D8.4	
Version	1.0
Author	URJC, ATOS
Dissemination	PU
Date	31-12-2019
Status	FINAL

ElasTest

D8.4 ElasTest Open Source Software Community 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	WP8
WP leader	Malena Donato
Deliverable nature	PU
Lead editor	Francisco Gortázar, Malena Donato
Planned delivery date	31-12-2019
Actual delivery date	31-12-2019
Keywords	ElasTest Community, Governance model, how to contribute, Resources, ElasTest components and definitions

Funded by the European Union

License

This is a public deliverable that is provided to the community under a **Creative Commons Attribution-ShareAlike 4.0 International** License:

http://creativecommons.org/licenses/by-sa/4.0/

You are free to:

Share — copy and redistribute the material in any medium or format.

Adapt — remix, transform, and build upon the material for any purpose, even commercially.

The licensor cannot revoke these freedoms as long as you follow the license terms.

Under the following terms:

Attribution — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.

ShareAlike — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.

No additional restrictions — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.

Notices:

You do not have to comply with the license for elements of the material in the public domain or where your use is permitted by an applicable exception or limitation.

No warranties are given. The license may not give you all of the permissions necessary for your intended use. For example, other rights such as publicity, privacy, or moral rights may limit how you use the material.

For a full description of the license legal terms, please refer to:

http://creativecommons.org/licenses/by-sa/4.0/legalcode

Contributors

Name	Affiliation
Francisco Gortázar	URJC
Micael Gallego	URJC

Version History

Version	Date	Author(s)	Description of changes
0.1	07/11/2019	Francisco Gortázar	Initial TOC
0.2	15/11/2019	Francisco Gortázar	Initial content. KPIs from previous deliverable (up to M18)
0.3	16/12/2019	Francisco Gortázar	Sections 2, 3 and 4.
1.0	31/12/2019	Micael Gallego	Final revision

Table of Contents

List of figures

List of tables

Glossary of Acronyms

Acronym	Definition
CLA	Contributor License Agreement
DoA	Description of Action
FOSS	Free and Open Source Software
GA	Grant Agreement
WP	Work Package

1 Executive Summary

This deliverable summarizes the activities we did around the **ElasTest Open source community** and the numbers we got for the second period. During the first period we revised and refined our strategy, established the community, the rules of governance and code of conduct, and the necessary channels like website, YouTube channel, Twitter account, and mailing list. In this second period we worked towards dissemination of the achievements within the project and tried to gather visitors to our community channels.

2 Introduction

Building an open source community is hard. Open source software is based on two main forces: individuals willing to participate during their spare time because they find the project interesting, employers who want to push a project forward because it is core to their business. The former is already participating on open source software projects, and most probably they don't have enough time to be involved in a new project. The latter will only join the community if they find the project key to their business.

During this second period, project members have defined a **strategy** to bringing people to the community with different carefully selected complementary actions:

- **Attending and presenting at industrial events:** In that way, potential users can know how ElasTest can improve their testing strategy to build better products with less defects.
- **Maintaining a good cadence of releases:** So users feel the product evolving to meet their needs and to be in line with advances in the state of the art of testing tools and practices
- **Encouraging users to fill in issues and request new features through our GitHub issues system:** So the project members can define the evolution of the product taking into account users' feedback.
- **Bring companies to the product and build synergies with those companies:** To better understand their needs and to make specific improvements over the product for those companies.
- **Encouraging users to ask questions publicly through our mailing list, instead of asking them in private emails:** This action creates a sense of community as some users start answering questions for other users.
- **Rebuilding the documentation page on its entirety:** To make easier to discover features with guided, step by step, tutorials
- **Providing an ElasTest live instance:** For better assessment of the capabilities of the product avoiding the need to download and install it
- **Complete rebuild of the UI of the product:** Based more on common uses cases and following our users' recommendations to improve discoverability of the features.
- **Rebuilding the communication strategy for the product**

The next section presents each of the actions in more detail. Section 3 presents how ElasTest open source community has been promoted. Section 4 introduces the actual numbers of the ElasTest community. Finally, Section 5 introduces the future of the community and the project, and the plans to make the project sustainable.

3 Promotion of the ElasTest Open Source Community

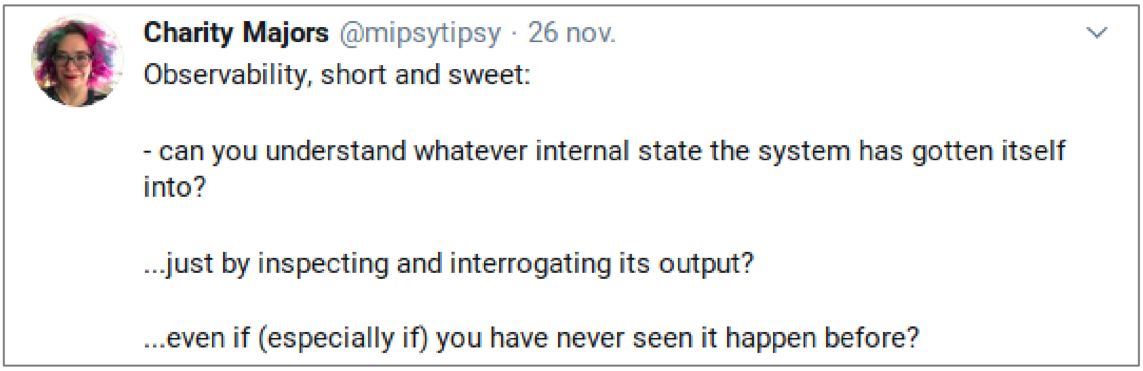
Since the end of the first period (from M1 to M18), it was clear that building an open source community around the project was no easy task. It is important to remark that since the beginning of the project we have been doing efforts on communicating about the project, even before ElasTest was available. We did a presentation at the international conference ExpoQA[1] in June 2017 (5 months and a half after the beginning of the project) where we already presented the main ideas around ElasTest. Through our social networks we have been insisting on the features that the project was going to provide. Therefore, by the time of the first review of the project ElasTest was well known within the circle of industrial contacts of the URJC group in Madrid and part of Spain, but no one was really joining the community and pairing with us to build the product together.

Before the summer of 2018, the URJC team was aware of a concept that, despite being old, was now being applied to distributed systems and specifically to understand problems within these systems. The concept was *observability* [1], and it was taken from control theory. Observability means trying to guess the status of a system, based on external observations, i.e., with no intervention and no modification of the system. This concept is being applied successfully to identify problems in highly available systems in production. We borrowed it from the operations teams and applied it to testing. In our testing context, observability means being able to localize a bug in a software product in a short time when a test fails without a need to debug the product (which for large distributed applications might be impossible at all).

We learned from the observability thing first by Charity Majors, founder and CTO of Honeycomb[2], a company that offers a pay-per-use observability platform for distributed applications. Since the concept was coined, she has explained several times what observability means when applied to distributed systems (see in Figure 1 a tweet explaining it).

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

[2] https://honeycomb.io/



Charity Majors @mipsytipsy · 26 nov.

Observability, short and sweet:

- can you understand whatever internal state the system has gotten itself into?

...just by inspecting and interrogating its output?

...even if (especially if) you have never seen it happen before?

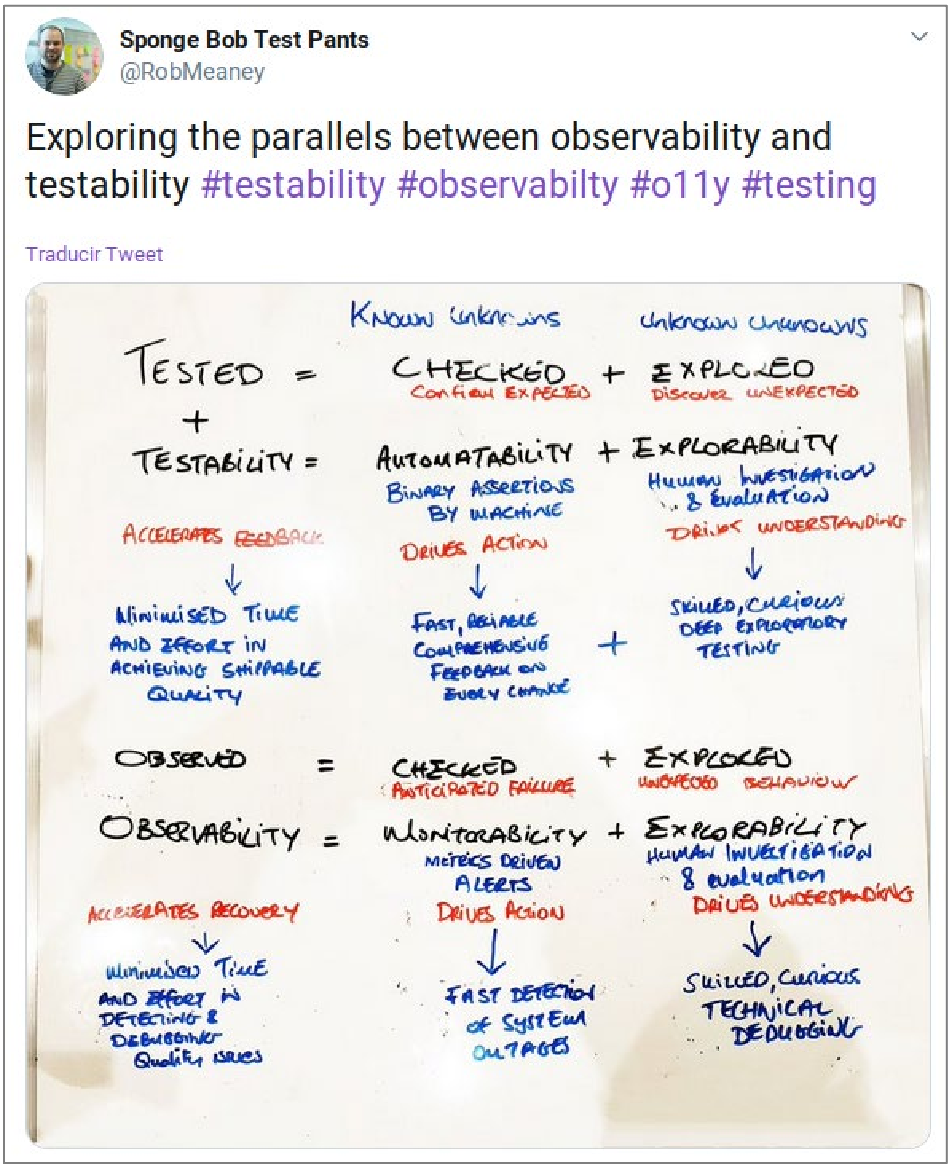
Figure 1. Charity Majors' description of observability[3].

Since then, many other people have taken observability into consideration, and linked it to testing. These people include Sarah Wells, Director for Operations and Reliability at Financial Times, in a keynote at European Testing Conference in February 2019, Abby Bangser, who gave a 2-day workshop on observability for distributed systems at Agile Testing Days 2019[4], or Robert Meaney, who's work on linking observability to testing has been shared across different events (see Figure 2).

The popularization of observability in the testing field has been a great opportunity to promote ElasTest features in the community. Following this new path, we steered many of our actions to build a sustainable open source community. In the following, we describe the actions taken and the outcomes.

[3] https://twitter.com/mipsytipsy/status/1199447610160902144

[4] https://agiletestingdays.com/2019/session/exploring-systems-quality-in-a-distributed-world/



Sponge Bob Test Pants
@RobMeaney

Exploring the parallels between observability and testability #testability #observabilty #o11y #testing

Traducir Tweet

KNOWN UNKNOWNS
UNKNOWN UNKNOWNS
TESTED = CHECKED + EXPLORED
CONFIRM EXPECTED
DISCOVER UNEXPECTED
+
TESTABILITY = AUTOMATABILITY + EXPLORABILITY
BINARY ASSERTIONS BY MACHINE
HUMAN INVESTIGATION & EVALUATION
ACCELERATES FEEDBACK
DRIVES ACTION
DRIVES UNDERSTANDING
MINIMISED TIME AND EFFORT IN ACHIEVING SHIPPABLE QUALITY
FAST, RELIABLE COMPREHENSIVE FEEDBACK ON EVERY CHANGE
+
SKILLED, CURIOUS DEEP EXPLORATORY TESTING
OBSERVED = CHECKED + EXPLORED
ANTICIPATED FAILURE
UNEXPECTED BEHAVIOW
OBSERVABILITY = MONITORABILITY + EXPLORABILITY
METRICS DRIVEN ALERTS
HUMAN INVESTIGATION & evaluation
ACCELERATES RECOVERY
DRIVES ACTION
DRIVES UNDERSTANDING
MINIMISED TIME AND EFFORT IN DETECTING & DEBUGGING Quality ISSUES
FAST DETECTION OF SYSTEM OUTAGES
SKILLED, CURIOUS TECHNICAL DEBUGGING

Figure 2. Robert Meany's linking between observability and testing[5].

3.1 Attending and Presenting at Industrial Events

We have presented ElasTest at many different industrial events, in an attempt to catch attention from the IT community and either make them join our open source community, or at least make them support the project economically through specific commercial agreements. Since September 2018, most of our dissemination activities to the IT industry were spreading this observability marketing idea. Table 1 shows the list of attended events during the second period. There’s been a huge effort on disseminating the ElasTest project.

[5] https://twitter.com/RobMeaney/status/1169394283385937925

Event	Partner(s)	Date
1st ElasTest Workshop	URJC, NAEVATEC, ATOS	Sep. 2018, Madrid
Fuseco Forum	URJC, FOKUS	Nov. 2018, Berlin
IT Directors Forum	RELATIONAL	Jul. 2018
European Testing Conference	URJC	Feb. 2019, Valencia
Artificial Intelligence in Industry	IBM	Oct. 2018
Test Bash Brighton	URJC	Apr. 2019, Brighton
Mobile World Congress	FOKUS	Feb. 2019, Barcelona
ARI Digital Show	ATOS	Nov. 2018, Madrid
IIX Speed Geeking Event	IBM	Nov. 2018
BTIC Innovation Approaches Show	ATOS	Jan. 2019
Steering Meeting	IBM	Jan. 2019
Voxxed Days Athens	RELATIONAL	Jun. 2019, Athens
Agile Greece Summit	RELATIONAL	Sep. 2019
SeleniumConf	URJC	Apr. 2019, Japan
QA Fest	URJC	Sep. 2019, Kyiv
Agile Automation Days	URJC	Oct. 2019, Poland
FIWARE Summit	NAEVATEC	Oct. 2019
Fuseco Forum	FOKUS, URJC	Nov. 2019
Software Freedom Day	URJC	Sep. 2019
Northern Quality webinar	URJC	Oct. 2019
ISOLA Conference	IMDEA	Nov. 2019
WebRTC Meetup	URJC	Jul. 2019
WebRTC Meetup	URJC	Jul. 2018

Table 1. Industrial events attended during the second period.

In addition to these events, URJC sent a proposal for the EUROStars conference 2018 that was rejected, and RELATIONAL sent another one to Voxxed Days Thessaloniki 2018 that was rejected as well.

It is important to note that the project made huge efforts to reach specific communities, like the Selenium open source community, FIWARE community or the WebRTC community. Selenium is the most widely used open source project for building web UI tests. ElasTest uses Selenium under the hoods, and URJC showcased ElasTest at SeleniumConf in Japan, the biggest conference around the Selenium community. The FIWARE foundation leads the FIWARE project where NAEVATEC offers a Generic Enabler for media. The project showcased ElasTest at the FIWARE Summit event, and there was a joint meeting with URJC, NAEVATEC, ATOS and the FIWARE Foundation to present the benefits that embracing ElasTest could have in the FIWARE Community. However, too many adaptations were envisioned in order to enable ElasTest for FIWARE Generic Enablers, and in the end no agreement was made. Finally, there was a big interest in Madrid area about WebRTC, so a WebRTC Meetup was established, although with no periodic meetings. URJC has participated in two of the meetings organized by the community, and synergies with some participants has been established.

3.2 Maintained a Regular Cadence of Releases

From the beginning, we have followed the "release early, release often" mantra. First ElasTest release[6] happened as early as September 2017. Since then, we have released a new version every month on average, with a total of 20 releases in 18 months (see Table 2).

Release	Date	Blog entry
1.0.0-beta1	12/09/2018	
1.0.0-beta2	04/10/2018	
1.0.0-beta3	31/10/2018	https://medium.com/me/stats/post/4a409f9fe063
1.0.0-beta4	03/12/2018	
1.0.0-beta5	20/12/2018	
1.0.0-beta6	23/01/2019	
1.0.0-beta7	31/01/2019	https://medium.com/me/stats/post/c0285e914f1f
1.0.0	13/02/2019	https://medium.com/me/stats/post/1f5de2a356c2
1.0.1	19/02/2019	https://medium.com/me/stats/post/a535adbc6420
1.1.0	01/03/2019	
1.2.0	07/03/2019	https://medium.com/@elastest/elastest-1-2-0-b0054e68ef92
1.3.0	20/03/2019	https://medium.com/@elastest/elastest-1-3-0-83aba8fb45c9
1.3.1	28/03/2019	

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

1.4.0	08/04/2019	
1.4.1	09/04/2019	https://medium.com/@elastest/elastest-1-4-1-9d7821ebb313
1.5.0	29/04/2019	https://medium.com/@elastest/elastest-1-5-0-37f650330b5f
2.0.0	15/10/2019	https://medium.com/@elastest/elastest-2-0-0-fd88d0a1c3d3
2.0.1	22/10/2019	
2.1.0	14/11/2019	https://medium.com/@elastest/elastest-2-1-0-d08819677a33
2.1.1	11/12/2019	

Table 2. Releases for the second period.

Releases that included new features (those that change one of the first two numbers, starting from the left) were accompanied by a blog post entry in our blog.

3.3 Encouraging the Community to use GitHub Issues for Requests & Bugs

During this period, we insisted to all the users and through all the presentations we did that people used the issues available in out GitHub projects to suggests new features and notify us about bugs. During this period, we had 4 issues opened by external contributors.

We had an external contributor that contributed fixes to our documentation through a Pull Request that was accepted by the URJC team. Pull Requests are the usual way of getting involved in a project, i.e. contributing to the project, without being part of the project itself. Usually, only members of the project can commit changes to the project. Therefore, to foster adoption, projects usually accept contributions in the form of Pull Request. Given the open source nature of these projects, members of the community willing to contribute proceeds as follows:

- They fork the repository in their own personal account. Forking means that a copy of the repository is made into the account of the user, but the forked repository keeps linked to the original repository.
- The contributor makes the changes related to the contribution in its own repository.
- When changes are ready, the contributor opens a pull request against the original repository that is linked to the changes in the forked repository.
- Members of the project can review the pull request, and comment on it, accept it or reject it as they see fit.
- If the pull request is accepted, the changes are integrated into the original repository.

This is the process we followed with this contribution to our documentation: after a review, we accepted the pull request.

3.4 Encourage the Use of Mailing Lists

Some users where asking questions through our email (info@elastest.io). However, solving general questions in emails, is not a good way of building a community. Therefore, we encouraged people contacting us privately, to do so in the open, publicly, through our mailing lists[7].

In total we had 3 different questions asked in our mailing list, from two different people. These conversations evolved into threads with an average of 8 messages each.

3.5 Find Synergies with Companies

We have been endeavoring to involve companies from the IT sector in ElasTest to build fruitful collaborations. Our main objective was to have use cases first-hand, and in return steer the project towards fulfil some of those uses cases, thus complying with the lean approach set out at the proposal stage.

One of the first companies interested in the project was Panel Sistemas[8]. URJC was invited to present ElasTest at the Panel 101 Days as early as December 2017[9], before one year of the project. Later, other companies were interested as well. Zooplus, a German-based company, started to evaluate ElasTest in 2019, and a person within the company was working with the URJC team to make ElasTest work on their own premises and with their own applications. For this collaboration, a specific Slack workspace was used, which allowed both parties to quickly and easily talk to each other to solve questions and report bugs or problems.

Other companies talked to us in industrial events or through email noting their interest in ElasTest, but no collaboration has been started yet. This is the case of Idealista and TLG, two Spain-based companies.

3.6 Rebuilding the Communication Strategy for the Product

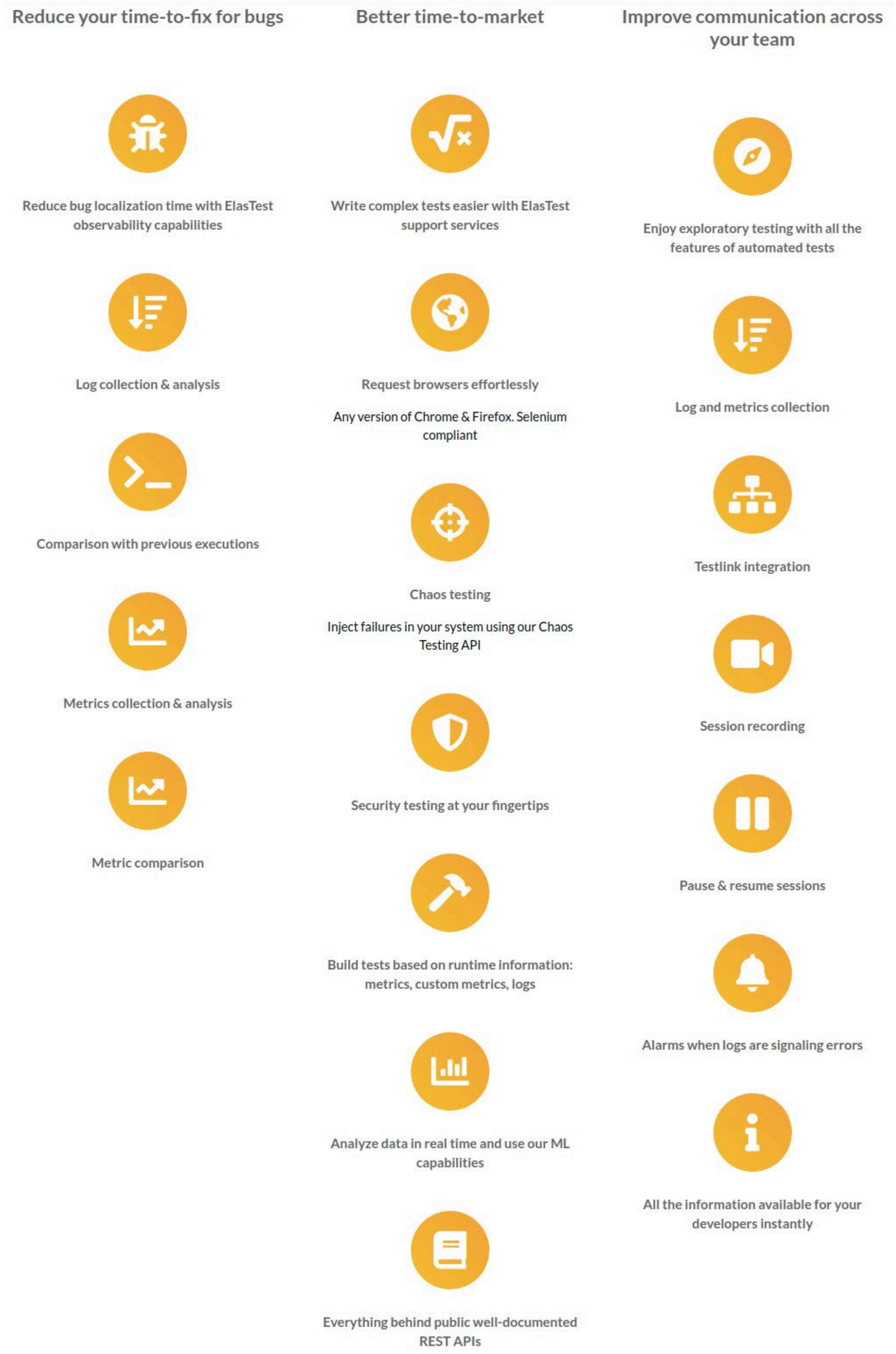
As we specified at the beginning of the Section, observability has been the main focus of the project. We rebuild the communication strategy around this concept, and we rebuild the website completely to explain the message through it. We based the website in three concepts, as can be seen in Figure 3:

- **Reduce time-to-fix for bugs**. By providing observability tools like the log comparator and the log analyzer, and being test aware, bug localization can improve and therefore, the time to detect and fix the bug reduced.

[7] https://groups.google.com/forum/#!forum/elastest-users

[8] https://www.panel.es/

[9] https://www.youtube.com/watch?v=C5s9ExDlEHA



Reduce your time-to-fix for bugs

Reduce bug localization time with ElasTest observability capabilities

Log collection & analysis

Comparison with previous executions

Metrics collection & analysis

Metric comparison

Better time-to-market

Write complex tests easier with ElasTest support services

Request browsers effortlessly

Any version of Chrome & Firefox. Selenium compliant

Chaos testing

Inject failures in your system using our Chaos Testing API

Security testing at your fingertips

Build tests based on runtime information: metrics, custom metrics, logs

Analyze data in real time and use our ML capabilities

Everything behind public well-documented REST APIs

Improve communication across your team

Enjoy exploratory testing with all the features of automated tests

Log and metrics collection

Testlink integration

Session recording

Pause & resume sessions

Alarms when logs are signaling errors

All the information available for your developers instantly

Figure 3. Use cases highlighted at the ElasTest community website.

- **Better time to market**. By providing test support services and engines, writing test cases is faster and easier, thus reducing the time to market.
- **Improve communication across the team**. By providing the same observability features to manual testing that are available to automated testing, members of the team can share much more information among them, making easier for developers to analyze problems detected by testers.

3.7 Rebuilding the Entire Documentation Page

In order to bring more people to the website and stay for longer, so to have better conversions, we rebuilt the web page. Specifically, we did the following:

- Include a *live instance*, that do not allow to execute TJobs but have many demos preconfigured and executed so that visitors can play around and see what ElasTest has to offer for testing. This is related to the UI changes made to ElasTest that are explained in Subsection 3.9.
- Add a *Demos* section to the website, where the demos of the live instance are carefully explained, and links to the demos within the live instance are provided.
- Add a *Tutorials* section with step-by-step descriptions of some of the main features of ElasTest.

3.8 Providing a Live instance for Better Assessment of the Capabilities of the Product

A live public instance, running on URJC premises has been deployed, and is available for everyone to use. We disabled in this instance the TJob execution, to avoid problems with high loads of the machine if many visitors execute TJobs at the same time. However, we set up a lot of demo projects using different Test Support Services, and executions of these demo projects are available. The analytics tools provided by ElasTest can be used to have an idea of the help ElasTest can provide in testing. This Live instance can be reached from the ElasTest website (as seen in Figure 4). The demos included in this instance are the following:

- **Unit test:** shows a simple example running some unit tests. Main purpose of this example is to get familiar with the ElasTest UI.
- **Rest API:** shows how to exercise the Rest API of a SUT that is started by ElasTest. This example requires ElasTest to provide to the test the URL where the SUT can be reached.
- **Webapp:** shows how to exercise a SUT that is started by ElasTest through its GUI (the SUT is a web app) by means of browsers provided by ElasTest. This example requires ElasTest to provide the test with the URL of the SUT and the URL of the EUS, the service that provides browsers on demand.
- **OpenVidu WebRTC:** shows how to exercise a more complex SUT. The SUT consists of a videoconferencing application, and the test starts two different browsers and try to connect them through the videoconferencing application OpenVidu.

- **FullTeaching:** shows how to exercise a SUT comprising several services: the web app FullTeaching and a MySQL database. It highlights how different services can be shown in the ElasTest analytics tools separately if needed. Logs and metrics for all the services are retrieved.
- **Building assertions with monitoring data:** shows how to leverage the EMS to build assertions out of monitoring data coming from the SUT. The test expects some events to be triggered by the SUT that are sent to it by the EMS.
- **IoT device emulation example:** shows how to leverage the emulation of IoT devices to test IoT applications. The test can emulate the behavior of IoT devices that are provisioned and configured by it, and then check the behavior of the application by listening to events flowing to and through these devices.
- **Jenkins Examples:** show the ElasTest integration with the popular Jenkins CI server.
- **Browsers in AWS:** shows how to leverage browsers started in AWS machines through the EUS (the browsers as a service component). When the infrastructure where ElasTest is running is not big enough to start as many browsers (OpenVidu load tests require 200 browsers in parallel), EUS allows users to start the browsers on AWS, with potentially unlimited resources available.
- **Quality of Experience:** shows how to use the QoE algorithms provided by the EUS when analyzing quality of video streams. The EUS can record the videos and analyze them using 6 different QoE algorithms. For an explanation of these algorithms and their efficiency, see [2].

A screenshot of the live instance[10] is shown in Figure 5.

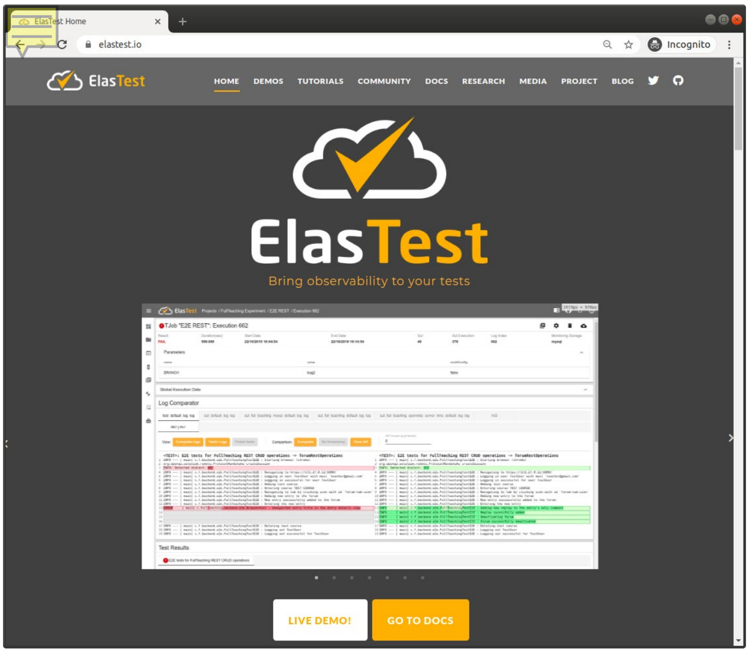
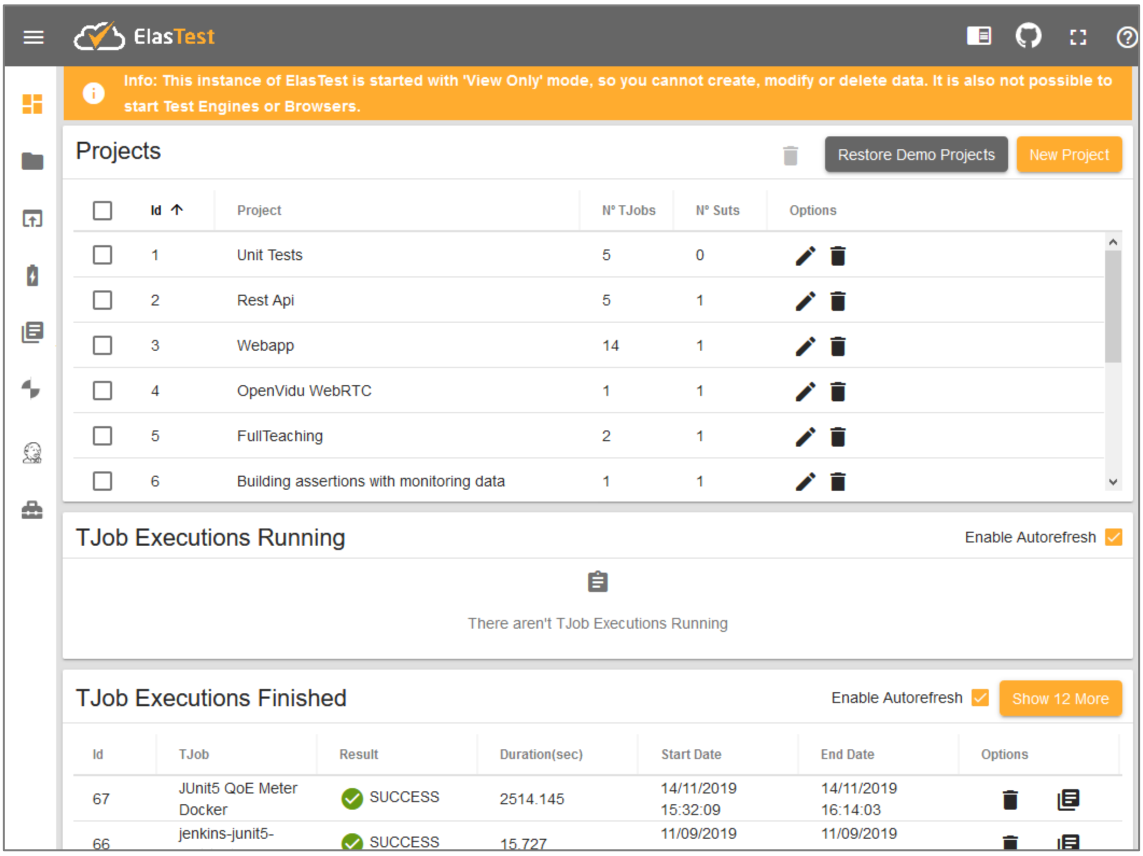


Figure 4. Button in the website leading to the ElasTest live instance.

[10] http://live.elastest.io:37000/



Info: This instance of ElasTest is started with 'View Only' mode, so you cannot create, modify or delete data. It is also not possible to start Test Engines or Browsers.

Projects

□	Id ↑	Project	Nº TJobs	Nº Suts	Options
□	1	Unit Tests	5	0	✎ 🗑
□	2	Rest Api	5	1	✎ 🗑
□	3	Webapp	14	1	✎ 🗑
□	4	OpenVidu WebRTC	1	1	✎ 🗑
□	5	FullTeaching	2	1	✎ 🗑
□	6	Building assertions with monitoring data	1	1	✎ 🗑

TJob Executions Running

Enable Autorefresh ☑

There aren't TJob Executions Running

TJob Executions Finished

Enable Autorefresh ☑

Id	TJob	Result	Duration(sec)	Start Date	End Date	Options
67	JUnit5 QoE Meter Docker	✓ SUCCESS	2514.145	14/11/2019 15:32:09	14/11/2019 16:14:03	🗑
66	jenkins-junit5-	✓ SUCCESS	15.727	11/09/2019	11/09/2019	🗑

Figure 5. ElasTest live instance. The projects are the different demos configured.

3.9 Complete Rebuild of the UI of the Product Following Our Users' Recommendations

During the second period, based on our things we learned from Observability and the feedback from the companies that were closely collaborating with us to try ElasTest internally, we rebuilt the UI, so that it was easier to find information about the behavior of the SUT during the different tests. Therefore, we grouped together the information related to each test case for better exploration (as usually, test cases are independent between them even if they are executed together as part of the same TJobs). In addition, we decided to show different information depending on the outcome of the test. If the test pass, we show a report focused on the first test case (each test case is shown in a different tab, so we highlight the first one), and within the information related to the test case we highlight the log analyzer (other tabs include details about the test run, files, and recordings from browser sessions). When a test fails, we highlight the filed test (i.e., the tab corresponding to the failed test grab the focus), and if there's information about previous executions of this test then instead of focusing on the log analyzer, we open a new tab with the log comparator and we show automatically a comparison between the last successful run of this test case and the current failed one. This gives a quick insight on what's different in this run with respect to other successful runs.

4 ElasTest Open Source community in numbers

The needs of the project are clear, they are defined in the Description of the Action (DoA) of the project, and partners shall pursue what it is stated there. In relation to the community, these are the objectives of the project:

- To create an open source software ecosystem and community around ElasTest suitable for guaranteeing the long-term sustainability of ElasTest technologies.
- Availability of a community site and of the community social channels.
- To have 1,000 individuals having participated in the community by month 36.
- To have, at least, 50 organization and/or individuals external to the project having actively used, contributed or supported the community by month 36.

In Section 3, we described the strategy and the actions undertaken during the second period to bring users to the platform, and from part of those users building the ElasTest community. As it can be seen in Figure 6, there were visitors in the community web page[11] during all the duration of the second period, some days with peaks of 100 visitors a day. This is the result of our dissemination actions that included a constant pace in writing blog posts (including specific blog posts for each of the new releases), tweets, and talking at industrial events.

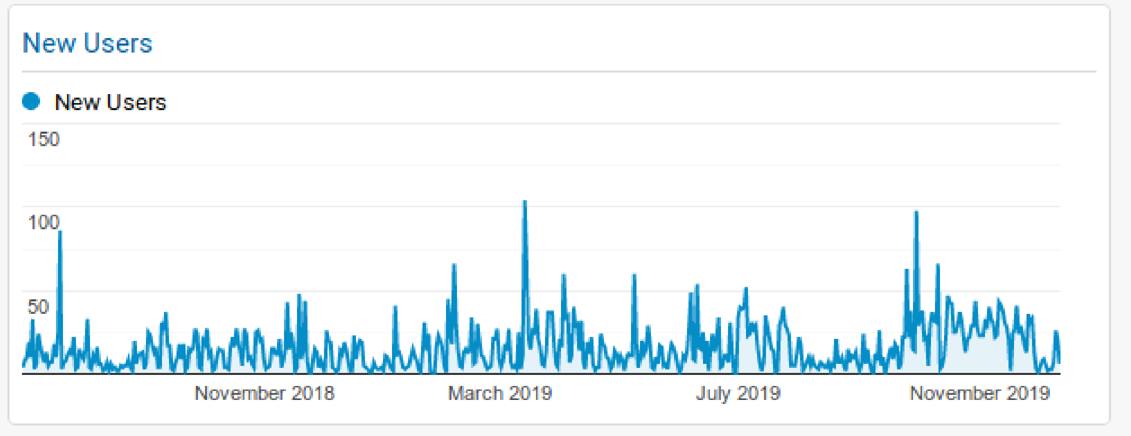


Figure 6. New users accessing the ElasTest.io website for the last year. In total 8,219 new users visited the page, all around the world, with peaks of 100 visitors on some days.

Figure 7 shows where our audience comes from. According to the analytics of our website, visitors are coming from all around the world, which is an indication that our dissemination activities had an impact worldwide. It seems, most visitors came from Spain, Ireland and US. Spain was no surprise, as there are three Spanish members that has been doing a strong dissemination at national events (see D8.5 for a detailed description of the number of industrial events attended). US is a large country, and therefore, it is reasonable that a lot of visitors came from this country. However, Ireland,

[11] https://elastest.io

despite having IBM in the consortium, was a surprise. IBM has been doing dissemination at local events as well.

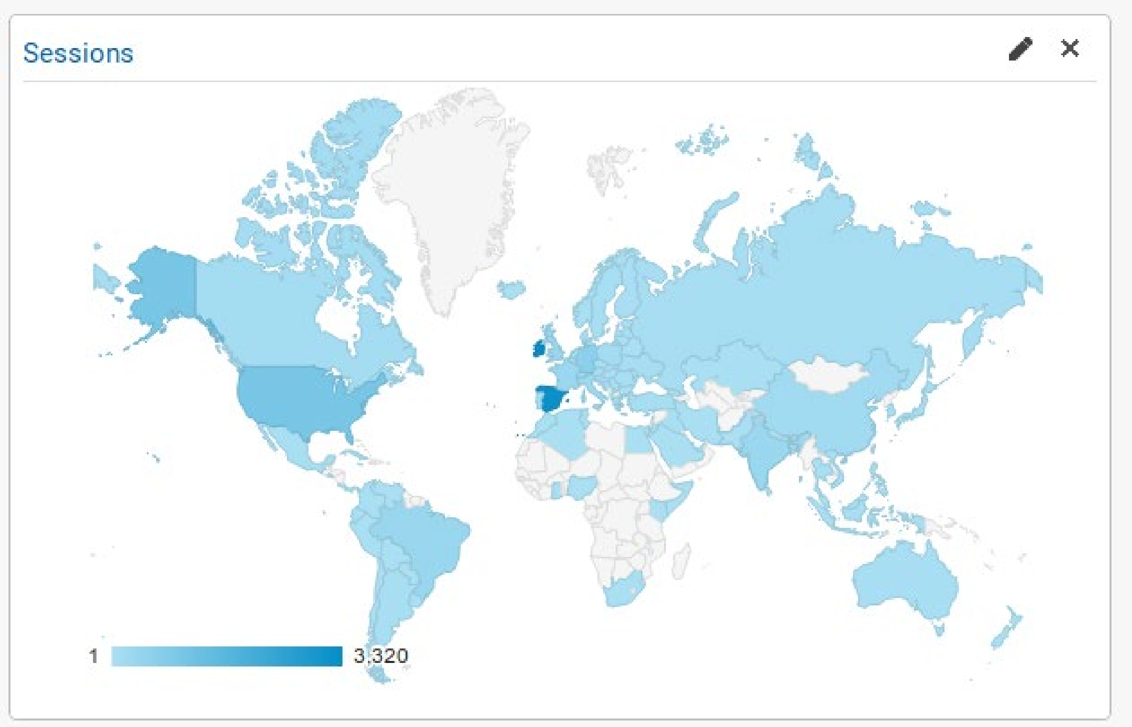


Figure 7. Visitors of the ElasTest website are all along the world, thus it seems the project had an impact worldwide.

We had also a look at top sources in Figure 8, and most of our visitors came directly to the website, meaning the already knew about the project. However, there's an important number of users that came from Internet searches, which means we are in good positions in searchers. It is important to remark that, as a consequence of giving a talk at the European Testing Conference, and industrial event, the Project Coordinator was interviewed and this interview was published in InfoQ, one of the news in software development with more impact worldwide. As a result, we see in the yellow that some visitors came from InfoQ webpage.

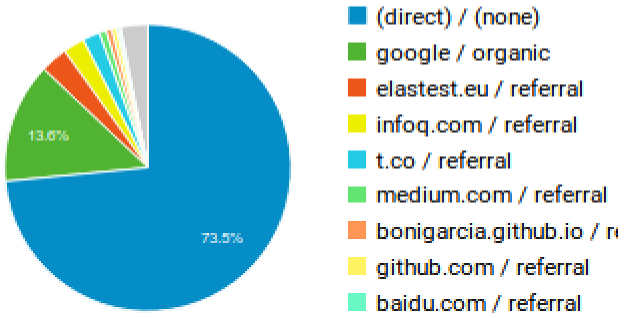
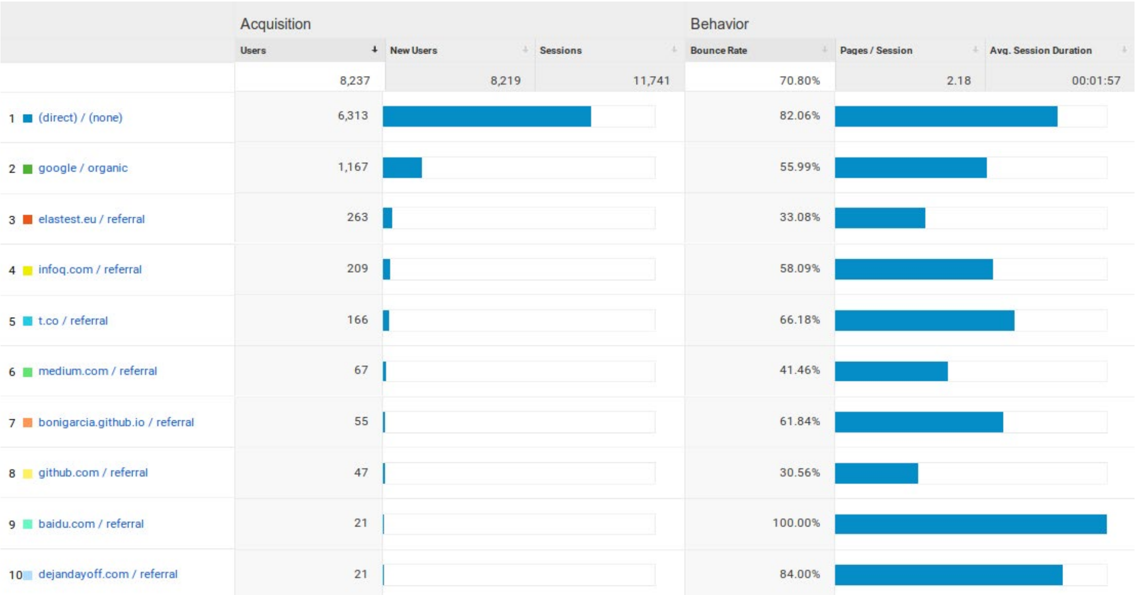


Figure 8. Top sources from where visitors came to the ElasTest webpage.

Figure 9 shows the data from the previous Figure, but in tabular form and highlighting the total number of visitors from each source.



	Acquisition			Behavior		
	Users	New Users	Sessions	Bounce Rate	Pages / Session	Avg. Session Duration
	8,237	8,219	11,741	70.80%	2.18	00:01:57
1 🟦 (direct) / (none)	6,313			82.06%		
2 🟩 google / organic	1,167			55.99%		
3 🟥 elastest.eu / referral	263			33.08%		
4 🟨 infoq.com / referral	209			58.09%		
5 🟦 t.co / referral	166			66.18%		
6 🟩 medium.com / referral	67			41.46%		
7 🟧 bonigarcia.github.io / referral	55			61.84%		
8 🟨 github.com / referral	47			30.56%		
9 🟩 baidu.com / referral	21			100.00%		
10 🟦 dejandayoff.com / referral	21			84.00%		

Figure 9. Tabular data on top sources with the number of visitors.

Finally, Figure 10 shows the number of interactions within the website. Most users (8,100) reached the main page of the website (denoted as “/”), and they never progressed any further. However, some of them were curious enough to visit other pages in our website. Specifically, 975 users navigated to the documentation page, 524 jumped to the project page (where details of the EU funding and the consortium are presented), 208 progressed towards the try-elastest page that provides installation instructions, and 120 navigated to the releases page where features and bugs solved for each release are described. Giving this numbers, it seems some visitors were interested in the project and stayed in our website for long enough to have a look at some details. For instance, more than the third of visitors that visited our documentation page, had a look at the try-elastest page. Of the original 8,100 visitors, 1,400 had three interactions with the website, that is, they visited three different pages within our website.

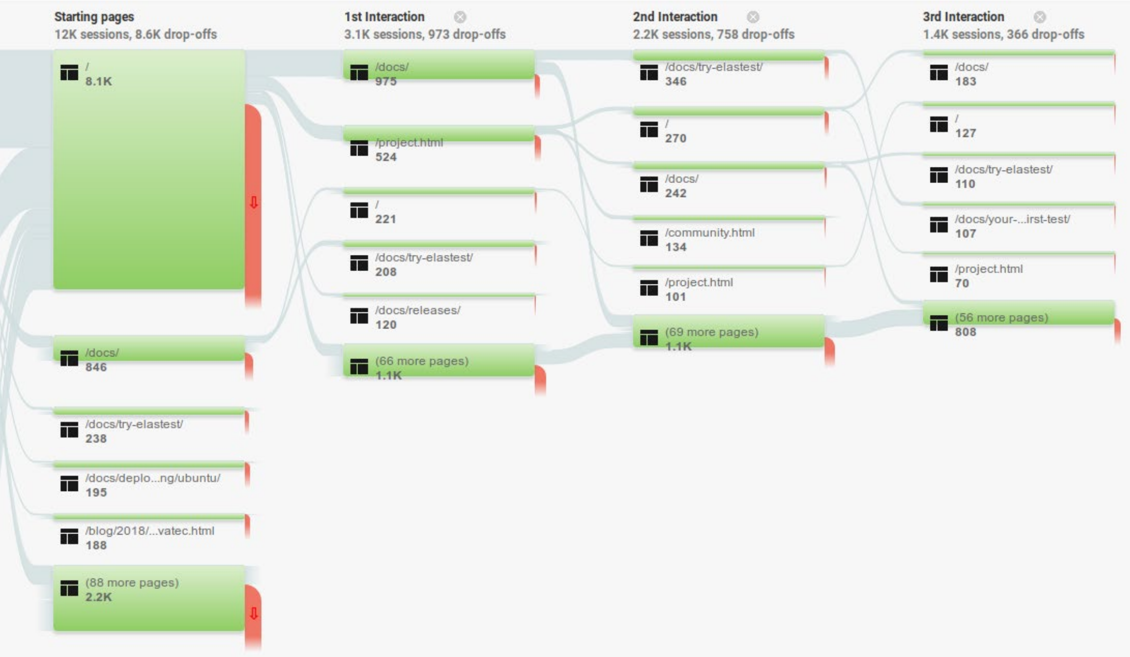


Figure 10. Flow of interactions.

In addition to the website, we actively used our blog, Twitter and YouTube channel to communicate about the project and the community, and we have some participants that used the mailing list or GitHub to contact us. At M36, the number of followers of the @elastestio handle on Twitter is 513, the number of followers on our blog is 15 with 953 views and 518 reads of our blog posts, and in YouTube with have 34 subscribers and more than 2,300 views of our videos. There were two users asking questions in our mailing list, and three GitHub users opening issues and pull request in GitHub. If we account exclusively for those users having active interactions with our community (following out twitter account, reading our blog posts or watching our videos), we have a total of 8,100+513+518+2,300+2+3 = 11,436 people having participated in the community.

KPI	DoA	Data at M36
Individuals participating in the community	1,000	11,436 Twitter: 513 followers Website: 8,100 new visitors Medium: 15 followers YouTube: 2,300 views Medium: 953 views, 518 reads, 379 fans Mailing list: 2 GitHub: 3

Companies or individuals having used, contributed or supported ElasTest	50	10 5 companies 5 individuals

Table 3. ElasTest community numbers

5 The future of the ElasTest Open Source community and project

Despite having made an impact worldwide, as the analytics of our webpage and reads of our blog pots indicates, not many people decided to be directly involved in the project. Rising an open source community is a tough task, and some people have written before giving some tips on actions that could help in this goal. We followed some of them, like providing a carefully designed website and writing proper documentation, and we were also very responsive whenever people came to ask through our channels. We also did talk about the project in as many events as we could.

Right now, there are two different companies that are actively working with the URJC team to include ElasTest in their software development lifecycle processes, one of them is in addition collaborating with us to include ElasTest within the portfolio of testing services they offer to their clients. At some point, we might see these companies devoting some time to the project, by helping us in maintaining or developing the platform.

In the meanwhile, the URJC team will maintain the community on its own. This effort will be sustained on a PhD that is currently using ElasTest to do research on context-aware prioritization (see D4.3 for further details), and on the adaptation of ElasTest to the specificities of the OpenVidu project, led by the CodeURJC group at URJC. OpenVidu is an open source project for building videoconferencing applications with ease and has been used widely in ElasTest to showcase specific features (like browsers and more recently, QoE measurement for video). OpenVidu is offered also under a commercial license that includes features not available in the open source product, like elasticity and resiliency. We expect that part of the incomes from OpenVidu be devoted to maintain and develop specific features in ElasTest tailored towards testing this project.

In order to guarantee the sustainability of the project, the URJC team has designed a roadmap consisting on the following tasks to be performed during a duration of 2 months after the end of the project with the goals explained above:

- ElasTest is a huge project that is easy to install but requires a lot of resources. We plan to **reduce the friction and offer ElasTest directly through the Jenkins plugin**. This task is comprised of the following subtasks:
    - Extract the ETM and EUS, as they are implemented in the mini mode, into their own project. This will be the core ElasTest platform, avoiding any other component.

  - Remove any TJob execution feature from the ETM. Jobs will be always executed from Jenkins. ElasTest will be started on demand when a Jenkins job requests it.
  - Rely on an external Elasticsearch (ES) installation ElasTest will use to gather data. The specificities of this ES installation will be properly documented in the webpage.
  - Rely on the user sending events to Elasticsearch. This will be properly documented on the webpage.
  - Document the data model and provide dashboards on top of Kibana.
  - The Kubernetes support will be made JenkinsX compatible, thus making possible to use ElasTest in the context of this next generation CI server of the Jenkins ecosystem.
- ElasTest provides specific WebRTC testing capabilities. However, user experience can be improved, and the reporting enriched. We plan to bring observability to WebRTC applications, by accomplishing the following subtasks:
  - Provide over-the-time visualization of Quality of Experience for the tests exercising the media capabilities of WebRTC applications
  - Provide comparison with other network metrics such as latency, jitter, and bandwidth.
  - Provide visualization of internal OpenVidu metrics through integrations with the OpenVidu Inspector[12] and OpenVidu Dashboard[13].

These tasks are aimed at reducing the maintenance costs, and at the same time, be tailored towards the needs of OpenVidu. ElasTest will still be useful for standard web applications through the Jenkins plugin installing other specific tooling. Users will be able to use it just by installing a Jenkins plugin, thus making much easier to give it a try.

The full ElasTest stack will be available and active maintenance efforts will be performed if new community members are interested on it or some companies are willing to pay for it.

6 References

[1] Sridharan, C. Distributed Systems Observability: A Guide to Building Robust Systems. *O'Reilly Media*, **2018**.

[2] García, B.; López-Fernández, L.; Gortázar, F.; Gallego, M. Practical Evaluation of VMAF Perceptual Video Quality for WebRTC Applications. *Electronics* **2019**, *8*, 854.

[12] https://openvidu.io/docs/openvidu-pro/openvidu-inspector/

[13] https://openvidu.io/docs/openvidu-pro/detailed-session-monitoring/