStack Consortium.

Table of Contents

Table of Contents	4
List of tables	5
List of figures	7
1 Executive Summary	8
2 Introduction	9
2.1 Method	10
2.2 Organization	13
3 Business Stakeholders and Goals.....	14
3.1 Stakeholder Categories	14
3.2 Business Model	15
3.3 Business Outcomes	16
3.4 Business Goals	17
4 Use Case Requirements and Scenarios	20
4.1 Real-time Ship Management	20
4.2 Connected Consumer	27
4.3 Smart Insurance	36
5 Platform Roles.....	41
6 Infrastructure-Data Management Requirements	43
7 Data as a Service Requirements.....	56
8 Dimensioning, Modelling & Interaction Services Requirements	67
9 Baseline Technologies	78
9.1 Computing Resources Management	78
9.2 Storage Resources Management.....	79
9.3 Data-driven Network Management.....	80
9.4 Dynamic Orchestrator	80
9.5 Triple Monitoring.....	81
9.6 Applications & Data Services Deployment	83
9.7 Distributed Storage & Analytics	87
9.8 Live Migration	88
9.9 Data Cleaning.....	88
9.10 Big Data Layout.....	90
9.11 Real-time CEP.....	92
9.12 Predictive and Process Analytics	93
9.13 Seamless Analytics Framework	93
9.14 Application Dimensioning Workbench	94
9.15 Process modelling framework	95
9.16 Data Toolkit	97
9.17 Adaptable Visualizations	98
10 Bibliography.....	100

List of tables

TABLE 1 – REQUIREMENTS ENGINEERING PROCESSES.....	10
TABLE 2 – LEVELS OF REQUIREMENT SPECIFICATION.....	12
TABLE 3 – STAKEHOLDERS.....	14
TABLE 4 – STAKEHOLDER CONCERNS.....	15
TABLE 5 – PRELIMINARY BUSINESS MODEL.....	16
TABLE 6 – STAKEHOLDER REQUIREMENTS.....	17
TABLE 7 – PRIVACY AND SECURITY (BUSINESS GOAL).....	17
TABLE 8 – ATTRACTIVE REVENUE AND BUSINESS MODEL (BUSINESS GOAL).....	18
TABLE 9 – HIGH PERFORMANCE, SCALABILITY AND SHARING (BUSINESS GOAL).....	18
TABLE 10 – PRODUCT INTEGRATION (BUSINESS GOAL).....	18
TABLE 11 – DIFFERENT ANALYTIC CAPABILITIES (BUSINESS GOAL).....	19
TABLE 12 – EASE OF USE (BUSINESS GOAL).....	19
TABLE 13 – MONITORING AND PREDICTIVE MAINTENANCE SCENARIO DESCRIPTION (SCENARIO).....	22
TABLE 14 – ORDER SUGGESTION AND DYNAMIC ROUTING (SCENARIO).....	23
TABLE 15 – MAIN ENGINE MONITORING (STAKEHOLDER REQUIREMENT).....	24
TABLE 16 – MALFUNCTION ALERT (STAKEHOLDER REQUIREMENT).....	25
TABLE 17 – ALERT INSPECTION (STAKEHOLDER REQUIREMENT).....	25
TABLE 18 – SPARE PART REQUISITION (STAKEHOLDER REQUIREMENT).....	26
TABLE 19 – REQUISITION PROCESS (STAKEHOLDER REQUIREMENT).....	26
TABLE 20 – DYNAMIC ROUTING (STAKEHOLDER REQUIREMENT).....	27
TABLE 21 – RETAIL RECOMMENDER (SCENARIO).....	30
TABLE 22 – RETAIL DEMONSTRATOR (SCENARIO).....	32
TABLE 23 – PREDICT PRODUCTS REQUIRED BY A RECURRENT USER (STAKEHOLDER REQUIREMENT).....	32
TABLE 24 – PREDICT PRODUCTS TO A NEW USER (STAKEHOLDER REQUIREMENT).....	33
TABLE 25 – RECOMMEND PERSONALIZED DISCOUNTS (STAKEHOLDER REQUIREMENT).....	33
TABLE 26 – DATA REQUIREMENT (STAKEHOLDER REQUIREMENT).....	34
TABLE 27 – CLIENTS REQUIREMENTS (STAKEHOLDER REQUIREMENT).....	34
TABLE 28 – PRODUCTS REQUIREMENTS (STAKEHOLDER REQUIREMENT).....	34
TABLE 29 – MULTI-DEVICE (STAKEHOLDER REQUIREMENT).....	35
TABLE 30 – EASY-TO-USE (STAKEHOLDER REQUIREMENT).....	35
TABLE 31 – MULTI-USER (STAKEHOLDER REQUIREMENT).....	35
TABLE 32 – DATA SECURITY (STAKEHOLDER REQUIREMENT).....	36
TABLE 33 – SERVICES SECURITY (STAKEHOLDER REQUIREMENT).....	36
TABLE 34 - CUSTOMERS SEGMENTATION SCENARIO.....	38
TABLE 35 - CUSTOMER LIFETIME VALUE PREDICTION SCENARIO.....	39
TABLE 36 – PROVIDE PERSONALIZED POLICIES REQUIRED BY CUSTOMER (STAKEHOLDER REQUIREMENT).....	39
TABLE 37 – PROVIDE PERSONALIZED GUARANTEES REQUIRED BY CUSTOMER (STAKEHOLDER REQUIREMENT).....	40
TABLE 38 – CUSTOMERS (STAKEHOLDER REQUIREMENT).....	40
TABLE 39 – POLICIES (STAKEHOLDER REQUIREMENT).....	40
TABLE 40 – GUARANTEES (STAKEHOLDER REQUIREMENT).....	41
TABLE 41 – EASY-TO-USE (STAKEHOLDER REQUIREMENT).....	41
TABLE 42 – DATA SECURITY (STAKEHOLDER REQUIREMENT).....	41
TABLE 43 – BIGDATASTACK PLATFORM ROLES.....	42
TABLE 44 - SUPPORT OPENSIFT INSTALLATION ON OPENSTACK VMs (SYSTEM REQUIREMENT).....	43
TABLE 45 - AVOID DOUBLE ENCAPSULATION OF NETWORK PACKAGES (SYSTEM REQUIREMENT).....	44
TABLE 46 - SPARK OPERATOR (SYSTEM REQUIREMENT).....	44
TABLE 47 - ACCEPT REQUESTS TO ALLOCATE ADDITIONAL RESOURCES TO THE STORAGE LAYER (SYSTEM REQUIREMENT).....	44
TABLE 48 - FORCE THE STORAGE LAYER TO RELEASE SOME OF ITS AVAILABLE RESOURCES (SYSTEM REQUIREMENT).....	45
TABLE 49 - CORRECTION OF REQUIREMENTS AND SLOS VIOLATIONS (STAKEHOLDER REQUIREMENT).....	45
TABLE 50 - DECISION EFFICIENCY (STAKEHOLDER REQUIREMENT).....	45
TABLE 51 - RESOURCES LIMITS (STAKEHOLDER REQUIREMENT).....	45
TABLE 52 - ORCHESTRATION FOR IMPROVEMENTS (STAKEHOLDER REQUIREMENT).....	46

TABLE 53 - INGEST CANDIDATE DEPLOYMENT PLAYBOOKS AND BENCHMARKING INFORMATION (SYSTEM REQUIREMENT).....	46
TABLE 54 - DEPLOYMENT SUITABILITY FEATURE EXTRACTION (SYSTEM REQUIREMENT).	46
TABLE 55 - CDP PLAYBOOK SCORING (HEURISTIC) (SYSTEM REQUIREMENT).	47
TABLE 56 - CDP PLAYBOOK SCORING (SUPERVISED) (SYSTEM REQUIREMENT).	47
TABLE 57 - CDP PLAYBOOK SELECTION (SYSTEM REQUIREMENT).....	47
TABLE 58 - SUPERVISED MODEL TRAINING (SYSTEM REQUIREMENT).	48
TABLE 59 - CDP PLAYBOOK RE-SCORING (SYSTEM REQUIREMENT).	48
TABLE 60 - PERFORMANCE MEASURABILITY (STAKEHOLDER REQUIREMENT).	48
TABLE 61 - STANDARDS-BASED PLAYBOOK (STAKEHOLDER REQUIREMENT).	49
TABLE 62 - STANDARD DEPLOYMENT INFORMATION (SYSTEM REQUIREMENT).	49
TABLE 63 - APPLICATION SCORING SYSTEM (SYSTEM REQUIREMENT).	49
TABLE 64 - COMPATIBILITY WITH KUBERNETES (SYSTEM REQUIREMENT).....	50
TABLE 65 - SYNCHRONOUS COMMUNICATION (SYSTEM REQUIREMENT).	50
TABLE 66 - REGULAR RECORDING OF DEPLOYMENT QoS INFORMATION (STAKEHOLDER REQUIREMENT).....	50
TABLE 67 - QoS VIOLATION NOTIFICATION (STAKEHOLDER REQUIREMENT).....	51
TABLE 68 - QoS VIOLATION MONITORING (STAKEHOLDER REQUIREMENT).....	51
TABLE 69 - METRICS PUSHER (SYSTEM REQUIREMENT).	51
TABLE 70 - MONITORING METRICS API REST (SYSTEM REQUIREMENT).	51
TABLE 71 - MONITORING METRICS GETTER (SOFTWARE REQUIREMENT).....	52
TABLE 72 - SPARK COMPATIBILITY (SOFTWARE REQUIREMENT).....	52
TABLE 73 - LEANXCALE COMPATIBILITY (SOFTWARE REQUIREMENT).....	52
TABLE 74 - OKD COMPATIBILITY (SOFTWARE REQUIREMENT).	53
TABLE 75 - CEP COMPATIBILITY (SOFTWARE REQUIREMENT).....	53
TABLE 76 - MINIO COMPATIBILITY (SOFTWARE REQUIREMENT).....	53
TABLE 77 - OPENSTACK NETWORKING SERVICES COMPATIBILITY (SOFTWARE REQUIREMENT).....	53
TABLE 78 - MONITORING DATABASE (SOFTWARE REQUIREMENT).	54
TABLE 79 - MONITORING PUSHGATEWAY (SOFTWARE REQUIREMENT).....	54
TABLE 80 - METRICS VISUALIZATION (SOFTWARE REQUIREMENT).	54
TABLE 81 - METRICS VISUALIZATION (SOFTWARE REQUIREMENT).	54
TABLE 82 - NETWORK POLICIES BASED ON TYPE OF DATA (SOFTWARE REQUIREMENT).	55
TABLE 83 - NETWORK POLICIES BASED ON APPLICATION (SOFTWARE REQUIREMENT).	55
TABLE 84 - REQUIREMENT REQ-BDL-01 FOR BIG DATA LAYOUT.....	56
TABLE 85 - REQUIREMENT REQ-BDL-02 FOR BIG DATA LAYOUT.....	56
TABLE 86 - REQUIREMENT REQ-BDL-03 FOR BIG DATA LAYOUT.....	57
TABLE 87 - REQUIREMENT REQ-BDL-04 FOR BIG DATA LAYOUT.....	57
TABLE 88 - REQUIREMENT REQ-ADS-01 FOR ADAPTABLE DISTRIBUTED STORAGE.....	57
TABLE 89 - REQUIREMENT REQ-ADS-02 FOR ADAPTABLE DISTRIBUTED STORAGE.....	58
TABLE 90 - REQUIREMENT REQ-ADS-03 FOR ADAPTABLE DISTRIBUTED STORAGE.....	58
TABLE 91 - REQUIREMENT REQ-ADS-04 FOR ADAPTABLE DISTRIBUTED STORAGE.....	58
TABLE 92 - REQUIREMENT REQ-ADS-05 FOR ADAPTABLE DISTRIBUTED STORAGE.....	59
TABLE 93 - REQUIREMENT REQ-ADS-06 FOR ADAPTABLE DISTRIBUTED STORAGE.....	59
TABLE 94 - REQUIREMENT REQ-ADS-07 FOR ADAPTABLE DISTRIBUTED STORAGE.....	60
TABLE 95 - REQUIREMENT REQ-ADS-08 FOR ADAPTABLE DISTRIBUTED STORAGE.....	60
TABLE 96 - REQUIREMENT REQ-SDAF-01 FOR SEAMLESS DATA ANALYTICS.....	61
TABLE 97 - REQUIREMENT REQ-SDAF-02 FOR SEAMLESS DATA ANALYTICS.....	61
TABLE 98 - REQUIREMENT REQ-SDAF-03 FOR SEAMLESS DATA ANALYTICS.....	61
TABLE 99 - REQUIREMENT REQ-SDAF-04 FOR SEAMLESS DATA ANALYTICS.....	62
TABLE 100 - REQUIREMENT REQ-SDAF-05 FOR SEAMLESS DATA ANALYTICS.....	62
TABLE 101 - REQUIREMENT REQ-DQAI-01 FOR DATA QUALITY ASSESSMENT & IMPROVEMENT.....	63
TABLE 102 - REQUIREMENT REQ-DQAI-02 FOR DATA QUALITY ASSESSMENT & IMPROVEMENT.....	63
TABLE 103 - REQUIREMENT REQ-DQAI-03 FOR DATA QUALITY ASSESSMENT & IMPROVEMENT.....	63
TABLE 104 - REQUIREMENT REQ-DQAI-04 FOR DATA QUALITY ASSESSMENT & IMPROVEMENT.....	63
TABLE 105 - REQUIREMENT REQ-DQAI-05 FOR DATA QUALITY ASSESSMENT & IMPROVEMENT.....	64
TABLE 106 - REQUIREMENT REQ-DQAI-06 FOR DATA QUALITY ASSESSMENT & IMPROVEMENT.....	64

TABLE 107 - REQUIREMENT REQ-RD-01 FOR PREDICTIVE & PROCESS ANALYTICS	64
TABLE 108 - REQUIREMENT REQ-RD-02 FOR PREDICTIVE & PROCESS ANALYTICS	64
TABLE 109 - REQUIREMENT REQ-RD-03 FOR PREDICTIVE & PROCESS ANALYTICS	65
TABLE 110 - REQUIREMENT REQ-RD-04 FOR PREDICTIVE & PROCESS ANALYTICS	65
TABLE 111 - REQUIREMENT REQ-CEP-01 FOR CEP	65
TABLE 112 - REQUIREMENT REQ-CEP-02 FOR CEP	66
TABLE 113 - REQUIREMENT REQ-CEP-03 FOR CEP	66
TABLE 114 - REQUIREMENT REQ-CEP-04 FOR CEP	66
TABLE 115 – SYSTEM REQUIREMENT (1) FOR PROCESS MODELLING FRAMEWORK.....	67
TABLE 116 – SYSTEM REQUIREMENT (2) FOR PROCESS MODELLING FRAMEWORK.....	67
TABLE 117 – SYSTEM REQUIREMENT (3) FOR PROCESS MODELLING FRAMEWORK.....	68
TABLE 118 – SYSTEM REQUIREMENT (4) FOR PROCESS MODELLING FRAMEWORK.....	68
TABLE 119 – SYSTEM REQUIREMENT (5) FOR PROCESS MODELLING FRAMEWORK.....	68
TABLE 120 – SYSTEM REQUIREMENT (6) FOR PROCESS MODELLING FRAMEWORK.....	69
TABLE 121 – SYSTEM REQUIREMENT (7) FOR PROCESS MODELLING FRAMEWORK.....	69
TABLE 122 – SYSTEM REQUIREMENT (8) FOR PROCESS MODELLING FRAMEWORK.....	69
TABLE 123 – SYSTEM REQUIREMENT (1) FOR PROCESS MAPPING.....	69
TABLE 124 – SYSTEM REQUIREMENT (2) FOR PROCESS MAPPING.....	70
TABLE 125 – SYSTEM REQUIREMENT (3) FOR PROCESS MAPPING.....	70
TABLE 126 – SYSTEM REQUIREMENT (4) FOR PROCESS MAPPING.....	70
TABLE 127 – SYSTEM REQUIREMENT (1) FOR DATA TOOLKIT.....	70
TABLE 128 – SYSTEM REQUIREMENT (2) FOR DATA TOOLKIT.....	71
TABLE 129 – SYSTEM REQUIREMENT (3) FOR DATA TOOLKIT.....	71
TABLE 130 – SYSTEM REQUIREMENT (4) FOR DATA TOOLKIT.....	71
TABLE 131 – SYSTEM REQUIREMENT (1) FOR PATTERN GENERATOR.....	72
TABLE 132 – SYSTEM REQUIREMENT (2) FOR PATTERN GENERATOR.....	72
TABLE 133 – SYSTEM REQUIREMENT (3) FOR PATTERN GENERATOR.....	72
TABLE 134 – SYSTEM REQUIREMENT (4) FOR PATTERN GENERATOR.....	72
TABLE 135 – SYSTEM REQUIREMENT (5) FOR PATTERN GENERATOR.....	73
TABLE 136 – SYSTEM REQUIREMENT (6) FOR PATTERN GENERATOR.....	73
TABLE 137 – SYSTEM REQUIREMENT (1) FOR ADW CORE.....	74
TABLE 138 – SYSTEM REQUIREMENT (2) FOR ADW CORE.....	74
TABLE 139 – SYSTEM REQUIREMENT (3) FOR ADW CORE.....	74
TABLE 140 – SYSTEM REQUIREMENT (4) FOR ADW CORE.....	75
TABLE 141 – SYSTEM REQUIREMENT (5) FOR ADW CORE.....	75
TABLE 142 – SYSTEM REQUIREMENT (6) FOR ADW CORE.....	75
TABLE 143 – SYSTEM REQUIREMENT (7) FOR ADW CORE.....	76
TABLE 144 – SYSTEM REQUIREMENT (8) FOR ADW CORE.....	76
TABLE 145 – SYSTEM REQUIREMENT (9) FOR ADW CORE.....	76
TABLE 146 – SYSTEM REQUIREMENT (1) FOR ADAPTABLE VISUALIZATIONS.....	77
TABLE 147 – SYSTEM REQUIREMENT (2) FOR ADAPTABLE VISUALIZATIONS.....	77
TABLE 148 – SYSTEM REQUIREMENT (3) FOR ADAPTABLE VISUALIZATIONS.....	77

List of figures

FIGURE 1. REQUIREMENTS ENGINEERING METHOD	10
FIGURE 2. BIGDATASTACK CORE PLATFORM CAPABILITIES	13
FIGURE 3. NETDATA ROLE IN TRIPLE MONITORING.....	83
FIGURE 4. DATA INGESTION IN WATSON IOT PLATFORM	94
FIGURE 5. NODE-RED PROGRAMMING EXAMPLE	96

1 Executive Summary

This is the second version of a series of three deliverables specifying the stakeholder as well as technical (software and technology) requirements for BigDataStack. These three versions of the requirements specification are delivered at M6 (D2.1), M11 (D2.2, the present version) and M22 (D2.3).

In the requirements analysis shown in this document, a top-down approach is taken with respect to the user requirements, which have been collected through the BigDataStack use case providers. This is complemented with a bottom-up approach aiming to identify, collect, and analyse the rest of stakeholder requirements as well as technical requirements from BigDataStack technology providers.

The analysis has produced measurable and unambiguous requirements, which inform and drive architectural and design decisions at different levels of the BigDataStack platform: 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 (and brought as-is from the previous version of the requirements specification, D2.1). 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.

Note that the set of requirements contained in this deliverable supersedes those specified in the first version of the requirements specification (D2.1) at M6, mainly to capture the rationale behind the architecture decisions described in D2.4 (M8), and design decisions described in D3.1, D4.1 and D5.1 (M11). As new requirements and constraints are expected to emerge during the ongoing implementation and experimentation, the present requirements specification will be superseded by a new version (D2.3) at M22.

2 Introduction

The main purpose of this deliverable is to describe the set of measurable and unambiguous **business and technical requirements for the BigDataStack environment** as known at M11. This set will be further tracked and validate the architecture development and implementation during the project lifetime.

This report represents the second 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 as well as component design and implementation activities** of the project.

Task 2.1 started at M1 and produced the first version of the requirements analysis (D2.1) at M6. The continued work on requirements analysis has produced a new version of the requirements at M11, giving rise to the present deliverable D2.2. A third and last version of the requirements (refining the present version) is expected to be delivered at M22 as D2.3.

The main contribution (and difference) of this deliverable with respect to the first requirements specification (D2.1) is the following:

- a) Redefinition of GFT's use case in the context of Smart Insurance instead of Intelligent Multi-Channel Baking ([Section 4.3](#)).
- b) Redefinition of the stakeholder, system and technology requirements ([Sections 6, 7 and 8](#)).
- c) Better explanation of the organization of the document ([Section 2.2](#)).

Like in the first requirements analysis (D2.1), this elicitation and analysis of the requirements has been carried out considering the needs and concerns coming from the communities and end users related to the BigDataStack's use case providers 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). The method is fully explained at Section 2.1.

To contextualize the analysis of requirements, Section 3 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 9). In fact, Sections 3 and 9 are a refinement of the sections dedicated respectively to business goals and baseline technologies in the internal joint (with architecture) report completed at M3 (and brought as-is from D2.1).

The rest of the document is organized as follows: Section 2.1 explains the requirements engineering method and Section 2.2 the organization of requirements in this deliverable; Section 4 specifies business scenarios and use case requirements related to three use cases; Section 5 introduces the most relevant roles that the actors (stakeholders) interacting with the platform may take; Sections 6, 7 and 8 specify the rest of stakeholder requirements as well as technical (system and software) requirements; finally, Section 9 describes baseline technologies relevant for BigDataStack.

It is important to note that the requirements specified in this deliverable were brought into D3.1, D4.1 and D5.1 for the reader's convenience, i.e., for a better understanding of the

BigDataStack capabilities design described in those deliverables: **Data-driven Infrastructure Management (D3.1)**, **Data as a Service (D4.1)**, and **Dimensioning, Modelling and Interaction Services (D5.1)**. Nevertheless, the present document should be considered the single source of all requirements and be considered the master version of them in case of discrepancies.

2.1 Method

The requirements engineering method will follow ISO/IEC/IEEE 29148:2011¹ 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 (<i>StRS</i>)
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.	<i>System Requirements Specification (SyRS)</i>
		<i>Software Requirements Specification (SRS)</i>

Table 1 – Requirements engineering processes

BigDataStack Requirements Engineering¹ Iterative and recursive method

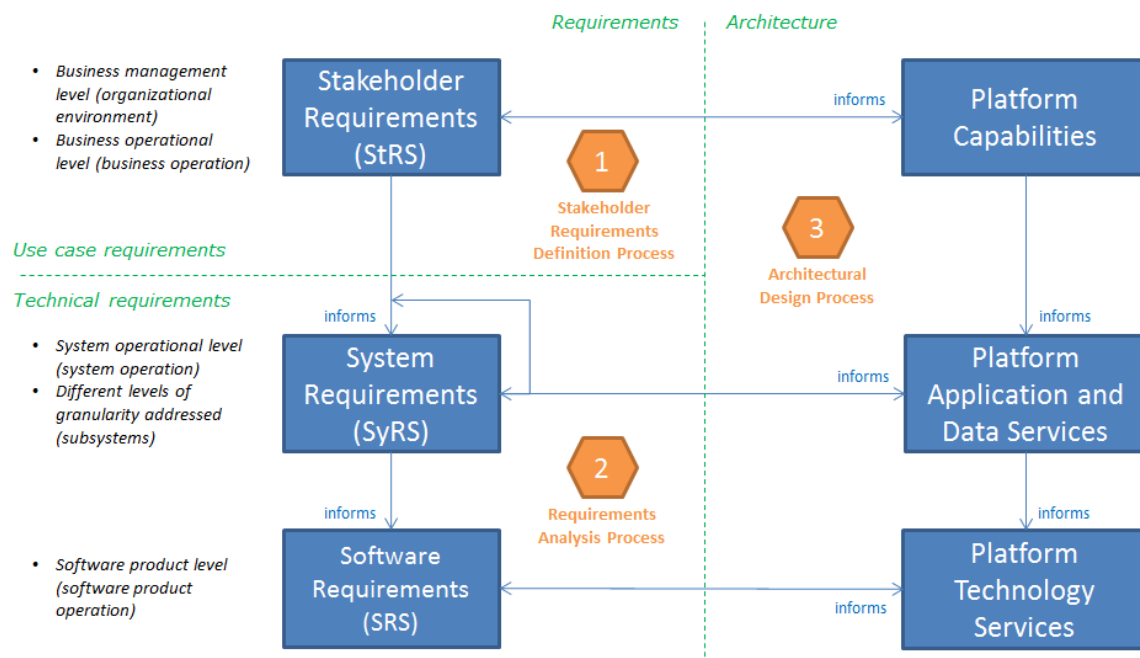


Figure 1. Requirements engineering method

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

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 (extracted from ISO/IEC/IEEE 29148:2011 [41]), including the architecture domain whose decisions are informed by them.

Work product	Acronym	Description	Informed architecture domain
Stakeholder Requirements Specification	StRS	<p>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.</p> <p>It specifies:</p> <ul style="list-style-type: none"> A. The required system characteristics and context of use of the product (platform) business functions and services, and operational concepts are specified. B. The constraints on a system solution are defined. C. Traceability of stakeholder requirements to stakeholders and their needs is achieved. D. The stakeholder requirements are defined from the stakeholder’s perspective. E. Stakeholder requirements for validation are identified. 	Platform Capabilities (business architecture)
System Requirements Specification	SyRS	<p>Technical specifications for the selected system of-interest and usability for the envisaged human-system interaction. It characterises system requirements because:</p> <ul style="list-style-type: none"> F. It represents a system (including interfaces of functions and services) that will meet stakeholder requirements. G. Allows lower levels of granularity (recursion), i.e., subsystems. H. Does not imply any specific implementation. <p>It specifies:</p> <ul style="list-style-type: none"> I. The future system requirements from the domain perspective, background information about the overall objectives for the system, and its target environment. J. A statement of the constraints, assumptions and non-functional requirements. 	Platform Applications and Data Services Architecture

		K. Measurable system requirements specifying, from the supplier's perspective, what characteristics and with what magnitude it is to possess to satisfy stakeholder requirements.	
Software Requirements Specification	SRS	<p>A specification for a software product, program, or set of programs) that performs certain functions in a specific environment.</p> <p>The SRS may be written by one or more representatives of the supplier, one or more representatives of the acquirer, or by both.</p> <p>Typically,</p> <ul style="list-style-type: none"> L. there will be a requirement specification that will state the interfaces between the system and a software portion; M. it will place external performance as well as functionality requirements upon the software portion; N. it defines all the required features (e.g., functions) of the specified software product to which it applies; and O. it documents the conditions and constraints under which the software portion must perform, and the intended verification approaches for the requirements. 	Platform Technology Architecture

Table 2 – Levels of requirement specification

Finally, to identify requirements from all stakeholders' point of view, we have taken inspiration from the *TOGAF® Series Guide²: 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>

2.2 Organization

In this document, we firstly describe the stakeholder requirements of the **User Enterprises** (see Table 3 – Stakeholders) to which the use cases belong. This is done through business scenarios which represent business problems for the customer organization of a developed Big Data solution (see Section 4 - Use Case Requirements and Scenarios).

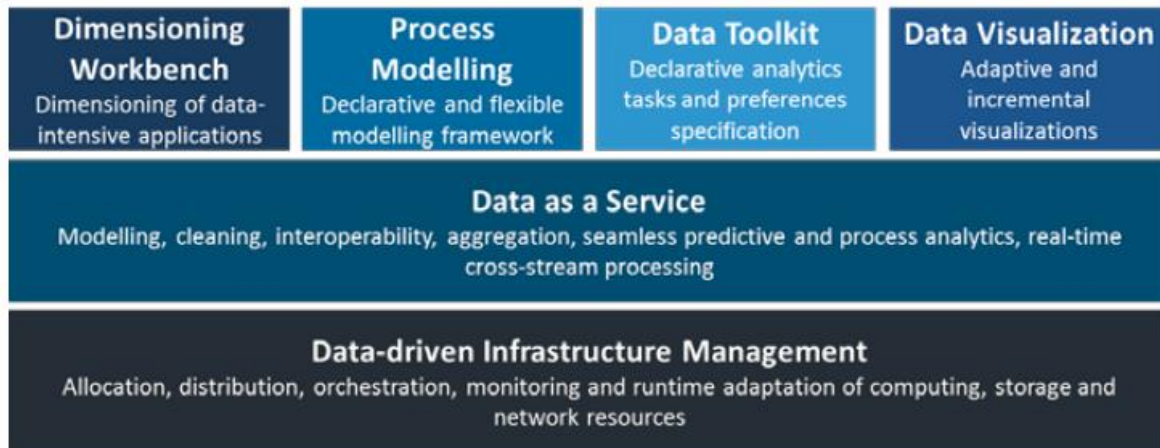


Figure 2. BigDataStack core platform capabilities

Secondly, we present the rest of stakeholder requirements as well as the system and technology (software) requirements organized in terms of the envisioned BigDataStack platform capabilities (see Figure 2):

- **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.
- **Dimensioning, Modelling and Interaction Services:**
 - Data Visualization 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 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 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 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.

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 M18³ 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

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

		data protection mechanism is needed. For businesses this is a key point, since 89% of companies avoid doing business 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

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

the envisioned BigDataStack platform value proposition for customers at each side of it (demand and supply sides) as well as the revenue model proposed for them⁶:

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 <i>technology</i> , <i>applications</i> and <i>data</i> 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.

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.

	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	<i>Infrastructure providers, Data providers, Application providers, Data practitioners, Citizens, SMEs and Large industries, Public Organisations, Entrepreneurs, Decision makers.</i>

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	<i>Citizens, SMEs and Large Industries, Public Organisations, Entrepreneurs, Decision makers.</i>

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	<i>Infrastructure providers, Data providers, Application providers, Data practitioners.</i>

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	<ul style="list-style-type: none"> - 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

	<p>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.</p>
Variations	<p>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.</p>
Post-condition	<p>If a malfunction pattern is identified, the actor is informed via an alert.</p>
Diagrams	<pre> graph TD C[Coordinator] --> MEM([Main Engine Monitoring]) C --> MA([Malfunction Alert]) C --> AI([Alert Inspection]) FM[Fleet Manager] --> MEM FM --> MA FM --> AI MA -.-> MEM AI -.-> MA </pre>

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

Section	Description
Id	SCE-RSM-02
Title	Requisition and dynamic routing
Description	<p>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.</p>
Actors	Coordinator, Fleet manager, Supplies coordinator

Objectives	<ul style="list-style-type: none"> - 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	<pre> graph TD C[Coordinator] --> SR(Spare Part Requisition) C --> DR(Dynamic Routing) FM[Fleet Manager] --> SR FM --> DR SC[Supplies Coordinator] --> RP(Requisition Process) RP -.-> DR DR -.-> SR </pre>

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	<p>Data requirements (indicative):</p> <ul style="list-style-type: none"> - 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: <ul style="list-style-type: none"> - 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: <ul style="list-style-type: none"> - 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: <ul style="list-style-type: none"> - 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 <i>DANAOS</i> 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: <ul style="list-style-type: none"> - 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: <ul style="list-style-type: none"> - 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	<p>This scenario is distributed in three steps:</p> <ul style="list-style-type: none"> - data collection, - calculate recommendations, and - show predictions. <p>The first step, data collection, provides services to update those entities needed for the recommender with data coming from external systems:</p> <ol style="list-style-type: none"> a) Product Service (products, categories, products x category) b) Sales Service (orders) c) Client service d) Events service (used by the external systems to provide feedback about the visualization of the recommendations) <p>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.</p> <p>Therefore, we can split the recommendation process in two phases:</p> <ul style="list-style-type: none"> - Calculate the products that a user would be interested to buy, based on: <ol style="list-style-type: none"> 1. Product sales frequency by user 2. Product sales frequency by product 3. Product seasonality 4. Product cross-selling 5. User feedback (registered with events) - Calculate the promotions that will be recommended to the final users. The promotions are calculated in base to: <ul style="list-style-type: none"> - Representative products that the user could buy (calculated in “calculate products” use case) - Possible promotions, with a priority ranking - User feedback (registered with events)

	<p>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.</p>
Actors	External System, Trigger recommender
Objectives	<p>The main objective of this scenario is to calculate the most interesting products and promotions to recommend.</p> <p>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.</p>
Pre-conditions	<p>The system needs, at least, 2 years of data history to do good predictive recommendations. This data includes:</p> <ol style="list-style-type: none"> 1) Sales 2) Clients 3) Products 4) Categories <p>Additionally, we can also have some other data that can be included in the recommendation process, such as user events, user feedback, etc.</p>
Process Description	<ul style="list-style-type: none"> - Data collection <ul style="list-style-type: none"> - External system invokes service - System stores the data provided by the external system - Calculate recommendations <ul style="list-style-type: none"> - 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 - Data preparation <ul style="list-style-type: none"> - Some input data arrives to the system - The system initiates a denoising process - The system initiates a modelling process - Denoising data <ul style="list-style-type: none"> - The process for denoising is called with some data - The data noise is removed - Denoised data is returned to the caller - Data modelling <ul style="list-style-type: none"> - The process for modelling is called with some data - The data is modelled - Modelled data is returned to the caller - Calculate products

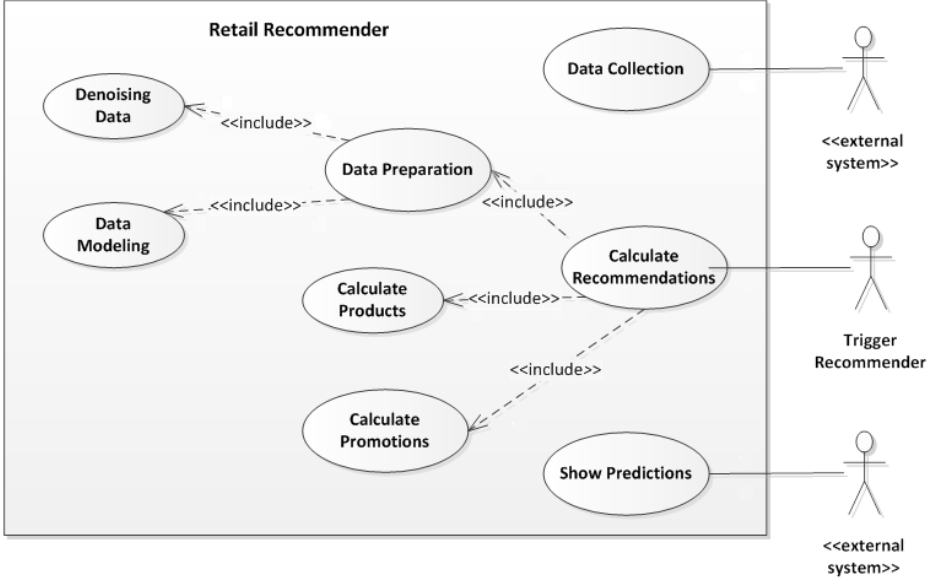
	<ul style="list-style-type: none"> - The process of calculate products is called with some data - The system calculates recommended products for every user - Calculate promotions <ul style="list-style-type: none"> - The process of calculate promotions is called with some data. - If the data does not include the suggested products calculated, the system calculates recommended products for every user. - The system calculates recommended promotions. - Show Predictions <ul style="list-style-type: none"> - External system invokes service - System provides the predictions requested by the external system
Variations	N/A
Post-condition	<p>The suggested products and promotions shouldn't be null. The system must store in the DB:</p> <ul style="list-style-type: none"> - 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	 <p>The diagram shows a system boundary labeled 'Retail Recommender'. Inside, there are several use cases: 'Data Collection', 'Data Preparation', 'Calculate Recommendations', 'Calculate Products', 'Calculate Promotions', 'Show Predictions', 'Denoising Data', and 'Data Modeling'. 'Data Collection' is connected to an external actor '<<external system>>'. 'Calculate Recommendations' is connected to an actor 'Trigger Recommender'. 'Show Predictions' is connected to an external actor '<<external system>>'. 'Data Preparation' includes 'Denoising Data' and 'Data Modeling'. 'Calculate Recommendations' includes 'Calculate Products' and 'Calculate Promotions'. 'Calculate Promotions' includes 'Show Predictions'.</p>

Table 21 – Retail Recommender (scenario)

Section	Description
Id	SCE-CC-02
Title	Retail Demonstrator
Description	<p>This scenario is distributed in three modules: login, View Predicted Products and View Predicted Promotions.</p> <ul style="list-style-type: none"> - The login module logs into the demonstrator application for a given customer. - The View Predicted Products module displays the products that

	<p>the system would suggest to the customer.</p> <ul style="list-style-type: none"> - The View Predicted Promotions module displays the personalized promotions that the system would suggest to the customer. <p>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.</p>
Actors	Customer, Products Recommender
Objectives	<ul style="list-style-type: none"> - 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	<ol style="list-style-type: none"> 1. Display predicted products <ul style="list-style-type: none"> - User logs into the system - System verify username exists - User selects <i>My predicted products</i> - 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 <ul style="list-style-type: none"> - 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	<p>The user is logged in the demonstrator, and the user can view the following in the application:</p> <ul style="list-style-type: none"> - List of products predicted for the user - List of promotions predicted for the user
Diagrams	

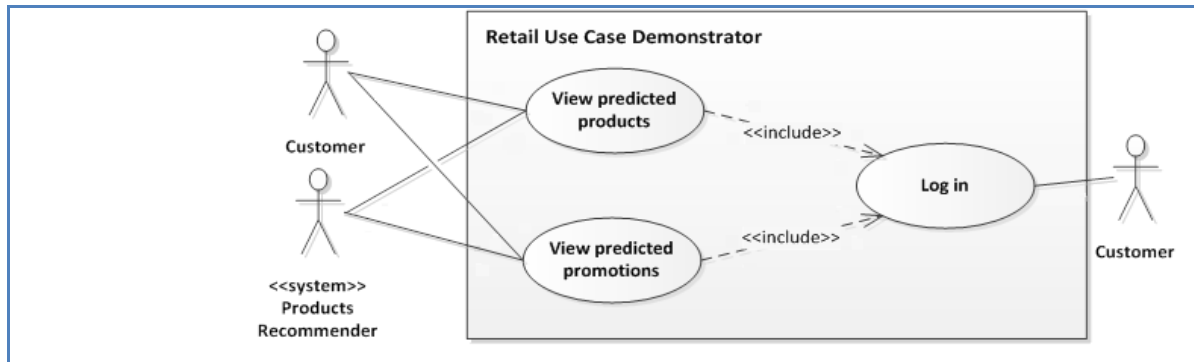


Table 22 – Retail Demonstrator (scenario)

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	<p>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:</p> <ul style="list-style-type: none"> - 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	<p>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.</p>
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: <ul style="list-style-type: none"> - 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: <ul style="list-style-type: none"> - 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: <ul style="list-style-type: none"> - 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: <ul style="list-style-type: none"> - Client Id - Order Date - Items <ul style="list-style-type: none"> ○ 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 <i>Eroski</i> customers should be loaded at least once per day. Customers should have at least the following information: <ul style="list-style-type: none"> - 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 <i>Eroski</i> products should be loaded at least once per day. Products information should have at least the following information: <ul style="list-style-type: none"> - 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 Smart Insurance⁸

4.3.1 Introduction

This section refers to the use case of Smart Insurance: Customers segmentation and Customer Lifetime Value (CLV) prediction. Here, we discuss the usage scenarios by giving a description of the use cases (Section 4.3.2) along with a high-level representation of functional requirements (Section 4.3.3).

The use case focuses on the development of solution for Insurance companies, developing software and systems addressing the needs of such institutions based on a data-centric paradigm and addressing the provision of services according to the customer “tailored” requirements.

The main goal is to allow insurance companies that focus on customer management, to provide personalized services for their customers, as well as new corporate services for the handling of customers’ profitability and retention.

This scenario is realized in collaboration with **HDI Assicurazioni**, which is part of a large German insurance group, of international standing, the **Talanx Group of Hannover**⁹ (born HDI group). Talanx is the third insurance company in Germany and operates in 150 countries.

⁸ Note that with respect to deliverable D2.1, this use case has been redefined, according to a new context (insurance instead of banking) and scenarios.

⁹ <http://www.talanx.com/>

4.3.2 Scenarios

Section	Description
Id	SCE-SI-01
Title	Customers segmentation
Description	<p>The scenario focuses on customers' segmentation according to their financial sophistication, age, location, etc. Thus, all the customers are classified into groups by spotting coincidences in their personal information, preferences or behaviour. This grouping allows developing attitude and solutions especially relevant for the particular customers. As a result, target cross-selling (recommendations of products to customers based on what other customers bought) and upselling (recommendations of more advanced products to customers based on what they have bought) strategies may be developed and personal services may be tailored for each particular segment (such as lower priced premiums).</p> <p>This scenario is distributed in three steps:</p> <ul style="list-style-type: none"> - Data collection. - Optimizing the product configuration, and recommendations. - Show recommended products.
Actors	Customer, Service Provider
Objectives	<ul style="list-style-type: none"> - Provide a clusterization of the insurance company customers according to predefined variables. - Optimize product configuration or suggestions for campaigns and cross selling/upselling strategies.
Pre-conditions	N/A
Process Description	<ul style="list-style-type: none"> - Data collection: the system collects information related to customers and their policies/guarantees. - Data processing: the intelligent system module analyses the collected data and elaborates personalized policies and guarantees for the different customers. - Data visualization: the results from the previous phase are optimized and prepared for the presentation.
Variations	N/A
Post-condition	<ul style="list-style-type: none"> - The suggested recommendations and optimizations should not be null. - The system must store in the DB the optimizations and recommendations for every customer.

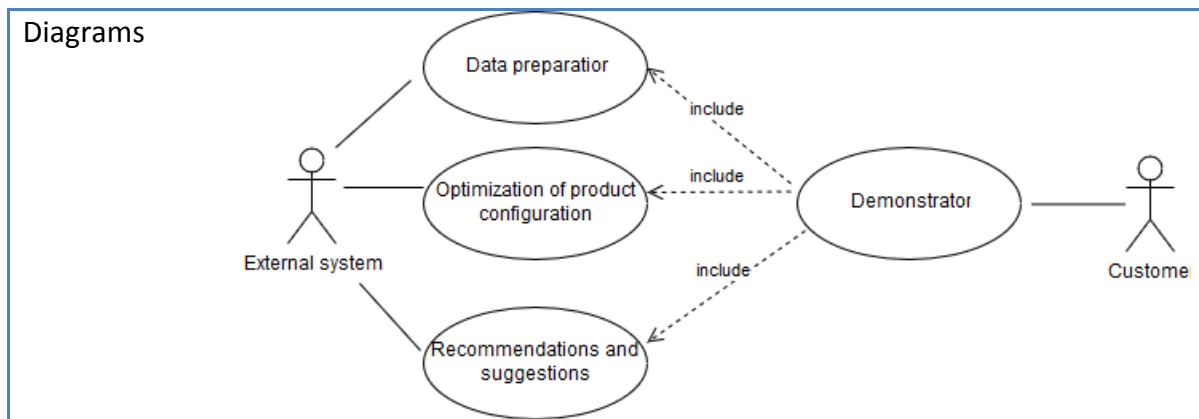


Table 34 - Customers segmentation scenario

Section	Description
Id	SCE-SI-02
Title	Customer Lifetime Value prediction
Description	Customers Lifetime Value is a complex phenomenon representing the value of a customer to a company in the form of the difference between the revenues gained and the expenses made projected into the entire future relationship with a customer. Prediction of the CLV is typically assessed via customer behaviour data in order to predict the customer’s profitability for the insurer. Thus, the behaviour-based models will be applied to forecast the customer retention . This allows forecasting the likelihood of the customers’ behaviour and attitude, as well as churn prevention (identify which customers are likely to cancel contracts in the near future).
Actors	Customer, Service Provider
Objectives	<ul style="list-style-type: none"> - Compute and dynamically update the CLV. - Forecast which customers are likely to cancel contracts in the near future.
Pre-conditions	The prediction approach needs at least 1 year of historical data for efficient predictions.
Process Description	The key aspects in this case are related to data analytics in order to predict the different customers value (e.g., most profitable), analyse present and future profitability, identify target customers, and predict which customers are not satisfied and are likely to cancel their contracts in the future. This information allows enhancing the process described in the first scenario, and helps providing better recommendations to customers.
Variations	N/A
Post-condition	<ul style="list-style-type: none"> - The computed CLV should not be null. - The system must store the predicted results about customers.

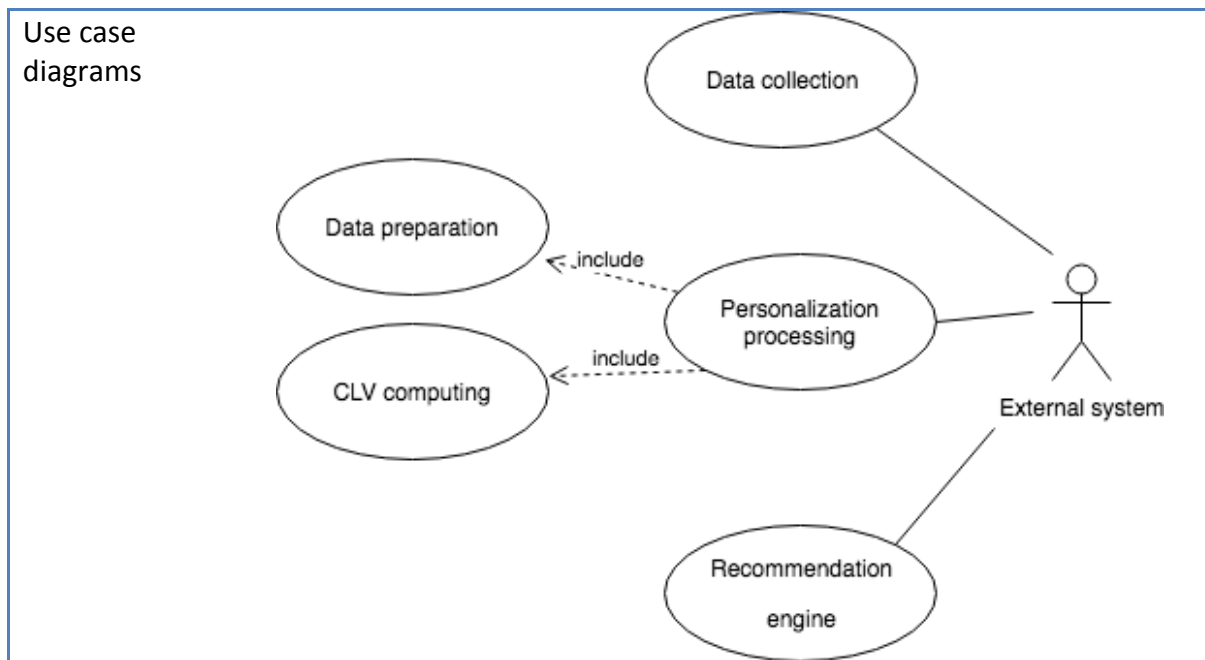


Table 35 - Customer lifetime value prediction scenario

4.3.3 Requirements

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

Section	Description
Id	REQ-SI-01
Type	FUNC
Short name	Provide personalized policies required by customer
Description	For any customer, the system should be able to predict a set of personalized policies for the customer. The calculation should consider the historic of product purchases made by the customer.
Additional Information	N/A
Priority	MAN

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

Section	Description
Id	REQ-SI-02
Type	FUNC
Short name	Provide personalized guarantees required by customer
Description	For any customer, the system should be able to predict a set of personalized guarantees for the customer.

	The calculation should consider the historic of product purchases made by the customer.
Additional Information	N/A
Priority	MAN

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

Section	Description
Id	REQ-SI-03
Type	DATA
Short name	Customers
Description	Customers should have at least the following information: <ul style="list-style-type: none"> - Client Id
Additional Information	N/A
Priority	MAN

Table 38 – Customers (stakeholder requirement)

Section	Description
Id	REQ-SI-04
Type	DATA
Short name	Policies
Description	Policies' information should have at least the following information: <ul style="list-style-type: none"> - Policy reference - Policy category
Additional Information	N/A
Priority	MAN

Table 39 – Policies (stakeholder requirement)

Section	Description
Id	REQ-SI-05
Type	DATA
Short name	Guarantees
Description	Guarantees' information should have at least the following information: <ul style="list-style-type: none"> - Guarantee reference - Guarantee category

Additional Information	N/A
Priority	MAN

Table 40 – Guarantees (stakeholder requirement)

Section	Description
Id	REQ-SI-06
Type	Stakeholder
Short name	Usability
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-SI-07
Type	Stakeholder
Short name	Security
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 Infrastructure-Data Management Requirements

To facilitate the understanding of the design as well as the challenges addressed by this component, the requirements related to this component were brought into D3.1 and literally included from this section. Therefore, note the following requirement tables also appear as-is in such deliverable for the reader's convenience.

	Id ¹⁰	Level of detail ¹¹	Type ¹²	Actor ¹³	Priority ¹⁴
	REQ-CM-01	System	FUNC	Developer	MAN
Name	Support OpenShift installation on OpenStack VMs				
Description	Include the needed steps on the OpenShift installer to handle OpenShift cluster installation on top of OpenStack resources, i.e, VMs, networks, volumes, etc.				
Additional Information	This needs to be done in the 'upstream' way so that it is supported also after the project lifecycle. It entails modification to different repositories, not only the openshift/installer (https://github.com/openshift/installer) but also other related such as: <ul style="list-style-type: none"> - cluster-network-operator¹⁵ - cluster-api-provider-openstack¹⁶ - gophercloud¹⁷ 				

Table 44 - Support OpenShift installation on OpenStack VMs (system requirement).

	Id	Level of detail	Type	Actor	Priority
	REQ-CM-02	System	PERF	Developer	MAN
Name	Avoid double encapsulation of network packages				
Description	Integrate Kuryr on the OpenShift installer to avoid the double encapsulation problem due to using 2 different overlays (OpenStack SDN and OpenShift SDN on top). Kuryr enables containers running on top of OpenStack VMs to use the same SDN as the VMs itself, i.e., the OpenStack SDN. Thus, avoiding				

¹⁰**Identifier:** To be used in D2.2 to allow for the correct traceability of requirements.

¹¹**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).

¹²**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).

¹³**Actor:** It needs to be either one of the BigDataStack platform roles identified in Section 5 or a system actor, e.g. another component or service.

¹⁴**Priority:** Requirements can have different priorities: MAN (mandatory requirement), DES (desirable requirement), OPT (optional requirement), ENH (possible future enhancement).

¹⁵ <https://github.com/openshift/cluster-network-operator>

¹⁶ <https://github.com/kubernetes-sigs/cluster-api-provider-openstack>

¹⁷ <https://github.com/gophercloud/gophercloud>

	the double encapsulation and enabling a remarkable throughput gain, needed for handling the data at the BigDataStack components.
Additional Information	Similarly, to REQ-CM-01, this needs to be done in the ‘upstream’ way so that it is supported after the project. It entails modifications to the same repositories plus the addition of a kuryr operator that will handle the kuryr related operational actions,

Table 45 - Avoid double encapsulation of network packages (system requirement).

	Id	Level of detail	Type	Actor	Priority
	REQ-CM-03	System	ENV	Developer	DES
Name	Spark operator				
Description	This operator will be responsible for handling the spark cluster, not only its installation but also the scaling actions. It will offer an API to the spark management through the OpenShift API.				
Additional Information	This is related to the dynamic orchestrator, as the optimization actions could be then simply triggered through standard OpenShift API commands (e.g., modifying the information at the associated spark ConfigMap)				

Table 46 - Spark Operator (system requirement).

	Id	Level of detail	Type	Actor	Priority
	REQ-CM-04	System	ENV	Developer	DES
Name	Accept requests to allocate additional resources to one of the storage layer components				
Description	The Adaptable Distributed Storage component can be scaled in/out independently, considering decisions based on its internal metrics and handle on its own the reconfiguration of the internal data regions. Due to this, it is necessary from the Cluster Management to provide a mechanism that allows the storage layer to request for additional resources or the release of already provided ones.				
Additional Information	This is closely related to requirement REQ-ADS-04 “Be able to request additional resources from the infrastructure layer,” described in D4.1.				

Table 47 - Accept requests to allocate additional resources to the storage layer (system requirement).

	Id	Level of detail	Type	Actor	Priority
	REQ-CM-05	System	ENV	Developer	OPT
Name	Force the storage layer to release some of its available resources				
Description	The cluster management might identify that the overall BigDataStack platform is running out of available resources. To ensure the execution of crucial components, it might decide to reduce some of the already allocated resources for some services, for the benefits of others. Due to this, it should be able to request the release of the storage resources and wait for its				

	proper response. The storage should be able to reject such requests, in cases that could lead to data loss.
Additional Information	This is close related with requirement REQ-ADS-05 “Being able to release resources and adapt if resources are deallocated from the infrastructure,” as described in more details in D4.1.

Table 48 - Force the storage layer to release some of its available resources (system requirement).

	Id	Level of detail	Type	Actor	Priority
	REQ-DO-01	Stakeholder	FUNC	Developer	MAN
Name	Correction of Requirements or SLOs Violations				
Description	When an application or service is running, the orchestrator shall detect the violation of an application requirement or service level objective (SLO) and send a signal to the ADS-ranker to trigger a change in the deployment to try to satisfy the requirements or SLOs.				
Additional Information	N/A				

Table 49 - Correction of Requirements and SLOs Violations (stakeholder requirement).

	Id	Level of detail	Type	Actor	Priority
	REQ-DO-02	Stakeholder	FUNC	Developer	MAN
Name	Decision Efficiency				
Description	When the violation of a requirement has been detected, the orchestrator shall be able to decide what modification to the deployment (e.g. change the number of replicas or the number of vCPUs) has the highest probability of improving the requirements or SLOs satisfaction, as long as any change is possible (i.e. all resources are at its full capacity due to limits).				
Additional Information	N/A				

Table 50 - Decision Efficiency (stakeholder requirement).

	Id	Level of detail	Type	Actor	Priority
	REQ-DO-03	System	FUNC	Developer	MAN
Name	Resources Limits				
Description	The orchestrator shall be able to receive a trigger from the ADS-Ranker when a deployment parameter, such as the number of replicas, the number of vCPUs or the assigned cluster memory, cannot be further increased or decreased (i.e. this resource has reached its maximum or minimum possible value) and use this information in its own decisions.				
Additional Information	The complete list of deployment parameters to be taken into account still needs to be determined.				

Table 51 - Resources Limits (stakeholder requirement).

	Id	Level of detail	Type	Actor	Priority
	REQ-DO-04	Stakeholder	FUNC	Developer	DES
Name	Orchestration for Improvements				
Description	When an application or service is running, the orchestrator shall detect changes in the system status or inputs (e.g. less new events per minute) and trigger a change in the deployment that results in lower costs (e.g. to use less replicas) without compromising the application functioning.				
Additional Information	N/A				

Table 52 - Orchestration for Improvements (stakeholder requirement).

	Id	Level of detail	Type	Actor	Priority
	REQ-ADSR-01	System	FUNC	Application Dimensioning Workbench	MAN
Name	Ingest Candidate Deployment Playbooks and Benchmarking Information				
Description	The Application Dimensioning Workbench sends a series of candidate deployment patterns (CDP) playbooks and benchmarking information to the ADS Ranking component. ADS Ranking needs to collect all these patterns for subsequent scoring/ranking based on the user requirements and preferences.				
Additional Information	Ingestion occurs via a common publisher/subscriber platform (RabbitMQ).				

Table 53 - Ingest Candidate Deployment Playbooks and Benchmarking Information (system requirement).

	Id	Level of detail	Type	Actor	Priority
	REQ-ADSR-02	System	FUNC	Developer	MAN
Name	Deployment Suitability Feature Extraction				
Description	Once a series of candidate deployment pattern playbooks and associated benchmarking information has been received, the next step is to determine how each pattern is predicted to perform based on the benchmarking information. In effect, this involves defining a series of functions that relate individual or groups of user requirements to the predicted performances produced by benchmarking. The output of this step is a vector representation for each CDP playbook, representing how that playbook is predicted to fair under different user requirements.				
Additional Information	Features produced here are dependent on the capabilities of the benchmarking system and the amount of information the user provides in terms of requirements and preferences.				

Table 54 - Deployment Suitability Feature Extraction (system requirement).

	Id	Level of detail	Type	Actor	Priority
	REQ-ADSR-03	System	FUNC	Developer	MAN
Name	CDP Playbook Scoring (Heuristic)				
Description	Given a vector representation for a CDP Playbook, we next need to map this vector into a single score, representing how suitable that playbook will be overall (such that we can compare different CDP Playbooks). This involves combining the different elements within the vector (that each represent some aspect of pattern suitability, such as cost, or predicted compute wastage). The first version of this will use a hand-tuned linear combination.				
Additional Information	N/A				

Table 55 - CDP Playbook Scoring (Heuristic) (system requirement).

	Id	Level of detail	Type	Actor	Priority
	REQ-ADSR-04	System	FUNC	Developer	DES
Name	CDP Playbook Scoring (Supervised)				
Description	Given a vector representation for a CDP Playbook, we next need to map this vector into a single score, representing how suitable that playbook will be overall (such that we can compare different CDP Playbooks). This involves combining the different elements within the vector (that each represent some aspect of pattern suitability, such as cost, or predicted compute wastage). The second version of this will learn how to combine the elements based on logging information from past deployments. Models may be non-linear in nature.				
Additional Information	Depends on REQ-ADSR-06.				

Table 56 - CDP Playbook Scoring (Supervised) (system requirement).

	Id	Level of detail	Type	Actor	Priority
	REQ-ADSR-05	System	FUNC	Developer	MAN
Name	CDP Playbook Selection				
Description	Once all candidate deployment patterns have been scored, the final step is to select one of those patterns to pass to ADS Deployment. In many cases this will simply involve selecting the highest scoring pattern. However, the user may have the option to select an alternative configuration at this stage.				
Additional Information	N/A				

Table 57 - CDP Playbook Selection (system requirement).

	Id	Level of detail	Type	Actor	Priority
	REQ-ADSR-06	System	FUNC	Developer	DES
Name	Supervised Model Training				
Description	To support REQ-ADSR-04, a supervised scoring model is needed. To react to changes in the deployment environment over time, this model needs to be frequently updated based on new information from current deployments. This model needs to be trained based on logging data being collected by the Triple Monitoring Framework.				
Additional Information	Requires logging information produced by the Triple Monitoring Framework and stored in the Central Decision Tracker.				

Table 58 - Supervised Model Training (system requirement).

	Id	Level of detail	Type	Actor	Priority
	REQ-ADSR-07	System	FUNC	Developer	MAN
Name	CDP Playbook Re-Scoring				
Description	It is envisaged that in (rare) scenarios, an ongoing application deployment will fail to meet the user's quality of service requirements. This might occur due to assumptions on data input volumes being violated for instance. In this case, we may not be able to solve this issue without fully redeploying the user application with different resources. To support such re-deployment activities, ADS Ranking supports a re-scoring function, where a previous set of CDP playbooks for a user's application can be re-scored based on updated preferences provided by the Big Data Stack Orchestrator, as well as live data about how the previous deployment performed (and failed).				
Additional Information	N/A				

Table 59 - CDP Playbook Re-Scoring (system requirement).

	Id	Level of detail	Type	Actor	Priority
	REQ-ADSD-01	Stakeholder	FUNC	Application developers	MAN
Name	Performance Measurability				
Description	Each environment should be measurable according to a set of characteristics, that is, Key Performance Indicators (KPIs).				
Additional Information	The KPIs considered must include: <ul style="list-style-type: none"> - vCPUs - Memory 				

Table 60 - Performance Measurability (stakeholder requirement).

	Id	Level of detail	Type	Actor	Priority
	REQ-ADSD-02	Stakeholder	FUNC	System	MAN
Name	Standards-based Playbook				
Description	The description of the environments and deployments (i.e., playbooks) will follow a standard specification language				
Additional Information	N/A				

Table 61 - Standards-based Playbook (stakeholder requirement).

	Id	Level of detail	Type	Actor	Priority
	REQ-ADSD-03	System	FUNC	System	MAN
Name	Standard deployment information				
Description	When communicating with other components, as described in Section 7.2, these components will use the playbook standard defined in REQ-RD-02.				
Additional Information	N/A				

Table 62 - Standard deployment information (system requirement).

	Id	Level of detail	Type	Actor	Priority
	REQ-ADSD-04	System	FUNC	System	MAN
Name	Application Scoring System				
Description	The ranking system evaluates each environment's deployment, which keeps track of the most suitable configuration for each application. When trying a deployment configuration for a new application, this ranking will be used to select the most suitable one.				
Additional Information	The evaluation will be done following the measurements defined in REQ-RD-01.				

Table 63 - Application Scoring System (system requirement).

	Id	Level of detail	Type	Actor	Priority
	REQ-ADSD-05	System	FUNC	Cluster management component	MAN
Name	Compatibility with Kubernetes				
Description	Since the technology used to run and orchestrate the applications is based in Kubernetes (OKD ¹⁸). Thus, the ADS-Deployment component is required to be compatible with Kubernetes.				

¹⁸ OKD - <https://www.okd.io/>

Additional Information	The ADS-Deploy component should translate from the playbook standard defined in REQ-RD-01 into Kubernetes primitives.
-------------------------------	---

Table 64 - Compatibility with Kubernetes (system requirement).

	Id	Level of detail	Type	Actor	Priority
	REQ-ADSD-06	System	FUNC		MAN
Name	Synchronous communication				
Description	The communication with and within both components should be done through an API REST.				
Additional Information	N/A				

Table 65 - Synchronous Communication (system requirement).

	Id	Level of detail	Type	Actor	Priority
	REQ-TM-01	Stakeholder	FUNC	Developer	MAN
Name	Regular recording of QoS metrics				
Description	<p>When a user's application is deployed, the Triple Monitoring Framework monitors that application, tracking statistical information about its operation and associated QoS data, including network, data storage, virtualization layers, etc.</p> <p>This data is needed to support the learning of ranking models by ADS-Ranking service (part of Application and Service Deployment; see REQ-ADSR-03) and regularly saved in a centralised data store for later access.</p>				
Additional Information	<p>Input:</p> <ul style="list-style-type: none"> - Candidate Deployment Pattern (application identifier from this is the primary key for saving monitoring data for an application) <p>Output:</p> <ul style="list-style-type: none"> - Deployment QoS Snapshot (monitoring/QoS data, every few mins) <p>Service Dependencies:</p> <ul style="list-style-type: none"> - Centralised Data Store (Storage Service) - <p>This is implemented over Prometheus¹⁹ as the monitoring collector.</p>				

Table 66 - Regular recording of deployment QoS information (stakeholder requirement).

	Id	Level of detail	Type	Actor	Priority
	REQ-TM-02	Stakeholder	FUNC	Developer	MAN
Name	QoS violation alert				
Description	If the system does not respect the agreed QoS, an alert is raised.				

¹⁹ Prometheus. <https://prometheus.io/>

Additional Information	This alert is used internally to evaluate the performance of an environment, relating to REQ-RD-004.
-------------------------------	--

Table 67 - QoS violation notification (stakeholder requirement).

	Id	Level of detail	Type	Actor	Priority
	REQ-TM-03	Stakeholder	FUNC	Developer	DES
Name	QoS violation monitoring				
Description	QoS violations are also monitored and shown to the user/admin.				
Additional Information	N/A				

Table 68 - QoS violation monitoring (stakeholder requirement).

	Id	Level of detail	Type	Actor	Priority
	REQ-TM-04	System	FUNC	Developer	MAN
Name	Metrics pusher				
Description	The metric pusher retrieves KPI data, clean them and ingest them into the monitoring collector (<i>Prometheus</i>).				
Additional Information	The metrics pusher is used when the exporter approach is impossible to apply. This solution will be very useful for getting application specific metrics (it's not approved yet).				

Table 69 - Metrics pusher (system requirement).

	Id	Level of detail	Type	Actor	Priority
	REQ-TM-05	System	FUNC	Developer	DES
Name	API REST for accessing the collected monitoring metrics				
Description	The metrics are accessible through an API REST.				
Additional Information	This component translates client's requests to Prometheus request compatible. Grafana ²⁰ will be used for visualization.				

Table 70 - Monitoring metrics API REST (system requirement).

	Id	Level of detail	Type	Actor	Priority
	REQ-TM-06	Software	FUNC	Developer	MAN
Name	Pub/Sub Mechanism for Metrics				
Description	This component queries the metrics repository periodically and publishes this information through a publisher/subscriber mechanism. Each client sends subscription requests to the system.				

²⁰ Grafana. <https://grafana.com/>

Additional Information	The monitoring metrics getter is implemented on RabbitMQ ²¹
-------------------------------	--

Table 71 - Monitoring metrics getter (software requirement).

	Id	Level of detail	Type	Actor	Priority
	REQ-TM-07	Software	FUNC	Developer	DES
Name	Spark compatible				
Description	The triple monitoring engine monitors the performance of Apache Spark ²² , which is used in the BigDataStack project as an analytics engine for Big Data, thus needs to be compatible with this technology.				
Additional Information	Monitoring Spark is done using Spark measure project, which can be embedded in spark application allowing the collection of some metrics after each SQL execution. Those metrics are sent to <i>push gateway</i> to be exported to Prometheus.				

Table 72 - Spark compatibility (software requirement).

	Id	Level of detail	Type	Actor	Priority
	REQ-TM-08	Software	FUNC	Developer	DES
Name	LeanXcale compatibility				
Description	LeanXcale database ²³ already uses Prometheus for its monitoring subsystem. However, the integration is relied on static deployments. Thus, it should be extended to consider re-deployments in cases when an elasticity action takes places which leads to a scale in/out of the resources. In these scenarios, LeanXcale should reconfigure its integration with the existing Prometheus deployment on the run-time and provide monitoring information for the new nodes				
Additional Information	N/A				

Table 73 - LeanXcale compatibility (software requirement).

	Id	Level of detail	Type	Actor	Priority
	REQ-TM-09	Software	FUNC	Developer	DES
Name	OKD compatibility				
Description	The Triple Monitoring engine monitors the performance of Openshift OKD ²⁴ , which is the baseline technology used in the orchestration of containers. Therefore, the triple monitoring engine needs to be compatible with this technology.				

²¹ RabbitMQ. <https://www.rabbitmq.com/>

²² Apache Spark. <https://spark.apache.org/>

²³ LeanXcale. <https://www.leanxcale.com/>

²⁴ Openshift OKD (Origin Kubernetes Distribution). <https://www.okd.io/>

Additional Information	N/A
-------------------------------	-----

Table 74 - OKD compatibility (software requirement).

	Id	Level of detail	Type	Actor	Priority
	REQ-TM-10	Software	FUNC	Developer	DES
Name	CEP compatibility				
Description	The triple monitoring engine monitors the performance of CEP, which is used in the BigDataStack project as a streaming engine for processing data in real-time. Therefore, the triple monitoring engine needs to be compatible with this technology.				
Additional Information	The CEP exposes several monitoring metrics that are exported to Prometheus.				

Table 75 - CEP compatibility (software requirement).

	Id	Level of detail	Type	Actor	Priority
	REQ-TM-11	Software	FUNC	Developer	DES
Name	Minio compatibility				
Description	The triple monitoring engine monitors the performance of Minio ²⁵ , which is used for object storage in the system. Therefore, the triple monitoring engine needs to be compatible with this technology.				
Additional Information	N/A				

Table 76 - Minio compatibility (software requirement).

	Id	Level of detail	Type	Actor	Priority
	REQ-TM-12	Software	FUNC	Developer	DES
Name	OpenStack Networking Services compatibility				
Description	The Triple Monitoring engine monitors the performance of the internal network connecting the different containers inside an application. BigDataStack uses the OpenStack networking services for managing this network communications, so the triple monitoring engine needs to be compatible with this technology.				
Additional Information	N/A				

Table 77 - OpenStack Networking Services compatibility (software requirement).

²⁵ Minio Private Cloud Storage- <https://www.minio.io/>

	Id	Level of detail	Type	Actor	Priority
	REQ-TM-13	Software	FUNC	Developer	MAN
Name	Persistently store the monitoring metrics				
Description	The triple monitoring engine should use a database for persistently storing monitoring metrics and is connected to Prometheus by http.				
Additional Information	This database is based on influxDB24.				

Table 78 - Monitoring database (software requirement).

	Id	Level of detail	Type	Actor	Priority
	REQ-TM-14	Software	FUNC	Developer	ENH
Name	Spark Monitoring Pushgateway				
Description	This component is used to gather metrics from Spark and ingest them into the metrics collector.				
Additional Information	The connection between this component and the applications use http.				

Table 79 - Monitoring Pushgateway (software requirement).

	Id	Level of detail	Type	Actor	Priority
	REQ-TM-16	Software	FUNC	Developer	ENH
Name	Metrics visualization				
Description	The metrics must be shown to the end-user via a graphical interface. Grafana is used for metrics' visualization.				
Additional Information	Grafana ²⁶ is configured for receiving metrics from two sources (Prometheus, InfluxDB).				

Table 80 - Metrics visualization (software requirement).

	Id	Level of detail	Type	Actor	Priority
	REQ-TM-17	System	FUNC	Dynamic Orchestrator	ENH
Name	Asynchronous rich notification of SLA violations				
Description	SLA violations should be notified by means of a publish/subscribe mechanism, together with the metrics (KPIs) upon which the SLA imposes restrictions.				
Additional Information	The main consumer of the SLA violations notifications is the Dynamic Orchestrator.				

Table 81 - Metrics visualization (software requirement).

²⁶ Grafana - <https://grafana.com/>

	Id	Level of detail	Type	Actor	Priority
	REQ-IN-01	Software	FUNC	ROL-02	MAN
Name	Information-Driven Networking based on type of data				
Description	The Information-Driven Networking mechanisms enforce a set of policies by specifying the rules of how two or more components can communicate (send/receive data) with each other according to the available resources.				
Additional Information	A different policy is enforced based on different incoming data requirements, following the type of processing requirements (stream, micro-batch, batch) and the type of data (structured, semi-structured, unstructured).				

Table 82 - Network Policies based on type of data (software requirement).

	Id	Level of detail	Type	Actor	Priority
	REQ-IN-02	Software	FUNC	ROL-02	MAN
Name	Information-Driven Networking based on application requirements				
Description	<p>The Information-Driven Networking mechanisms enforce a set of policies by specifying the rules of how to handle applications with different requirements according to the available resources. For instance, an application with analytics requiring real-time data processing may impose time-critical constraints on the handling, operation and transformation of data.</p> <p>To support online analytics and decision making in time-critical conditions specific network policies need to be applied to deliver the results within predefined time constraints.</p>				
Additional Information	The Data Scientist can set an “allow/deny access” policy regarding the set of applications and their requirements (real-time, close to real-time needs) accessing the backend services of the BigDataStack environment to prioritize/isolate the set of ingress/egress workloads that are enabled/disabled based on their IP & Port in order to achieve efficient services interaction.				

Table 83 - Network policies based on application (software requirement).

7 Data as a Service Requirements

To facilitate the understanding of the design as well as the challenges addressed by this component, the requirements related to this component were brought into D4.1 and literally included from this section. Therefore, note the following requirement tables also appear as-is in such deliverable for the reader's convenience.

	Id ²⁷	Level of detail ²⁸	Type ²⁹	Actor ³⁰	Priority ³¹
	REQ-BDL-01	Software	FUNC	Developer	MAN
Name	Support data skipping for arbitrary query predicates				
Description	The query predicate could comprise UDFs and AND/OR/NOT. Example UDFs could be geospatial or temporal functions.				
Additional Information	This functionality is important for the ship management use case, which requires geospatial UDFs.				

Table 84 - requirement REQ-BDL-01 for Big Data Layout

	Id	Level of detail	Type	Actor	Priority
	REQ-BDL-02	Software	FUNC	Developer	MAN
Name	Support a truly pluggable architecture for data skipping				
Description	The goal of this requirement is to enable the addition of new data skipping index types without changing the core data skipping library. This is needed for requirement REQ-BDL-01 since supporting new UDFs may require new index types.				
Additional Information	External users can also exploit this capability				

Table 85 - requirement REQ-BDL-02 for Big Data Layout

	Id	Level of detail	Type	Actor	Priority
	REQ-BDL-03	Software	FUNC	Developer	MAN
Name	Enable layout change for (part of) a dataset				

²⁷**Identifier:** To be used in D2.2 to allow for the correct traceability of requirements.

²⁸**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).

²⁹**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).

³⁰**Actor:** It needs to be either one of the BigDataStack platform roles identified in Section 5 or a system actor, e.g. another component or service.

³¹**Priority:** Requirements can have different priorities: MAN (mandatory requirement), DES (desirable requirement), OPT (optional requirement), ENH (possible future enhancement).

Description	There is a strong relationship between how a dataset is laid out in the object store and the performance of data skipping against this data set. Moreover, this performance may be also very dependent on the queries. Hence the need to adapt the layout, not only for future data but also for heavily queried data already in object store.
Additional Information	N/A

Table 86 - requirement REQ-BDL-03 for Big Data Layout

	Id	Level of detail	Type	Actor	Priority
	REQ-BDL-04	Software	FUNC	Developer	MAN
Name	Enable on-line data layout				
Description	Layout is critical for the data skipping performance. As of now data is stored as is and possibly laid out again offline. The need is to upload dataset chunks with the best-known layout as data is ingested.				
Additional Information	N/A				

Table 87 - requirement REQ-BDL-04 for Big Data Layout

	Id	Level of detail	Type	Actor	Priority
	REQ-ADS-01	System	DATA	Developer	MAN
Name	Being able to fragment a dataset and move the data fragments across different nodes.				
Description	The adaptable distributed storage should be able to split a dataset into different regions, and move these regions to different data nodes, in order to adapt in case of increased load (both in terms of user workload or data load) so as to achieve efficient consumption, based on the provided resources.				
Additional Information	When a movement (move, split, join) of a data fragment occurs, the storage must not suffer from a down-time. On the contrary, it must remain operational with minimum overhead on the overall performance.				

Table 88 - requirement REQ-ADS-01 for Adaptable Distributed Storage

	Id	Level of detail	Type	Actor	Priority
	REQ-ADS-02	System	ENV	Developer	MAN
Name	Identify data nodes that are overprovisioning.				
Description	The adaptable storage must be able to identify data nodes that are overprovisioning their available resources and send internal alerts to trigger a dynamic reconfiguration of the deployment of the data fragments.				
Additional Information	N/A				

Table 89 - requirement REQ-ADS-02 for Adaptable Distributed Storage

	Id	Level of detail	Type	Actor	Priority
	REQ-ADS-03	System	FUNC	Developer	DES
Name	Solve the non-linear resource allocation problem to suggest alternative deployment of the data fragments.				
Description	According to the available resources for the deployment of the data nodes and the stored data set, along with its split points that define data fragments, there is a non-linear resource allocation problem for the optimal deployment of the data fragments.				
Additional Information	As a non-linear, the solution of the resource allocation problem requires exponential time to be solved, which is not acceptable for run-time requirements. The provided solution should take into account possible acceptable solutions that can solve the problem and improve the resource consumption, under a minimum time interval.				

Table 90 - requirement REQ-ADS-03 for Adaptable Distributed Storage

	Id	Level of detail	Type	Actor	Priority
	REQ-ADS-04	System	ENV	Developer	DES
Name	Be able to request additional resources from the infrastructure layer.				
Description	In case of overprovisioning of the resources, the adaptable distributed storage should be able to request additional resources from the infrastructure of BigDataStack.				
Additional Information	<p>As noted in REQ-ADS-02, the adaptable storage must identify data nodes that are overprovisioning, and using REQ-ADS-03, it can suggest different distribution of the data fragments. However, there might be cases that this is not possible due to the overprovision of the whole system, and in such case, a horizontal scale out must take place. The adaptable storage should request additional resources, and grant them, if they are available. The communication should be as follows:</p> <ul style="list-style-type: none"> - The adaptable storage requests an additional node with the specific requirements for resources. - The infrastructure responds if it can allocate additional resources for the storage. - The infrastructure informs the storage that the additional resources are now available. <p>This requirement also includes the need from the adaptable storage to inform the infrastructure that it can release resources that are not needed.</p>				

Table 91 - requirement REQ-ADS-04 for Adaptable Distributed Storage

	Id	Level of detail	Type	Actor	Priority
	REQ-ADS-05	System	ENV	Developer	OPT
Name	Being able to release resources and adapt if resources are deallocated from the infrastructure.				
Description	There might be cases where the whole infrastructure is overprovisioning there are no more resources to be allocated to tasks. Then, the infrastructure might decide to reduce the overall resources of specific components, in favour of others that might execute some critical operations, or they have biggest priority at that point. The adaptable storage engine should be listening to the infrastructure for such cases and adapt accordingly.				
Additional Information	Once the adaptable distributed storage receives a request to release some of its nodes, then it should inform if it is capable of doing so: releasing some the data nodes, might result to not have the required amount of storage available for the dataset. In such cases, it should be responding that this is not permitted, as this would lead to data loss. In case that this is permitted, then it should re-distribute its data load, and inform the infrastructure that the node is ready to be released.				

Table 92 - requirement REQ-ADS-05 for Adaptable Distributed Storage

	Id	Level of detail	Type	Actor	Priority
	REQ-ADS-06	System	ENV	Developer	DES
Name	Inform the re-deployment component regarding reconfigurations of the data fragments.				
Description	As it is up to the storage itself to decide its optimal configuration of its data load, the re-deployment component cannot be aware of possible reconfigurations, that might affect the overall deployment of an application. Therefore, the storage should inform the re-deployment component about these actions.				
Additional Information	A message should be sent just before the re-configuration takes place, along with the setup, so that the re-deployment component can be notified and not take into account possible outlier monitoring information coming from this subcomponent. During this time, the re-deployment component should not modify any deployments that rely on the data set that is being re-configured. When the reconfiguration is finished, the adaptable storage should notify the redeployment component again, in order for the latter to start looking on the new monitoring information and decide upon possible redeployment of existed applications as well.				

Table 93 - requirement REQ-ADS-06 for Adaptable Distributed Storage.

	Id	Level of detail	Type	Actor	Priority
	REQ-ADS-07	System	ENV	Developer	MAN

Name	Re-establish connectivity with the monitoring subcomponent when a horizontal scaling action takes place
Description	The adaptable storage engine exports its monitoring data to a specific place where the Prometheus, part of the monitoring subcomponent of BigDataStack can periodically pull and gather this information. Prometheus can be configured on where to pull this information upon its initialization. However, in cases of a runtime redeployment that takes place after a horizontal scaling action, information regarding the newly deployed nodes should also reach the monitoring component.
Additional Information	There should be a monitoring proxy of the adaptable storage that will take the responsibility to send monitoring information to the target component. This proxy should encapsulate the details of the underlying deployment. It should gather all information of the data nodes, reconfigure itself to take into account newly deployed data nodes, and send everything to the Prometheus.

Table 94 - requirement REQ-ADS-07 for Adaptable Distributed Storage

	Id	Level of detail	Type	Actor	Priority
	REQ-ADS-08	System	ENV	Developer	MAN
Name	Enable a deployment of the data node component using Kubernetes				
Description	As the infrastructure of BigDataStack uses Kubernetes for deploying the various application/platform components, the adaptable distributed engine must be able to deploy and configure additional data nodes via this technology.				
Additional Information	N/A				

Table 95 - requirement REQ-ADS-08 for Adaptable Distributed Storage

	Id	Level of detail	Type	Actor	Priority
	REQ-SDAF-01	Software	FUNC	Developer	MAN
Name	Provide access to data stores via a single and common interface.				
Description	BigDataStack includes two different data stores: the LeanXcale relational data store and IBM object store. The dataset can be fragmented and distributed over the two data stores (historic data being moved to object store). However, the application should be kept unaware of these internal data transfers. The application needs a common interface to submit queries, without having to specify where the data is stored.				
Additional Information	A federation mechanism is required that will encapsulate the process of data retrieval from the two data stores. The LeanXcale access point will act as the federator between the relational and the Object Storage. The LeanXcale data base already provides a common JDBC interface for data connectivity. The federator will receive the query and execute it in both data stores. For the object store, the access would be via Spark SQL, which also provides a JDBC interface. The federator will take into consideration the operations that can				

	be supported in order to push down the operations accordingly. Regarding the relational store, all operations will be pushed down to the store. At the very end, the federator will merge the results and return back the result set. It shouldn't count data that appears in both data stores twice.
--	---

Table 96 - requirement REQ-SDAF-01 for Seamless Data Analytics

	Id	Level of detail	Type	Actor	Priority
	REQ-SDAF-02	System	DATA	Developer	MAN
Name	Move historical data from the relational data store to the object store.				
Description	Data ingested by the use cases will be stored into the relational datastore, as they are operational, in order to ensure data consistency in terms of ACID properties. After a configurable period of time, called the <i>freshness window</i> (which depends on the data set), the data becomes outdated and is no longer used by operational workloads. However, this historical data is still valuable and can be exploited by Big Data analytics algorithms. This data should be moved from the LeanXcale data base to the IBM object store.				
Additional Information	The LeanXcale data base provides a mechanism that allows to periodically produce a dumb snapshot of the modified data. This information will be transformed accordingly and will be pushed to an Apache Kafka queue. A Kafka based connector will, periodically pull this information and import the historical data to the object store.				

Table 97 - requirement REQ-SDAF-02 for Seamless Data Analytics

	Id	Level of detail	Type	Actor	Priority
	REQ-SDAF-03	Software	DATA	Developer	MAN
Name	Inform the LeanXcale data store when data are imported to the object store.				
Description	When data are pushed to the Apache Kafka queue, the LeanXcale data base can drop them. However, due to the asynchronous design, the LeanXcale data base cannot know when the data has been made available to the object store. As a result, the object store must inform the LeanXcale data base regarding the successful insertion of the data, so that the LeanXcale data base can safely drop these data.				
Additional Information	One possible solution to deal with this requirement will be the introduction of marking the data to be transferred to the object store by additional timestamps. Data that is being flushed and exported to the Kafka queue can be marked that way, so that later on, the object store can inform the LeanXcale data base that this bunch of data has been successfully imported. By doing so, the <i>federator</i> component can push down operations accordingly, and only request specific data from the underlying data stores. Data that are known to the LeanXcale data base that has been previously uploaded to the object store, will not be retrieved by the <i>federator</i> and can be safely discarded by the <i>vacuum</i> process of the LeanXcale data base.				

Table 98 - requirement REQ-SDAF-03 for Seamless Data Analytics

	Id	Level of detail	Type	Actor	Priority
	REQ-SDAF-04	Software	DATA	Developer	OPT
Name	Optimize query execution				
Description	The federator receives a query and executes it into the different stores. The federator will be based on the LeanXscale query engine. The latter provides a query optimizer, which allows it to examine the different execution plans that can be produced in order to execute a query. However, it has been implemented to evaluate plans to be executed locally. It should be extended in order to take into consideration the operations that can be pushed down to the object store, and whether or not it is worth for an operator to be pushed down, according to the response time of the execution from Spark SQL, the amount of data that will be retrieved to the federator etc.				
Additional Information	N/A				

Table 99 - requirement REQ-SDAF-04 for Seamless Data Analytics

	Id	Level of detail	Type	Actor	Priority
	REQ-SDAF-05	Software	DATA	Developer	OPT
Name	Optimize access to Object Storage.				
Description	<p>In order to perform analytics efficiently on Object Storage, a client-side caching/acceleration layer is needed. This is critical for a hybrid cloud scenario, where some of the customer data is on premise (potentially the LeanXscale data base and Spark) and some is in the cloud (potentially IBM COS). In such a scenario, when performing analytics, data needs to move from COS to Spark across the WAN, therefore minimizing the amount of data movement when part of the data is retrieved multiple times is of utmost importance.</p> <p>A similar scenario involves multi-cloud, where a dataset may be distributed among more than one cloud, also requiring data transfer across the WAN for the purposes of analytics.</p>				
Additional Information	This complements data skipping and data layout techniques to further reduce the KPI measuring the number of bytes sent from Object Storage to Spark.				

Table 100 - requirement REQ-SDAF-05 for Seamless Data Analytics

	Id	Level of detail	Type	Actor	Priority
	REQ-DQAI-01	Software	DATA	Developer	MAN
Name	Infer data schema				
Description	The Data Quality Assessment and Improvement module should be able to infer a data schema for a given dataset. The data schema should describe				

	the name of each field, its type (e.g., integer, floating number, string, etc.) and its presence (mandatory or optional).
Additional Information	The schema will be stored in a sharable format (e.g. JSON document) and the system should be able to recall it and compare a new dataset against it, to discover lurking anomalies.

Table 101 - requirement REQ-DQAI-01 for Data Quality Assessment & Improvement

	Id	Level of detail	Type	Actor	Priority
	REQ-DQAI-02	Software	DATA	Developer	OPT
Name	Create schema environments				
Description	The Data Quality Assessment and Improvement module should be able to different schema environments, for example for training and serving datasets.				
Additional Information	This way the user should be able to feed the system slightly different datasets for training, validation, testing and serving purposes.				

Table 102 - requirement REQ-DQAI-02 for Data Quality Assessment & Improvement

	Id	Level of detail	Type	Actor	Priority
	REQ-DQAI-03	Software	DATA	Developer	MAN
Name	Check data anomalies				
Description	The Data Quality Assessment and Improvement module should be able to compare a given dataset to a restored data schema and produce a data anomaly assessment, e.g., discovering missing columns or wrong data types.				
Additional Information	The final analysis will be stored in a sharable format (e.g. JSON document) and the system should be able to recall it and present it to the user to act.				

Table 103 - requirement REQ-DQAI-03 for Data Quality Assessment & Improvement

	Id	Level of detail	Type	Actor	Priority
	REQ-DQAI-04	Software	DATA	Developer	MAN
Name	Check data skew and drift				
Description	The Data Quality Assessment and Improvement module should be able to detect skew between training and serving data, as well as drift between training datasets in different model versions.				
Additional Information	The final analysis will be stored in a sharable format (e.g. JSON document) and the system should be able to recall it and present it to the user to act.				

Table 104 - requirement REQ-DQAI-04 for Data Quality Assessment & Improvement

	Id	Level of detail	Type	Actor	Priority
	REQ-DQAI-05	Software	DATA	Developer	DES

Name	Detect data source deterioration
Description	The Data Quality Assessment and Improvement module should be able to detect if a data source, e.g., an IoT sensor, is not malfunctioning and always emits corrupted data.
Additional Information	The final analysis will be stored in a sharable format (e.g. JSON document) and the system should be able to recall it and present it to the user to act.

Table 105 - requirement REQ-DQAI-05 for Data Quality Assessment & Improvement

	Id	Level of detail	Type	Actor	Priority
	REQ-DQAI-06	Software	DATA	Developer	MAN
Name	Detect unexpected range of values				
Description	The Data Quality Assessment and Improvement module should be able to detect unexpected range of values in any field of the dataset, given a specific context, i.e., the values in neighbouring columns.				
Additional Information	The final analysis will be stored in a sharable format (e.g. JSON document) and the system should be able to recall it and present it to the user to act.				

Table 106 - requirement REQ-DQAI-06 for Data Quality Assessment & Improvement

	Id	Level of detail	Type	Actor	Priority
	REQ-RD-01	Stakeholder	FUNC	Developer	ENH
Name	Global event tracker connection				
Description	A connection to the Global Event Tracker (GET) is needed for the Predictive & Process Analytics component.				
Additional Information	The information stored in GET is crucial to the implementation of this module.				

Table 107 - requirement REQ-RD-01 for Predictive & Process Analytics

	Id	Level of detail	Type	Actor	Priority
	REQ-RD-02	Stakeholder	FUNC	Developer	ENH
Name	Connection to the Process Modelling Framework				
Description	A connection between this component and the Process Modelling Framework needs to be established, so information can be sent and received.				
Additional Information	The recommendations made by this component will be in real time, as the Business Analyst – Data engineer is modelling the process.				

Table 108 - requirement REQ-RD-02 for Predictive & Process Analytics

	Id	Level of detail	Type	Actor	Priority
	REQ-RD-03	Stakeholder	FUNC	Developer	ENH
Name	Data pre-processing				

Description	The data ingested by this component needs to be in an eXtensible Event Stream ³² (XES) file format. A tool is created, depending on the format of the Global Event Tracker.
Additional Information	The XES standard defines a grammar for a tag-based language whose aim is to provide designers of information systems with a unified and extensible methodology for capturing systems behaviours by means of event logs and event streams is defined in the XES standard. An XML Schema describing the structure of an XES event log/stream and an XML Schema describing the structure of an extension of such a log/stream are included in this standard. Moreover, a basic collection of so-called XES extension prototypes that provide semantics to certain attributes as recorded in the event log/stream is included in this standard.

Table 109 - requirement REQ-RD-03 for Predictive & Process Analytics

	Id	Level of detail	Type	Actor	Priority
	REQ-RD-04	Stakeholder	FUNC	Developer	ENH
Name	ProM framework				
Description	ProM is an extensible framework that supports a wide variety of process mining techniques in the form of plug-ins.				
Additional Information	The process mining techniques used will be utilized to derive metrics of the event log, to create the semantics needed between events for the recommendation process.				

Table 110 - requirement REQ-RD-04 for Predictive & Process Analytics

	Id	Level of detail	Type	Actor	Priority
	REQ-CEP-01	System	FUNC	Developer	MAN
Name	Manage data from different sources to generate alarms if required.				
Description	The CEP will process data on the fly coming from sensors. Each sensor sends events each minute. CEP will analyse the data according to a set of rules and generate alarms.				
Additional Information	The processing will be both stateless and over windows of time and number of events.				

Table 111 - requirement REQ-CEP-01 for CEP

	Id	Level of detail	Type	Actor	Priority
	REQ-CEP-02	System	FUNC	Developer	MAN
Name	Send alarms and data from each node of the distributed environment to the data centre.				

³² <http://www.xes-standard.org/>

Description	Once metrics have been analysed, the CEP will send the alarms and the data to a central location (data centre).
Additional Information	The CEP will run on nodes of a geographically distributed environment.

Table 112 - requirement REQ-CEP-02 for CEP

	Id	Level of detail	Type	Actor	Priority
	REQ-CEP-03	System	PERF	Developer	MAN
Name	Data from distributed nodes is aggregate at a central location.				
Description	Further processing over remote data will be done at a central location.				
Additional Information	The CEP processing will scale to tens streams coming from different remote sources.				

Table 113 - requirement REQ-CEP-03 for CEP

	Id	Level of detail	Type	Actor	Priority
	REQ-CEP-04	Stakeholder	PERF	Developer	ENH
Name	Store data on the data store				
Description	The CEP will store the data at the rate is being produced.				
Additional Information	Both CEP and LX will run at the same location.				

Table 114 - requirement REQ-CEP-04 for CEP

8 Dimensioning, Modelling & Interaction Services Requirements

To facilitate the understanding of the design as well as the challenges addressed by this set of services, the requirements related to them were brought into D5.1 and literally included from this section. Therefore, note the following requirement tables also appear as-is in such deliverable for the reader's convenience.

	Id ³³	Level of detail ³⁴	Type ³⁵	Actor ³⁶	Priority ³⁷
	REQ-PMF-01	System and Software	USE	ROL-04	MAN
Name	UI/UX experience				
Description	The system should guide the users to complete the business diagram / flow with easy steps. It should clearly indicate what connections – interactions are possible and provide comprehensive error messages.				
Additional Information	N/A				

Table 115 – System Requirement (1) for Process Modelling Framework

	Id	Level of detail	Type	Actor	Priority
	REQ-PMF-02	System and Software	FUNC	ROL-04	MAN
Name	Multi-user support				
Description	Multiple users should be able to use the Process Modelling Framework and create diagrams at the same time. It should also support different roles: business analysts and data analysts. A business analyst will define a process in a higher level and a data analyst will provide the concrete implementations				
Additional Information	N/A				

Table 116 – System Requirement (2) for Process Modelling Framework

³³**Identifier:** To be used in D2.2 to allow for the correct traceability of requirements.

³⁴**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).

³⁵**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).

³⁶**Actor:** It needs to be either one of the BigDataStack platform roles identified in Section 5 or a system actor, e.g. another component or service.

³⁷**Priority:** Requirements can have different priorities: MAN (mandatory requirement), DES (desirable requirement), OPT (optional requirement), ENH (possible future enhancement).

	Id	Level of detail	Type	Actor	Priority
	REQ-PMF-03	System and Software	FUNC	Business Analyst	MAN
Name	Process workflow creation				
Description	A business analyst should be able to create a process workflow in a higher level. The analyst will select nodes from a catalogue and using a drag-and-drop interface will link them together to create the flow.				
Additional Information	N/A				

Table 117 – System Requirement (3) for Process Modelling Framework

	Id	Level of detail	Type	Actor	Priority
	REQ-PMF-04	System and Software	FUNC	Data Analyst	MAN
Name	Process workflow configuration				
Description	The data analyst should be able to configure a process workflow with all the required details. The data analyst will set up the nodes parameters and define the rules for moving from one node to another.				
Additional Information	N/A				

Table 118 – System Requirement (4) for Process Modelling Framework

	Id	Level of detail	Type	Actor	Priority
	REQ-PMF-05	System and Software	FUNC	Data Analyst	MAN
Name	Process workflow export				
Description	The data analyst should be able to export the process workflow in BigDataStack format.				
Additional Information	The default format of the export will be in JSON. It will include information regarding the flows and their interconnections. Alternative export formats (YAML, Dockerfile) will be considered based on the requirements of other components. The user should be able to select the appropriate export format.				

Table 119 – System Requirement (5) for Process Modelling Framework

	Id	Level of detail	Type	Actor	Priority
	REQ-PMF-06	System and Software	FUNC	Business Analyst	MAN
Name	Support for end-to-end (in terms of process workflow) objectives				
Description	The business analyst should be able to define end-to-end objectives. These objectives do not apply to a single process, but to the workflow as a whole.				

Additional Information	N/A
-------------------------------	-----

Table 120 – System Requirement (6) for Process Modelling Framework

	Id	Level of detail	Type	Actor	Priority
	REQ-PMF-07	System and Software	FUNC	Business Analyst	MAN
Name	Process constraints				
Description	The business analyst should be able to set apply constraints per node / process of the workflow				
Additional Information	N/A				

Table 121 – System Requirement (7) for Process Modelling Framework

	Id	Level of detail	Type	Actor	Priority
	REQ-PMF-08	System and Software	FUNC	Business Analyst	MAN
Name	Edge constrains				
Description	The business analyst should be able to apply constraints / parameters per edge (i.e. connections between processes of the workflow).				
Additional Information	N/A				

Table 122 – System Requirement (8) for Process Modelling Framework

	Id	Level of detail	Type	Actor	Priority
	REQ-DO-01	Stakeholder	FUNC	ROL-04	MAN
Name	Compatibility with output of Process Modelling				
Description	The Process Mapping component is able to process the output of Process Modelling, in order to select appropriate ML algorithm(s) for specific Process steps.				
Additional Information	This requirement practically ascertains that the two components (Process Modelling and Process Mapping) are compatible and that the output of the first can be consumed by the second.				

Table 123 – System Requirement (1) for Process Mapping

	Id	Level of detail	Type	Actor	Priority
	REQ-DO-02	Stakeholder	FUNC	ROL-04	MAN
Name	Extraction of metadata				
Description	Given a dataset, extract a set of metadata that is sufficient in order to discover similarities between datasets, in particular regarding the underlying data distributions and other statistical properties.				

Additional Information	The metadata should cover at least statistical and information-theoretic characterization of a given dataset.
-------------------------------	---

Table 124 – System Requirement (2) for Process Mapping

	Id	Level of detail	Type	Actor	Priority
	REQ-DO-03	Stakeholder	FUNC	ROL-04	MAN
Name	Build and maintain a meta-knowledge repository				
Description	Collect and store information about datasets, metadata, and the performance of ML algorithms that have been executed on the datasets. This information is referred to as meta-knowledge, because it is essentially knowledge about the learning process. This meta-knowledge repository is going to be used for <i>meta-learning</i> , which is defined as the study of methods that exploit meta-knowledge to obtain efficient models and solutions by adapting machine learning processes.				
Additional Information	The meta-knowledge repository is augmented with information about the execution of ML algorithms on new datasets.				

Table 125 – System Requirement (3) for Process Mapping

	Id	Level of detail	Type	Actor	Priority
	REQ-DO-04	Stakeholder	FUNC	ROL-04	MAN
Name	ML algorithm selection				
Description	Given a machine learning task, a dataset, and a set of available ML algorithms that can handle the given task, select (or recommend) the subset of ML algorithms with best performance.				
Additional Information	It assumes the availability of a pool of ML algorithms (e.g., a ML library) and an execution environment for running ML algorithms on different datasets and evaluating their result quality.				

Table 126 – System Requirement (4) for Process Mapping

	Id	Level of detail	Type	Actor	Priority
	REQ-SY-DT-01	Software	FUNC	ROL-02, ROL-03	MAN
Name	Describe data mining and analysis processes through data workflows				
Description	Support for the description of data mining and analysis processes, interconnected to each other in terms of input/output data streams/objects. The corresponding metadata and an algorithms taxonomy for the categorisation of the analytic processes, type of data and connection details will be used to facilitate the description of individual nodes.				
Additional Information	The playbook must be represented in the form of a descriptor (e.g. through a yaml file) that can be incorporated into the Dimensioning Workbench as well as the Dynamic Orchestrator.				

Table 127 – System Requirement (1) for Data Toolkit

	Id	Level of detail	Type	Actor	Priority
	REQ-SY-DT-02	Software	FUNC	ROL-02, ROL-03	MAN
Name	Express data workflows through graphs using nodes and edges				
Description	Data workflows are 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 (i.e. edges). The links may include properties representing constraints, KPIs or objectives which are desirable at specific analytic stage.				
Additional Information	N/A				

Table 128 – System Requirement (2) for Data Toolkit

	Id	Level of detail	Type	Actor	Priority
	REQ-SY-DT-03	Software	FUNC	ROL-03	MAN
Name	Validate graph through chain-ability constraints				
Description	This requirement resolves chain-ability constraints through different nodes in the data workflows. The target is to produce a valid graph. This is the reason why a set of checks will be performed to meet these prerequisites. If these prerequisites are not met, the graph is not considered valid.				
Additional Information	N/A				

Table 129 – System Requirement (3) for Data Toolkit

	Id	Level of detail	Type	Actor	Priority
	REQ-SY-DT-04	Software	FUNC	ROL-03	MAN
Name	Link valid graphs with viable executables for Big Data analytic processes				
Description	This step links the graph with the actual executable image. In order to cope with the problem of vendor lock-in format of the executable the container format has been chosen. To this end, the actual container pulling will be performed.				
Additional Information	N/A				

Table 130 – System Requirement (4) for Data Toolkit

	Id	Level of detail	Type	Actor	Priority
	REQ-T5.1-PG-01	System and Software	FUNC	ROL-04	MAN
Name	Ingest Playbook				
Description	The Data Toolkit sends to the Pattern Generation a Playbook containing the graph of the user's application. The Pattern Generation receives the playbook and initiates creation of candidate deployment patterns.				

Additional Information	N/A
-------------------------------	-----

Table 131 – System Requirement (1) for Pattern Generator

	Id	Level of detail	Type	Actor	Priority
	REQ-T5.1-PG-02	System and Software	FUNC	ROL-04	MAN
Name	Load Hardware Directory (File)				
Description	To produce candidate deployment patterns, Pattern Generation needs to know what hardware is available to deploy the components of the user's application upon. Initial versions will load this information from a static file.				
Additional Information	N/A				

Table 132 – System Requirement (2) for Pattern Generator

	Id	Level of detail	Type	Actor	Priority
	REQ-T5.1-PG-03	System and Software	FUNC	ROL-04	MAN
Name	Load Hardware Directory				
Description	To produce candidate deployment patterns, Pattern Generation needs to know what hardware is available to deploy the components of the user's application upon.				
Additional Information	N/A				

Table 133 – System Requirement (3) for Pattern Generator

	Id	Level of detail	Type	Actor	Priority
	REQ-T5.1-PG-04	System and Software	FUNC	ROL-04	MAN
Name	Service-Hardware Mapping (1-1)				
Description	The main process in Pattern Generation is mapping the different components (services) to potentially suitable hardware. The first version of this functionality produces only 1-1 mappings, i.e. one service is mapped to one piece of hardware (e.g. machine).				
Additional Information	N/A				

Table 134 – System Requirement (4) for Pattern Generator

	Id	Level of detail	Type	Actor	Priority
	REQ-T5.1-PG-05	System and Software	FUNC	ROL-04	MAN

Name	Service-Hardware Mapping (1-M)
Description	The main process in Pattern Generation is mapping the different components (services) to potentially suitable hardware. The second version of this functionality produces only one to many mappings, i.e. one service can be mapped to multiple piece of hardware (e.g. spread over multiple machines). This may be advantageous in cases such as were a single 'big' machine is more expensive than multiple smaller machines.
Additional Information	N/A

Table 135 – System Requirement (5) for Pattern Generator

	Id	Level of detail	Type	Actor	Priority
	REQ-T5.1-PG-06	System and Software	FUNC	ROL-04	DES
Name	Service-Hardware Mapping (M-1/Pods)				
Description	The main process in Pattern Generation is mapping the different components (services) to potentially suitable hardware. The third version of this functionality produces only many to one mappings, i.e. multiple services can be co-located on a single piece of hardware. This may be advantageous when services perform high-volume data transfers that would be expensive over a network.				
Additional Information	N/A				

Table 136 – System Requirement (6) for Pattern Generator

	Id	Level of detail	Type	Actor	Priority
	REQ-SY-DW-01	System	PERF/ NONFUNC	ROL-02	MAN
Name	Response Time and Workload				
Description	The service provided by the data applications (e.g. recommender system) must have enough speed so consumers will not notice the time taken by the request. This implies that the Data Scientist should be able to dictate what are the required levels of QoS, selecting them from available metrics and appropriate levels for them.				
Additional Information	This requirement poses initially the feature of metric selection and insertion at the Data Toolkit layer, for the Data Scientist to express their desires. Then the annotated Playbook gets passed to the following components (primarily ADW). Inside the Application Dimensioning Workbench, an initial candidate solution set is created, its estimated QoS level is enriched and the solution set is returned to the Data Scientist for final selection. Workload features (e.g. maximum/average etc. number of concurrent users) should also be able to be specified in order for the system to estimate the anticipated QoS levels for the desired range of application level workload.				

	This indicates that per category of data service or data service+analytics function a suitable selection of workload and QoS metrics should be performed and supported across the system (including also other components like monitoring)
--	--

Table 137 – System Requirement (1) for ADW Core

	Id	Level of detail	Type	Actor	Priority
	REQ-SY-DW-02	System	NONFUNC / PERF	ROL-04	MAN
Name	Scalability and configurability of stress tests for load injection				
Description	The system should have knowledge of a mapping between workload and QoS levels of the data services and algorithms (in order also to support REQ-SY-DW-02). Therefore, it should be able to launch stress tests against the data services that can easily scale to support the client sizes needed. Furthermore, different parameters of workload should be able to be determined				
Additional Information	Given that different data services exist in the project ecosystem, different baseline benchmarking tools should be identified per case. Following their selection, they need to be configured based on the respective workload parameters and scaled based on an abstracted generic approach (e.g. Docker containerization and Docker swarm approach)				

Table 138 – System Requirement (2) for ADW Core

	Id	Level of detail	Type	Actor	Priority
	REQ-SY-DW-03	System	FUNC	ROL-04	MAN
Name	Dimensioning output				
Description	The Dimensioning workbench should provide a list of candidate dimensioning suggestions along with the expected QoS levels towards the ADS Deploy component (and eventually the Application Engineer role), for the former to filter them based on an extra set of criteria and the latter to perform the final selection.				
Additional Information	Upon reception of the playbook with the service graph, ADW needs to estimate QoS level based on the results obtained through REQ-SYS-DW-02 and populate the respective fields. The operation should be offered through a REST service interface for automating the process and hiding complexities.				

Table 139 – System Requirement (3) for ADW Core

	Id	Level of detail	Type	Actor	Priority
	REQ-SY-DW-04	System	FUNC	ROL-04	MAN
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. This applies to both actual runtime and benchmarking phase
Additional Information	Relevant Tools affected: Data services, application components, triple monitoring engine.

Table 140 – System Requirement (4) for ADW Core

	Id	Level of detail	Type	Actor	Priority
	REQ-SO- ADW-01	Software	FUNC	ROL-04	MAN
Name	Load injector dockerization				
Description	To support a generic load injection process as indicated by REQ-SY-DW-02, “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				

Table 141 – System Requirement (5) for ADW Core

	Id	Level of detail	Type	Actor	Priority
	REQ-SO- ADW-02	Software	FUNC	ROL-04	MAN
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. The same applies for the benchmarking runs, which should be based on the same format (even without the inclusion of the PM and Data Toolkits). All requirements needed for deploying the benchmarking environment should be described using this common agreed standard.				
Additional Information	N/A				

Table 142 – System Requirement (6) for ADW Core

	Id	Level of Type detail		Actor	Priority
	REQ-SO- ADW-03	Software	FUNC	ROL-04	MAN
Name	Representative nature of gathered data samples				
Description	In order to create representative and accurate performance models, dataset creation from benchmarking should take into account different conditions such as applied workloads, configuration aspects of the service, deployment options etc. In this way different bottlenecks may be examined, and the final decision making can be adapted per case of service usage.				
Additional Information	N/A				

Table 143 – System Requirement (7) for ADW Core

	Id	Level of Type detail		Actor	Priority
	REQ-SO- ADW-04	Software	FUNC	ROL-04	ENH
Name	Deployment time for stress tests				
Description	The overhead added by the benchmarking setup should be negligible and not included in the measurement process.				
Additional Information	Since the deployment phase is done in a containerized manner, the time used in instructions different than launching the benchmark or storing data should not be significant.				

Table 144 – System Requirement (8) for ADW Core

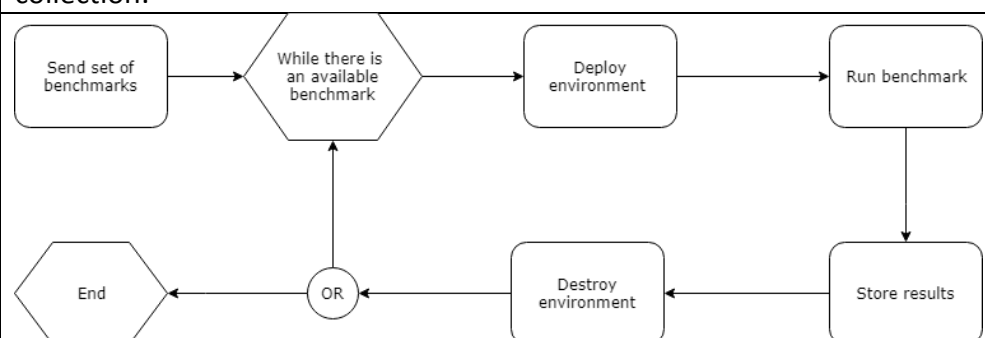
	Id	Level of Type detail		Actor	Priority
	REQ-SO- ADW-05	Software	FUNC	ROL-04	ENH
Name	Benchmarking Workflow implementation				
Description	During the benchmarking phase, there should be a controlled manner in which the various combinations described in REQ-SY-DW-02 and REQ-SO-ADW-03 are enforced during an automated process in order to ease data collection.				
Additional Information	 <pre> graph TD A[Send set of benchmarks] --> B{While there is an available benchmark} B --> C[Deploy environment] C --> D[Run benchmark] D --> E[Store results] E --> F[Destroy environment] F --> G((OR)) G --> B G --> H{End} </pre>				

Table 145 – System Requirement (9) for ADW Core

	Id	Level of detail	Type	Actor	Priority
	REQ-AV-01	System and Software	USE	ROL-04	MAN
Name	Interactive and Responsive UI				
Description	The system should provide an interactive UI that should adapt to different devices and displays in order to provide a proper operation of the solution and a good user experience.				
Additional Information	N/A				

Table 146 – System Requirement (1) for Adaptable Visualizations

	Id	Level of detail	Type	Actor	Priority
	REQ-AV-02	System and Software	FUNC	ROL-04	MAN
Name	Automatic graph selection				
Description	Appropriate graphs and reports should automatically be selected for different data sets.				
Additional Information	N/A				

Table 147 – System Requirement (2) for Adaptable Visualizations

	Id	Level of detail	Type	Actor	Priority
	REQ-AV-03	System and Software	FUNC	ROL-04	MAN
Name	Live data for different data sources				
Description	The system should be able to display live data obtained from application probes, resource probes and data operations probes.				
Additional Information	Adaptable selection of sources should be possible both in terms of application, resources or data operations, as well as in terms of the datasets selected and visualized per each one of these cases. Combinations should also be enabled.				

Table 148 – System Requirement (3) for Adaptable Visualizations

9 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.

9.1 Computing Resources Management

Public clouds (such as **GCP**³⁸ or **IBM Cloud**³⁹) 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).

OpenStack⁴⁰ 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**⁴¹, **Docker Swarm**⁴², **Amazon ECS**⁴³, or **Mesosphere Marathon**⁴⁴. 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.

Kubernetes is the most popular and widespread container orchestration platform; its main purpose is to schedule container (process) execution in a single machine or a cluster of machines and optimize (maximize the utilization) of the resources of the cluster. Everything in the platform is treated as an API object representing a concrete instance of a resource type in the cluster:

- Kubernetes Pods are the simplest object to be managed by Kubernetes. A pod is a group of related containers (processes) collocated on a Kubernetes cluster's node, sharing

³⁸ GCP (Google Cloud Platform). <https://cloud.google.com>

³⁹ IBM Cloud. <https://www.ibm.com/cloud/>

⁴⁰ Open Stack. <https://www.openstack.org/>

⁴¹ Kubernetes. <https://kubernetes.io/>

⁴² Docker Swarm. <https://docs.docker.com/engine/swarm/>

⁴³ Amazon ECS (Elastic Container Service). <https://aws.amazon.com/en/ecs/>

⁴⁴ Mesosphere Marathon. <https://mesosphere.github.io/marathon/>

- storage/network and a specification for how to run the containers⁴⁵. Replica sets are the number of pod instances running and represent the way to scale out/in deployments.
- Kubernetes Namespaces support 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 to isolate environments and better ensure Quality of Service (QoS).
 - Kubernetes Services describe how to access sets of pods and can describe ports and load-balancers.

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 [53]: The *request value* specifies the min value you will be guaranteed; the *limit value* specifies the max value you can consume (i.e., a CPU quota or a memory limit) and 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 [53]: *Best-Effort*, *Guaranteed* and *Burstable*.

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

9.2 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.

⁴⁵ <https://kubernetes.io/docs/concepts/workloads/pods/pod/>

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

9.3 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*⁴⁶ 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, guaranteeing the support of novel and intelligent network management mechanisms, applicable to data-intensive and network-intensive processes.

9.4 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

⁴⁶ <https://www.opennetworking.org/technical-communities/areas/specification/open-datapath/>

using such data-related synergies for all system orchestration aspects by defining machine-readable profile specifications (hereinafter referred to as Synergy Profile) for profiling homogeneously 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.

9.5 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 BigDataStack, 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 reasons⁴⁷:

- **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.

⁴⁷ Prometheus. <https://prometheus.io/>

- Push gateway: In case it's impossible to scrape metrics (using probe), metrics can be exposed to the Prometheus collector by this mechanism⁴⁸.

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 following⁴⁹:

- 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.

- 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. 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.
- 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.

⁴⁸ Prometheus <https://prometheus.io/>

⁴⁹ RabbitMQ <https://www.rabbitmq.com/>

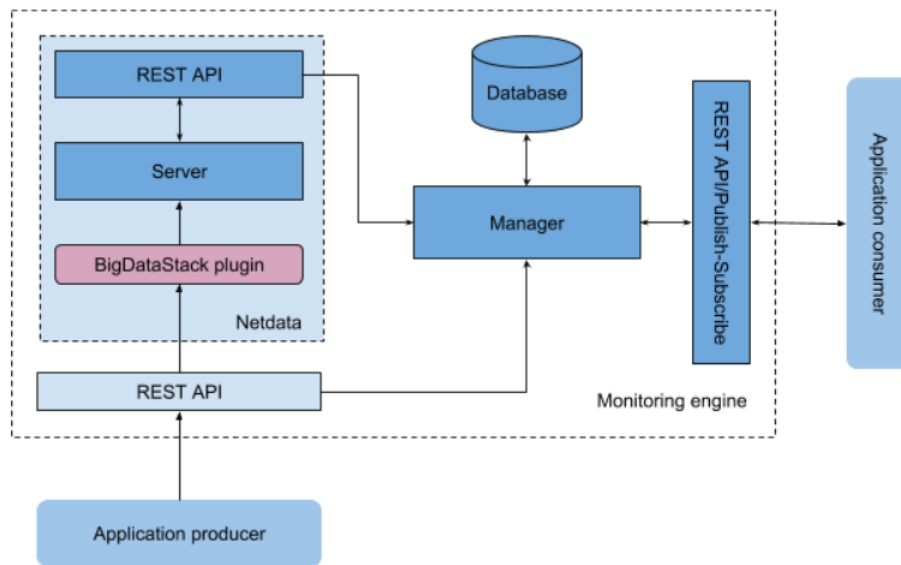


Figure 3. Netdata role in triple monitoring

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.

9.6 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.

9.6.1 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 MLlib.

9.6.2 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 Heat⁵⁰ and **AWS CloudFormation**⁵¹ are based on specific language descriptors. TOSCA⁵² specification provides a language to describe application service components and their relationships using a service topology. This specification also provides mechanisms to

⁵⁰ Openstack Heat. <https://wiki.openstack.org/wiki/Heat>,

⁵¹ AWS CloudFormation. <https://aws.amazon.com/cloudformation/>,

⁵² OASIS Topology and Orchestration Specification for Cloud Applications (TOSCA) , https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=tosca

describe management procedures that create or modify services using orchestration processes. Popular cloud tools such as OpenStack Heat conform to this specification.

jClouds⁵³ 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.

Docker⁵⁴ 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**⁵⁵ 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.

Kubernetes⁵⁶ 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**⁵⁷ 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**⁵⁸, supporting application SLA (service level agreement) specification, and **Serf**⁵⁹, offering a decentralized solution for cluster management, failure detection, and orchestration; **Cloudify**⁶⁰, **Chef**⁶¹ and **Puppet**⁶², on the other hand, are general purpose infrastructure configuration solutions which provide very flexible automated application lifecycle management.

⁵³ Apache jClouds. <https://jclouds.apache.org/>

⁵⁴ Docker. <https://www.docker.com/>

⁵⁵ Docker Swarm. <https://docs.docker.com/engine/swarm/>

⁵⁶ Kubernetes. <http://kubernetes.io/>

⁵⁷ Helm. <https://helm.sh/>

⁵⁸ Alien4Cloud. <http://alien4cloud.github.io/>

⁵⁹ Serf. <https://www.serf.io/>

⁶⁰ Cloudify. <http://docs.getcloudify.org/3.3.1/intro/what-is-cloudify/>

⁶¹ Chef. <https://www.chef.io/chef/>

⁶² Puppet. <https://puppet.com/>

9.7 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-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*⁶³, 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.

9.8 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.

9.9 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 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.

⁶³ <https://www.leanxcale.com/>

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]:

1. what to detect,
2. how to detect it, and
3. 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**⁶⁴. Several implementations like Yahoo!'s **TensorFlowOnSpark**⁶⁵ or **SparkNet**⁶⁶ 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].

9.10 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 Athena⁶⁷, 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 Parquet⁶⁸ and ORC⁶⁹. 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 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 csv⁷⁰.

⁶⁴ <https://www.tensorflow.org/>

⁶⁵ <https://github.com/yahoo/TensorFlowOnSpark>

⁶⁶ <https://github.com/amplab/SparkNet>

⁶⁷ <https://aws.amazon.com/athena/pricing/>

⁶⁸ <https://parquet.apache.org/>

⁶⁹ <https://orc.apache.org/>

⁷⁰ <http://blog.cloudera.com/blog/2016/04/benchmarking-apache-parquet-the-allstate-experience/>

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 post⁷¹).

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 project⁷². 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 presentation⁷³ 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 curves⁷⁴ such as Z-order curves (or Morton curves⁷⁵) 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 is 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 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).

⁷¹ <https://www.ibm.com/blogs/bluemix/2018/06/big-data-layout/>

⁷² <http://iostack.eu/>

⁷³ <https://databricks.com/session/using-pluggable-apache-spark-sql-filters-to-help-gridpocket-users-keep-up-with-the-jones-and-save-the-planet>

⁷⁴ http://www.mayr.informatik.tu-muenchen.de/konferenzen/Jass05/courses/2/Valgaerts/Valgaerts_paper.pdf

⁷⁵ “Z-order curves.” https://en.wikipedia.org/wiki/Z-order_curve

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 formats⁷⁶.

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 partitioning⁷⁷. 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.

9.11 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*⁷⁸ and later *Apache Flink*⁷⁹ 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 later⁸⁰. *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.

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

⁷⁶ Skipping-oriented partitioning for columnar layouts VLDB 2016 <https://dl.acm.org/citation.cfm?id=3025123>

⁷⁷ IOStack. <http://iostack.eu/deliverables/send/3-deliverables/31-d4-3-summary-and-demonstration-of-results>

⁷⁸ <http://storm.apache.org/>

⁷⁹ Apache Flink. <https://flink.apache.org/>

⁸⁰ Apache Spark. <https://spark.apache.org/streaming>

- 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.

9.12 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.

9.12.1 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*⁸¹, *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 MLlib* 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.

9.12.2 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.

9.13 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,

⁸¹ Spark SQL. <https://spark.apache.org/docs/2.2.0/sql-programming-guide.html#supported-hive-features>

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 API⁸² (a pluggable mechanism for accessing structured data through 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.

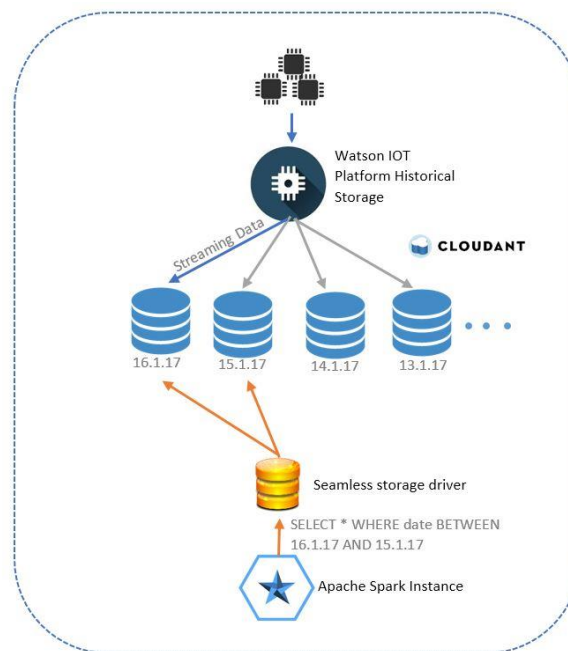


Figure 4. Data ingestion in Watson IoT Platform

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

⁸² <https://databricks.com/blog/2015/01/09/spark-sql-data-sources-api-unified-data-access-for-the-spark-platform.html>

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.

9.15 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-RED⁸³ 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.

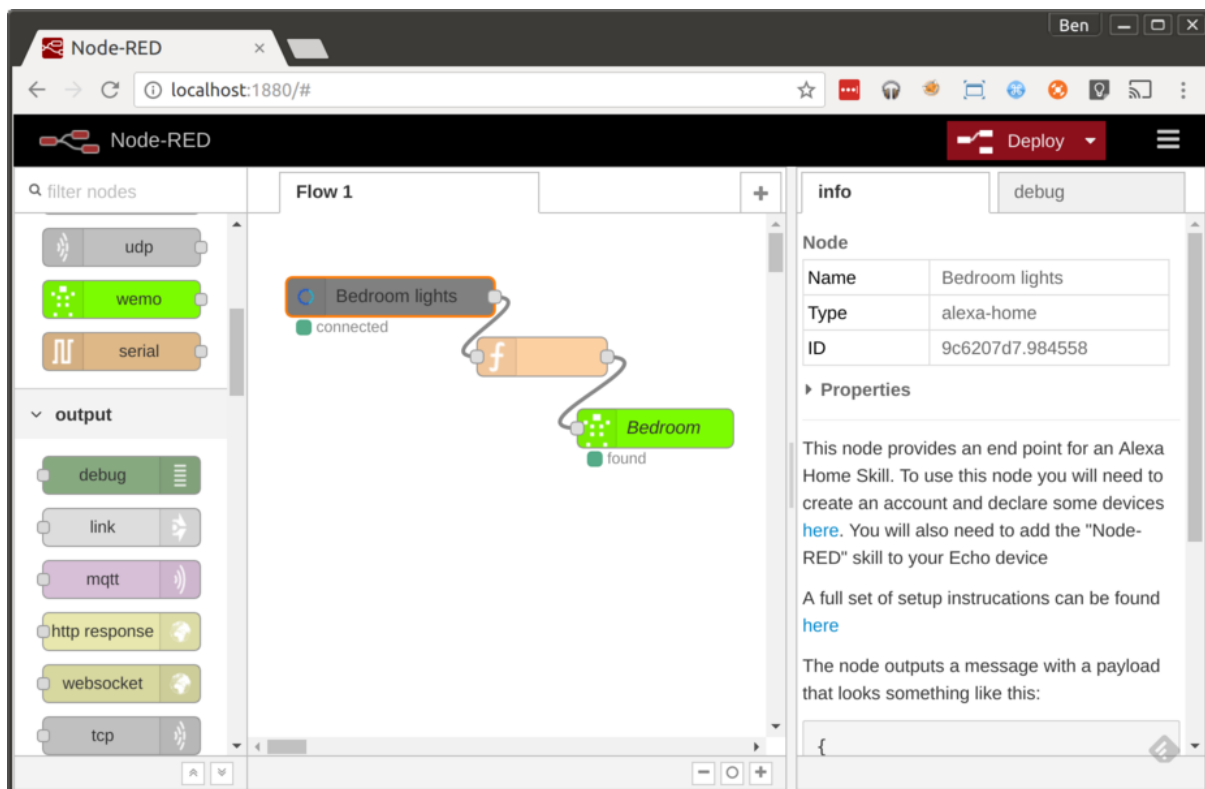


Figure 5. Node-red programming example

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.

⁸³ <https://nodered.org/>

9.15.1 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.

9.15.2 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**⁸⁴ (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.

9.16 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 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

⁸⁴ <https://www.drools.org/>

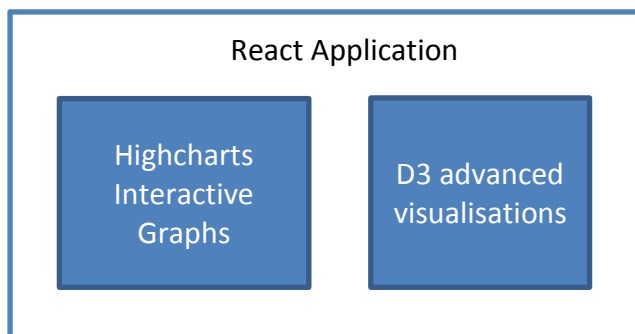
requirements and the software packages. Following, interconnection of such nodes leads to the production of the playbook (overall graph).

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**⁸⁵ that supports orchestration of micro-services-based process flows, **OpenCPU**⁸⁶ 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.

9.17 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*⁸⁷ 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*⁸⁸ 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*⁸⁹ library will be employed. For both Highcharts and D3.js ready-made components for React are available.

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

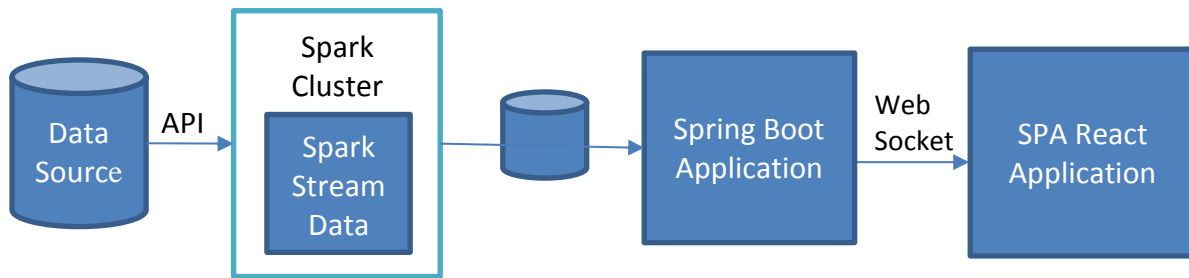
⁸⁵ <https://netflix.github.io/conductor/>

⁸⁶ <https://www.opencpu.org/>

⁸⁷ <https://reactjs.org/>

⁸⁸ <https://www.highcharts.com/>

⁸⁹ <https://d3js.org/>



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.

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 at <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.*, “Bigdancing: 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 at <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, “Adaptodb: Adaptive partitioning for distributed joins,” VLDB, 2017.
- [53] D. McPherson, “Managing Compute Resources with OpenShift/Kubernetes,” August 2016. Red Hat. <https://blog.openshift.com/managing-compute-resources-openshiftkubernetes/> [Accessed June 2018].