A Software Architecture for Extreme-ScaLe
Big-Data AnalyticS in Fog CompuTIng Ecosystems

D3.2 Refined requirements and integration plan

Version 1.0

Document Information

Contract Number	825473
Project Website	https://elastic-project.eu/
Contractual Deadline	M15, February 2020
Dissemination Level	PU
Nature	R
Author(s)	Maria A. Serrano, Elli Kartsakli (BSC)
Contributor(s)	Cristina Zubia, Marco González, Álvaro González, Xabier Pérez (IKL); Luis Miguel Pinho, Luis Nogueira (ISEP); Cristóvão Cordeiro (SIX); César Marin (ICE); Marco Merlini (THALIT); Anna Queralt (BSC)
Reviewer(s)	Cristóvão Cordeiro (SIX)
Keywords	software development ecosystem, software architecture, integration plan

Notices: *The ELASTIC project has received funding from the European Union's Horizon 2020 research and innovation programme under the grant agreement Nº 825473.*

Change Log

Version	Author	Description of Change
V0.1	Maria A. Serrano (BSC)	Initial Draft
V0.2	Elli Kartsakli (BSC)	Revised version, missing the integration sprints
V0.3	Elli Kartsakli (BSC)	Included integration sprints, missing two contributions
V0.4	Cristóvão Cordeiro, (SIX)	Revised version, missing one contribution
V1.0	BSC	Final Version. Ready to be submitted.

Table of contents

1. Executive Summary

This deliverable covers part of the work done during the second phase of the project (M7-M15) within WP3, mainly regarding Task 3.1 "Software architecture requirements specification", to reach milestone MS2.

Specifically, this deliverable provides a description of the work carried out within the frame of the integration plan during the second phase of the project. The deliverable will also provide a short summary of the software architecture requirements, since no modifications have been made with respect to their initial definition in D3.1 [1].

The second milestone of Task 3.1 has been carried out successfully and all objectives of MS2 have been reached and documented in this deliverable.

2. Requirements of the ELASTIC Software Architecture

The technical requirements of the ELASTIC Software Architecture have been extensively defined in Deliverable D3.1 [1]. In particular, after taking into consideration the main challenges for Big Data stakeholder ecosystems defined within the Strategic Research and Innovation Agenda (SRIA), a set of six business goals and four technical requirements has been identified and described in D3.1, and summarized for convenience in Table 1.

D3.1 provided a first description of the ELASTIC Software Development ecosystem and its software components to accomplish the aforementioned technical requirements and business goals. A redefined version of this ecosystem, still fulfilling the same set of requirements, is provided in Deliverable D3.3 "ELASTIC software architecture – First release" [2], where all modifications to the software components and interfaces with respect to the initial planning of D3.1 are fully described.

Table 1. Business goals and technical requirements of the ELASTIC Software Architecture

Business goals	Technical requirements
BG1. Interoperability	REQ-SWARCH-TR1. Increase Software Productivity
BG2. Easy-to-use	
BG3. Scalability and Performance	
BG4. Real-time Requirements	REQ-SWARCH-TR2. Fulfilment of Non-Functional Requirements
BG5. IT Infrastructure Cost Reduction	REQ-SWARCH-TR3. Enable Flexibility and Elasticity
BG6. Privacy and Security	REQ-SWARCH-TR4. Privacy and Security Mechanisms to Guarantee the Legal Framework REQ-SWARCH-TR2. Fulfilment of Non-Functional Requirements

3. Development and integration plan

D3.1 [1] introduced the development and integration plan defined for the ELASTIC project. In this regard, this section describes the work done since milestone MS1 to reach MS2, including both methodologies and tools.

3.1 Scrum-based methodology

The distributed nature of the several teams involved in the ELASTIC project led us to define a Scrum-based methodology [3] for the development of the project in D3.1. This strategy allowed us to define common goals together with a series of tasks to identify and execute the work to be done. The following subsections describe the sprints considered by each partner in the phase 2 of the project (months 7 to 16). Each team has worked in semi-isolation to develop the functionalities of the different components, based on the requirements and APIs defined during Phase 1. For that reason, the sprints are defined separately for the development of different technological components of the software architecture. Furthermore, sprints for the ADAS/NGAP use case are also presented, since scrum-based methodology has been also adopted for the preparation of the software and hardware platform of this specific use case.

3.1.1 Distributed data analytics platform

Distributed data analytics platform - Sprint 1	
Duration	June 1 – September 30, 2019
Goal	Definition of solution for distributed analytics
Planning	Define an architecture for analytics tools based on Spark.
Review	Completed
Retrospective	Architecture slightly out of the scope. Redesign

Distributed data analytics platform - Sprint 2	
Duration	October 1 – November 30, 2019
Goal	Redefinition of solution for a distributed data analytics platform (DDAP)
Planning	Define an architecture for a platform for distributed analytics. Solution to be supported by Spark and Druid.
Review	Completed
Retrospective	dataClay still has to be considered as part of the architecture

Distributed data analytics platform - Sprint 3	
Duration	December 1-31, 2019
Goal	First try at ingesting data using DDAP
Planning	First attempt at deploying Druid and ingesting some data available from the transport infrastructure.
Review	Completed
Retrospective	dataClay still have to be considered as part of the architecture

Distributed data analytics platform - Sprint 4	
Duration	January 1-31, 2020
Goal	Redesign of DDAP to include dataClay
Planning	Update architecture picture to include dataClay as part of DDAP, including deployment at transport infrastructure.
Review	Completed
Retrospective	-

Distributed data analytics platform - Sprint 5	
Duration	February 1, 2020 (ongoing)
Goal	Attempt to deploy DDAP at ICE and make it available to partners
Planning	Deployment of DDAP components, mainly Druid and Spark. Ingest data from transport infrastructure on a more regular basis.
Review	In progress
Retrospective	dataClay still not deployed

3.1.2 COMPSs orchestrator

COMPSs orchestrator - Sprint 1	
Duration	June 1-30, 2019
Goal	Design heuristics algorithms
Planning	Identify and adapt several heuristics to the scheduling process in COMPSs.
Review	Completed
Retrospective	-

COMPSs orchestrator - Sprint 2	
Duration	July 1-31, 2019
Goal	Implement adaptive feature for the COMPSs scheduler
Planning	Implement the designed algorithms, considering both online and offline timing analysis, to be integrated as an eligible COMPSs scheduler strategy.
Review	Completed
Retrospective	-

COMPSs orchestrator - Sprint 3	
Duration	August 1-31, 2019
Goal	Design the containerized deployment of a COMPSs workflow
Planning	Research on container technologies and approaches for containerized deployments and on how to adapt COMPSs to containerized deployment.
Review	Completed
Retrospective	-

COMPSs orchestrator - Sprint 4	
Duration	September 1-30, 2019
Goal	Design of the integration between the NFR tool and COMPSs by using dataClay
Planning	Develop the data model needed to communicate COMPSs and the NFR monitoring tool, to enable the monitoring of the available resources.
Review	Completed, a data model has been agreed among the involved partners
Retrospective	-

COMPSs orchestrator - Sprint 5	
Duration	October 1 – November 30, 2019
Goal	Migrate containerized deployment of COMPSs to a generic cloud scenario
Planning	Research on the most popular cloud platform and environments and how they work. Design an integration with COMPSs.
Review	Completed
Retrospective	-

COMPSs orchestrator - Sprint 6	
Duration	December 1-31, 2019
Goal	Implement the initial NFR tool integration with COMPSs
Planning	Implement the interaction between COMPSs and the NFR tool in order to update the list of available resources able to execute COMPSs tasks.
Review	Complete
Retrospective	This is the initial integration for MS2.

COMPSs orchestrator - Sprint 7	
Duration	February 1-29, 2020
Goal	Prepare documentation
Planning	Completion of the deliverables with the outcome of the previous tasks.
Review	Completed
Retrospective	-

3.1.3 dataClay – Distributed Storage

dataClay - Sprint 1	
Duration	June 1-30, 2019
Goal	Allow federated devices to leave the infrastructure
Planning	Provide an un-federation mechanism so that data can be kept if needed, but not synchronized with devices that left.
Review	Completed
Retrospective	Un-federation method as well as auxiliary methods to facilitate re-federation with other devices provided. Extensive testing of various foreseen situations with moving devices.

dataClay - Sprint 2	
Duration	July 1-31, 2019
Goal	Run dataClay on Jetson boards
Planning	Reduce footprint and improve performance of dataClay.
Review	Completed
Retrospective	Extensive testing of a dataClay instance in isolation and federated with other instances

dataClay - Sprint 3	
Duration	August 1-31, 2019
Goal	Improve deployment with Docker
Planning	Simplify Docker compose, minimize ports exposed, add health-check, move configuration variables from file to Docker compose.
Review	Completed
Retrospective	-

dataClay - Sprint 4	
Duration	September 1-30, 2019
Goal	Dockerize dataClay management functionalities
Planning	Containerize the dataClay command-line utility for model and user management so that Java or Python don't need to be installed in the client machine.
Review	Completed
Retrospective	-

dataClay - Sprint 5	
Duration	September 1 – October 31, 2019
Goal	Facilitate synchronization to the developer
Planning	Provide a pre-defined synchronization mechanism that can be incorporated to the necessary classes in the model, by importing a class (Java) or inheriting from a mixin (Python) already implementing the behavior.
Review	Completed
Retrospective	-

dataClay - Sprint 6	
Duration	October 1 – November 30, 2019
Goal	Provide an integrated deployment of COMPSs and dataClay
Planning	Facilitate deployment of COMPSs and dataClay using dockers, and develop examples of applications using both tools simultaneously so that other partners can easily use them in combination.
Review	Completed
Retrospective	-

dataClay - Sprint 7	
Duration	December 1-31, 2019
Goal	Publish dataClay in Maven central repository
Planning	Facilitate the use of dataClay from Java applications.
Review	Completed
Retrospective	-

dataClay - Sprint 8	
Duration	December 1, 2019 – January 31, 2020
Goal	Design integration with the DDAP
Planning	Design how the different tools involved (COMPSs, Spark, Druid and dataClay) can be integrated to provide the appropriate Distributed Data Analytics functionalities required by the use cases.
Review	Completed
Retrospective	Done in collaboration with ICE

dataClay - Sprint 9	
Duration	January 1 - February 29, 2020
Goal	Improve performance of storing new objects and not limit their size
Planning	Optimize the workflow by reducing interactions between dataClay components, removing unnecessary functionality, and using more efficient data structures. Enable storage of objects as big as allowed by the memory of the machine.
Review	Completed
Retrospective	-

dataClay - Sprint 10	
Duration	February 1-29, 2020
Goal	Implement a dataClay model for the integration of the NFR tool and COMPSs
Planning	Initial version of the model agreed by the involved components so that they can share the control data required to perform their functions. Sample NFR-inspired application provided as a guide.
Review	Completed
Retrospective	-

3.1.4 Energy and time non-functional requirements – NFR tool

Energy and time non-functional requirements - Sprint 1	
Duration	June 1-30, 2019
Goal	Timing and CPU load tool analysis
Planning	Research on the generally available Linux tools to determine execution times (per process) and CPU load (per process and per node).
Review	Completed
Retrospective	-

Energy and time non-functional requirements - Sprint 2	
Duration	July 1-31, 2019
Goal	Development of execution time and CPU load probes
Planning	Using selected Linux tools from previous analysis, develop software probes that report the execution times of processes and CPU load (per process and per node).
Review	Completed
Retrospective	-

Energy and time non-functional requirements - Sprint 3	
Duration	September 1 – October 31, 2019
Goal	Power and energy consumption tool analysis
Planning	Research on the generally available Linux tools and techniques to determine/infer power and energy consumption (per process and/or per node) and temperature (per core and per node).
Review	Completed
Retrospective	-

Energy and time non-functional requirements - Sprint 4	
Duration	November 1-30, 2019
Goal	Development of power and energy consumption probes
Planning	Using selected Linux tools from previous analysis, develop software probes that report the power and energy consumption (per process and/or per node).
Review	Completed
Retrospective	-

Energy and time non-functional requirements - Sprint 5	
Duration	December 1, 2019 – January 31, 2020
Goal	Integration of probes with NFR tool
Planning	Development of an NFR tool that based on detected execution time and CPU load NFR violations determines an ELASTIC redeployment of tasks. Development of an NFR tool that based on detected power and energy consumption NFR violations determines an ELASTIC redeployment of tasks.
Review	Completed
Retrospective	-

Energy and time non-functional requirements - Sprint 6	
Duration	February 1-29, 2020
Goal	Integration of NFR tool with COMPs
Planning	Integration of NFR tool with the COMPs orchestrator. Decisions computed by the NFR tool are made available in dataClay service, to the application orchestrator (COMPs).
Review	Completed
Retrospective	-

3.1.5 Communication and security non-functional requirements – NFR tool

Communication and security non-functional requirements- Sprint 1	
Duration	June 1-30, 2019
Goal	Requirements analysis
Planning	Definition of tasks based on the agreed requirements and research on the tools to be used.
Review	Completed
Retrospective	-

Communication and security non-functional requirements - Sprint 2	
Duration	July 1-31, 2019
Goal	Testing of different available tools
Planning	Perform tests on different hardware in order to check the optimal performance of the solutions.
Review	Completed
Retrospective	-

Communication and security non-functional requirements - Sprint 3	
Duration	September 1-30, 2019
Goal	Choice and validation of the analyzed tools
Planning	Decision of the best solution based on proofs of concept in the proposed hardware.
Review	Completed
Retrospective	-

Communication and security non-functional requirements - Sprint 4	
Duration	October 1-31, 2019
Goal	Containerization of OpenSCAP (open source Security Content Automation Protocol)
Planning	Adapt the available OpenSCAP container images to the specific requirements of the project.
Review	Completed
Retrospective	-

Communication and security non-functional requirements - Sprint 5	
Duration	November 1-30, 2019
Goal	Validation of OpenSCAP in the Jetson TX2
Planning	Use of the OpenSCAP tool and consider different software environments.
Review	Completed
Retrospective	-

Communication and security non-functional requirements - Sprint 6	
Duration	December 1-31, 2019
Goal	Implementation of cost algorithm for NFR monitor
Planning	Study and proposition of a cost algorithm, quantifying the performance parameters restricting the communications.
Review	Completed
Retrospective	-

Communication and security non-functional requirements - Sprint 7	
Duration	January 1 2020 (ongoing)
Goal	Validation of security tools with dataClay
Planning	Verification of the functionality of dataClay with regard to the security analysis performed by OpenSCAP.
Review	In progress
Retrospective	-

Communication and security non-functional requirements - Sprint 8	
Duration	February 1-29, 2020
Goal	Documentation
Planning	Completion of the deliverable documents with the outcome of the previous tasks.
Review	Completed
Retrospective	-

3.1.6 Communication middleware – Fog computing architecture

Communication middleware - Sprint 1	
Duration	June 1-30, 2019
Goal	Requirements analysis
Planning	Definition of tasks based on the agreed requirements and research on the tools to be used.
Review	Completed
Retrospective	-

Communication middleware - Sprint 2	
Duration	July 1-31, 2019
Goal	Information classification considering the use case scenarios
Planning	From the use case scenarios, define the data that will be transmitted and the destination of the data flow.
Review	Completed
Retrospective	-

Communication middleware - Sprint 3	
Duration	September 1-30, 2019
Goal	Definition of data streams
Planning	Classification of the different data streams that will be received and grouping as a function of their requirements.
Review	Completed
Retrospective	-

Communication middleware - Sprint 4	
Duration	October 1-31, 2019
Goal	Assignation of network interfaces for communication
Planning	Development of a decision table considering a network interface for each data stream.
Review	Completed
Retrospective	-

Communication middleware - Sprint 5	
Duration	November 1-30, 2019
Goal	Definition of communications middleware
Planning	Proposition of the communications middleware architecture and differentiation between the different phases of the project.
Review	Completed
Retrospective	-

Communication middleware - Sprint 6	
Duration	December 1-31, 2019
Goal	Design of interoperable data model
Planning	Design of the initial data model to be used by the edge/fog node for the communication between nodes.
Review	Completed
Retrospective	-

Communication middleware - Sprint 7	
Duration	January 1-31, 2020
Goal	Implementation of the interoperable data model
Planning	Implementation of the basic data msgType of the initial data model in KonnektBox.
Review	Completed
Retrospective	-

Communication middleware - Sprint 8	
Duration	February 1-28, 2020
Goal	Documentation
Planning	Completion of the deliverable documents with the outcome of the previous tasks.
Review	Completed
Retrospective	-

3.1.7 KonnektBox – Fog computing architecture

KonnektBox - Sprint 1	
Duration	June 1-30, 2019
Goal	Requirements analysis
Planning	Definition of tasks based on the agreed requirements and research on the tools to be used.
Review	Completed
Retrospective	-

KonnektBox - Sprint 2	
Duration	July 1-31, 2019
Goal	KonnektBox integration with Nvidia Jetson TX2
Planning	Installation and configuration of a KonnektBox in an Nvidia Jetson TX2 evaluation board.
Review	Completed
Retrospective	-

KonnektBox - Sprint 3	
Duration	September 1-30, 2019
Goal	4G modem support
Planning	Add support for LTE/4G modems in KonnektBox (control and monitoring of the connection).
Review	Completed
Retrospective	-

KonnektBox - Sprint 4	
Duration	October 1-31, 2019
Goal	KonnektBox data pipelines
Planning	Add service to deploy custom data pipelines inside a KonnektBox (with custom functions to perform data transformation and custom export clients)
Review	Completed
Retrospective	-

KonnektBox - Sprint 5	
Duration	November 1-30, 2019
Goal	KonnektBox-Nuvla integration
Planning	Integration of KonnektBox in Sixsq Nuvla.io platform to remotely deploy services and monitor KonnektBox.
Review	Completed
Retrospective	-

KonnektBox - Sprint 6	
Duration	December 1-31, 2019
Goal	Design of interoperable data model
Planning	Design of the initial data model to be used by the edge/fog node for the communication between nodes.
Review	Completed
Retrospective	-

KonnektBox - Sprint 7	
Duration	January 1-31, 2020
Goal	Integration of dataClay component
Planning	Development of dataClay client REST API and integration of dataClay REST component inside KonnektBox data pipeline.
Review	Completed
Retrospective	-

KonnektBox - Sprint 8	
Duration	February 1-29, 2020
Goal	Documentation
Planning	Completion of the deliverable documents with the outcome of the previous tasks.
Review	Completed
Retrospective	-

3.1.8 Nuvla/NuvlaBox – Fog computing architecture

Nuvla/NuvlaBox - Sprint 1	
Duration	June 1-30, 2019
Goal	Ability to register an existing container infrastructure into Nuvla
Planning	Develop the data models, API workflows and GUI to allow users to register new CaaS infrastructure in Nuvla
Review	Completed
Retrospective	-

Nuvla/NuvlaBox - Sprint 2	
Duration	July 1-31, 2019
Goal	Create and release the first container-based version of the NuvlaBox
Planning	Develop set of microservices that are capable of turning any Docker-compatible device into an edge device that can be managed from Nuvla
Review	Completed
Retrospective	-

Nuvla/NuvlaBox - Sprint 3	
Duration	September 1-30, 2019
Goal	Deploy the NuvlaBox into an NVidia Jetson TX2
Planning	Check that the NuvlaBox can be deployed into the Use Cases devices (Jetson TX2) and make use of GPUs via Docker.
Review	Completed
Retrospective	-

Nuvla/NuvlaBox - Sprint 4	
Duration	October 1-31, 2019
Goal	Deploy the NuvlaBox into an NVidia Jetson TX2
Planning	Check that the NuvlaBox can be deployed into the use cases devices (Jetson TX2) and make use of GPUs via Docker.
Review	Completed
Retrospective	-

Nuvla/NuvlaBox - Sprint 5	
Duration	November 1-30, 2019
Goal	Add Nuvla support for Kubernetes infrastructures
Planning	Apart from Docker CaaS infrastructures, let users also register Kubernetes CaaS and deploy Kubernetes apps.
Review	Completed
Retrospective	-

Nuvla/NuvlaBox - Sprint 6	
Duration	December 1, 2019 – February 29, 2020
Goal	Add Data Gateway to NuvlaBox
Planning	Implement a data routing mechanism to allow automatic ingestion and digestion of raw sensor data, via MQTT.
Review	Completed
Retrospective	-

Nuvla/NuvlaBox - Sprint 7	
Duration	January 15-31, 2020
Goal	Deploy dataClay into a NuvlaBox, from Nuvla
Planning	Take the dataClay compose file, register it as a new Nuvla application and deploy it to a NuvlaBox. Check successful installation of dataClay.
Review	Completed
Retrospective	Had to convert the DataClay compose file to be Docker Swarm compatible

Nuvla/NuvlaBox - Sprint 8	
Duration	January 1 2020 (ongoing)
Goal	Provide Nuvla GUI for managing NuvlaBox Data Gateway
Planning	Implement the API calls to NuvlaBox such that users can turn on/off the data routing, and also define the data models.
Review	In progress
Retrospective	-

Nuvla/NuvlaBox - Sprint 9	
Duration	February 1, 2020 (ongoing)
Goal	Allow application deployments with private Docker registries
Planning	Develop authentication models and workflow for user applications that need to pull Docker images from private registries.
Review	In progress
Retrospective	-

Nuvla/NuvlaBox - Sprint 10	
Duration	February 1, 2020 (ongoing)
Goal	Collect all fog metrics necessary for the NFR tool
Planning	Extend the existing fog manager/telemetry inside the NuvlaBox to also publish the metrics needed by the NFR tool.
Review	In progress
Retrospective	-

Nuvla/NuvlaBox - Sprint 11	
Duration	February 1-29, 2020
Goal	Documentation
Planning	Completion of the deliverable documents with the outcome of the previous tasks.
Review	Completed
Retrospective	-

3.1.9 Software architecture validation for the use cases

Software architecture validation - Sprint 1	
Duration	October 1 – November 30, 2019
Goal	Platform
Planning	Multi-platform delivery environment: Linux OS, toolchain, deployment, tooling of software
Review	Completed
Retrospective	-

Software architecture validation - Sprint 2	
Duration	December 1 - January 31, 2019
Goal	Software architecture review
Planning	Software architecture review for optimization and software preparation for future enhancements
Review	Completed
Retrospective	-

3.1.10 Use case related sprints

Scrum-based methodology has also been adopted to develop the software and hardware components of the Next Generation Autonomous Positioning (NGAP) and the Advanced Driving Assistant System (ADAS) use cases by THALIT.

3.1.10.1 ADAS Use case

ADAS use case - Sprint 1	
Duration	March 1-31, 2019
Goal	Sensor: camera selection
Planning	Key features definition, selection, starting purchasing activities.
Review	Completed
Retrospective	-

ADAS use case - Sprint 2	
Duration	April 1-30, 2019
Goal	Sensor: camera testing / Software tool for measurement analysis
Planning	Lab testing of camera. Software tool for sensor data analysis.
Review	Completed
Retrospective	-

ADAS use case - Sprint 3	
Duration	May 1-31, 2019
Goal	Documentation
Planning	Documents finalization. Data fusion study and architecture definition.
Review	Completed
Retrospective	-

ADAS use case - Sprint 4	
Duration	June 1-30, 2019
Goal	Radar characterization
Planning	Radar testing. Cluster vs objects outputs analysis.
Review	Completed
Retrospective	-

ADAS use case - Sprint 5	
Duration	July 1-31, 2019
Goal	Camera characterization
Planning	Camera study, lab testing, homography mathematics.
Review	Completed
Retrospective	-

ADAS use case - Sprint 6	
Duration	August 1-31, 2019
Goal	Sensor acquisition
Planning	More radar acquisition. Lidar definition and key features definition.
Review	Completed
Retrospective	-

ADAS use case - Sprint 7	
Duration	September 1-30, 2019
Goal	Software tool upgrade
Planning	Improvement of measurements analysis tool.
Review	Completed
Retrospective	-

ADAS use case - Sprint 8	
Duration	October 1-31, 2019
Goal	Installation and testing
Planning	Radar installation on tram vehicle 1013. GEST/Hitachi meetings and technical agreements for equipment installation.
Review	Completed /In progress
Retrospective	Radar installed. Hitachi agreement still pending, expected to be finalized in March 2020.

ADAS use case - Sprint 9	
Duration	November 1-30, 2019
Goal	LIDAR characterization
Planning	Lidar lab tests and analysis. Design of bracket system to install sensor on tram vehicles
Review	Completed
Retrospective	-

ADAS use case - Sprint 10	
Duration	December 1-31, 2019
Goal	Data fusion algorithm and installation on vehicle
Planning	Initial data fusion study and analysis. Installation on UNIMOG (special vehicle).
Review	Completed
Retrospective	-

ADAS use case - Sprint 11	
Duration	January 1-31, 2020
Goal	Data fusion. UNIMOG
Planning	Data fusion study and analysis, architecture finalization. Field test with UNIMOG. Project deliverable finalization.
Review	Completed
Retrospective	-

ADAS use case - Sprint 12	
Duration	February 1, 2020 (ongoing)
Goal	Dataset acquisition software
Planning	Software upgrade for dataset acquisition and sensor driver upgrade. Field test with vehicle 1013.
Review	In progress
Retrospective	-

3.1.10.2 NGAP use case

NGAP use case - Sprint 1	
Duration	March 1-31, 2019
Goal	Real Time Kinematic (RTK) technique
Planning	RTK study. Provider definition and agreement. Tool for measurement analysis design.
Review	Completed
Retrospective	-

NGAP use case - Sprint 2	
Duration	April 1-30, 2019
Goal	RTK, data fusion
Planning	RTK finalization and implementation of tool for measurement analysis. Sensor fusion study and analysis. Rail track modelling.
Review	Completed
Retrospective	-

NGAP use case - Sprint 3	
Duration	May 1-31, 2019
Goal	Splines and project deliverables
Planning	Tool for measurement analysis testing. Sensor fusion architecture. Rail track modelling with splines.
Review	Completed
Retrospective	-

NGAP use case - Sprint 4	
Duration	June 1-30, 2019
Goal	Loop data
Planning	Along line loop data retrieval system definition for hardware and software.
Review	Completed
Retrospective	-

NGAP use case - Sprint 5	
Duration	July 1-31, 2019
Goal	RADAR installation
Planning	RADAR installation on 1013, testing and measurement analysis. Tuning
Review	Completed
Retrospective	-

NGAP use case - Sprint 6	
Duration	August 1-31, 2019
Goal	Test and software tuning
Planning	Field testing and measurements analysis. Tuning of software application.
Review	Completed (testing is always ongoing because sensors continuously collect data and they are retrieved by LTE channel).
Retrospective	-

NGAP use case - Sprint 7	
Duration	September 1 - October 31, 2019
Goal	NGAP software tool
Planning	Upgrade and tuning of measurement tool.
Review	Completed
Retrospective	-

NGAP use case - Sprint 8	
Duration	November 1 – December 31, 2019
Goal	Sensor fusion implementation upgrade
Planning	Sensor fusion algorithm implementation upgrade and output verification
Review	In progress
Retrospective	-

NGAP use case - Sprint 9	
Duration	January 1, 2020 (ongoing)
Goal	RTK installation
Planning	NGAP performance estimation in comparison with ground truth system (RTK)
Review	In progress
Retrospective	Verification of output will go on till sensors are installed and data collected

NGAP use case - Sprint 10	
Duration	February 1, 2020 (ongoing)
Goal	Test field
Planning	Sensor fusion integration, testing and tuning
Review	In progress
Retrospective	Testing will go on

3.2 Version control systems

Version control systems provide means for distributed teams to work collaboratively together on shared documents. In this regard, the ELASTIC project started using two different platforms: Apache Subversion (SVN) [4] and Git [5]. The SVN repository was created at the beginning of the project, February 2019, in order to share documentation (meeting minutes, papers, presentations, etc.), and it is still used for the same purpose.

All partners stated the need to use a more complete platform for software development, i.e., Git, because it joins several DevOps functionalities (e.g., version control, issue tracking, distribution, etc.). BSC has set up a GitLab [6] instance hosted in the BSC premises. In detail, a virtual machine was setup, with public access (https://bsccselastic01.bsc.es) and login-protected. The machine has 4 virtual cores, 4GB RAM plus 4GB SWAP, a 60GB hard drive and Ubuntu 16.04 as the OS. Currently, the GitLab instance installed in this machine, as shown in Figure 1, contains three groups with different projects:

- Documents: for work-in-progress deliverables;
- ELASTIC-SA: for the development of the different software components of the ELASTIC Software Architecture;
- Examples: for examples of applications and demos of the different software components.

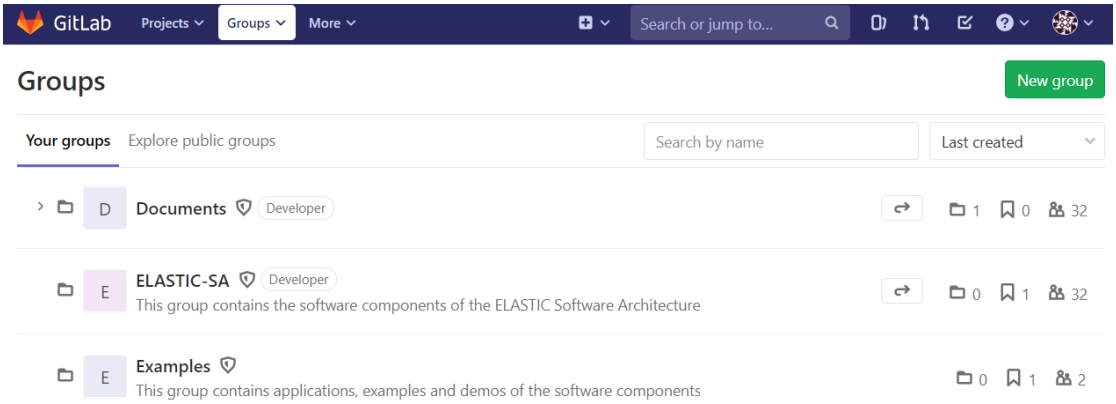

Figure 1. Screenshot of the ELASTIC GitLab groups

3.3 Continuous Integration (CI) system

A Jenkins [7] server has been installed in the BSC machine hosting the GitLab instance, in order to test both some isolated components (e.g., COMPSs, NFR tool), as well as the integration among them. The Jenkins interface is shown in Figure 2.

Even though no tests have been included so far, the Jenkins platform will be used during the third phase of the project, along with any other CI tools (e.g., GitLab CI) already used by each of the ELASTIC partners.

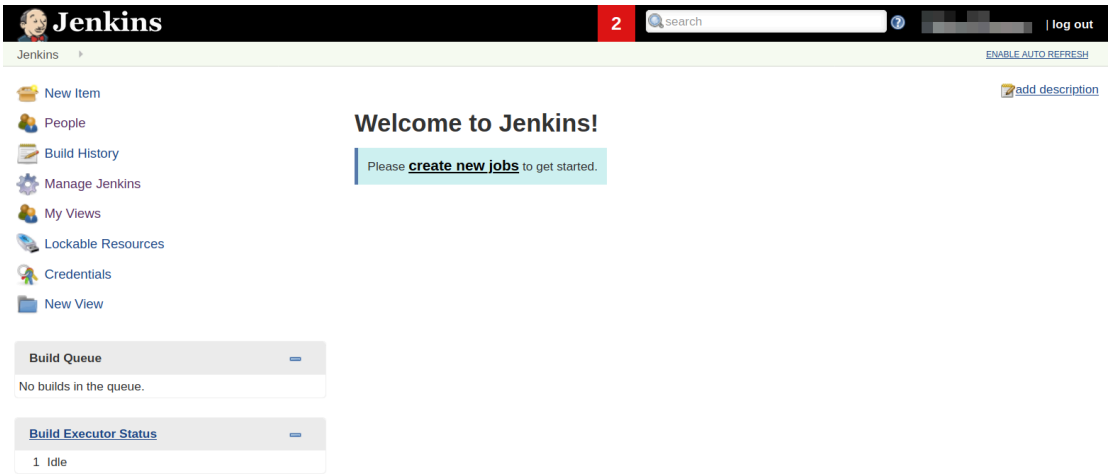

Figure 2. Screenshot of the ELASTIC Jenkins server.

3.4 Instant messaging and transparency

The distributed nature of the ELASTIC team enforces a mechanism to ensure agile communications and transparency. In this regard, BSC has set up a Slack [8] workspace on October 2019 to integrate team communications in one place. Slack offers, among others, the possibility of creating channels of communication, organized by topics, or to send direct messages, as shown in Figure 3.

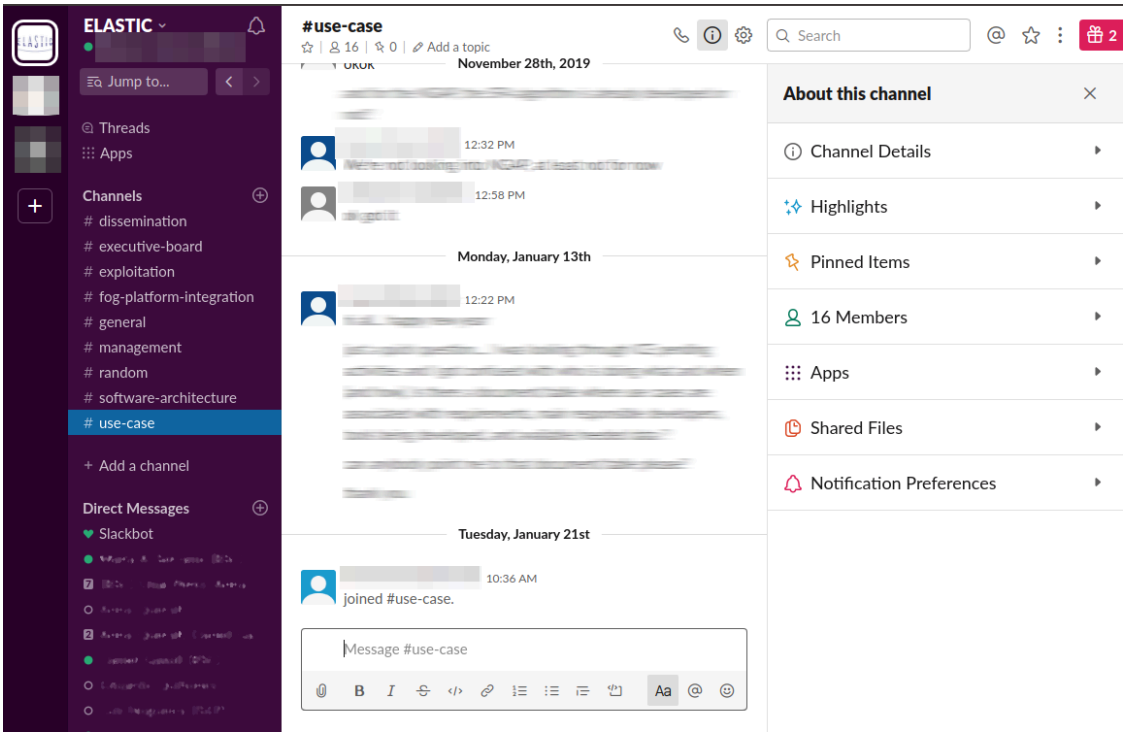

Figure 3. Screenshot of the ELASTIC Slack workspace.

During phase 2, the ELASTIC partners have used Slack to share code snippets, documents and messages. Some usage statistics towards the end of phase 2 are shown as an example in Figure 4.

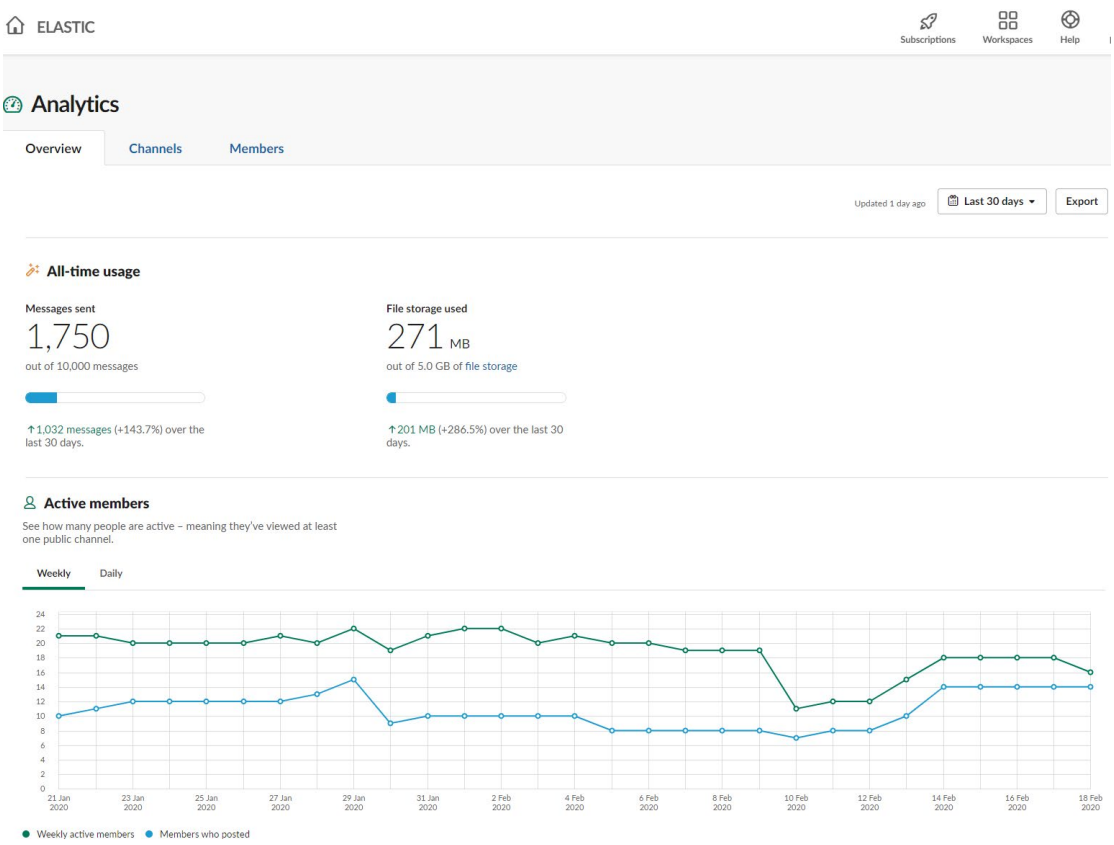

Figure 4. Usage analytics for the ELASTIC Slack workspace during the end of phase 2

3.5 Pending issues

Some pending issues still remain regarding the integration plan, and are described below:

- ***Testing with CI system***: Even though some steps have been already made towards CI testing, the ELASTIC consortium is still in the process of properly automating the testing process for the evaluation of the current implementation. It is expected that during phase 3, the partners involved in the integration of the ELASTIC software architecture will implement unit tests on the Jenkins platform (or in any other CI system that may be more suitable to the partners' workflows).
- ***Bug and issue tracking***: So far, the partners have used their own internal systems to track issues. However, during the third phase of ELASTIC, it is planned to include all bug and issue tracking information in the project's GitLab, thus centralizing the development of the code and facilitating the collaboration between partners.

4. Acronyms and Abbreviations

Each term should be bulleted with a definition.

Below is an initial list that should be adapted to the given deliverable.

- ADAS - Advanced Driving Assistant System
- API – Application Program Interface
- CaaS – Container as a Service
- CI – Continuous Integration
- D – deliverable
- DDAP – Distributed Data Analytics Platform
- GPU – Graphics Processing Unit
- GUI – Graphical User Interface
- LIDAR – Light Detection and Ranging
- M – Month
- MQTT – MQ Telemetry Transport
- MS – Milestones
- NFR – Non-Functional Requirements
- NGAP - Next Generation Autonomous Positioning
- RTK – Real Time Kinematics
- SCAP – Security Content Automation Protocol
- SRIA - Strategic Research and Innovation Agenda (SRIA)
- SVN - Apache Subversion
- WP – Work Package

5. References

[1] ELASTIC, "D3.1 Software architecture requirements and integration plan," May 2019.

[2] ELASTIC, "D3.3 ELASTIC software architecture – First release," February 2020.

[3] K. Schwaber and M. Beedle, Agile Software Development with Scrum, NJ, United States: Prentice Hall PTR, Upper Saddle River, ISBN: 978-0-13-067634-4, 2001.

[4] "Apache Subversion "Enterprise-class centralized version control for the masses"," [Online]. Available: https://subversion.apache.org/.

[5] git. [Online]. Available: https://git-scm.com/.

[6] "Gitlab. The entire DevOps lifecycle in one application.," [Online]. Available: https://about.gitlab.com/.

[7] "Jenkins," 2018. [Online]. Available: https://jenkins.io.

[8] "Slack, «Where work happens»," 2019. [Online]. Available: www.slack.com.