D2.5	
Version	1.1
Author	TUB
Dissemination	PU
Date	31-12-2019
Status	FINAL

D2.5 ElasTest requirements, use-cases and architecture v2

Project acronym	ELASTEST
Project title	ElasTest: an elastic platform for testing complex distributed large software systems
Project duration	01-01-2017 to 31-12-2019
Project type	H2020-ICT-2016-1. Software Technologies
Project reference	731535
Project website	http://elastest.eu/
Work package	WP2
WP leader	TUB
Deliverable nature	PUBLIC
Lead editor	Varun Gowtham
Planned delivery date	31-12-2019
Actual delivery date	27-12-2019
Keywords	Open source software, cloud computing, software engineering, operating systems, computer languages, software design & development

Funded by the European Union

License

Contributors

Name	Affiliation
Francisco Gortázar Bellas	URJC
Micael Gallego Carrillo	URJC
Orlando Avila García	ATOS
Varun Gowtham	TUB
Eduardo Jiménez	URJC

Version history

Version	Date	Author(s)	Description of changes
0.0	11-10-2019	Orlando Avila García, Francisco Gortázar Bellas, Varun Gowtham	Final ToC
0.1	27-11-2019	Varun Gowtham	Add initial content
0.2	02-12-2019	Eduardo Jiménez	Release mappings and high-level to technical requirements mapping.
0.3	11-12-2019	Varun Gowtham	Add requirements table
0.4	12-12-2019	Francisco Gortázar	Add architecture diagrams. Conclusions. High level requirements.
0.5	16-12-2019	Varun Gowtham	Formatting and editing
0.6	18-12-2019	Francisco Gortázar Bellas, Micael Gallego Carrillo	Internal deliverable review
1.0	27-12-2019	Varun Gowtham	Formatting and editing
1.1	27-12-2019	Varun Gowtham	Formatting and preparation of final version.

Table of contents

List of figures

List of tables

Glossary of acronyms

Acronym	Definition
CI (Continuous Integration)	This refers to the software development practice with that name.
FOSS (Free Open Source Software)	This refers to software released under open source licenses.
IaaS (Infrastructure as a Service), PaaS (Platform as a Service), SaaS (Software as a Service), MaaS (Mobile as a Service) and Baas (Browser as a Service).	This refers to different models of exposing cloud capabilities and services to third parties.
Instrumentation	This refers to extending the interface exposed by a software system for achieving enhanced controllability (i.e. the ability to modify behavior and runtime status) and observability (i.e. the ability to infer information about the runtime internal state of the system).
QoS (Quality of Service) and QoE (Quality of Experience)	In this proposal, QoS and QoE refer to nonfunctional attributes of systems. QoS is related to objective quality metrics such as latency or packet loss. QoE is related to the subjective quality perception of users. In ElasTest, QoS and QoE are particularly important for the characterization of multimedia systems and applications through custom metrics.
SiL (Systems in the Large)	A SiL is a large distributed system exposing applications and services involving complex architectures on highly interconnected and heterogeneous environments. SiLs are typically created interconnecting, scaling and orchestrating different SiS. For example, a complex microservice-architected system deployed in a cloud environment and providing a service with elastic scalability is considered a SiL.
SiS (Systems in the Small)	SiS are systems basing on monolithic (i.e. non distributed) architectures. For us, a SiS can be seen as a component that provides a specific functional capability to a larger system.
SuT (Software under Test)	This refers to the software that a test is validating. In this project, SuT typically refers to a SiL that is under validation.
TO (Test Orchestration)	The term orchestration typically refers to test

	orchestration understood as a technique for executing tests in coordination. This should not be confused with *cloud orchestration*, which is a completely different concept related to the orchestration of systems in a cloud environment.
TORM (Test Orchestration and Recommendation Manager)	Is an ElasTest functional component that abstracts and exposes to testers the capabilities of the ElasTest orchestration and recommendation engines.
TJob (Testing Job)	We define a TJob as a monolithic (i.e. single process) program devoted to validating some specific attribute of a system. Current Continuous Integration tools are designed for automating the execution of TJobs. TJobs may have different flavors such as unit tests, which validate a specific function of a SiS, or integration and system tests, which may validate properties on a SiL as a whole.
TiL (Test in the Large)	A TiL refers to a set of tests that execute in coordination and that are suitable for validating complex functional and-or non-functional properties of a SiL on realistic operational conditions. We understand that a TiL can be created by orchestrating the execution of several TJobs.
Test Support Service (TSS)	We define a TSS as a tool which aides in the implementation of tests in different contexts.
ETM (ElasTest Tests Manager)	A core component of ElasTest.
EPM (ElasTest Platform Manager)	A core component of ElasTest.
ESM (ElasTest Service Manager)	A core component of ElasTest.
EDM (ElasTest Data Manager)	A core component of ElasTest.
EIM (ElasTest Instrumentation Manager)	A core component of ElasTest.
ECE (ElasTest Cost Engine)	A test engine provided by ElasTest.
ERE (ElasTest Recommendation Engine)	A test engine provided by ElasTest.
EQE (ElasTest Question & Answer Engine)	A test engine provided by ElasTest.
EOE (ElasTest Orchestration Engine)	A test engine provided by ElasTest.
EUS (ElasTest User Emulator Service)	A test support service provided by ElasTest.

EDS (ElasTest Device Emulator Service)	A test support service provided by ElasTest.
ESS (ElasTest Security Service)	A test support service provided by ElasTest.
EBS (ElasTest Big-Data Service)	A test support service provided by ElasTest.
EMS (ElasTest monitoring Service)	A test support service provided by ElasTest.
ET (ElasTest ToolBox)	A toolbox provided to deploy ElasTest with external tools.
EJ (ElasTest Jenkins Plugin)	A plugin provided for integration of ElasTest Jenkins CI System.
CRUD (Create Read Update Delete)	Standard operations that can be performed on/to a software.
DoA (Description of Action)	A document which lists and described the actions to be performed in a project.
FMC (Fundamental Modelling Concepts)	Framework that provides comprehensive description of software intensive systems.
UML (Unified Modelling Language)	A general purpose, developmental, modelling language in the field of software engineering.
AWS (Amazon Web Services)	A cloud services platform.
AAA (Authentication, authorization and accounting)	Is a framework for intelligently controlling access to computer resources, enforcing policies, auditing usage and providing the information to bill for services.
API (Application Programming Interface)	A set of functions and procedures that allow the creation of applications which access the features or data of an operating system, application, or other service.
UI (User Interface) and GUI (Graphical User Interface)	The UI is an information device which a user can use to interact with a machine. Similarly, GUI is a type of UI that allows users to interact with electronic devices through graphical icons and visual indicators.
OAI (Open API Initiative)	An effort to standardize the description of REST API.
Release Management Meeting (RMM)	Face to face meeting held by the consortium at the end of a 4 month agile development cycle.

1 Executive summary

ElasTest is a cloud platform designed for helping developers to test and validate large software systems while maintaining compatibility with current continuous integration practices and tools. The platform embraces a microservice like architecture, collectively providing facilities for the tester to deploy testing processes as separate entities. A combination of such testing processes can be leveraged or reused to form a larger testing process which counters the monolithic testing approach.

This deliverable, outlines the efforts invested as part of tasks 2.2 and 2.3 during the second term of the project and extends the previous deliverable D2.3 presented at M18. Emphasis has been given to document traceability of requirements that has enriched the platform. An updated architecture diagram of the platform is presented.

2 Introduction

Nowadays, complex large software systems are proliferating due to the commodity of cloud and the need of elastic applications which pushes developers towards resilient software architectures like microservices. In this project, we concentrate on testing large software systems (i.e. SiL) created by the orchestration of simple components (i.e. SiS). Typically, those software systems are validated using CI tools and methodologies. This approach provides some minimal guarantees in relation to the correctness of the functional properties of the software, but it has very relevant limitations when evaluating other attributes of a software system in real production environments. For example, whenever developers want to validate non-functional features such as scalability, fault-tolerance or data consistency; they need to create complex testing architectures customizing the cloud orchestration mechanisms and managing test scalability by themselves. Things become even more complex when trying to reproduce real-world operational conditions. For example, tasks such as finding out how the system performs in real-networks (e.g. congestion, packet loss, latency, etc.) or evaluating how latency and other QoE parameters degrade with the number of users are relevant challenges. This becomes even more complex when systems manage special types of traffic such as sensor data or multimedia communications, which may follow complex binary protocols with real-time requirements and where the evaluation of QoS and QoE requires complex data processing.

ElasTest is an elastic cloud platform designed for helping developers to test and validate SiL (see definitions above), while maintaining compatibility with current CI practices and tools. For this, ElasTest bases on three principles:

- **Instrumentation of SuT:** ElasTest offers the facility to instrument the SuT based on the tester requirements. Such SuTs can be deployed on a native machine or on a cloud.
- **Test Orchestration:** ElasTest provides the facility to orchestrate one or more TJobs that assess the SuT. The orchestration is at the heart of the platform able

to apply novel techniques to form Test in Large (TiL) as a combination of TJobs. Furthermore, it exploits the reuse of TJobs.
- **Test Recommendation:** To ease the tester's job, ElasTest offers a novel solution of recommending tests to a user. This feature optimizes the tester productivity.

These principles are complemented by a set of tools aimed at supporting testing on different contexts:

- Browsers as a service, for UI testing.
- Emulators and actuators as a service, for testing of IoT applications.
- Security as a service, for assessing the security properties of large software systems.
- Monitoring as a service, for providing dynamic probes in a domain specific language capable of capturing the high level behaviour of the system and raising alarms.
- Big Data as a service, for capturing and processing all the data of the different services.

2.1 Core Concepts and design principles

The microservice in the context of ElasTest and the rest of the document is referred to as component, this is due to the fact that we do not follow the microservice architecture closely, and favor flexibility over formality. The platform is dynamic in nature in which the composition of the platform depends on which components that are active at any given time depending on the testing process (TJob) that is running.

The nomenclature of components and the relevance to ElasTest is detailed in Section 3.1. In this subsection we describe the basic principles used when designing the individual components of ElasTest.

We followed a requirements-driven development in the project. The initial set of use cases were taken as the basis to provide technical requirements for the components. The Release Management Meetings (RMMs) which took place every 4 months in the course of the project, helped to steer further requirements, based on agile development methodology. The requirements generally were collected at two levels:

1. Platform level requirements provided by vertical demonstrators and end users of the platform. The project also adopted requirements on prevailing technologies in the market.
2. Technical requirements in general, aided in development of features from the components such that one or more such technical requirements could address a platform level requirement.

The RMMs enabled the consortium to align and plan development efforts until the next RMM. This method of approach lead to the present architecture, as result of series of evolution steps during the project.

2.2 Structure of the document

The rest of the document summarizes the efforts of objectives T2.2 and T2.3 together. Section 3 elaborates on the agile methodology, together with explaining the resulting evolution of the architecture based on decisions taken during periodic RMMs. Section

4 documents the platform level and technical requirements. Section 5 documents the architecture of the platform at month 36. Section 6 concludes the document with emphasis on future research and lessons learned. Section 7 provides list of references and section 8 provides the complete set of technical requirements of the components.

3 Methodology

Figure 1 shows the ElasTest agile methodology and its applications on the work packages.

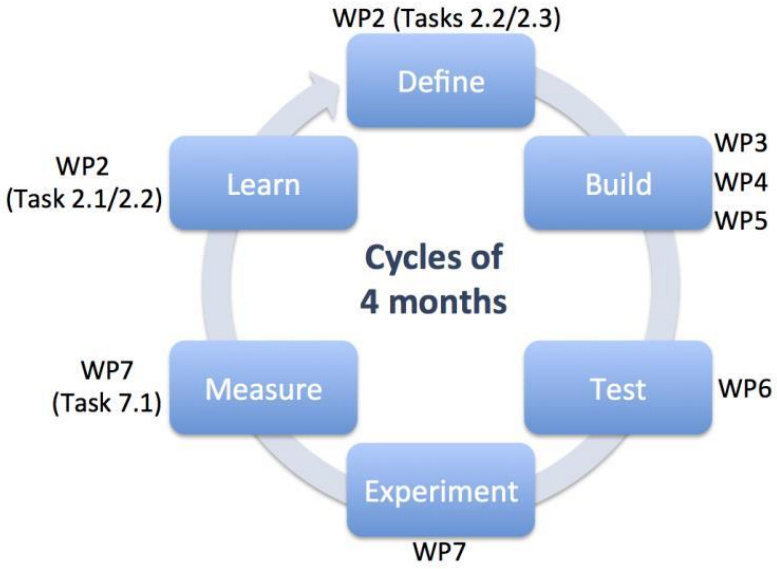

Figure 1 ElasTest agile methodology

The agile methodology selected is based on executing incremental iterations on a *Build-Measure-Learn* feedback loop which validates that the implemented technology is valuable and responds to real needs. On every of these iterations, the built technologies are discussed between the technical WPs (WP3, WP4, WP5) and the vertical demonstrators (WP7) in order to refine their roadmap.

The cycles have a duration of 4 months, we call each of these cycles a *Release (R)*, a total of 9 releases have been planned during the project duration. The development tasks until R6 shall provide the platform validated in a lab, while the last 3 shall demonstrate it through the vertical demonstrators.

During the first cycle we have focused on developing, based on our initial component designs, proof and concepts as well as performing adaptations to the previous baseline technologies/components in order to identify and start solving the integration issues that the consortium were able to anticipate at this stage. Hence, in an initial stage most components only implement a small subset of the requirements depicted in section 4.2. Within the second cycle, and once the CI environment was ready, we have focused mainly on providing value to the platform users by implementing the required component functionalities while at the same time we organized regular meetings keeping an eye on the market and vertical demonstrators needs, in order to maintain the project aligned with the industry requirements. At month 18, we were able to

cover four of these incremental cycles which include the release of the first integrated version of the platform (Milestone5).
At month 36, we were able to cover the rest of the 5 release cycles, totally covering nine release cycles.
In order to achieve our goals, common conventions and approaches have been agreed within the consortium. You can find below the most relevant conventions:

- Fundamental Model Concept (FMC) has been selected to provide understandable block diagrams of the platform as well as for each of the submodules that constitutes the platform.
- Unified Modeling Language (UML) is the general-purpose modeling language selected to define the Data Model diagrams and Sequence diagrams across the components.
- Discussions are promoted across technical WPs in order to ensure that all components have the same level of understanding on the platform. The discussions have been organized in the following small working groups:
    - Persistence Working Group
    - Monitoring Working Group
    - Test Management Working Group
    - Data Management Working Group
- To facilitate the communication across developers, different Slack channels (#) are used in the project.
- For the fine-grained management of the previous and ongoing tasks each WP leader manages a Trello board.
- The platform is designed as a Service Oriented Architecture (SOA) where the direct interaction between software modules uses to be synchronous through REST APIs, however for certain cases the systems within ElasTest will be able to react asynchronously based on events forwarded by other modules or systems of the platform.
- The interactions and the information exchanged between components have been captured early during the design phase through the specification of the interfaces exposed by the components following the *OpenAPI* initiative.
- Software components releases follow Semantic Versioning approach which proposes a simple set of rules and requirements that dictate how version numbers are assigned and incremented.

3.1 ElasTest Architecture (Functional Architecture)

For the purpose of fluency in understanding, a functional architecture of the platform is provided. A detailed architecture reference can be found in section 5.

ElasTest is a cloud platform designed for helping developers to test and validate large software systems while maintaining compatibility with current continuous integration practices and tools. For this, ElasTest bases on three principles:

1) Instrumentation of the software under test through observability and controllability agents so that it reproduces real-world operational behavior.
2) Test orchestration combining intelligently testing units for creating a more complete test suite.

3) Test recommendation using machine learning and cognitive computing techniques for recommending testing actions and providing testers with friendly interactive facilities for decision making.

ElasTest enables developers to test large software systems through complex test suites created by orchestrating simple testing units (so-called TJobs). This orchestration mechanism is one of the main novelties of the ElasTest project and its precise conception, formalization and consolidation is one of our main research objectives. From the perspective of the tester, a TJob is software that, upon execution, performs some testing actions against the software under test. From this perspective, the TJob is a "testing unit". In order of not to constrain the freedom and flexibility of the tester, we do not assume any kind of property for the TJob neither from the technological (i.e. language, framework, etc.) nor from the semantics perspective (i.e. model, behavior, etc.) Our only assumption is that the TJob accepts some input parameters and that, upon execution, generates an outcome (i.e. output parameters). The expected values of such outcome constitute the TJob oracle.

The conceptual representation of the ElasTest architecture is shown in Figure 2.This conceptual representation, created at the time of the proposal, was the starting point of our architecture design. It consists of a number of software modules that testers can install into public or private clouds.

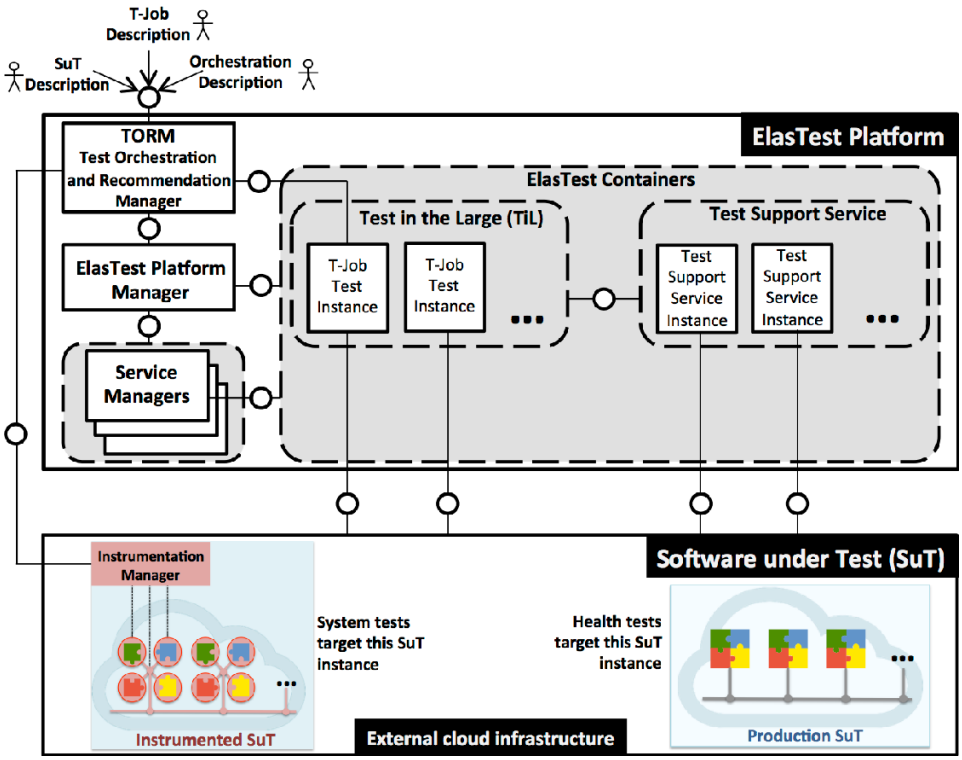

Figure 2 Conceptual representation of the ElasTest architecture and its relation with the SuT.

A more detailed overview of the functional architecture of the ElasTest platform is shown in Figure 3. The intermediate architecture design presented within this document has been produced after accomplishing the procedures defined in the methodology described in section 4.3 of this report. Both figures, the conceptual

representation and the functional architecture overview are based on the Fundamental Model Concept (FMC) which primarily provides a framework for the comprehensive description of software-intensive systems. It is based on a precise terminology and supported by a graphical notation which can be easily understood. In order to know how to interpret the block diagrams and their communications, please refer to the FMC cheat sheet [2].

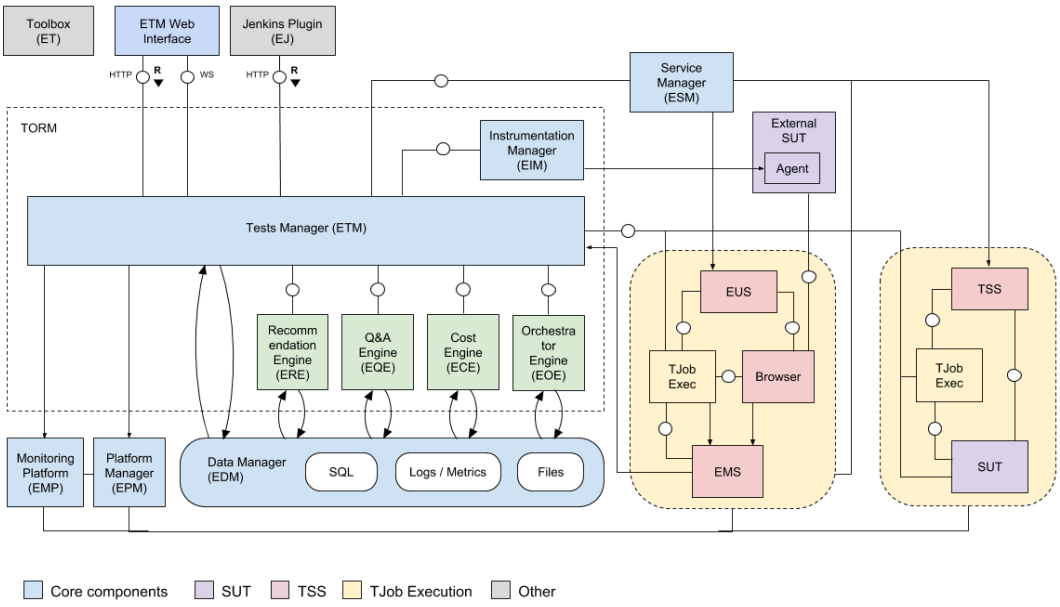

Figure 3 Functional architecture overview of the ElasTest platform.

Table 1 illustrates the building blocks of the ElasTest system; the individual software components of the platform maps with the blocks depicted in the aforementioned figure, each of them constitutes a fine-grained SOA.

We envisaged the ElasTest platform to be implemented as a distributed and scalable system, which allows the testing of large software systems created by the orchestration of simple components. Nowadays, those large systems are mainly validating the correctness of the software under evaluation using CI tools and DevOps methodologies among other available options. Despite this, very relevant limitations exist when we try to evaluate other attributes of the software such as non-functional features in real production environments or real-world operational conditions. ElasTest is a cloud platform designed for helping developers to test and validate large software systems while maintaining compatibility with current continuous integration practices and tools.

The resultant components are categorized as follows:

- ElasTest Core Components: These components constitute the enablers of the platform. They have the responsibility of providing management mechanism for the platform, the tests jobs and the software under evaluation.
- ElasTest Test Engines: The engines offer additional capabilities that can be used by the platform or the test support services, thanks to our modular architecture different engines may be plugged.

- ElasTest Test Support Service (TSS): These comprise reusable cloud services used to support the testing of the software under evaluation.
- ElasTest Integrations with External Tools: These comprise of the tools and plugins used for integration of ElasTest with external tools.

Table 1 Building blocks of ElasTest.

Component Name	Role
Core Components	
Test Manager (ETM)	It is the brain of ElasTest and the main entry point for developers.
Platform Manager (EPM)	It is the interface between ElasTest components and the cloud infrastructure.
Platform Monitoring (EMP)	It is a service that monitors the core components of ElasTest platform.
Service Manager (ESM)	It delivers, on request/demand, service instances of particular service types.
Data Manager (EDM)	It provides the persistence layer services for all components.
Instrumentation Manager (EIM)	It controls and orchestrates the agents that are deployed on the software under test.
Test Engines	
Cost Engine (ECE)	It estimates the cost to make developers cost aware of running a test.
Recommendation Engine (ERE)	It is a cognitive system designed to leverage recommendations based on learned knowledge.
Question & Answer Engine (EQE)	It accepts questions asked in natural language and tries to identify user's intentions and needs.
Orchestrator Engine (EOE)	It orchestrates and executes in coordination a set of TJobs for creating more complex test suite.
Test Support Services	
User Impersonation Service (EUS)	It is devoted to provide the mechanism for emulation of users in end-to-end tests. Specifically, it provides web browsers to tests.
Device Emulator Service (EDS)	It emulates devices used in Internet of Things (IoT) applications.
Security Service (ESS)	It facilitates the security testing of the software under test.
Big-Data Service (EBS)	It provides a scalable computing engine based on

	big-data technologies
Monitoring Service (EMS)	It provides a monitoring service suitable for inspecting the execution of the software under test.
ElasTest Integrations with External Tools	
ElasTest Jenkins plugin	It is devoted to provide the mechanism for using ElasTest via Jenkins CI system.
ElasTest Toolbox	This provides tools to install and configure ElasTest in the easiest way possible.

The current architecture reference diagram can be found in a later diagram.

3.2 Roadmap

Release Management Meetings (RMM) were held at the end of every 4-month release cycle. During the meeting, emphasis was laid on to discuss the learning from the retrospective and came up with a roadmap for the next release. The roadmap took into account the requirements from the vertical demonstrators within the project, and the requirements from industries and market external to the project. Through the course of RMMs, we were able to steer the project based on requirements through the agile development methodology as shown in Figure 1. During the RMM and also through our internal Slack communication channels, we were able to prioritize the development of features based on the mandatory requirements for vertical demonstrators and market. All in all, the second period revolved around three different pillars:

- **Support for verticals:** in order to guarantee a proper evaluation of the project, we paid special attention to the needs from our verticals. This involvement allowed us to carefully plan the features needed in order to have everything in place by the time the validation activities started (see D7.2 for a detailed explanation of the different validation activities carried on.) It is worth noting that manual testing is a big feature not considered in the project DoA but developed in the context of the project to meet requirements of one of the verticals.
- **Elasticity:** we focused on building elasticity within the platform. We steered from building our own scheduler, to delegating to Jenkins to finally resorting to Kubernetes.
- **Observability and analytics:** since the very beginning ElasTest was able to gather data from the different services involved during the testing process. However, we were lacking actionable visualizations of such data. During this second period we focused on building an actionable UI that allowed end-users to fast and easily find the cause of an error when tests failed. The term observability has gained momentum in highly distributed systems to describe

the availability of information that can be used to understand the system behavior at runtime. We adapted and used it in the context of testing, and we gave several talks about how ElasTest brings observability to the testing process at different events (European Testing Conference, February 2019, TestBash Brighton, April 2019, ExpoQA, June 2019, or Fuseco Forum, November 2019, among others).

A brief summary of the roadmap can be found in Table 2. The table summarizes the following:

1. **Document type:** Specifies whether the directions towards a decision were made available through the medium of meeting minutes of RMM or as an internal communication.
2. **Pivot or Persevere:** During Release Management Meeting (RMM), the consortium identifies if it is required to change direction by pivoting or pursue in the given direction by method of persevering. It can be seen that in some cases, both pivot and persevere directions appear.
3. **Decision on architecture:** Describes briefly what changes were introduced as a consequence of agreeing on a pivot or persevere direction.

Table 2 ElasTest roadmap summary

Serial Number	Document	Date	Pivot or Persevere	Decisions on architecture	Remarks
1	KO Meeting - Madrid	26th - 27th January 2017			
	WP2 Requirements			Initial architecture planning: ElasTest platform requirements: 1.deploy and manage, T-Jobs and TSSs. 2.Initial planning on service manager to deploy TSS on demand by the TJobs. 3. Uses Docker.	
2	RMM-1 - Berlin	4th - 5th May 2017	Pivot - Use Docker and hence microservice like REST API approach	1. Components named per work package. 2. Fix API for component. 3. CI system used to build the CI. 4. License badges and coverage report. 5. Documentation on Github.	
3	Roadmap - R2	25th August 2017		Initial integrated platform, integrated core components, TSS and engines.	

4	RMM-2 - Madrid	5th – 6th September 2017	Persevere - Work on integration	1. EPM integration with ETM, ESM and EMP. 2. Monitoring integration 3. EBS integration 4. Persistence integration. 5. ESM integrates TSS using elastestservice.json 6. Engines integration	
5	Roadmap - R2.5	25th September 2017	Persevere - Lite version to have demonstrable platform by December 2017	Platform features: 1. Web browsers 2. Log analysis 3. Jenkins integration 4. TJobs	ElasTest-Lite available.
6	Roadmap - R3	29th November 2017	Persevere - Work on full-fledged platform	Assign specific tasks to each component towards complete integration.	Beta versions published in R3, focusing on development in shorter mini-cycles.
7	RMM-3 - Pisa	10th – 11th January 2018	Persevere - 1. Components improve pending action points	1. Suggestion, if EMS can be used for TJob orchestration. 2. Components show their	

			2. EPM to plan on clusterization. Pivot - 1.Test Link integration for ATOS vertical 2. EOE planning on supporting FOKUS vertical. 3. Toolbox	working on nightly. 3. Proposal for AAA from ESM.	
8	RMM-4 Madrid	3rd – 4rd May 2018	Pivot: 1. Test orchestration 2. Test link and Jira integration? 3. JMeter	1. AAA proposal and discussion 2. Multitenancy proposal 3. EMS integration 4. ECE cost model integration	
9	RMM-5 Barcelona	27th – 28thSeptember 2018	Pivot: 1. Platform testing 2 phases. Plan Jenkins integration and empirical survey 2. Map tests with requirements 3. Test orchestration 4. Test prioritization	1. Plugins as a collective name for services and engines. 2. EOE configurations and high level management of TJobs. 3. EPM proposal for clusterization.	1. More tests needed because requirements are not covered.

			5. Mandatory features for QoE for Naevatec	
10	Roadmap - R6	28th November 2018	Pivot - Initial efforts on Kubernetes cluster	EPM initial efforts on Kubernetes.
11	RMM-6 Madrid	15th – 16th January 2019	Pivot - Towards Kubernetes in the requirements	Component demo shown in preparation for QE.
11	Roadmap - R7a	7th February 2019		1. Include all project components in ElasTest 1.0 2. Document all user-visible components 3. ElasTest available in 3 flavors: mini, singlenode and cluster (multi-node and kubernetes)
12	Roadmanp - R7b	10th April 2019		1. SuT deployment as Kubernetes app. 2. Deploy ElasTest as Kubernetes cluster using helm charts similar to Docker compose. 3. Compare several independent

				executions - required by 5G vertical. 4. Send test container start and end events to ECE. 5. Allow using EMS with external SuT 6. Stop saving metrics in external SuT when TJob has finished its execution. 7. TestLink integration 8. EUS - exploring WEBRTC QoE support (delayed) 9. EPM : polish management of elastic Kubernetes cluster, decide on EPM API to deploy docker-compose TSS in Kubernetes or not, execute some load tests to show elasticity.
13	RMM-7 Athens	22nd – 23rd May 2019	Persevere - EPM Kubernetes integration with ETM, ESM	1. Key improvements to platform esp. log analyzer and usability.
14	RMM-8 Madrid	3rd - 4th September 2019	Persevere – Kubernetes integration	1. Platform features frozen 2. Elasticity in progress and

		components to adopt to Kubernetes 3. Planned for final review

The envisioned 9 release (R1-R9) cycles have been finished at the end of month 36. These releases were mainly referred to as the internal releases. However, during the course of the certain internal release cycles were provided as public releases. The public release documentation can found online on ElasTest website[1]. A mapping of internal releases with public release during the second period is provided in the following table.

Table 3 Internal to public release mapping

Internal Release Number	Public Release Number	Release date
R5	0.9.1	09/05/2018
R5	1.0.0-beta1	12/09/2018
R6	1.0.0-beta2	04/10/2018
R6	1.0.0-beta3	31/10/2018
R6	1.0.0-beta4	03/12/2018
R6	1.0.0-beta5	20/12/2018
R7a	1.0.0-beta6	23/01/2019
R7a	1.0.0-beta7	31/01/2019
R7a	1.0.0	13/02/2019
R7b	1.0.1	19/02/2019
R7b	1.1.0	01/03/2019
R7b	1.2.0	07/03/2019
R7b	1.3.0	20/03/2019
R7b	1.3.1	28/03/2019
R7b	1.4.0	08/04/2019
R7b	1.4.1	09/04/2019
R7b	1.5.0	29/04/2019
R8	2.0.0	15/10/2019
R8	2.0.1	22/10/2019
R8	2.1.0	14/11/2019

At the end of month 36, the project has completed all milestones available in the Description of Action (DoA). A more detailed explanation can be found in deliverables of WP6 [3].

3.3 Traceability

In a nut shell, the development efforts carried out in the project, used a requirement driven approach. In other words, that is any feature available in the platform can be traced to a set of one or more requirements. The requirements can be classified as follows:

[1] https://elastest.io/docs/releases/

1) **High-level requirements:** These requirements are related to the platform in general, typically requested from end users of the platform such the vertical demonstrators, stakeholders and market trends. Therefore, we can further sub-divide the high level requirements into:
    a. **Requirements from vertical demonstrators:** Platform features requested by the 4 vertical demonstrators in the project for carrying out validation experiments in WP7.
    b. **Requirements from stakeholders and market:** Platform features added or requested from outside the project. These requirements were collected from external users interested in the project or added by the project based on the market trends.
2) **Low-level technical requirements:** A set of one or more low-level requirements drive the implementation details of a high-level requirement. A single low-level requirement also called as component requirement is specific to a component, which enhances the features offered by a component.

While it is possible to provide features corresponding to a requirement, either at platform or component level, it is also important to validate such features. To this end, we used a common requirements spreadsheet in cooperation with WP6 and WP7. Component owners were able to build specific tests and validate the component requirements through tests developed as part of efforts from WP6. From WP7, the vertical demonstrators were able to validate the availability of the requested high level requirements. Furthermore, the validation of high-level requirements from market was validated internally.

Fine grained details of traceability of requirements can be found in section 4.

4 Use cases and requirements

In the earlier phases the use cases listed in section 3 of D2.3, served as a generic set of use cases for the platform, defining a framework that enables end-user interaction with the platform. Based on the explanation given in subsection 3.3, in this section we document the requirements collected in the duration of the project, which were collected in the form of tables and coordinated in the consortium using a common spreadsheet. SMART criteria[2] was used in documenting the requirements such that they are specific and measurable. A general structure of table for requirements documentation is provided below:

1) Column [1]: **ID:** A unique identification string given to each requirement. Using this unique string it is possible to trace the origin of the requirement. For example, ETM1, is the 1st requirement of the ETM core component of ElasTest.
2) Columns [2-5]: **User Story:** The user story is presented as title on column 2, followed by columns 3 to 5, explaining user story in 3 parts which are; which type of user, what goal is to be achieved and for what reason.

[2] SMART criteria, https://en.wikipedia.org/wiki/SMART_criteria

3) Column [6]: **Status:** Specifies the status of requirement which can be assigned one of the following status:
    a. AVAILABLE: The requirement was made available as feature and adopted into the platform.
    b. BACKLOG: The requirement is planned to be made available in the future.
    c. PROGRESS: The requirement is currently in the phase of implementation.
    d. DROPPED: The requirement, though documented, may not be important and therefore has been dropped.
4) Column [7]: **Release:** Specifies which internal release, the feature specified by the requirement is made available or will be made available, corresponding to the value in the column [6] Status.
5) Column [8]: **Technical Requirement ID**: This is a special column that differentiates between technical requirements and high level requirements. It lists one or more identification strings of the technical requirements of components, that make the requested feature available in the platform. The high level requirements which are not available are marked, "Not implemented" in Table 4.

We present the following requirements tables:

1) Table 4: Requirements from vertical demonstrators: This table lists requirements from the four vertical demonstrators. The requirement ID strings can be shortly summarized as follows:
    a. Requirements ID from IIoT demonstrator from TUB, starts with the string "IIoT".
    b. Requirements ID from WebRTC demonstrator from NAEVATEC, starts with the string "NAEVA".
    c. Requirements ID from online banking demonstrator from ATOS, starts with the string "ATOS".
    d. Requirements ID from 5G demonstrator from FOKUS, starts with the string "FOKUS".
2) Table 5: Requirements provided by core component owners to enable features in the platform.
3) Table 6: Requirements provided by component owners of the test support services (TSSs), to enable features required to configure TJobs.
4) Table 7: Requirements provided by the component owners of the test engines (TEs), to enable features into the platform.
5) Table 8: Requirements from component owners which enable integrations with external tools such as Jenkins.

Table 4 High level requirements table provided by vertical demonstrators

ID	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Expected Release	Technical Requirement ID
IIoT-1	Provide a variety of basic sensors	ElasTest User	a manifest of emulated sensors	I can use them in my application towards building an IIoT application	AVAILABLE	R3	EDS9
IIoT-2	Provide device models	ElasTest User	a method to change the behavior of the devices	I can test my IIoT application with varying inputs in run time.	AVAILABLE	R3	EDS11
IIoT-3	Keep the SuT initiated by ElasTest alive instead of terminating it on termination of TJob	ElasTest User	a way to keep a live SuT initiated by ElasTest	I can use the same SuT across multiple TJobs initiated across different timelines.	BACKLOG	-	Not implemented
IIoT-4	Ability to launch multiple TJobs on a single SuT	ElasTest User	a way to launch multiple TJobs on a single SuT in ElasTest	I can test different methods on the same SuT simultaneously.	BACKLOG	-	Not implemented

ID	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Expected Release	Technical Requirement ID
IIoT-5	Provide the ability to request devices	ElasTest User	the ability to request emulated devices	I can use them in my IIoT application.	AVAILABLE	R4	EDS3
IIoT-6	Provide the ability to form IIoT applications with emulated devices	ElasTest User	the ability to wire the emulated devices	I can use them with an application logic	AVAILABLE	R4	EDS4
IIoT-6	Provide ability to dynamically reconfigure the behavior of emulated devices during run time	ElasTest User	the ability to dynamically reconfigure the emulated devices during run time	I can inject faults into the system dynamically.	AVAILABLE	R4	EDS11
NAEVA-Web-01	Copy TJobs from one project to another	ElasTest User	Copy TJobs from one project to another	So I can even reuse the whole TJob	BACKLOG	-	Not implemented
NAEVA-Web-02	Browser and version selection	ElasTest User	to select from the TJob Matrix the Browsers and versions to tests	I can reuse one single TJob and TestSuite to test multiple configurations and compatibilities.	AVAILABLE	R4	ETM32

ID	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Expected Release	Technical Requirement ID
NAEVA-Web-03	Compare TJob execution against different SuTs	ElasTest User	to select in the TJob Matrix the Sut against which the TJob should be executed	I can compare the tests results against different configurations or versions of the same SW	AVAILABLE	R7a	ETM28
NAEVA-Sec-01	Parallel security analysis	ElasTest User	when ESS is selected in a TJob, run security tests on the request, forms, etc,	I can get security testing over the running TJobs, taking advantage of the possible complex navigation or interaction with the SW that can be necessary to reach parts of the application.	AVAILABLE	R5	ETM9, ETM40
NAEVA-Gen-01	Export TJob results	ElasTest User	to export TJob results, logs, files and videos	I can share them with clients or being used into automatic	AVAILABLE	R1-R2	ETM46

ID	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Expected Release	Technical Requirement ID
				reports			
NAEVA-WebRTC-01	Adapt behavior and result of the TJob to webRTC statistics	ElasTest User	some mechanism for dynamically interact with the WebRTC stats within the test and the TJOB	I can set for example thresholds on the WebRTC statistics.	AVAILABLE	R5	EUS7
NAEVA-WebRTC-02	Simulate network conditions.	ElasTest User	to simulate network conditions. Network types network changes or even connection losses	I can test how a real-time application behaves under different conditions.	AVAILABLE	R8	EIM14, EIM15, EIM16
NAEVA-WebRTC-03	Load and stress tests (JMeter like)	ElasTest User	the possibility of integrate Load an stress tests (JMeter) on a ElasTest Project	I can really have all my tests in one place	AVAILABLE	R9	EUS11,ETM43
ATOS-Web-01	Browser and version selection for test plan execution	ElasTest User	to be allowed to select which Browser and Version to use for an specific test plan	I can execute my test plan against different platforms	AVAILABLE	R7a	ETM31

ID	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Expected Release	Technical Requirement ID
			execution				
ATOS-Web-02	Browser synchronization testing	ElasTest User	the possibility to perform browser synchronized testing in several browsers simultaneously (https://browsersync.io/)	I can manually test in several browsers at the same time	AVAILABLE	R8	ETM44
ATOS-Web-03	Security	ElasTest User	to have a kind of security catalog that could be selected for each TJob	activated security tests are executed when I am running functional TJobs	AVAILABLE	R8	ETM45
ATOS-Web-04	Semi-Automated exploratory testing / Automated Steps (Snippets) for manual testing support	ElasTest User	to be able to record / execute some steps automatically when manually testing	I can perform regressions or exploratory testing much faster	BACKLOG	-	Not implemented

ID	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Expected Release	Technical Requirement ID
ATOS-Web-05	Integration with TestLink	ElasTest User	to be able to define test cases using TestLink	all my information regarding testing is inside the ame tool	AVAILABLE	R4	ETM12
ATOS-Web-06	Integration with Jira	ElasTest User	to be able to raise new bugs in JIRA from Elastest	traceability of bugs and test cases exist	BACKLOG	-	Not implemented
ATOS-Web-07	Execution of manual test cases in different versions of browsers	ElasTest User	to be able to do manual testing in a browser and version I specifically choose	I can reproduce bugs raised by final users in the specific web browser and version	AVAILABLE	R7a	ETM13
ATOS-Web-08	Execution of automated test cases in different versions of browsers	ElasTest User	to be able to do automated testing in a browser and version I specifically choose	I can reproduce bugs raised by final users in the specific web browser and version	AVAILABLE	R4	ETM6

ID	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Expected Release	Technical Requirement ID
ATOS-Web-09	Execution of automated test cases concurrently for performance test: To take advantage of automated test cases with Selenium (test cases already automated) it would be good to be able to run those test cases simultaneously in different browsers just to check the time required to execute each one of those test case when several instances of that test case are running at the same time.	ElasTest User	to be able to do performance test using end2end tests defined in Selenium	I can ensure the SuT is able to work properly when several concurrent users are working at the same time	BACKLOG	R6-Final	Not implemented
ATOS-Web-10	Execution of performance test similarly to Jmeter executions	ElasTest User	to be able to import my own JMeter projects and run them from Elastest	I can have all tests running in the same platform	BACKLOG	-	Not implemented
ATOS-Web-11	Execution of basic security test	ElasTest User	to be able to execute basic security testing	my websites have always a minimum security	AVAILABLE	R7a	ESS1

ID	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Expected Release	Technical Requirement ID
				validations			
FOKUS - 1	Openstack SUT integration	ElasTest User	to be able to deploy Elastest on Openstack	that we can more easily have cloud deployments integration	AVAILABLE	R5	EIM11
FOKUS - 2	External SUT Monitoring	ElasTest User	to be able to monitor external SuTs	that it is easier to write performance conditioned tests	AVAILABLE	R5	ETM25
FOKUS - 3	Test Orchestration	ElasTest User	to be able to orchestrate tests (sequential and/or parallel)	that it's easier to compare different SuTs and have conformance test suites	AVAILABLE	R6	EOE1/EOE2
FOKUS - 4	Openbaton Integration	ElasTest User	to be able to deploy SuT via Elastest-OpenBaton integration	it's easier to deploy and manage Cloud	BACKLOG	-	Not Implemented

ID	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Expected Release	Technical Requirement ID
				SuTs			
FOKUS - 5	Data export	ElasTest User	to be able to export monitoring data	it's easier to disseminate and exploit testing data	AVAILABLE	R8	EJ11/EJ12
FOKUS - 6	E2E Elastest support services integration	ElasTest User	to be able to use all support services	the tester has increased productivity and better feedback	AVAILABLE	R7b	ETM9
FOKUS – 7	Standard-aligned test description	ElasTest User	to be able to write standardized tests	it's easier to use the Platform as a standard aligned testing tool	BACKLOG	-	Not implemented

5 Architecture

During the first reporting period, D2.3, architecture of the platform and specifications of the components were presented (see Figure 4). The component specifications presented in D2.3 are relevant for the second reporting period as well. In this version of deliverable, we are going to address the evolved architecture of the platform. For a more detailed understanding the evolution of architecture of components, we refer the reader to the corresponding deliverables of technical work packages WP3, WP4 and WP5 [4][5][6][7].

As Table 2 explains, the architecture was evolved during the second period. We discussed, and even explored, different approaches to make the architecture elastic. Initially, we explored the possibility of making an even deeper integration with Jenkins, and allowing Jenkins orchestrate TJobs, thus relying on the distributed nature of this CI server. However, soon it was clear that distributed applications were moving to Kubernetes. This idea was reinforced by JenkinsX (an open source project that was working on adapting Jenkins to the Kubernetes context) being included as part of the Jenkins ecosystem. Therefore, we decided to design an architecture suitable for Kubernetes.

As a result, we ended up with the following four deployment modes, two for plain Docker support, two with specific support for Kubernetes:

- ElasTest Mini
- ElasTest Singlenode
- ElasTest EK (ElasTest mini on Kubernetes)
- ElasTest HEK (Highly scalable ElasTest on Kubernetes)

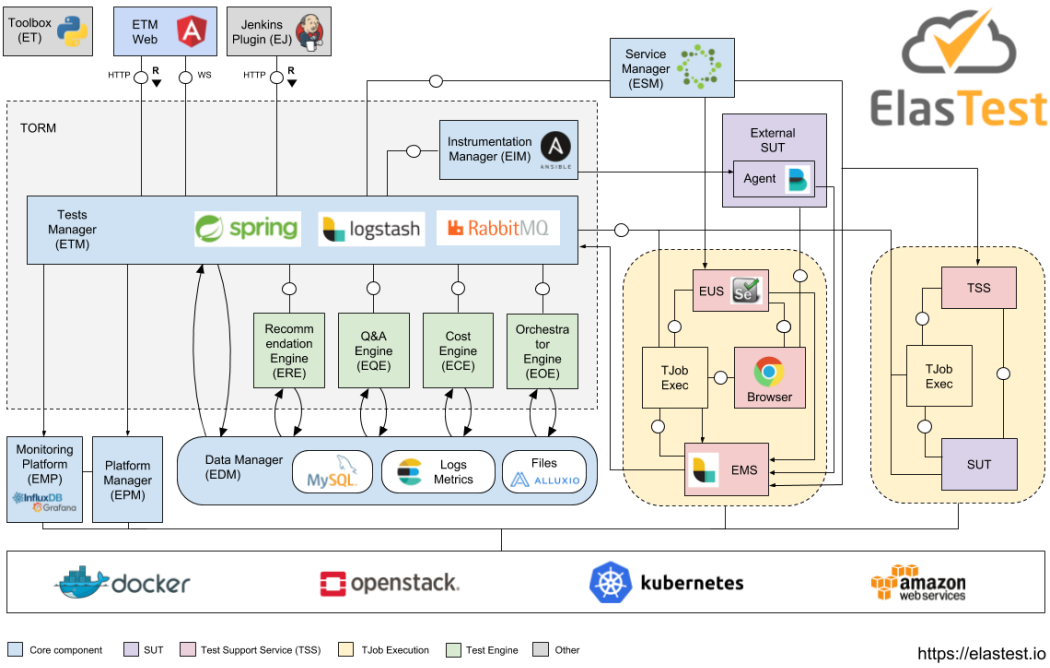

Figure 4 Architecture reference - support systems overview

The ElasTest Singlenode deployment was demonstrated at the first review, and its architecture corresponds to that ofFigure 4. ElasTest Mini was in development during

the first review and was released by September 2018. The main differences with the Singlenode deployment was its reduced memory footprint that made ElasTest suitable for being installed on 8Gb machines. This was key to foster adoption of the tool. Although this deployment mode is not suitable for production, allowed end-users to try out the product and see if it fitted well for them. The reference architecture of the ElasTest Mini mode is shown in Figure 5. Note that the EPM, ESM and the EUS are included within the ETM component, as modules, instead of running as independent services. Also, on the persistence side (see EDM below the ETM), we skipped completely ElasticSearch, and wrote a Logstash to MySQL adaptor that ingests logs and metrics directly into MySQL. ElasticSearch was one of the components requiring more memory, and this reduced enormously the memory requirements, at the cost of slightly lengthier searches, and losing the elastic nature of ElasticSearch.

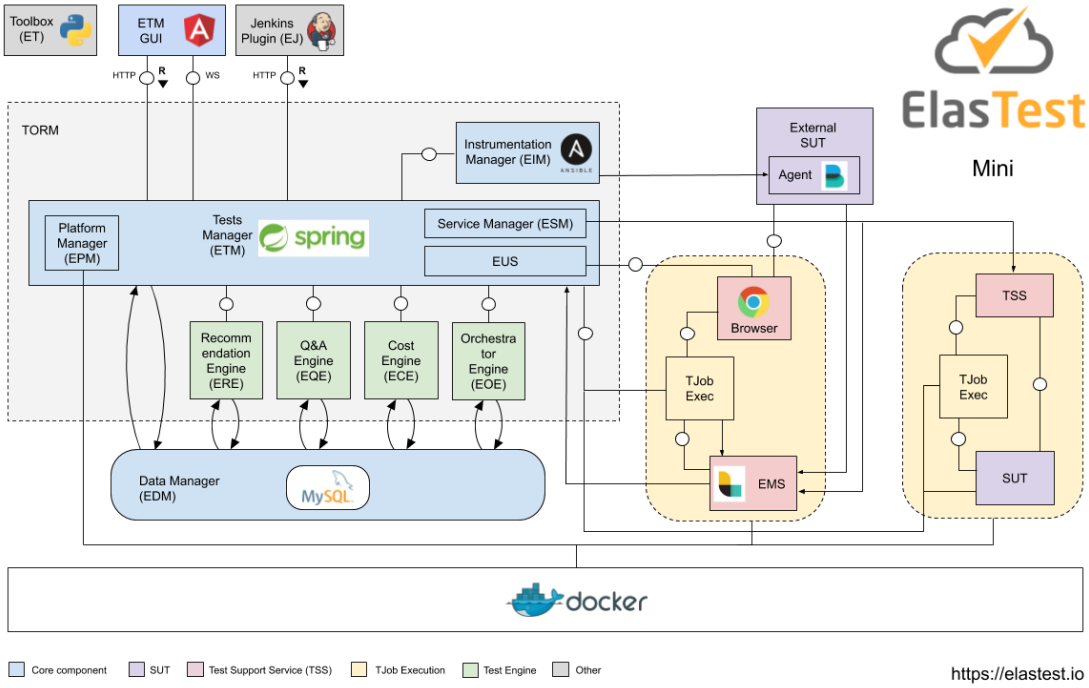

Figure 5. ElasTest Mini architecture diagram

The ElasTest EK mode depicted in Figure 6 is basically the ElasTest Mini mode deployed on top of Kubernetes. We had to rewrite completely the way ElasTest is deployed to comply with the deployment mechanisms of Kubernetes. We also had to rethink the concepts of TJob and associated services (TSS) and SUTs in order to comply with how Kubernetes manages networks and services. This required a lot of efforts, mainly on the ETM side (see D4.3) and the toolbox (see D6.4) responsible for the deployment of ElasTest. Note how the ETM talks to the Kubernetes cluster through the EPM (included in this mode as part of the ETM itself) to start and stop services and doing the monitoring. In this mode, we made use of Kubernetes namespaces that enables isolation of resources. Each TJob and related services are deployed in their own namespace (yellow dotted box around TJobs.) This avoids unwanted interactions between TJobs and allows services within a namespace talk to each other by name instead of IP address.

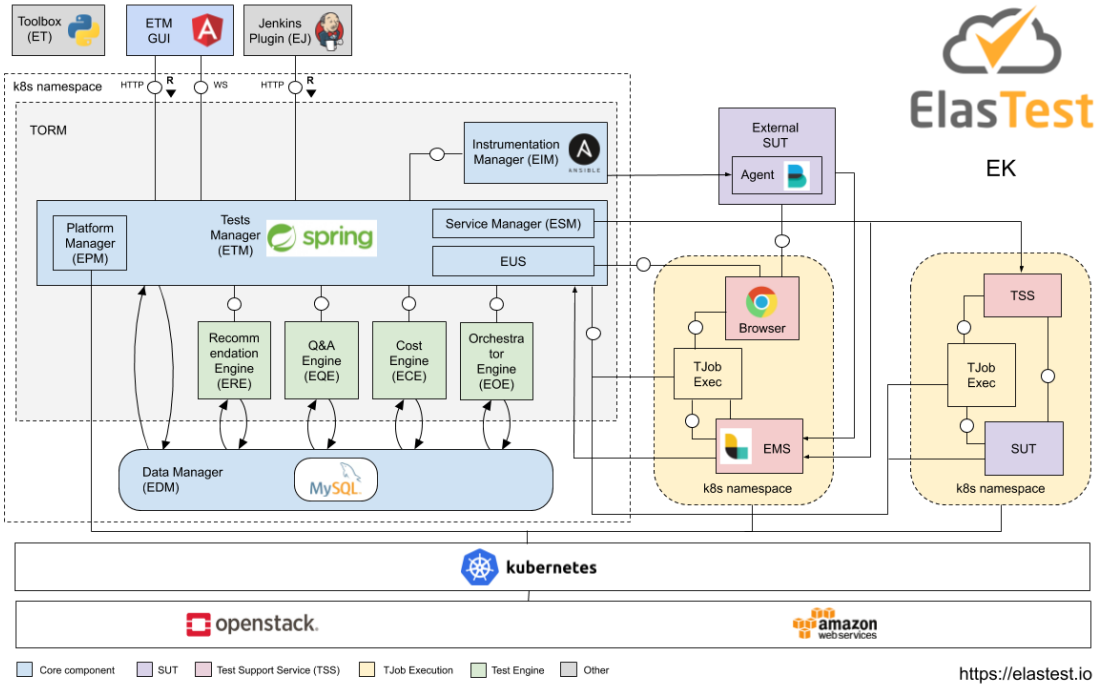

Figure 6. ElasTest EK architecture diagram

Finally, we developed the ElasTest HEK mode, standing for Highly-scalable ElasTest on Kubernetes, where ElasTest was deployed in a fully elastic mode. This mode resembles the Singlenode mode, but components deployments are elastic. For instance, the ElasticSearch can span through several nodes in the cluster, thus scaling as needed. Other core components can also be started on any node in the cluster, depending on availability, and everything works as if they were running on a single node, i.e., it is transparent to the end-user where in the Kubernetes cluster their jobs are running. The architecture diagram for ElasTest HEK mode is depicted in Figure 7. Note that the ESM, EPM and EUS are no longer within the ETM, but they run independently. ElasTest core components (EPM, ETM, EIM, ESM, EDM) and Test Engines (TEs) all share the same namespace. This namespace is known to any other ElasTest service (TSS, TJobs, SUTs) so that every component can talk with ElasTest core components.

Regarding elasticity, Figure 8 shows how different TJobs can span through several nodes. In the figure, a first TJob (*TJob1*) with its corresponding SUT (*SUT1*) is deployed within *worker1* (a node of the Kubernetes cluster). However, for a second TJob (*TJob2*), only the SUT could be started in *worker1*, due to lack of resources, and the remaining services needed for the TJob (namely, the test itself, *TJob2*, and the *EUS*), were deployed on *worker2*, a second Kubernetes node. This happened in a transparent way for the ElasTest end-user, and the TJob worked as usual, independently of where the resources were deployed.

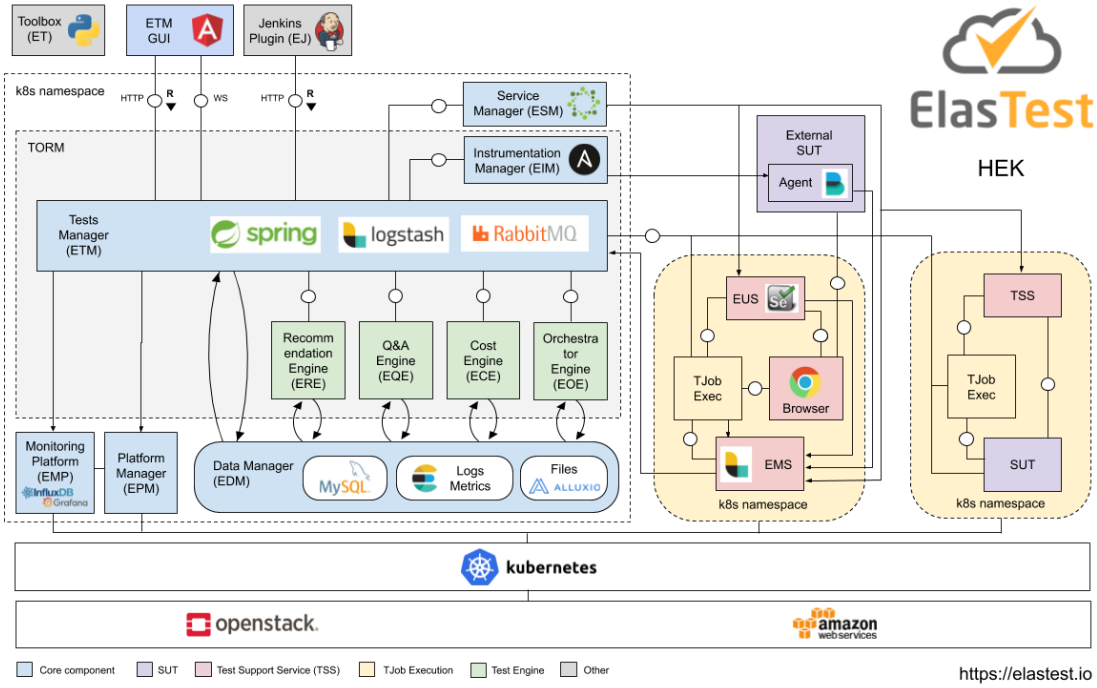

Figure 7. ElasTest HEK architecture diagram

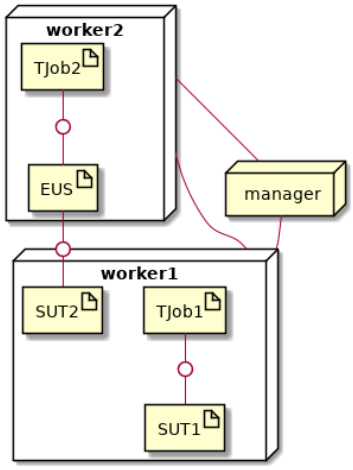

Figure 8. A TJob deployed through two different nodes (worker1 and worker2)

6 Conclusion

On a research perspective, there's still room for improvement. Modern distributed systems raised many challenges from a testing perspective. Early Kubernetes adopters, for instance, are starting to claim that this container orchestrator put a lot of complexity on the deployment side. Kubernetes is a complex system of systems, and this complexity is being faced by the teams that deploy their applications on top of it. Simplifying the testing process, and providing accurate and actionable information about this process, is key for testing Kubernetes-enabled applications.

Furthermore, testing in production is becoming more and more popular. This approach consists on deploying a new version of the application and routing a small percentage of actual traffic to the new version and see what happens. Some proponents say that this is the only way to know if your application will behave properly under real conditions: by exposing it to real traffic. Paraphrasing Robert Meany on Twitter: "If you're not testing in production then your users are testing in production."

However, when testing in production is sometimes difficult to know how the system is behaving. ElasTest could develop specific tools aimed at segregating actual traffic that belongs to the testing in production process, from any other kind of traffic, and helping to better identify the problems when they appear.

Several lessons have been learnt by the partners during the lifetime of the project. First, and most important, is that there are lots of schedulers for running tasks out there. At the beginning of the project we resorted to build our own CI server, capable of running TJobs defined as Docker containers. That was a mistake. No end-user will migrate their job definitions in whatever the CI server they have to ElasTest. We should have delegated that to Jenkins and focus on what was more interesting for those that were curious about ElasTest: observability and analytics. During the second period we have built interesting features like the log comparator, and we did a complete rewrite of the user interface, to make it more task focused so that it could highlight interesting events. The idea of evolving the architecture towards Kubernetes has been a big success, making some companies interested in the project, like Zooplus and Okteto. Only the future will tell what the use cases of these companies will turn ElasTest into.

7 References

[1] ElasTest project Description of Action (DoA) – part B. Amendment 3.

[2] Compositional Structures, Block diagrams – reference sheet, http://www.fmc-modeling.org/download/notation_reference/Reference_Sheet-Block_Diagram.pdf

[3] ElasTest Public Deliverable D6.3, ElasTest CI & validation system v2

[4] ElasTest Public Deliverable D3.2, ElasTest Platform cloud modules v2

[5] ElasTest Public Deliverable D4.3, Test Orchestration v2

[6] ElasTest Public Deliverable D4.4, Test Recommendation Engines v2

[7] ElasTest Public Deliverable D5.2, ElasTest Test Support Services v2

[8] Bertolino, A., 2007, May. Software testing research: Achievements, challenges, dreams. In *2007 Future of Software Engineering* (pp. 85-103). IEEE Computer Society.

[9] Apache 2.0 license terms. https://www.apache.org/licenses/LICENSE-2.0. Accessed on 07 March 2017.

[10] Grant Agreement number: 731535 - ELASTEST - H2020-ICT-2016-2017/H2020-ICT-2016-1. EUROPEAN COMMISSION. Communications Networks, Content and Technology. 11 November 2016.

8 ANNEX

Table 5 Technical Requirements List - ElasTest Core Components

ID	Component	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Release
ETM1	ETM	Manage projects	ElasTest user	CRUD operations on projects	create, edit, remove and update test projects to group TJobs and SuTs	AVAILABLE	R1-R2
ETM2	ETM	Create SuTs	ElasTest user	to create SuTs	I can specify how to start a SuT with the following options: Deployed by ElasTest (Docker, Docker-compose, commands) or Deployed Elsewhere.	AVAILABLE	R3
ETM3	ETM	Manage SuTs	ElasTest user	CRUD operations on SuTs	I can create, edit, remove and update SuTs	AVAILABLE	R3
ETM4	ETM	Create TJobs	ElasTest user	to create TJobs	I can specify what SuT should be tested and how to execute tests against it	AVAILABLE	R3
ETM5	ETM	Manage TJobs	ElasTest user	CRUD operations on TJobs	I can create, edit, remove and update TJobs	AVAILABLE	R3
ETM6	ETM	Execute TJobs	ElasTest user	to execute a TJob	logs, metrics and tests results can be recorded for further	AVAILABLE	R3

ID	Component	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Release
					inspection		
ETM7	ETM	Dashboard	ElasTest user	to see projects and last TJob executions in a single screen	I can have an overview of the status of the platform	AVAILABLE	R3
ETM8	ETM	Review TJob executions	ElasTest user	to review finished TJob executions	I can see what happened, especially in executions with failed tests	AVAILABLE	R3
ETM9	ETM	Test Support Services	ElasTest user	to specify what TSSs must be ready to use when a TJob is executed	tests in TJob can use selected TSS when testing the SuT	AVAILABLE	R1-R2
ETM10	ETM	Log analyzer	ElasTest user	to analyse, filter and mark logs gathered during TJob execution	troubleshooting a problem is easier than looking to plain log	AVAILABLE	R3
ETM11	ETM	Test case execution	ElasTest user	to review easily all information gathered during one specific test (logs, events and files)	I can focus on information related to a test (possible failed)	AVAILABLE	R4
ETM12	ETM	TestLink info management	ElasTest user	to see TestLink projects, test cases, suites, builds and test plans in ElasTest interface	I can see that information integrated with other TJobs and projects	AVAILABLE	R4

ID	Component	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Release
ETM13	ETM	TL Test plan execution	ElasTest user	to execute TL Test plans using browsers provided by ElasTest and recording all information from SuT and browsers	I can associate all that information to a bug report in case of test failure	AVAILABLE	R4
ETM14	ETM	Test Engines	ElasTest user	to start, use and stop a Test Engine	I can start the engine only when needed	AVAILABLE	R4
ETM15	ETM	Show platform information	ElasTest admin	to see the version and compilation date of ElasTest components	I can see if platform is updated or not	AVAILABLE	R3
ETM16	ETM	Core components integration	ElasTest user	to see core components' GUI integrated in the main ElasTest GUI	I can see the platform integrated	AVAILABLE	R4
ETM17	ETM	TSS Definition	TSS Creator	to know how to create the metadata file for TSSs	the TSS can be included in the list of available TSSs in the GUI	BACKLOG	
ETM18	ETM	Show logs and metrics in real-time	ElasTest user	to see logs and metrics from SuT and Tests execution	I can know what happened with SuT and Tests in case I want to solve any problem	AVAILABLE	R1-R2
ETM19	ETM	ESM Integration	ETM developer	to use ESM services	I can manage lifecycle of TSS	AVAILABLE	R1-R2

ID	Component	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Release
ETM20	ETM	EPM Integration	ETM developer	to use EPM services		BACKLOG	
ETM21	ETM	ElasTest micro	ElasTest user	a reduced version of ElasTest	I can try it with a very reduced resources requirements	BACKLOG	
ETM22	ETM	Update to Angular7	ElasTest user	a latest version of Angular		AVAILABLE	R7a
ETM23	ETM	Start EUS in singlenode mode when starting ETM and enable WebBrowsers section in this mode (Actually the test of this test is implemented in the EUS e2e tests)	ElasTest user	to start browsers manually	I can interact with a browser and watch the recording later	AVAILABLE	R7a
ETM24	ETM	Add auto-refresh to GUI Dashboard	ElasTest user	to refresh TJob Executions tables	I can see the status of the executions without having to refresh the page manually	AVAILABLE	R7a
ETM25	ETM	Instrumentalize a SuT in the EIM only during the	ElasTest user	to obtain monitoring traces of an External SuT	I can get monitoring traces of an external SuT during execution and stop receiving	AVAILABLE	R7a

ID	Component	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Release
		execution of a TJob			them if there is no execution in progress.		
ETM26	ETM	Use the EUS started when starting ElasTest for all runs in singlenode mode	ElasTest user	to reduce waiting time at the start of each TJob run and resource consumption in singlenode mode	I can use Web Browsers in my tests initiated by the same instance of EUS	AVAILABLE	R7a
ETM27	ETM	Establish new modes of execution of ElasTest (mini and singlenode)	ElasTest user	to have different options depending on the system I use	I can use ElasTest whatever the technical specifications of my system.	AVAILABLE	R7a
ETM28	ETM	Executions Comparator	ElasTest user	to be able to select two or more executions and compare	I can see the differences between the selected executions of a Job and compare their logs	AVAILABLE	R7a
ETM29	ETM	Configure an External ElastiSearch for a SuT	ElasTest user	to retrieve the SuT logs and metrics from an external Elasticsearch	I can see the logs and metrics for an external SuT retrieving them from an external Elasticsearch	AVAILABLE	R7a
ETM30	ETM	Add hosts to browsers used for	ElasTest user	to add a list of host necessaries for the browser	I can access to a SuT with an unknown name	AVAILABLE	R7b

ID	Component	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Release
		a manual test of TL		used in a manual test			
ETM31	ETM	Select browser for manual test execution (TestLink)	ElasTest user	to show the list of available browsers	I can select the browser that I want for my test	AVAILABLE	R7b
ETM32	ETM	Multi Axis TJob	Elastest user	to allow TJobs to have several configurations	with a single execution of a TJob will run once per configuration. You will be able to compare executions with different configurations between them.	AVAILABLE	R7a
ETM33	ETM	Run a TJob with parameters	ElasTest user	to be able to execute a TJob with parameters and be able to edit its values for each execution	I can achieve different results	AVAILABLE	R3
ETM34	ETM	Add attachments to a TJob execution	ElasTest user	to be able to add new evidences to a TJob execution using the ETM's API	I can store additional data apart from those generated by the ETM	AVAILABLE	R7b
ETM35	ETM	Elasticsearch Indices Management	ETM developer	to be able to remove indices in red state	I can fix the problems with the logs in singlenode	AVAILABLE	R7b

ID	Component	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Release
ETM36	ETM	Deploy ElasTest Mini on Kubernetes (EK)	ETM developer	to be able to deploy ElasTest Mini on k8s	I can provide elasticity to ElasTest	AVAILABLE	R7b
ETM37	ETM	Deploy ElasTest Mini on Kubernetes (EK) from the ElasTest Platform	ETM developer	to be able to deploy ElasTest Mini on Kubernetes using the platform and the EPM	I can configure ElasTest and ask the EPM for a k8s cluster to deploy Elastest	IN PROGRESS	
ETM38	ETM	Run TJobs on Kubernetes	ETM developer	to be able to deploy a TJob on Kubernetes	ElasTest can use more resources if doesn't have enough	AVAILABLE	R7b
ETM39	ETM	Runt SuTs on Kubernetes	ETM developer	to be able to deploy a SuTs on Kubernetes	ElasTest can use more resources if doesn't have enough	AVAILABLE	R7b
ETM40	ETM	Deploy TSSs on Kubernetes in EK	ETM developer	to be able to deploy a TSSs on Kubernetes	ElasTest can use more resources if doesn't have enough	AVAILABLE	R7b
ETM41	ETM	Deploy TEs on Kubernetes in EK	ETM developer	to be able to deploy a TEs on Kubernetes	ElasTest can use more resources if doesn't have enough	AVAILABLE	R7b
ETM42	ETM	Deploy Integrated	ETM	to be able to deploy the	ElasTest can use more	AVAILABLE	R7b

ID	Component	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Release
		Jenkins on Kubernetes and create Kubernetes examples	developer	Jenkins integrated on Kubernetes and also run Jobs on Kubernetes	resources if doesn't have enough		
ETM43	ETM	Deploy ElasTest Singlenode on Kubernetes (HEK)	ETM developer	to be able to deploy ElasTest Singlenode on Kubernetes	I can provide elasticity to ElasTest singlenode	IN PROGRESS	
ETM44	ETM	TL Crossbrowser	ElasTest user	to be able to execute a Test plan with multiple browsers at once, synchronizing actions	I can save time in running a test	AVAILABLE	R8
ETM45	ETM	ESS in TL	ElasTest user	to be able to use ESS in a Test plan execution		AVAILABLE	R8
ETM46	ETM	Export logs & metrics	ElasTest user	to be able to download the results information	I can inspect them in different tools/perform specific analysis	AVAILABLE	R8
EDM1	EDM	Provide MySQL Relational Database	ElasTest component	to store my data in a relational database	I can persist and query my structured data	DROPPED	
EDM2	EDM	Provide Elasticsearch	ElasTest component	to index my data in an Elasticsearch instance	I can index and search my unstructured data	DROPPED	
EDM3	EDM	Provide HDFS	ElasTest	to store my file data in a	I can safely store and retrieve	AVAILABLE	R1-R2

ID	Component	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Release
			component	scalable, fault tolerant filesystem	my file data		
EDM4	EDM	Provide Kibana	ElasTest component	to have access to a visualization tool	I can create and store visualizations of my data	DROPPED	
EDM5	EDM	Provide Alluxio	ElasTest component	to be able to store my file data in various backend filesystems (on premise or cloud) and retrieve the data at memory speeds	I can change backend filesystems transparently depending on my needs	DROPPED	
EDM6	EDM	Provide Cerebro	ElasTest component	to have access to a monitoring tool	I can monitor my Elasticsearch instance	DROPPED	
EDM7	EDM	Provide a REST API for backup/restore	ElasTest component	to be able to backup/restore all my data using a REST API	I can backup my data when I uninstall the Elastest platform and restore it later if I need to	DROPPED	
EDM8	EDM	Provide CloudFormation to deploy in AWS	Admin	to deploy EDM in AWS with elasticity	I can use the needed resources	BACKLOG	
EDM9	EPM	Provide Heat script to deploy in OpenStack	Admin	to deploy EDM in OpenStack with elasticity	I can use the needed resources	BACKLOG	
EDM10	EDM	EDM should be	Admin	to know the EDM health	I can check if services are	BACKLOG	

ID	Component	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Release
		monitored			working		
ESM1	ESM	Create a support service instance	TORM	to create a service instance	additional test functionality can be used	AVAILABLE	R1-R2
ESM2	ESM	Info on instance	TORM	to get a list of my service instances	I can reuse and understand what I have running	AVAILABLE	R1-R2
ESM3	ESM	Delete instance	TORM	to delete a service instance	I don't get charged for it	AVAILABLE	R1-R2
ESM4	ESM	Configure instance	TORM	to configure a service instance with new or updated parameters	the software can run as desired by end user	AVAILABLE	R6
ESM5	ESM	Register TSS offer	TORM	to register a TSS	I can offer my software as a service to the TORM	AVAILABLE	R1-R2
ESM6	ESM	Update TSS offer	TORM	to update a TSS's technical and business description	I can change my offer	AVAILABLE	R1-R2
ESM7	ESM	Delete TSS offer	TORM	to delete a TSS	I do not offer it anymore	AVAILABLE	R1-R2
ESM8	ESM	Register service instance with monitoring	ESM	to register the service instance endpoint with a monitoring service	the TSS provider can ensure guarantees offered to the TORM/end-user are met and adjustments can be made to ensure this	AVAILABLE	R6

ID	Component	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Release
ESM9	ESM	Public catalog to allow end users to "install" and "unsinstall" services in a current ElasTest instance	User	to install a TSS from a public catalog/registry	new TSSs can be installed in an ElasTest instance	AVAILABLE	R7b
ESM10	ESM	Allow deployment of services with Kubernetes template	TORM	to register a service with a Kubernetes YAML description	Kubernetes and docker-compose can be supported by the ESM	AVAILABLE	R7b
EIM1	EIM	Non-Intrusive	ElasTest user	instrumentation agents to be as less intrusive as possible	produce low overhead of the instrumentation on the software under test (SuT).	AVAILABLE	R1-R2
EIM2	EIM	Lightweight	ElasTest user	instrumentation agents to be as lightweight as possible	deploy them within the software under test (SuT).	AVAILABLE	R1-R2
EIM3	EIM	Configuration Management	ElastTest component (TORM)	to deploy automatically instrumentation agents in the target infrastructure	achieve automated installation of instrumentation agents across target compute environments, such as bare metal, VMs, cloud instances (IaaS such as AWS or	AVAILABLE	R3

ID	Component	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Release
					OpenStack) and container platforms (such as Kubernetes).		
EIM4	EIM	Interoperability	ElasTest user	to maintain interoperability across different operating systems (OS) and distributions	the agents should be designed to consume well-established operating system interfaces to guarantee interoperability; supporting Linux systems at least.	AVAILABLE	R1-R2
EIM5	EIM	Agent management	ElastTest component (TORM)	to perform CRUD operations to manage the lifecycle of instrumentation agents	EIM offers northbound interfaces which controls and orchestrates the operation of instrumentation agents.	AVAILABLE	R3
EIM6	EIM	Persistence	ElastTest component	to store configuration data in structured database (MySQL-like)	I can persist and query my structured data about the agents. (R3 only support MongoDB).	AVAILABLE	R7a
EIM7	EIM	Observability	ElasTest user	to extend the interface exposed by a software system for archiving enhanced observability of the software under test	I can have the ability to collect the logs and metrics and performance data from the software under test (SuT) through the instrumentation	AVAILABLE	R7a

ID	Component	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Release
				(SuT)	agents.		
EIM8	EIM	Portability	ElastTest component	to be able to install the instrumentation agents and manager across different compute environments.	To enable the installation, configuration and provisioning of the EIM along the rest of the ElastTest platform, and its instrumentation agents in the supported SuT environments.	AVAILABLE	R5
EIM9	EIM	Scalability	ElastTest component	to provide a scalable solution to the instrumentation of the software under test for both observability and controllability	To avoid the degradation of test (and the overall ElastTest platform) performance when running an increasing number of intrumentalized SuTs.	AVAILABLE	R5
EIM10	EIM	Allow instrument AWS resources	ElasTest user	to tnstrument AWS resources (VM, RDS, S3...) using native AWS monitoring capabilities (CloudWatch)	I can inspect what happens in those resources from ElasTest when execute tests against that AWS SuT	DROPPED	
EIM11	EIM	Allow instrument OpenStack resources	ElasTest user	to instrument OpenStack resources (VMs, ObjectStorage, networking) using native OpenStack monitoring capabilities	I can inspect what happens in those resources from ElasTest when execute tests against that OpenStack SuT	BACKLOG	

ID	Component	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Release
				(Ceilometer)			
EIM12	EIM	Allow instrument Kubernetes resources	ElasTest user	to instrument kubernetes resources (Pods, services, containers, nodes) using native Kubernetes monitoring capabilities (Prometheus)	I can inspect what happens in those resources from ElasTest when execute tests against that Kubernetes SuT	BACKLOG	
EIM13	EIM	Provide fine grain configuration of Beats agents	ElasTest user	to configure the event stream names	I can recognize the events in ElasTest GUI	BACKLOG	
EIM14	EIM	Controllability of CPU overload	ElasTest user	to achieve enhanced controllability of the stress level of the CPU	I can simulate a rise in the load/use of the CPU of the machine running a software under test (SuT)	AVAILABLE	R7b
EIM15	EIM	Controllability of Network failures	ElasTest user	to achieve enhanced controllability of the stress level of the network interface	I can simulate network packet loss of the machine running a software under test (SuT)	AVAILABLE	R7b
EIM16	EIM	Controllability of Container failures	ElasTest user	to control the failures (breakdown) of containerized software components	I can simulate failures of some or all replicas of some or all components of a software under test (SuT)	AVAILABLE	R7

ID	Component	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Release
EPM1	EPM	Abstraction of underlying virtualization technologies	ElasTest Component (TORM, Test Support Services, etc.)	to be able to orchestrate and manage virtualized resources (compute, network, storage) on any virtualization technology under consideration	the consumer does not need to care about the target virtualization technology but can define needed resources in a generic way. Finally, the EPM can easily support various virtualization technologies by making use of an adapter mechanism which will be used by the EPM in order to communicate with the virtualization technology of interest.	AVAILABLE	R1-R2
EPM2	EPM	Providing Northbound API	ElasTest Component (TORM, Test Support Services, etc.)	to interact with the EPM	the consumer can make use of the EPM via a ReSTful API	AVAILABLE	R1-R2
EPM3	EPM	Providing SDKs	ElasTest Component (TORM, Test Support Services, etc.)	to interact with the EPM	developers of other components can easily integrate with the EPM by making use of SDKs (libraries) provided for different	AVAILABLE	R1-R2

ID	Component	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Release
					languages (e.g. java, python)		
EPM4	EPM	Image operations	ElasTest Component (TORM, Test Support Services, etc.)	to retrieve information and manage images in the target virtualization environment	the EPM can check that images exist already and pull the image if needed	AVAILABLE	R1-R2
EPM5	EPM	Instance lifecycle operations	ElasTest Component (TORM, Test Support Services, etc.)	to be able to execute lifecycle operations such as start/stop, remove instances and retrieving information of the instance at runtime	the consumer of the EPM has full flexibility of executing lifecycle operations with the virtualized instances for a proper management at runtime	AVAILABLE	R1-R2
EPM6	EPM	Instance management operations	ElasTest Component (TORM, Test Support Services, etc.)	to be able to execute operations such as executing commands inside the instances and downloading/uploading files	the consumer of the EPM has full flexibility of accessing and interact with the virtualized instances	AVAILABLE	R1-R2
EPM7	EPM	Platform - Linux support	User	to run the EPM in Linux as the OS with native Docker	the user of ElasTest has the free choice of the underlying OS where ElasTest is running	AVAILABLE	R3
EPM8	EPM	Platform - Mac	User	to run the EPM in Mac OS	the user of ElasTest has the	AVAILABLE	R1-R2

ID	Component	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Release
		support		as the OS with Docker for Mac	free choice of the underlying OS where ElasTest is running		
EPM9	EPM	Platform - Windows support	User	to run the EPM in Windows with Docker toolbox as the OS and Docker for Windows	the user of ElasTest has the free choice of the underlying OS where ElasTest is running	BACKLOG	
EPM10	EPM	Containers monitoring	ElasTest Component (TORM, Test Support Services, etc.)	to monitor instances managed by the EPM	the consumer is able to retrieve monitoring information for further evaluation and troubleshooting	AVAILABLE	R1-R2
EPM11	EPM	Log forwarding	ElasTest Component (TORM, Test Support Services, etc.)	to forward logs to the configured endpoint	other parties can access those logs which can be used for further troubleshooting and debugging	AVAILABLE	R1-R2
EPM12	EPM	Support for Docker	ElasTest Component (TORM, Test Support Services, etc.)	to be able to allocate, manage and terminate Containers via Docker	the consumer can make use of Docker as a virtualization technology	AVAILABLE	R1-R2
EPM13	EPM	Support for Docker-compose	ElasTest Component	to be able to use Docker-compose templates	the consumer of the EPM can make use of docker-compose	AVAILABLE	R3

ID	Component	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Release
			(TORM, Test Support Services, etc.)		files provided in a package which will be used for resource management in a Docker environment		
EPM14	EPM	Platform Elasticity	User/ElasTest Component (TORM, Test Support Services, etc.)	elasticity provided by the EPM	either other ElasTest components can be scaled dynamically or the virtualized resources requested by other ElasTest components themself	AVAILABLE	R5
EPM15	EPM	Support for OpenStack	ElasTest Component (TORM, Test Support Services, etc.)	to be able to allocate, manage and terminate VMs via OpenStack	the consumer can make use of OpenStack as a virtualization technology	AVAILABLE	R5
EPM16	EPM	Kubernetes Cluster on Openstack	ElasTest Component (TORM, Test Support Services, etc.)	to be able to deploy Kubernetes in OpenStack	the consumer of the EPM want to deploy SUTs on top of Kubernetes	AVAILABLE	R7a
EPM17	EPM	Support for AWS	User, ElasTest Component (TORM, Test	to be able to allocate, manage and terminate VMs via AWS	the consumer can make use of AWS as a virtualization technology	AVAILABLE	R5

ID	Component	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Release
			Support Services, etc.)				
EPM18	EPM	Management of external machines	ElasTest Component (TORM, Test Support Services, etc.)	to be able to manage external machines which are not deployed by the EPM itself	the EPM can manage those machines via ssh in order to execute certain lifecycle operations to integrate them in the testing procedures	AVAILABLE	R4
EPM19	EPM	Kubernetes Cluster Management	ElasTest Component (TORM, Test Support Services, etc.)	to be able to manage a Kubernetes Cluster	the ElasTest platform can allocate virtual resources in an EPM on demand managed Kubernetes Cluster.	AVAILABLE	R7a
EPM20	EPM	Deployment of an OpenStack environment	User	to be able to deploy OpenStack on a physical machine	the ElasTest platform can allocate virtual resources in an OpenStack environment through the EPM	DROPPED	
EPM21	EPM	Kubernetes Cluster Scaling	ElasTest Component (TORM, Test Support Services, etc.)	a scalable Kubernetes cluster	EPM can manually scale out and in of workers	AVAILABLE	R7a
EPM22	EPM	Support for Open	User	to be able to deploy an	I can make use of OpenBaton	DROPPED	

ID	Component	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Release
		Baton		external SUT via OpenBaton	as an Orchestrator		
EPM23	EPM	Automatic start of adapter	ElasTest Component (TORM, Test Support Services, etc.)	to be able to request provisioning of resources without starting an adapter	the EPM can seamlessly provision resources	BACKLOG	
EPM24	EPM	Support for Ansible	User	to be able to use Ansible play	I can make use of Ansible to deploy Kubernetes	AVAILABLE	R7-Final
EMP1	EMP	Monitoring spaces	complex application / system developer / integrator	to be able to specify a separate monitoring space for the overall application	i can get easy, properly segregated access to my overall metric / log data	AVAILABLE	R1-R2
EMP2	EMP	monitoring subspaces	complex application / system developer / integrator	to be able to further separate metric and log stream of an application sub-component / microservice from rest of the components	I can easily locate the data stream coming from one component versus looking at a large set of data points from all possible metric generation sources in my large application (possibly distributed)	AVAILABLE	R1-R2
EMP3	EMP	API authentication and authorization	monitoring system user	my access to be authenticated and properly	no one else to be able to access the data streams from	AVAILABLE	R1-R2

ID	Component	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Release
				logged for safety as well as auditing purposes	my services as they may contain sensitive data		
EMP4	EMP	Receive system metric streams	ElasTest platform operator / and or any monitoring user	to be able to send relevant system metrics into the monitoring system	I can analyze data trends later or in real time	AVAILABLE	R1-R2
EMP5	EMP	Persist system metric streams	ElasTest platform operator / and or any monitoring user	my data points to be stored for a specified period in time	I can do detailed offline analysis of trends and / or investigate bottlenecks / problem areas with my application	AVAILABLE	R1-R2
EMP6	EMP	receive application log streams	ElasTest platform operator / and or any monitoring user	to use same service preferably to send my log messages too	I can do a proper correlation study of service degradation as observed from logs and the environment metric data	AVAILABLE	R7
EMP7	EMP	Persist application log streams	ElasTest platform operator /	my data points aka log parts to be stored for a specified	I can do offline / historical data analysis	AVAILABLE	R7

ID	Component	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Release
			and or any monitoring user	period of time			
EMP8	EMP	Data query capability	monitoring user	to be able to see stored data points	I can analyze data trends and observe system trends	AVAILABLE	R1-R2
EMP9	EMP	Metric visualization	monitoring user / application developer / operator	to be able to see charts / graphs visualizing data points in a meaningful way	I can comprehend quickly trends over time from metric streams from my services	AVAILABLE	R4
EMP10	EMP	Cross space/subspace correlated query capability	monitoring user / application developer / operator	to perform advance inter-space/domain data query	I can gain insight into correlation among various services on each other's performance	IN PROGRESS	
EMP11	EMP	Health check capability	application developer / operator	to be able to monitor the liveness of set of target services	I know as soon as possible when a service is down in order to react in a timely manner for restoring it	AVAILABLE	R3
EMP12	EMP	Alerting capability	operator	to be alerted if one of my services becomes dead	I can react quickly to restore the service	AVAILABLE	R3

ID	Component	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Release
EMP13	EMP	Online expression evaluation against metric data stream	operator	to ensure that the minimum service availability and contracts with my users are supported at all times and if any violation is notified to me as soon as possible	I satisfy my service level agreement terms	DROPPED	
EMP14	EMP	RESTful APIs	monitoring user / application developer / operator	easily integrate with the monitoring service with clearly defined interfaces	I can send metrics and perform control operations through my application code logic rather than interacting with the monitoring service in a standalone detached mode	AVAILABLE	R1-R2
EMP15	EMP	Availability of commonly used metric collectors (agents)	operator / application developer	to easily collect and send most commonly used system metrics into the monitoring platform	I can concentrate more on my system specific instrumentation and monitoring	AVAILABLE	R4
EMP16	EMP	Showing in ElasTest GUI monitoring information of all components	user	See all metrics and other monitoring information in ElasTest GUI	I can know the status of the system	AVAILABLE	R7
EMP17	EMP	API for querying TJob resource consumption	elastest cost engine	to be given a TJob ID, resource consumption data	I can compute the true cost of TJob execution based on cost	AVAILABLE	R7a

ID	Component	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Release
		parameters	process	across all known executions	models defined in ESM		
EMP18	EMP	Monitor Kubernetes clusters	ElasTest operator	to quickly see the status of my Kubernetes clusters	I can quickly identify hotspots and take corrective measures	AVAILABLE	R7b
EMP19	EMP	Individual container metrics visualization	monitoring system user	to see CPU, memory and networking stats for any container and not just the core components of ElasTest	I can visually see any abnormal spikes in data	AVAILABLE	R7

Table 6 Technical requirements list - ElasTest Test Support Services

ID	Component	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Release
EMS1	EMS	Receive beats	SuT / ElasTest component	to feed events to the EMS using the beats elasticsearch infrastructure	they can be processed by the monitoring machines	AVAILABLE	R1-R2
EMS2	EMS	Subscribe an elasticsearch instance to receive events	Tester / ElasTest user	my instance of elasticsearch to receive events flowing through a certain channel	I can detect anomalies and follow the running test's trace	AVAILABLE	R1-R2
EMS3	EMS	Subscribe a RabbitMq instance to events	Tester / ElasTest user	to receive events flowing through a certain channel using the rabbitmq infrastructure	I can detect anomalies and follow the running test's trace	AVAILABLE	R1-R2
EMS4	EMS	Subscribe the dashboard to events	Tester / ElasTest user	the dashboard to receive events automatically, without having to implement functionality to communicate my (dashboard) end-point	I can visualize the data from the events	AVAILABLE	R4
EMS5	EMS	Subscribe persistence to events	Tester / ElasTest user	the persistence manager to receive events automatically, without having to implement functionality to communicate my (EDM) end-point	I can extract information from the data offline	AVAILABLE	R4

ID	Component	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Release
EMS6	EMS	Specify rules to route events through channels	Tester / ElasTest user	to be able to route events to subscribers depending only on the "channel" stamped in the input event	I can select which input channels to listen to	AVAILABLE	R4
EMS7	EMS	Simple filtering rules	Testers / Elastest user	to be able to add subscriptions so that EMS filters events according to conditions expressed in my subscriptions	I can better select which events to receive	AVAILABLE	R4
EMS8	EMS	Deploy sampled-based and signal-based signals	Tester / ElasTest user	to extract a field from certain types of events to be considered as sampled values from a signal, which I want to reconstruct and work with.	I can build on it, aggregating its values and combining them with each other to synthesize useful information	AVAILABLE	R5
EMS9	EMS	Deploy correlation machines	Tester / ElasTest user	to create notification events based on received events, their relative timing and arrival and other contextual information	I can delegate to the EMS the finding of patterns of events in the stream of observations from the test	AVAILABLE	R5
EMS10	EMS	Unsubscribe endpoints	Tester / ElasTest user	to stop the flow of events to a subscribed endpoint	I can remove it safely	AVAILABLE	R5
EMS11	EMS	Undeploy routing rules	Tester / ElasTest user	to stop applying certain rules of routing	I can regroup the events in new ways	AVAILABLE	R5

ID	Component	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Release
EMS12	EMS	Undeploy machines	Tester / ElasTest user	to stop executing certain monitoring machine	The system avoids computing data which I no longer need	AVAILABLE	R5
EMS13	EMS	Subscribe websockets	Tester / ElasTest user	to receive events over a websocket	I can easily describe tests that guide the testing process depending on the observations of the current test	AVAILABLE	R4
EMS14	EMS	Receive beats in many formats	SuT / ElasTest component	to feed events to the EMS using other infrastructure (like Zabbix)	I can easily produce events within the SuT to assess the outcome of tests	BACKLOG	
EMS15	EMS	Extend the Monitoring Machines language	Tester / ElasTest user	to have extra features in the Monitoring Machines language	I can describe my test	IN PROGRESS	
EMS16	EMS	JSON template in Monitoring Machines	Tester / ElasTest user	to receive events with rich data	I can analyse them more easily	AVAILABLE	R7b
EMS17	EMS	Emission of structured output as JSON	ElasTest user	output events to be structured types	The TJob receives rich information which makes it easier to investigate the cause of a test failure	AVAILABLE	R7b
EMS18	EMS	Monitoring	ElasTest user	to calculate durations and	define tests in terms of	AVAILABLE	R7b

ID	Component	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Release
		Machines can interpret a timestamp in a string field as a numeric value		delays	absence or presence of events within an interval of time.		
EMS19	EMS	Accessing the value of streams at previous instants	ElasTest user	to perform a computation between successive values of the same stream	I can compute the delay between two successive events	AVAILABLE	R7b
EMS20	EMS	Vector notation for streams	ElasTest user	to scale the tests easily	I reduce the duplication of streams	AVAILABLE	R7b
EMS21	EMS	Output events at websocket channel at port 8181	ElasTest user	to connect to the websocket endpoint and receive the events	I can get the verdicts from the EMS specifications	AVAILABLE	R7b
EMS22	EMS	Offline Monitoring	ElasTest user	to compute a specification on a dump of a Tjob execution	I can evaluate specifications without the need to re-run the same actions on a SuT	AVAILABLE	R7b
EMS23	EMS	If-then-else expression in EMS language	ElasTest user	to assign the value of an if-then-else expression to a stream	I can express richer streams more easily	AVAILABLE	R7b

ID	Component	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Release
ESS1	ESS	Security testing via TJobs	Tester / ElasTest user	to do provide tests of my SuT as TJobs and get a security scan report	I can integrate security testing to my tests without additional effort	AVAILABLE	R1-R2
ESS2	ESS	Unprotected url detection	Tester / ElasTest user	to receive list of unprotected URLs	I can protect them and prevent attackers from stealing sensitive information of the users of my web site	AVAILABLE	R3
ESS3	ESS	Insecure cookie detection	Tester / ElasTest user	to receive list of insecure cookies (sensitive cookie values sent via http channel)	I can secure them and prevent attackers from impersonating the users of my web site	AVAILABLE	R3
ESS4	ESS	Scanning for common vulnerabilities (unauthenticated)	Tester / ElasTest user	to be able to automatically test whether my Web Application is vulnerable to common Web Application security weaknesses	I can avoid the situation in which malicious actors cannot easily hack my web site by exploiting the most common vulnerabilities .	AVAILABLE	R4
ESS5	ESS	Deep scanning for common vulnerabilities (authenticated)	Tester / ElasTest user	to do a deep security scan covering parts of my Web application that are otherwise difficult to be reached by automatic Web vulnerability scanners	I can avoid the situation in which common Web application vulnerabilities are not missed by my security scanner due to the reachability issue	AVAILABLE	R5
ESS6	ESS	Privacy check	Tester /	to be able to detect privacy	the privacy of my users are not	AVAILABLE	R6

ID	Component	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Release
		(without GUI)	ElasTest user	leaks in my Web application	compromised to malicious third party web sites		
ESS7	ESS	Privacy check (with GUI)	Tester / ElasTest user	to be able to detect privacy leaks in my Web application	the privacy of my users are not compromised to malicious third party web sites	IN PROGRESS	
EUS1	EUS	W3C WebDriver compatibility	Tester / ElasTest user	to support standard W3C WebDriver API (based on JSON messages over REST)	EUS is backwards compatible with existing technologies such as Selenium and Appium	AVAILABLE	R1-R2
EUS2	EUS	Basic media evaluation	Tester / ElasTest user	to read audio level and RGB colors of given UI elements	they can be used as test oracle to feed test assertion	BACKLOG	
EUS3	EUS	Event subscription	Tester / ElasTest user	to subscribe to UI elements	tests can receive events notification	BACKLOG	
EUS4	EUS	Measure end-to-end latency of a WebRTC session	Tester / ElasTest user	to measure end-to-end latency	that tests can know whether or not a WebRTC service has operational real-time performance rates	BACKLOG	
EUS5	EUS	Measure quality (audio|video) of a WebRTC session	Tester / ElasTest user	to measure full-reference QoE indicators both for audio and video	that tests can find out the quality of the WebRTC media in an easy way	BACKLOG	
EUS6	EUS	Remote control	Tester /	to monitor remote sessions for	I can watch in real-time and	AVAILABLE	R1-R2

ID	Component	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Release
			ElasTest user	browsers and mobile	interact with browser/mobile sessions		
EUS7	EUS	WebRTC stats	Tester / ElasTest user	to read WebRTC statistics	test can read WebRTC QoS indicator in a seamless way	AVAILABLE	R4
EUS8	EUS	Browser logging gathering	Tester / ElasTest user	to read browser logs	tests are aware on the underlying logging info to trace potential failures	AVAILABLE	R3
EUS9	EUS	Mobile logging gathering	Tester / ElasTest user	to read mobile logs	tests are aware on the underlying logging info to trace potential failures	BACKLOG	
EUS10	EUS	Upload a file to browser context	Tester / ElasTest user	to upload a file to browser context	It can be used in browser context	AVAILABLE	R7
EUS11	EUS	Deploy ElasTest browsers in AWS	Tester / ElasTest user	to start, stop and record browsers in AWS and get logs		AVAILABLE	R7b
EUS12	EUS	Deploy ElasTest browsers on K8s	EUS developer	to deploy browsers on Kubernetes and get recordings and logs	Elastest be able to ask Kubernetes for more resources if it doesn't have enough	AVAILABLE	R7b
EDS1	EDS	Minimal orchestrator	ElasTest User	to orchestrates sensors and actuators	demonstrator can initiate and connect sensors and actuators	AVAILABLE	R4

ID	Component	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Release
		EDS			and connect them with a logic		
EDS2	EDS	Min. EDS orchestration	ElasTest User	to call and initiate required sensors and actuators	the sensors and actuators are live and can provide data	AVAILABLE	R4
EDS3	EDS	Demonstrator logic	ElasTest User	to connect sensors and actuators via logic	an IoT application can be realized	AVAILABLE	R4
EDS4	EDS	EDS communication activities	ElasTest User	to communicate with oneM2M standard	IoT application can be realized in an industrial context	AVAILABLE	R4
EDS5	EDS	Min. EDS tracks life cycle of sensors and actuators	ElasTest User	efficient resource handling	clean up and track activities of min. EDS	AVAILABLE	R4
EDS6	EDS	Scalability of an application	ElasTest User	combination of demonstrator applications	to test an SiL	AVAILABLE	R4
EDS7	EDS	Reusability of an application	ElasTest User	combination of demonstrator applications	to form an SiL as a combination of SiS	AVAILABLE	R4
EDS8	EDS	Mechanisms for collecting QoS	ElasTest User	to collect QoS metrics	to analyze QoS	DROPPED	
EDS9	EDS	Add basic set of sensors (about 7	ElasTest User	can use various sensors	I can them in an IIoT application	AVAILABLE	R7b

ID	Component	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Release
		types)					
EDS10	EDS	Add basic set of actuators	ElasTest User	can use various actuators	use them in an IIoT application	AVAILABLE	R5
EDS11	EDS	Basic configuration for actuators and sensors.	ElasTest user	set of meaningful and configurable parameters for devices	I can configure the devices during run time to emulate device behavior.	AVAILABLE	R7a
EBS1	EBS	Provide Spark	ElasTest component	to have access to a big data processing framework	I can process my data using various big data algorithms	AVAILABLE	R1-R2
EBS2	EBS	Provide a REST API for healthcheck of Spark	ElasTest component	to be able to monitor the health of my Spark cluster	I can take appropriate action in the case of failure	AVAILABLE	R3

Table 7 Technical requirements list - ElasTest Test Engines

ID	Component	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Release
EOE1	EOE	Topology generation	Tester / ElasTest user	to define some kind of test orchestration notation	users can define a TiL by aggregating different T-Jobs	AVAILABLE	R5
EOE2	EOE	Jenkins DSL notation	Tester / ElasTest user	to leverage Jenkins shared library technology to create orchestration topology	users can define a TiL by aggregating different T-Jobs	AVAILABLE	R6
EOE3	EOE	EOE DSL parser	Tester / ElasTest user	EOE capable of parsing Jenkins notation	EOE is integrated in ElasTest	BACKLOG	
EOE4	EOE	EOE	Tester /	EOE able to support data-	EOE is integrated in ElasTest	BACKLOG	

ID	Component	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Release
		communication manager	ElasTest user	driven orchestration approach			
EOE5	EOE	EOE proxy	Tester / ElasTest user	EOE to intercept requests from ETM to TSSs	shared sessions among different tests	BACKLOG	
EOE6	EOE	Reference implementation	Tester / ElasTest user	to create some reference implementation of the data-driven approach, for example using the JUnit 5 extension model	the adoption of the data-driven approach can be easy for ElasTest users	BACKLOG	
EOE7	EOE	Test augmentation	Tester / ElasTest user	new TJobs can be added to the orchestration	I can reproduce custom operational conditions of the SUT or non-functional attributes (such as performance, scalability or reliability)	BACKLOG	
EOE8	EOE	Include extra checkpoints	Tester / ElasTest user	to integrate techniques (new or existing) to include automated assertions in existing orchestrations	I can improve test coverage or orchestrated T-Jobs by adding extra checkpoints (especially in data-drive approach)	BACKLOG	
EOE9	EOE	Include resource information when orchestrating test	Tester / ElasTest user	to ensure my test cases when running in parallel fit within the resources	test cases don't fail because of a high load or scarce resources	BACKLOG	

ID	Component	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Release
		cases		available			
ERE1	ERE	Data Preprocessing	Admin user	to automatically preprocess user data	I can minimize time and effort spent on data preparation	AVAILABLE	R1-R2
ERE2	ERE	Data Load	Admin user	to upload user data to cloud	it can be fed to the machine learning model	AVAILABLE	R1-R2
ERE3	ERE	Training on User Data	Admin user	to launch the execution of ML algorithms	I can train machine learning model on user provided data	AVAILABLE	R1-R2
ERE4	ERE	Flexible Storage	Admin user	a flexible solution for storing user data	I can choose storage type that fits best the size of my datasets	AVAILABLE	R3
ERE5	ERE	Authentication	Admin / Tester	to log in and authenticate as a registered user	I can get access to proprietary services	BACKLOG	
ERE6	ERE	Admin Dashboard	Admin user	to separate role and UI for managing data load and training	I can ensure control over resource-consuming procedures	AVAILABLE	R3
ERE7	ERE	Configure default settings	Tester	to configure and save default settings for queries	I can choose the model that I want to query	AVAILABLE	R3
ERE8	ERE	Tester UI	Tester	a user interface	I can query for, view and interact with recommendations generated by ERE	AVAILABLE	R1-R2

ID	Component	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Release
ERE9	ERE	Recommend TJobs for reuse	Tester	to receive recommendations, based on a natural language descriptions, on automated testcases to reuse	I can increase code reusability, save time and effort	AVAILABLE	R1-R2
ERE10	ERE	Recommend manual test cases for reuse	Tester	to receive recommendation on manual test steps to reuse, based on functionality description,	I can increase knowledge reuse, improve test cases quality, support less experienced testers	BACKLOG	
ERE11	ERE	Recommend new TJobs from natural language description	Tester	to receive newly generated code for automated testcases, based on natural language description	I can save time and resources spent on test automation	AVAILABLE	R4
ERE12	ERE	Learning from tester feedback	Tester	a convenient way to amend received recommendations and return them to the system	the feedback is used for re-training to improve future recommendations	BACKLOG	
ERE13	ERE	Inline help	Tester	to have access to inline help as I navigate through	I can get immediate explanations and tips for using various features	AVAILABLE	R4

ID	Component	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Release
				the UI			
ERE14	ERE	End-to-end preprocessing pipeline	Admin user	to automatically crawl a software repository (Java) and extract relevant training data	I can eliminate manual effort required to gather and analyse data	AVAILABLE	R5
ERE15	ERE	Pre-trained models	Admin /Tester	ERE to provide off-the-shelf pretrained model that can be customized using my own data	I can leverage software engineering knowledge captured in large open source repositories, decrease training time and cost	AVAILABLE	R7
ERE16	ERE	Trial version of Recommender	Tester	to get free access to a subset of ERE functionality	I can try and evaluate the component	AVAILABLE	
ERE17	ERE	View more context for reusable test cases	Tester	to view class members corresponding to the recommended test methods and easily access parent repositories	I can fully understand the recommended code	AVAILABLE	R7
EQE1	EQE	Data Preprocessing	Admin	to automatically preprocess user data	I can minimize manual effort on data preparation	BACKLOG	
EQE2	EQE	Data Load	Admin	to load user data	it can be fed to a machine learning model	BACKLOG	
EQE3	EQE	Training on user	Admin	to launch training	I can generate a Q&A model	IN	

ID	Component	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Release
		data			trained on user data	PROGRESS	
EQE4	EQE	Interactive UI	Tester	an interactive UI	I can easily input questions and read responses	IN PROGRESS	
EQE5	EQE	Querying Q&A model	Tester	to send queries to the selected model and receive responses generated by the model	I can receive relevant responses based on the previous knowledge ingested and processed by EQE	IN PROGRESS	
EQE6	EQE	Pre-trained EQE model	Tester	EQE to provide a model pretrained on open data	I can benefit from EQE functionality even if I do not have own data	IN PROGRESS	
EQE7	EQE	Re-training with user data	Admin	to re-train off-the-shelf model using my custom data	I can receive responses that are better tailored to my domain while still leveraging open data	BACKLOG	
EQE8	EQE	Launching the component	Admin	to open/close the detached panel using the platform Dashboard	I can control the presence of detached panel	AVAILABLE	R7-Final
ECE1	ECE	Receive TJob information from TORM	ElasTest Component	to get TJob Information from the TORM	ECE can estimate a cost based on which services are used in the test	AVAILABLE	R3
ECE2	ECE	Receive TJob information from	ElasTest Component	to get the Service Type cost definitions	ECE can generate a Static cost estimation	AVAILABLE	R3

ID	Component	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Release
		ESM					
ECE3	ECE	Static Estimation of a TJob Cost	Tester	to estimate the TJob cost	I can see in beforehand a Static estimation of his TJob execution cost	AVAILABLE	R3
ECE4	ECE	Retrieve Monitoring information	ElasTest Component	to generate a real cost report based on the platform usage	the End-User sees the real cost of the TJob Execution	AVAILABLE	R6
ECE5	ECE	Use Lifecycle events	ElasTest Component	to generate a real cost report based on the platform usage inferred based on the start and stop events of a TJob	the End-User sees the real cost of the TJob Execution	DROPPED	
ECE6	ECE	Actual calculation of a TJob cost	Tester	to compute the true TJob cost based on monitored metric data	I can see in true TJob execution cost post execution	AVAILABLE	R7b
ECE7	ECE	Extend cost model to support all ElasTest support service	ElasTest Component	to capture service specific cost parameters with the model	I can use service specific nomenclature to define a reasonable cost model for my service	AVAILABLE	R5
ECE8	ECE	Make costs available via REST	ElasTest Component	to get the cost of running my tests programmatically and not only via the	as Elastest component I can query costs via an API to show to the	BACKLOG	

ID	Component	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Release
		APIs		dashboard	user in a consolidated view		
ECE9	ECE	Make rest cost feature optional depending on the availability of EMP service	Tester	to know the real cost feature is not supported in a release outright and not see a broken feature error	I know exactly what is supported and what is not	AVAILABLE	R7a

Table 8 Technical requirements list - ElasTest Integrations with External Tools

ID	Component	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Release
ET1	ET	Install full latest version	Any User (Tester, developer, operator...)	to install the latest full ElasTest	testers can configure SuTs and TJobs with all the capabilities	BACKLOG	
ET2	ET	Install lite latest version	Any User (Tester, developer, operator...)	to install the latest lite ElasTest	testers can configure SuTs and TJobs with and use the basic capabilities of ElasTest	BACKLOG	
ET3	ET	Install full specific version	Any User (Tester, developer, operator...)	to install a specific version of the full ElasTest	testers can configure SuTs and TJobs with all the capabilities	BACKLOG	
ET4	ET	Install lite specific version	Any User (Tester, developer, operator...)	to install a specific version of the lite ElasTest	testers can configure SuTs and TJobs with and use the basic capabilities of ElasTest	BACKLOG	
ET5	ET	Check ElasTest Status	Any User (Tester, developer, operator...)	to check the ElasTest status (running/stop/failed/unstable...)	the user can check if the platform is ready to be used	AVAILABLE	R1-R2
ET6	ET	Start full latest version	Any User (Tester, developer, operator...)	to start the latest full ElasTest docker image	testers can configure SuTs and TJobs with all the capabilities	AVAILABLE	R4
ET7	ET	Start lite latest	Any User (Tester,	to start the latest lite	testers can configure SuTs and	AVAILABLE	R1-R2

ID	Component	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Release
		version	developer, operator...)	ElasTest	TJobs with and use the basic capabilities of ElasTest		
ET8	ET	Start full specific version	Any User (Tester, developer, operator...)	to start a specific version of the full ElasTest	testers can configure SuTs and TJobs with all the capabilities	AVAILABLE	R4
ET9	ET	Start full latest version without ERE	Any User (Tester, developer, operator...)	to start the latest full ElasTest (without the ElastTest Recommendation Engine)	testers can configure SuTs and TJobs with all the capabilities except ElastTest Recommendation Engine	AVAILABLE	R1-R2
ET10	ET	Start lite specific version	Any User (Tester, developer, operator...)	to start a specific version of the lite ElasTest	testers can configure SuTs and TJobs with and use the basic capabilities of ElasTest	AVAILABLE	R1-R2
ET11	ET	Retrieve connection information	Any User (Tester, developer, operator...)	to know where to connect to access the ElasTest platform	users can connect to the Platform to operate	AVAILABLE	R1-R2
ET12	ET	Retrieve information of deployed components	Any User (Tester, developer, operator...)	to know which components are available in the running ElasTest	I can obtain information of each of the components, such as status, port, consuming resurces...	AVAILABLE	R3
ET13	ET	AWS Cloud Elastest Deployment	Any User (Tester, developer,	to configure and Run an Elastest Instance in the AWS	I am able to have a fully operating ElasTest running in	AVAILABLE	R4

ID	Component	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Release
			operator...)	cloud, with no or little effort	the cloud.		
ET14	ET	AWS Cloud Elastic Elastest	Any User (Tester, developer, operator...)	to configure and Run an Elastest that can be seamlessly elastic	I am able to launch (virtualy) any number of TJobs as the resources would be elastic	IN PROGRESS	
EJ1	EJ	Install plugin	Tester (Jenkins User)	to install ElasTest Plugin with default plugin installer	ElasTest configuration properties can be set on Jenkins Configuration and ElasTest can be used in Jenkins Jobs	AVAILABLE	R5
EJ2	EJ	Install plugin for pipelines	Tester (Jenkins User)	to install ElasTest Plugin with default plugin installer	ElasTest configuration properties can be set on Jenkins Configuration and ElasTest can be used in Jenkins Pipelines	AVAILABLE	R5
EJ3	EJ	Global configuration of ElasTest platform	Tester (Jenkins User)	to configure global ElasTest settings: - Version - type (lite/full)	the plugin can manage the ElasTest platform with the appropriate configuration	BACKLOG	
EJ4	EJ	Jenkins Managed ElasTest	Tester (Jenkins User)	ElasTest plugin to be able to launch a ElasTest with the specified global configuration	any job can be able to launch and use this ElasTest.	BACKLOG	
EJ5	EJ	External Managed	Tester (Jenkins	ElasTest plugin to be able to use an ElasTest runing in	any job can be able to use this	AVAILABLE	R5

ID	Component	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Release
		ElasTest	User)	other location (local or external)	ElasTest.		
EJ6	EJ	Configure Docker Image SuT	Tester (Jenkins User)	to configure in a Job, ElasTest to recognize and launch a provided an image of a SuT	ElasTest can work with that SuT	BACKLOG	
EJ7	EJ	Configure externally hosted SuT	Tester (Jenkins User)	to configure in a Job, ElasTest to recognize an externally hosted SuT	ElasTest can work with that SuT	BACKLOG	
EJ8	EJ	Configure personalized SuT	Tester (Jenkins User)	to provide an script (mvn, sh, py...) that the ElasTest plugin will use to launch a SuT	ElasTest can work with that SuT	BACKLOG	
EJ9	EJ	Configure Docker Image TJob	Tester (Jenkins User)	to configure ElasTest to recognize and launch a provided an image of a TJob	ElasTest can execute that TJob	BACKLOG	
EJ10	EJ	Configure personalized TJob	Tester (Jenkins User)	to provide an script (mvn, sh, py...) that the ElasTest plugin will use to launch a TJob	ElasTest can execute that TJob	BACKLOG	
EJ11	EJ	Export results	Tester (Jenkins	to export the result of the tests executed in a readable	user can read detailed results	AVAILABLE	R6

ID	Component	Title	As a <type of user>	I want <some goal>	so that <some reason>	Status	Release
			User)	format			
EJ12	EJ	Export logs	Tester (Jenkins User)	to export all the generated logs for SuT and TJobs	user can retrieve them for further operations.	AVAILABLE	R6
EJ13	EJ	Request for TSS	Tester (Jenkins User)	to use the TSS's functionalities	users can use browsers in their tests	AVAILABLE	R5
EJ14	EJ	Bind a Job to a specific ET Project	Tester (Jenkins User)	to create the TJob associated with a Job inside of a specific ET Project	the executions can be organized	AVAILABLE	R5
EJ15	EJ	Choose a SuT defined on ElasTest	Tester (Jenkins User)	to don't have to start the SuT in the jenkins pipeline	users can reuse SUTs defined on ElasTest	AVAILABLE	R5
EJ16	EJ	Monitoring SuTs started from a Jenkins pipeline (doesn't work on k8s)	Tester (Jenkins User)	to send logs and metrics from a SUT started on Jenkins to ElasTest	users can analyze those logs and metrics in ElasTest	AVAILABLE	R5