Horizon 2020
Communications Networks, Content & Technology
Cloud & Software
Research & Innovation Action
Grant agreement n° 731529

Title:	WP5 – D5.4 – Validation Roadmap and framework V2
Date:	July 31, 2018
Writer:	Jesús Gorroñogoitia Cruz - Atos Vincent Massol - XWiki Lars Thomas Boye – TellU Assad Montasser, Cedric Thomas – OW2 Mael Audren - ActiveEon
Reviewers:	Caroline Landry, Benoit Baudry, Valentina Di Giacomo

Table Of Content

1. Executive Summary

This document describes the objectives of the validation activities, the proposed evaluation method, the supporting conceptual and material evaluation framework and the adopted evaluation roadmap, which altogether drive and support the industrial validation of the STAMP toolset and services. The method collects and defines a set of configurable process fragments or tasks that, once tiled together, constitute a customized methodology for conducting the different validation activities. It is influenced by industrial standards and scientific guidelines to conduct and report assessment experiments in real industrial scenarios. The framework introduces the key evaluation concepts and elements, but also the supporting materials and tools required to conduct the validation activities following the proposed method. The roadmap schedules a set of proposed validation activities in a timeline within the project lifetime, which is aligned to the STAMP development roadmap, milestones and release plan, aiming at maximizing the impact of the evaluation feedback (e.g. findings, recommendations, reported issues, etc.), reported by stakeholders, into the quality of the STAMP toolset and services.

This document provides the final conceptualization for the evaluation method, framework and roadmap. This document improves the previous version (D5.2, M9) in different aspects, as a result of the discussions held during the period M9-M20 by the industrial UC partners themselves and between them and the technical providers. Improvements also come from the experience gained during the first evaluation phase (M9-M18). Moreover, the evaluation roadmap has been adjusted to address the interim review recommendations to anticipate the second phase evaluation report to M24.

The main improvements of this document with regards to the previous one, D5.2, are the following:

- The industrial expectations and objectives for STAMP that will constitute the target of this evaluation have been specified by each use case provider;
- The concrete STAMP results (tools, clients, services) target of this industrial evaluation have been specified with links to them and their supporting material (source code, releases, documentation, etc);

- The evaluation roadmap has been updated;
- Comparative experiments for precise in-lab controlled validation have been defined, aiming at comparing (based on KPIs measurement) the benefits of adopting STAMP methods and techniques (tool supported) to the current industrial state of practice adopted by the use case providers;
- The strategy to engage external developers within open-source communities in the usage and evaluation of the STAMP methods and tools have been defined.
- The quality model adopted for qualitative evaluation has been specified most precisely;
- Other aspects of the evaluation tasks and framework have been updated.

2. Revision History

Date	Version	Author	Comments
02-Jul-2018	0.1	Jesús Gorroñogoitia (Atos)	ToC
03-Jul-2018	0.2	Jesús Gorroñogoitia (Atos)	Initial contributions. Updates on different sections.
05-Jul-2018	0.3	Jesús Gorroñogoitia Cruz - Atos Vincent Massol - XWiki Lars Thomas Boye – TellU Assad Montasser – OW2 Mael Audren - ActiveEon	Contributions to section 7
16-Jul-2018	0.4	Jesús Gorroñogoitia Cruz - Atos	Description of comparative studies
18-Jul-2018	0.5	Jesús Gorroñogoitia Cruz - Atos	Updates in the evaluation roadmap
20-Jul-2018	0.6	Jesús Gorroñogoitia Cruz - Atos Cedric Thomas - OW2	Executive summary, Update on comparative studies. Open field trial description

23-Jul-2018	0.7	Jesús Gorroñogoitia Cruz - Atos	Update references, improvements in different sections, re-structure of the document, open field trial
24-Jul-2018	0.8	Jesús Gorroñogoitia Cruz - Atos	Peer-review version
17-Ago-2018	0.9	Jesús Gorroñogoitia Cruz - Atos	Addressed peer-review recommendations Polished document
17-Ago-2018	1.0	Jesús Gorroñogoitia Cruz - Atos	Final version
09-Sep-2018	1.1	Caroline Landry	Updating References

3. Objectives UPDATED

This document is the second (and final) release of the validation roadmap and framework specification, initially developed in D5.2 [1]. The objectives of this series of deliverables are two-fold. On the one hand, they define a conceptual and practical methodology and its supporting framework that is adopted to conduct the validation of the STAMP toolset and services in the context of the different industrial use cases (T5.3-T5.7) that participate in the project. This validation methodology also defines specific support for the external validation by potential adopters of the STAMP technology, among those developers of open communities who are interested on test amplification techniques and their potential benefits in software development and QA. This document elaborates on important concepts of the validation such as its objectives, participants, validation environments, validation activities, supporting materials, metrics and measurements or reporting.

On the other hand, this document defines a validation roadmap that schedules the validation activities in a timeline, in a suitable way for their alignment to the STAMP development activities and its plan for software releases. This roadmap tries to maximize the impact that the validation findings may have on the improvement of the quality and usability of the STAMP tools, particularly assisting the team during their development lifecycle, as well as facilitating their adoption by practitioners external to the project consortium.

This final release of this document further elaborates on some elements of the validation framework and roadmap providing additional material or refining the previous specification, based on the internal discussion held during the period M10-M20 and on the lessons learnt during the first evaluation phase (M10-M18). In particular, this document focuses on:

- The quality model adopted for assessment based on ISO-25010

- The set of STAMP tools target of evaluation
- The definition of experimental comparative experiments
- The definition of the evaluation method for empirical experiments
- The definition of the strategy for engaging as external evaluators the developers of open-source communities;
- The adaptation of the evaluation roadmap for remaining phases 2 and 3;
- The specification of the requirements and expectations that industrial use case providers have on the STAMP methods and techniques (supported by the tools) that will constitute the subject of this industrial evaluation.

Aiming at keeping this series of documents self-readable, we have opted to author this document as an update from D5.2, rather than as an independent document with multiple references to it. In order to identify new updates from D5.2, updated sections will be tagged with $^{\text{UPDATED}}$. New sections will be tagged as $^{\text{NEW}}$.

4. Introduction $^{\text{UPDATED}}$

This document defines the methodology and supporting framework adopted to conduct the validation of the STAMP techniques and its toolset in real-life industrial scenarios.

Software validation is an important part of the software development life-cycle, sometimes mistaken with a related concept, software verification, which is not targeted in this document:

- *Software validation* assesses whether or not a software product satisfies or fits the intended use, that is, the software meets the stakeholders' requirements. In other words, "developers have built the right system".
- *Software verification* assesses whether or not the designed and developed software is compliant to its specification, that is, "developers have built the system right". The verification of the STAMP toolset will be assessed by the development teams themselves and reported in the deliverables associated to the STAMP individual tools.

In summary, the **validation** of the STAMP toolset and services is the main scope of the methodology and framework described in this deliverable.

The validation activities adopted will combine two different methodological approaches:

- A validation conducted on continuous basis, adopting the STAMP toolset and services on the daily testing activities of the development life-cycle of the industrial software development projects included in the use cases. These validation activities will report feedback to STAMP development teams in continuous basis as well.
- A validation conducted on discrete basis, assessing the STAMP toolset and services in scheduled workshops conducted on laboratory controlled experimentation.

Besides the continuous validation feedback reported on continuous basis to the STAMP development teams, additional validation findings will be formally reported in official deliverables D5.5, D5.6 and D5.7

The decision of combining both validation approaches is justified to take advantage of the benefits of both approaches: a) influencing the STAMP toolset functional and usability specification and technical development with its duly validation on real testing phases of the industrial software development life-cycle and b) assessing the hypotheses associated to the proposed KPIs in controlled laboratory experimentation.

The remaining of this document is structured as follows. Section 5 collects the references cited within the document. Section 6 collects the abbreviations used within the documents. Section 7 explains the industrial-driven objectives of the STAMP validation process. Section 8 describes the validation roadmap and its alignment to the STAMP software release plan. Section 9 concludes the document. Appendix A introduces all the methodological concepts and constituents of the validation framework. Appendix B introduces the elements of the adopted evaluation method. Appendix C introduces the user survey questionnaire that will be used to collect initial feedback on the current users' practices for testing and test amplification.

5. References UPDATED

[0] Latest version of this document: d54_validation_roadmap_and_framework_v2.pdf

[1] J. Gorroñogoitia et al. D5.2 Validation Roadmap and framework. V1, STAMP report. 2017.: d52_validation_roadmap_and_framework.pdf

[2] M. R. Fine, Beta Testing for Better Software, ISBN 0-471 -25037-6, Wiley Computer Publishing, 2002

[3] Z. Yanga, S. Cai, Z. Zhou, N. Zhou, Development and validation of an instrument to measure user perceived service quality of information presenting Web portals, Information and Management, Elsevier, 2004

[4] J. Rubin, D. Chisnell, Handbook of Usability Testing - How to Plan, Design, and Conduct Effective Tests, ISBN: 978-0-470-18548-3, Wiley Publishing, 2008

[5] E. Almirall, M. Lee, J. Wareham, "Mapping Living Labs in the Landscape of Innovation Methodologies", Technology Innovation Management Review, 2012

[6] Barry W. Boehm, "Characteristics of Software Quality", North-Holland Pub. Co., 1978

[7] Jedlitschka, A., & Pfahl, D. (2005, November). Reporting guidelines for controlled experiments in software engineering. In Empirical Software Engineering, 2005. 2005 International Symposium on (pp. 10-pp). IEEE.

[8] C.J. DeLisle et al. D5.1 Industrial requirements and metrics for validation. V1, STAMP report. 2017

[9] Cohn, Mike. User stories applied: For agile software development. Addison-Wesley Professional, 2004.

[10] Rougemaille, Sylvain, et al. "Methodology fragments definition in SPEM for designing adaptive methodology: A first step." International Workshop on Agent-Oriented Software Engineering. Springer, Berlin, Heidelberg, 2008

[11] Björn Regnell, Richard Berntsson Svensson, and Thomas Olsson, “Supporting roadmapping of quality requirements”, Software, IEEE 25.2(2008): 42-47, 2008;

[12] V. Massol et al. D5.3 Industrial requirements and metrics for validation. V2, STAMP report. 2018

[13] P.Urso et al. D5.5 Use Cases Validation Report V1, STAMP report. 2018

[14] P. Urso et al. D6.3 Market Analysis, STAMP report. 2018

[15] Likert, Rensis (1932). "A Technique for the Measurement of Attitudes". Archives of Psychology. 140: 1–55.

[16] D. Gagliardi et al. D4.2 – First public version of API and initial implementation of

services and course-ware, STAMP report. 2018

6. Acronyms

CAMP	Configuration AMPlification
CI/CD	Continuous Integration / Continuous delivery
CLI	Command line interface
COTS	Commercial off the shelf
D<X.X>	Deliverable <X.X>

EC	European Commission
GA	Grant Agreement
IDE	Integrated Development Environment
IF	Integration Framework
HTML	Hypertext Markup Language
IT	Information Technologies
JS	Javascript
KPI	Key Performance Indicators
LDAP	Lightweight Directory Access Protocol
MAPE-K	Monitor, analysis, planning, execution, knowledge
MRL	Market Readiness Level
M<x>	Month <X>
MS<x>	Milestone <X>
OFT	Open Field Trial
OS	Operating System
PaaS	Platform as a Service

PWS	ProActive Workflows & Scheduling
QA	Quality assurance
R&D	Research & Development
SaaS	Software as a Service
SUT	System under test
UC	Use Case
UI	User Interface
UML	Unified Modeling Language
WAR	Web Archive
WP<X>	Work-package <X>

7. Validation Objectives

The overall goal of the validation task is to conduct a fit-for-purpose evaluation that assesses the STAMP toolset and services utility in real life scenarios that requires test amplification. The aim is to assess whether the STAMP scientific and technical achievements satisfy the needs and expectations of users.

The results of the project will be evaluated in the context of real life situations, starting with selected industrial use cases (UCs) owned by partners of the consortium, but also concluding with the involvement of communities of open-source software developers that will be invited to use and evaluate the STAMP toolset and tools in their own development activities. For this purpose STAMP will provide public downloadable releases of the standalone toolset and eventually links to publicly accessible instances of some services[1] – in the last phase of the project –, so potential adopters within those communities can download the tools (or eventually access the services) and evaluate them.

The approach used in the STAMP project is not defined entirely from scratch. Instead, partners involved in the evaluation consider and reuse parts of existing research and industry methodologies, methods, practices and standards related to requirements validation, requirements reviews, acceptance testing, beta testing [2], user-perceived service quality [3] and usability [4], considering their applicability for STAMP validation purposes. Some shared characteristics of participatory validation observed in the living lab methods [5] are derived, in particular the shared characteristic to involve users early in the process of validation in real-life environments and to incorporate the validation results into the toolset and service development.

The user needs are formalized as a set of requirements (and associated KPIs designed to assess their fulfillment). T5.1 "coordinates the elicitation of these industrial requirements for test amplification from the use case stakeholders, aiming at influencing the scientific and technical development of WP1-WP4 and the selection of results for industrial exploitation in WP6."

As explained in D5.1[8] and D5.3[12] and the GA, the final validation of the project will come through the validation of the objectives and KPIs. Evolution of the project as a whole will be validated through repeated collection of the KPI metrics by the use cases as they integrate the STAMP software and methodologies. In order to target the specific issues that are needed to improve the use case KPIs, we have developed a requirements and validation framework to allow use cases to express their needs clearly to the scientific and development work-packages.

Aiming at making this document focused the detailed description of the adopted evaluation framework and method has been placed in the appendices. Elements of the evaluation framework are introduced in Appendix A. Elements of the adopted evaluation method are introduced in Appendix B.

Validation objectives for industrial cases ^{NEW}

Next paragraphs briefly introduce the STAMP industrial cases in whose real context this evaluation of STAMP tools will be conducted. In particular, this description focuses on the expectations and objectives that the industrial cases lay on the STAMP results. The evaluation process will assess whether or not these industrial objectives and expectations are satisfied with the adoption of the STAMP tools, techniques and methods.

[1] STAMP Evaluation will focus on the evaluation of the STAMP standalone tools, setting aside the evaluation of the services to the moment they are available.

This content was anticipated in D5.5 [13], but added as well here, for self-consistency, since this content fits better within this section in order to highlight the overall objectives and expectations of the industrial stakeholders in the consortium that will be assessed by the validation process.

ActiveEon

ProActive Workflows & Scheduling (PWS) supports the execution of jobs and business applications, the monitoring of activity and the access to job results. It ensures more work done with fewer resources, managing heterogeneous platforms and multiple sites with advanced usage policies. PWS is a generic system, used in many different domains. Consequently, it requires a lot of tests on different configurations.

PWS is a distributed system built upon a microservice architecture pattern. The backend is mostly developed in Java, and the frontend in javascript. Jenkins is used to launch the jobs that build and test PWS: 150 jobs in order to manage the whole product life cycle. Gradle is used for build automation. In order to keep a high level of code writing quality, the Spotless Gradle plugin[2] is used to check and correct the formatting before building. SonarQube is used for continuous inspection of code quality. The dependency management Gradle plugin[3] is used to centralize dependencies in order to avoid conflicts between services.

Configuration testing of PWS is a time-consuming process, which is only achieved on a limited subset of configuration. PWS configuration testing is handled by Jenkins and is limited by the number of Jenkins slaves. CAMP will be used to test PWS on different OS platforms and with different databases improving a lot ActiveEon configuration testing capability by creating dedicated containers.

Customer support is an important activity of Activeeon developers. Bugs and exceptions are reported regularly by our clients. Those issues are mainly reported by mail. If the client issue is a crash, the developper usually fix it and doesn't report it in the issue tracker. The issue tracker is mainly used to keep a trace of the bugs with low priority that will not be fix. The best way in Activeeon to find a bug and his fix is to use our kanban board. In the board, a bug fixing task describes the issue and have the link to the fix commit. However there is no process to link client, issue and fix commit. As a consequence, our lack of process for client crashes makes it harder to find quickly the cause of the issues.

Identifying the origin and the exact contexts of client bugs is time-consuming and could induce delay in bug fixing due to exchanges with consumers. EvoCrash will help to reproduce customer issues and reduce investigation time. Also, the corresponding non-regression tests will be automatically produced. Moreover, thanks to our testing platforms, we have direct access (without

[2] https://github.com/diffplug/spotless

[3] https://github.com/spring-gradle-plugins/dependency-management-plugin

requiring authorisation or information from our consumers) to production logs. Using EvoCrash on such logs will allow to detect and prevent bug in a proactive way.

Atos

Atos participates in this industrial evaluation, using as evaluation context, several cases under development or industrial exploitation, which are the result of the participation of Atos in different R&D projects developed in the context of FIWARE and H2020.

CityGo is FIWARE IoT-based platform that offers a Web Portal and an Android App that citizens can use to obtain the best estimated transport route inside a city by combining several parameters such as user requirements, real time traffic data, location of vehicles, cars and bicycles parking availability and environmental data (weather forecast, social events in the city, etc.). The Web portal provides an informative dashboard that shows statistical data about the human and vehicle traffic inside the city.

CityGo is implemented in different technologies: Python and Angular for the backend and Web Portal, respectively, and Java for the Android App. A CityGo prototype is installed in Malaga city. City Go App is available in Google Play store[4] and FIWARE Marketplace[5].

SUPERSEDE develops a platform that assist software stakeholders on the adaptation for their software systems, based on gathered end-user feedback and system monitoring data, using a MAPE-K control loop. This use case focuses on: a) The Execution framework[6], which uses UML models@runtime to keep in sync the target adaptive system configuration, using pluggable enactors to interpret these UML models, generate and activate new target adaptive system configurations, b) The Integration Framework (IF)[7], that provides client-service communication among SUPERSEDE tools, using the WSO2 Integration Platform.

The SUPERSESE backend components have been developed in different languages, mostly using Java, but also Ruby and Javascript (NodeJS). Front-end components have been developed with JS/HTML.

Software Quality Assurance (QA) is an essential part of Atos IT industrial development life cycle, In the last lustrum, Atos R&D teams have been progressively adopting agile methodologies for development and DevOps pipelines for CI/CD. Testing activities have been incorporated within these practices although not as widely as on the commercial development projects.

4 https://play.google.com/apps/testing/net.atos.showcaseandroidapp2

5 http://marketplace.fiware.org/pages/solutions/6ccbccac9c87ab76c5f8ff61

6 https://github.com/supersede-project/dyn_adapt

7 https://github.com/supersede-project/integration

STAMP gives ARI a great opportunity to promote *testing* as an essential activity in the process development lifecycle and the DevOps pipeline in the development of R&D projects, with the adoption of testing best practices and test amplification techniques, which eventually can be promoted to commercial IT projects.

Tests Suites are designed and iteratively implemented and conducted during the development of these projects. However, the efficacy of these test suites to anticipate the detection of issues before delivery is largely improvable. Best practices for test planning and engineering need to be adopted, in order to cope with several testing situations that have been faced in Atos cases, including:

- Tests pass despite some runtime exceptions are thrown;
- Tests setups are not appropriate, particularly when the system-under-test (SUT) is not set to the correct precondition state;
- Tests assertions are not always correct or complete;
- Test coverage is low, hence the complete SUT is not adequately tested;
- Regression bugs are not detected by test suites, so they pop up among releases;
- We are suffering flaky tests because of race and environmental conditions (i.e. network cutoffs, unavailability of test input resources, etc);
- The analysis and reproduction of runtime failures are time-consuming and a human-specific task, where automation seems not to be possible;
- Testing maintenance is costly. Inadequate maintenance increases the occurrence of flaky tests, false positives, hiding of regression bugs, etc.

Test execution also faces a number of challenges detected in the context of Atos UCs. Test preparation and mocking requires significant engineering work and maintenance. Since most of test suites available in Atos UCs are integration and system tests, the delivery and configuration of the SUT requires complex DevOps pipelines for CI/CD. This is particularly problematic when test are executed locally (by individual developers), even when a target remote SUT is available. In particular, the eventual future new commercial distributions of CityGo will require the conduction of system and integration assessment in different configurations of the SUT, in particular:

- Configurations that modify the baseline dependencies of the SUT, including OS, databases, programming/execution frameworks, application containers, and other dependencies. Assessing the functional equivalence of these configurations (i.e. successful passing of sanity checks) is essential for the delivery of CityGo to new cities. These new configurations should embrace the technology baseline imposed by the infrastructure of the hosting city.
- Configurations that improve the capacity of CityGo backend and front-end (Android App, Web Portal) w.r.t. some non-functional properties (e.g. performance, resources consumption, availability, etc). This ability is required to tune CityGo parameterization to the execution needs (e.g. number of users, average workload per user) on the different CityGo instantiations.

OW2

In this section we present the context of the OW2 use case and OW2's specific validation criteria. OW2 is a non-profit organization which mission is to foster the growth of a portfolio of freely accessible open source software for enterprise information systems. The functional scope cover infrastructure software at large including middleware such as application servers, enterprise service bus, transaction monitor and portal, application platforms such as business intelligence, content management, identity management, cloud computing, incident and event management (SIEM), asset management, business process management, etc. and the tools to develop and manage them. From a language point of view, although there is no exclusive, most of the code base is written in Java which explains why OW2 invests in quality and testing tools for Java. The growth of the code base is managed by the community of project leaders who are all members of the technology council. They accept new projects in the code base, manage their evolution through the OW2 project lifecycle and decide new technology orientations. The governance constraint and community accountability is a significant aspect of this use case.

OW2's approach of the use case is both technical and business. OW2 is interested in validating the technical performance of the STAMP test tools but also their relevance in the context of OW2's governance and value proposal. The OW2 use case has four validation objectives:

- **Does STAMP bring value to OW2's projects and project leaders?** OW2 will leverage different projects to test the different STAMP tools. Validation will be achieved by working closely with the project leaders so as to help them improve the quality of their test suites and overall projects. Half a dozen projects have been identified and approached. While each project can be better suited to match a specific STAMP tool, all tools will be applied, when relevant to all projects. The use case will primarily concentrate on the following projects:
    - DSpot will be used on Sat4 (https://gitlab.ow2.org/sat4j/sat4j)j: Sat4j is a set of tools to solve SAT problems. It has a large collection of test files so it is a good candidate for test amplification with DSpot.
    - Descartes will be used on Authzforce (https://gitlab.ow2.org/authzforce): Authzforce is an authorization server, they put a high focus on the quality of their test suites. Descartes is a relevant choice for this project.
    - CAMP will be used on DocDoku (https://github.com/docdoku): DocDoku is a business data management platform, they have different kind of configurations available. CAMP, as configuration amplification, can use different docker sample files available on DocDoku Github repository.
    - For the EvoCrash tool two enterprise-class projects have been selected and approached: Bonita (https://github.com/bonitasoft), a BPM platform, and Knowage (https://github.com/KnowageLabs/Knowage-Server), a BI platform, because they

are In the OW2 code base among the projects which generate a high volume of logs.

- **Can OW2 integrate STAMP in the project lifecycle management?** The OW2 project lifecycle runs through three stages: incubation, mature and archive. To qualify as mature, a project must comply with ten categories of criteria. Software testing is one of these categories. OW2 has incorporated into its governance process a requirement that, for a project to be moved from incubation to mature, it must produce a report on the quality of its code and on its IP compliance. Other criteria include documentation, feature road-mapping, contributor management, etc. The second objective is to validate with the OW2 community whether the STAMP tools will help enhance the project life cycle testing criteria category.
- **Can OW2 offer STAMP as a service to its community?** OW2 could leverage the STAMP use case to incorporate the new testing tools as a service to its community with the strategic objective to enhance the market positioning of its projects. OW2 is currently developing a comprehensive approach targeted at conventional decision makers who are not necessarily open source supporters. The overall strategic objective is to make them perceive OW2 software as reliable as proprietary software. We have developed the OW2 Market Readiness Levels (OW2 MRL), a series of business-related situations that reflect the evolution of a project from initial software development to market leadership. The third objective is to validate whether STAMP can be offered as a service to help better defined the MRL rating of a project. The idea behind this is that STAMP can contribute to making projects more reliable and acceptable by conventional decision makers who need to be convinced of the quality of open source components.
- **Can OW2 technically integrate STAMP into OW2's technical infrastructure?** OW2 is an independent organization and independence is a key advantage in the world of open source software. An essential element of this independence is that OW2 runs its own technical infrastructure. Built around GitLab community edition, the infrastructure offers development teams some 15 services including source code management, issue tracking, continuous integration, contributor management, cloud deployment, wikis, mailing lists, etc. OW2's quality-oriented resources and services are grouped under the OSCAR generic name. OSCAR stands for Open Source Capability Assessment Radar, it is both a platform of technology resources and a methodology for providing decision support results. It is the technical platform which provides the data helping the technology council manage the projects life cycles. The infrastructure includes software analysis tools such as SonarQube and ScanCode. The fourth validation objective is to evaluate if and how STAMP can be integrated in the infrastructure.

Tellu

Tellu provides cloud services for collecting and processing data, with a focus on IoT. Tellu provides this to service providers and other partners in different domains, for integration in their solutions. Our current commercial focus is in e-health, communal care and personal safety and security domains. Tellu is currently operating two main commercial instances of the service, both of which are multi-tenant and hosting multiple service providers. One is running on a Norwegian hosting provider and one is running in Amazon's cloud infrastructure.

Tellu services are built on its own software platform: **TelluCloud.** This generic platform includes edges for communicating with a multitude of different devices, an internal data model and storage, processing by rule engine with resulting actions, APIs and a management web interface. The platform has undergone a large refactoring, switching to a micro-service architecture and splitting the main part of the system up into about a dozen micro-services. This 3.x branch of TelluCloud is Tellu's use case in STAMP. All core backend code is Java. The micro-service version is split into a large number of separate GIT repositories and Maven projects (around 30), and we have selected three key projects for the unit test level. The whole cloud system is the use case for configuration and runtime test amplification.

Splitting up the system into smaller parts has made the development and maintenance of each part more manageable. However, it has also introduced new challenges. The complexity of the system as a whole has increased, especially with respect to configuration and deployment, as there are many more parts to deploy, including queues and other infrastructure. It became very important that each micro-service has well-defined APIs and protocols. In addition to unit testing, testing of each micro-service and of the system as a whole became paramount.

Tellu's overall objective is to improve and automate testing and logging for TelluCloud 3.x, to ensure its correctness and robustness in a cost-effective manner so that Tellu can continue to develop and deploy new versions without introducing bugs which disrupt the services. Tellu's objectives in STAMP are described based on the three forms of amplification in the project.

For unit testing, Tellu's objectives are to increase the number of JUnit tests for the TelluCloud source code, increasing test coverage, as well as improving the quality of the tests. The first part is easy to quantify with tools measuring test coverage. For test quality, Tellu's initial objective was to learn about state of the art practices and tools and techniques such as mutation testing. With mutation testing Tellu gets a metric to help quantify test quality, so an objective is to compute this metric on TelluCloud projects and improve it. This should be integrated into the build chain.

For runtime test amplification, Tellu's initial objective is to get tools and methodologies to test the TelluCloud services at runtime, to check that running services are behaving according to specifications. More specifically, it is an objective to improve the quality of logging done by the system, and to get intelligent monitoring of the logs, to get automatic discovery and analysis of problems and inconsistencies.

Tellu is especially interested in configurability test amplification, with its aspects of managing diverse configurations and deploying a complex system in various configurations. TelluCloud is made to be deployed in different configurations. For deployment and test, it is important to be able to deploy both micro-services and full systems in a quick and easy manner. For production, it is important for Tellu to be able to move to different forms of cloud hosting depending on technological developments, pricing and customer needs. Tellu foresees the need to continue to support at least two different production deployments. Some customers are best served with an instance running in Amazon's cloud infrastructure. But Tellu is also heavily involved in e-health in Norway, and these customers want the data to be stored in Norway, requiring a more local solution.

There are many factors which may vary:

- Configuration of each type of micro-service.
- Number of instances of each service.
- Data persistence configurations.
- Connectivity between micro-services, configuration of queues.
- Other infrastructure, such as service registry.
- Connections to external services.
- Type of deployment: single local machine for testing, server infrastructure, auto-scaling cloud deployment.

The first objective here is to orchestrate and automate deployment of the system, to facilitate efficient and automated testing. The next objective is to amplify configurations and test inputs, generating new variations of configurations based on existing ones. Finally, there are objectives to run different types of tests on the system:

- Functional tests: Verify correct service and system outputs based on inputs.
- Quantify performance and scalability, to find optimal configurations (performance vs price) and verify correct distribution and load balancing.
- Stress testing, ensuring that the response times and throughput latencies are within specified limits.

Finally, and common to all forms of testing amplification, is the objective to operationalize the tools and technologies. Tellu's STAMP objectives are tied to a goal to move towards DevOps. With the move to a micro-service architecture Tellu wants to be able to release new versions of separate services often. Strong automated testing of both the service and the system is a prerequisite.

XWiki

XWiki is an advanced Enterprise open source wiki tool written in Java but it's also a generic web-development platform that can be used to develop any kind of web applications, especially if they relate to content. Its features are numerous and this makes it a large and complex platform.

XWiki currently boasts more than 700 extensions providing additional features (Blog, Meeting Manager, LDAP integration, etc).

For the STAMP project we've decided to focus on XWiki Standard. The code for XWiki Standard is spread across 3 repositories in GitHub (totalling about 650K of code):

- xwiki-commons: common modules that can be used outside of XWiki
- xwiki-rendering: the XWiki Rendering engine (i.e. libraries used to convert an input in a given syntax into another target syntax, e.g. from XWiki Markup to HTML)
- xwiki-platform: all the wiki features

The following use cases are of interest to the XWiki project:

- **Improving Flaky Test handling**: Several times per week, XWiki's Continuous Integration server (CI) sends false positives to the XWiki open source project's notifications mailing list. This list is used by XWiki developers working on the project. When they receive such emails, they usually stop what they're doing to analyze the problem, spending several minutes or hours to finally discover that the problem is actually either a build environment problem or a flickering test. This is wasting their time and preventing them from progressing faster on their task at hand. In addition, it decreases the value given to the CI, and, over time, developers stop caring about build failures coming from the CI till the day of the release and then all integration problems need to be fixed, this delaying the release a lot, and negating the exact purpose of a CI tool which is to discover integration problems as they happen.
- **Improving test coverage and test accuracy**: XWiki already has a relatively high level of test coverage (around 70% at the start of the STAMP project). However, the community raises bugs every day, showing that we still need more tests. Test coverage is not an absolute measure of the quality and getting to 100% is not the goal but increasing it to around 80% would help a lot. Especially in parts of the code where this coverage is low currently. Now test coverage doesn't guarantee the quality of the test (a test with no asserts will generate coverage but won't prove much). Thus it's also important to evaluate the quality of the tests. This is where mutation testing can help out. In addition it would be awesome to have tests automatically generated and that would increase automatically the test coverage/accuracy.
- **Configuration testing**: XWiki is web application deployed as a WAR which can be installed on any Servlet Container and running on any Database. It can also run in all browsers. Thus it's very important that the XWiki tests are executed in various environments to prove that the subset of supported environments work, and that no regression are brought to any of those environments when new code is added. In addition

and to make XWiki's distribution simpler to set up, we want to package those environment setups as Docker images.

- **Reduce time to debug runtime issues**: When XWiki users have a problem in production, they report it through the XWiki issue tracker. They are asked to include the stack trace in the issue report. It would be nice and interesting to make it simpler to reproduce the issue by having some tool generate automatically the test, that, when executed, leads to this stack trace. This is the ambition of EvoCrash and something the XWiki project is keen to test.

STAMP Tools/Features target of evaluation ^{NEW}

The evaluation task will assess the industrial fit-for-a-purpose of the tools/features developed by the STAMP project, collected in Table 1. At the time of writing, there are available stable versions for al thel STAMP tools. However, not all their features are available, in particular, not all the planned user interfaces (CLI, Maven, Gradle, Eclipse IDE, SaaS, Docker) are available for being referenced.

Table 1 STAMP Tools/Features target of the evaluation task

Tool/Feature		
DSpot	Description	DSpot is a tool that generates missing assertions in JUnit tests. DSpot takes as input a Java project with an existing test suite. As output, DSpot outputs new test cases on console. DSpot supports Java projects built with Maven and Gradle
	Documentation	https://github.com/STAMP-project/dspot/blob/master/README.md
	Tutorial for dummies	https://github.com/STAMP-project/dspot/blob/master/docs/dspot-for-dummies.md
	Source Code	https://github.com/STAMP-project/dspot

	Tracker	https://github.com/STAMP-project/dspot/issues	
	User Interface	CLI	Available (included in release)
		Maven	Available: https://github.com/STAMP-project/dspot/tree/master/dspot-maven
		Gradle	Planned
		Eclipse IDE	Available: https://github.com/STAMP-project/eclipse-ide
		SaaS	Planned
		Docker	Planned
Descartes	Description	Descartes evaluates the capability of your test suite to detect bugs using extreme mutation testing.	
	Documentation	https://github.com/STAMP-project/pitest-descartes/blob/master/README.md	
	Tutorial for dummies	https://github.com/STAMP-project/pitest-descartes/blob/master/docs/descartes-for-dummies-mvn.md	
	Source Code	https://github.com/STAMP-project/pitest-descartes	
	Tracker	https://github.com/STAMP-project/pitest-descartes/issues	
	User Interface	CLI	Available (based on Maven or Gradle)
		Maven	Available (included in release): https://github.com/STAMP-project/pitest-descartes#maven
		Gradle	Available (included in release):

			https://github.com/STAMP-project/pitest-descartes#gradle Tutorial for dummies: https://github.com/STAMP-project/pitest-descartes/blob/master/docs/descartes-for-dummies-gradle.md
		Eclipse IDE	Available: https://github.com/STAMP-project/eclipse-ide
		SaaS	Initial version of a github app https://github.com/STAMP-project/descartes-github-app
		Docker	Not planned
CAMP	Description	CAMP (Configuration AMPlification) takes as input a sample testing configuration and generates automatically a number of diverse configurations. The generation is guided by predefined features and constraints, and utilizes a set of reusable pieces. The current version of CAMP is focused on the Docker environment, and the input and output configurations are specified as Dockerfiles or docker-compose files.	
	Documentation	https://github.com/STAMP-project/camp/blob/master/README.md	
	Tutorial for dummies	N/A	
	Source Code	https://github.com/STAMP-project/camp	
	Tracker	https://github.com/STAMP-project/camp/issues	
	User Interface	CLI	Available (based on Docker)
		Maven	Planned

		Gradle	Planned
		Eclipse IDE	Planned: https://github.com/STAMP-project/eclipse-ide
		SaaS	Planned
		Docker	https://hub.docker.com/r/songhui/camp
Evocrash	Description	EvoCrash is a search-based approach to automated crash reproduction. EvoCrash receives a Java crash stack trace, and searches for a unit test case that can reproduce the crash.	
	Documentation	https://github.com/STAMP-project/EvoCrash/blob/master/README.md	
	Tutorial for dummies	N/A	
	Source Code	https://github.com/STAMP-project/EvoCrash	
	Tracker	https://github.com/STAMP-project/EvoCrash/issues	
	User Interface	CLI	Available (based on Gradle)
		Maven	Planned
		Gradle	Available (included in release)
		Eclipse IDE	Planned: https://github.com/STAMP-project/eclipse-ide
		SaaS	Planned
		Docker	Planned

Evaluation will target stable versions of these tools and their interfaces. Tool releases are available in their GitHub repositories, as shown in Table T2

Table T2 STAMP Tools latest releases

Tool	Release	Link
DSpot	1.1.0 16-04-18	https://github.com/STAMP-project/dspot/releases/tag/1.1.0
Descartes	1.2.2 28-06-18	https://github.com/STAMP-project/pitest-descartes/tree/descartes-1.2.4
CAMP	latest	https://github.com/STAMP-project/camp
Evocrash	latest	https://github.com/STAMP-project/EvoCrash

8. Validation Roadmap

8.1. Alignment to the STAMP Toolset Release plan

This section introduces the timeline of validation activities that will be conducted along the project lifetime, structured in a number of phases, which are aligned to the STAMP tool and service release plan (see Table 3). According to the STAMP GA, a number of tool and service releases will made available according to the following schedule:

- STAMP Initial prototype on M12-M14 (November 17-January 18);
- STAMP Enhanced prototype on M20-M24 (July 18 - November 18);
- STAMP Consolidated tool on M34 (Sep 19);
- STAMP Final services on M36 (Nov 19).

Table 3 STAMP Tool and Service releases

Release	Date/ Milestone	Deliverables (Responsible)
Initial prototype	M12/MS8 M14	D12 Initial prototype of the unit test amplification tool (INRIA) D22 Initial prototype on configuration test amplification (SINTEF) D32 Initial prototype of log optimization tool (TUD) D42 First public version of the API and initial implementation of services (ENG)
Enhanced prototype	M20/MS12 M24	D13 Enhanced prototype of the unit test amplification tool (INRIA) D23 Enhanced prototype of the configuration amplification (SINTEF) D33 Prototype of amplification tool for common and anomaly behaviors (TUD) D43 Second public version of the API and consolidated implementation of services (ENG)
Consolidated tool	M34/MS15	D14 Consolidated tool for the unit test amplification, selection and execution (INRIA) D24 Consolidated tool for the configuration amplification, selection and execution (SINTEF) D34 Consolidated services for online-test amplification (TUD)
Final services	M36	D44 Final public version of the API and consolidated implementation of services (ENG)

Above Table 3 further identifies the dates (and milestone) of each release, associated STAMP deliverables and responsible partners. Note that first and second releases in the table 3 aggregate the releases of the tools and services, which separately take place with 2 month span. This is done so, because the evaluation activities target the assessment of the complete toolset, not only core tools, but their industrialization with tool facades (e.g. Maven/Gradle, CLI, IDE clients, services) as well. The exception to this approach is the explicit split of releases: *Consolidate Tool* and *Final Services* at M34 (MS15) and M36. In this case, final evaluation activities target the *Consolidate Tool* release, and not the *Final Services,* since they are released at the end of the project. Nonetheless, intermediate releases of the *Final Services* will be evaluated during the final stages of the project.

8.2. Validation phases and timeline UPDATED

The validation timeline is organized in such a way that we address the following validation objectives:

- The validation activities provide **early and continuous feedback** to the development teams in technical WP1-WP4, during the entire development lifecycle, starting as early as initial prototypes are available for assessment.

- The validation activities are organized in such a way that they provide **iterative, increasing and agile feedback**:
  - Iterative:
    - **assessing new toolset releases** (internal for the consortium or external for the public) as soon as they are published. New tool releases will be made available within the *releases* tab of the tool repository at the GitHub STAMP portal;
    - **assessing new toolset minor versions**, as soon as they are patched in reaction to communicated bugs, defects or requested features. Patches will be notified by the tool development team within the thread discussion in the GitHub Issue Tracker associated to the tool repository;
  - Incremental:
    - incrementing the assessment precision and/or complexity by providing **deeper assessment in increasingly complex usage scenarios**;
  - agile:
    - **adapting the assessment process to the toolset maturity and the validation objectives** agreed between tool developers and evaluators.

For observing these considerations, the validation timeline has been organized in 3 phases that are aligned with the STAMP tools and services release plan, in a way that it optimizes the feedback provided to the development team at the suitable time that it can usefully support the development of the tools.

- **Phase 1 (M6-M18):** focuses on **validating the main concept and the functionality of the STAMP toolset**, from the industrial point of view, **engaging other potential industrial adopters**, as well as on providing early feedback to support the development of a usable and stable toolset;
- **Phase 2 (M19-M24)**: focuses on **validating the first stable STAMP toolset**, focusing of their usability, effectiveness, efficiency and the perceived fit-for-a-purpose benefit in a variety of industrial real scenarios. It also starts **conducting initial in-lab controlled experiments to verify the fulfillment of the hypothesis claimed in KPIs**;
- **Phase 3 (M25-M36)**: focuses on **validating the consolidated STAMP toolset**, focusing of their usability, effectiveness, efficiency and the perceived fit-for-a-purpose benefit in a variety of industrial real scenarios and on **engaging open communities of developers** for reporting feedback on the usage of the STAMP toolset on common daily development and testing activities. It also conducts **extensive In-Lab controlled studies aiming at verifying the fulfillment of the hypothesis and claims associated to the baseline of KPI** described in the GA. Eventually, STAMP KPIs can be compared to those KPIs measured when an industrial case adopts any previous state of practice technique that targets functional challenges similar to STAMP ones.

Phase 2 has been shortened with regards to the previous timeline defined in D5.2 in reaction to the interim review recommendation.

Next Table 4 lists the breakdown of evaluation activities included in the roadmap, structured according to the different phases. Each activity is positioned into the STAMP project calendar. Activity coordinator and contributors are also specified. The associated official deliverable for reporting the evaluation activities and findings is identified. See section 10.7 for a description of the validation activity types mentioned in the *Evaluation Activity* column.

Table 4 STAMP evaluation roadmap: breakdown of activities

Evaluation Activity	Objectives, Tool Target	Calendar	Coordinator	Contributors	Associated Reporting Deliverables
Phase 1	Main STAMP concept and functionality validation. Early feedback on tool prototypes **Tool Target**: Initial Prototype (D12, D22, D32, D42)	M6-M18	Aeon	Atos, TellU, XWiki, OW2	D5.5 UC Validation Report V1
Pilot Trial	Reporting usage experiences with initial prototypes: issues and potential bugs. Assessment of concrete requirements and KPIs	M6-M18	Aeon	Atos, TellU, XWiki, OW2	
User Survey	Collect feedback from potential external adopters about their testing practices and requirements for STAMP. The User survey questionnaire designed for this purpose is included in Appendix C (Section 12)	M10-M18	Atos	AEon, TellU, XWiki, OW2	
Focus Group	Refinement, formalization and prioritization of STAMP requirements and KPIs	M14-M18	XWiki	AEon, Atos, TellU, OW2	
Phase 2	Usability, effectiveness, efficiency and fit-for-purpose validation. Initial verification of KPIs' hypothesis **Tool Target**: Enhanced Prototype (D13, D23, D33, D43)	M19-M24	OW2	Aeon, Atos, TellU, XWiki	D5.6 UC Validation Report V2
Pilot Trial	Reporting usage experiences with enhanced tools: issues and potential bugs.	M19-M24	TellU	Aeon, Atos, OW2, XWiki	
User	Collect feedback about the STAMP approach, concept, toolset	M19-M24	Atos	AEon, TellU,	

Survey	functionality from potential external adopters.			XWiki, OW2	
Focus Group	Refinement, formalization and prioritization of STAMP requirements and KPIs	M19-M20	XWiki	AEon, Atos, TellU, OW2	
In-lab controlled experime nts	Assess the efficacy and efficiency of the STAMP enhanced tools in tasks conducted in common industrial scenarios. Initial assessment of the hypothesis associated to the baseline of KPIs. One or more in-lab (comparative) experiments will be scheduled during at the end of this period (M29-M30)	M21-M24	XWIKI	Aeon, Atos, TellU, OW2	
Phase 3	External validation from open communities of developers. Final verification of KPIs' hypothesis. **Tool Target**: Consolidated Tool (D14, D24, D34), Final Services (D44)	M25-M34	Atos	AEon, TellU, XWiki, OW2	D5.7 UC Validation Report V2
Pilot Trial	Reporting usage experiences with enhanced tools: issues and potential bugs.	M25-M34	XWiki	Aeon, Atos, OW2, TellU	
Open Field Trial	Evaluate the STAMP concepts, methods and tools in daily practical testing situations dealt with by developers engaged in open-source development communities	M31-M36	OW2	Aeon, Atos, TellU, XWiki	
In-lab controlled experime nts	Assess the efficacy and efficiency of the STAMP final tools in tasks conducted in common industrial scenarios. Final assessment of the hypothesis associated to the baseline of KPIs. One in-lab (comparative) experiment will be conducted in this period	M31-M36	Atos	AEon, TellU, XWiki, OW2	

Above evaluation activities will be conducted in the context of the following GA WP5 tasks (see GA):

- T5.3: ProActive Scheduling and Workflows (ActiveEon) case validation;

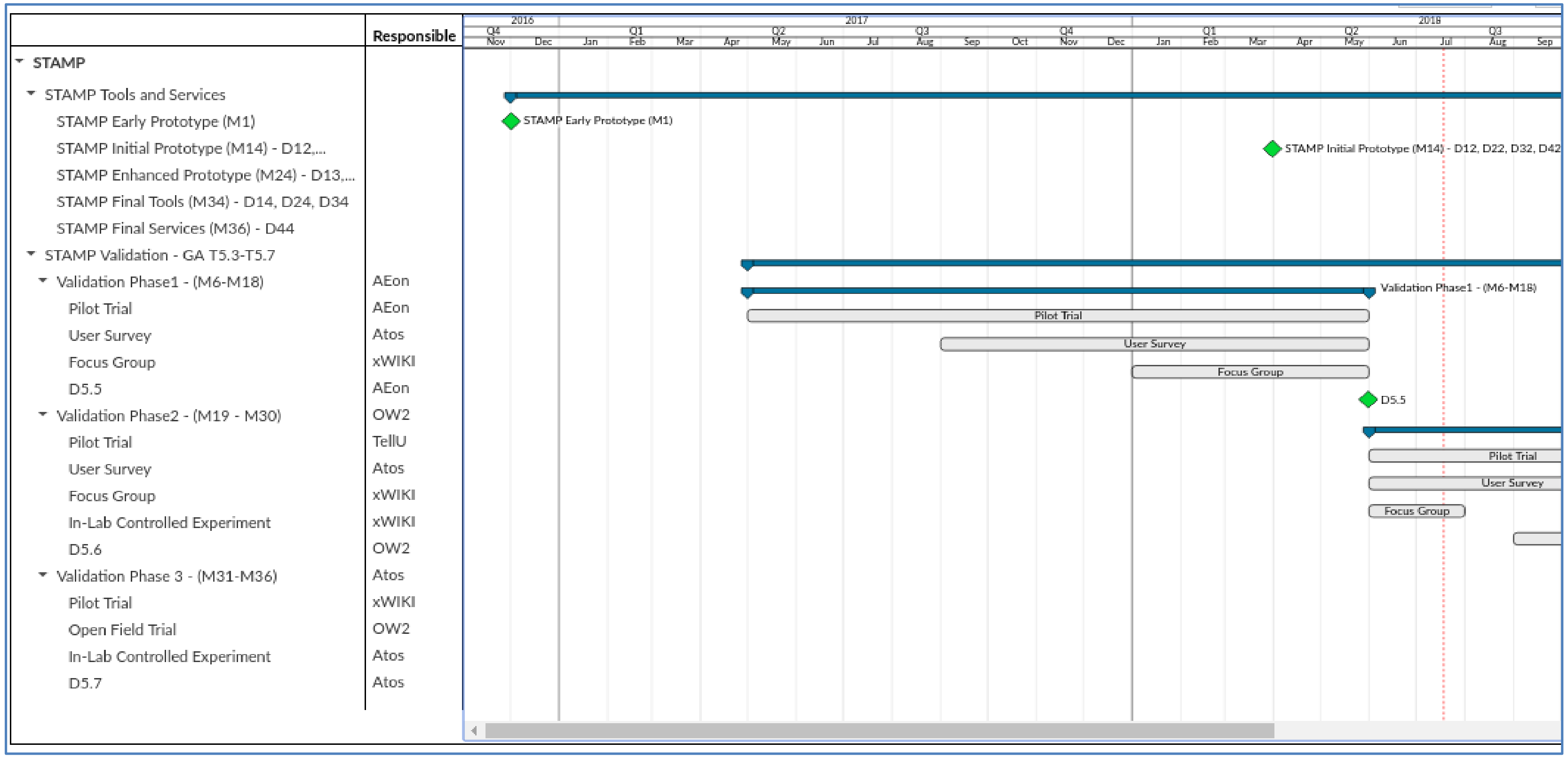
- T5.4: FIWARE Ecosystem (ATOS) case validation;
- T5.5: TellU case validation;
- T5.6: xWIKI case validation;
- T5.7: OW2 case validation.

While in the GA the validation tasks are structured per industrial case, above activity breakdown for the roadmap is structured by validation activity type (i.e. targeting different objectives) and phases. This breakdown is more adequate for the purpose of providing a detailed roadmap than the task structure for WP5 described in the GA. There is not a direct mapping between above roadmap activities and WP5 tasks. Nonetheless, all above roadmap activities will be conducted in the context of the five STAMP industrial case tasks. Hence, they will be contributed by all the STAMP industrial partners, namely, ActiveEon, Atos, TellU, xWiki, OW2 (see section 7 for brief introduction of use cases and overall objectives). Indeed, roadmap activities will make use of the resources available for each industrial partner in the context of the GA tasks T5.3-T5.7. In terms of periodic reporting, above evaluation activities in Table 4 will be justified in the context of GA T5.3-T5.7 as well.

Figure 1 and Figure 2 show the Gantt diagram of the evaluation roadmap[8], aligned to the STAMP release plan (shown at the top of the diagram). *Responsible* column shows the coordinator of each validation activity. Remaining industrial partners are contributing to each validation activity, as mentioned in Table 4[9].

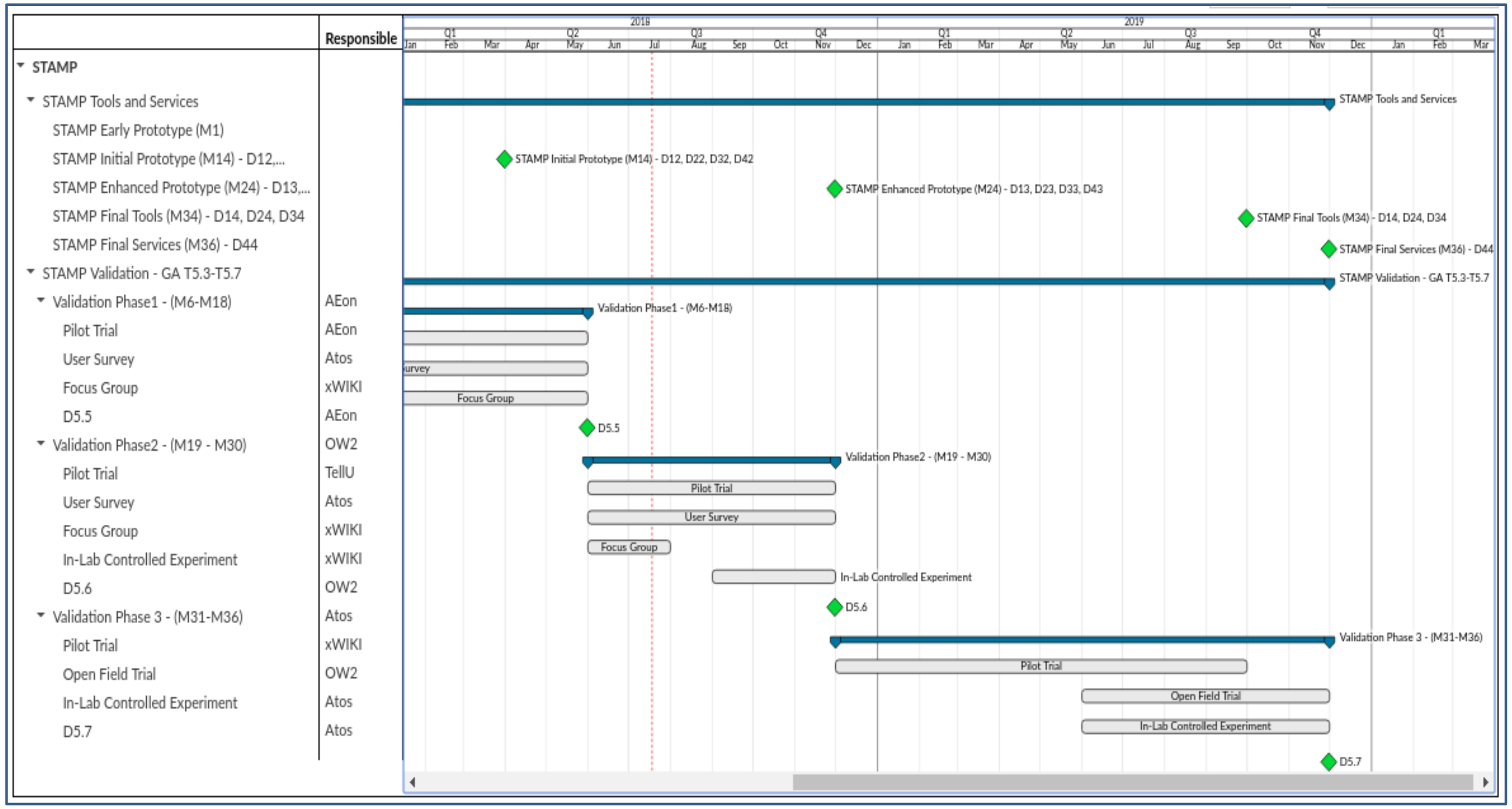
8 Gantt diagram is split into two figures in order to improve its readability

9 Contributors are not explicitly mentioned in the Gantt diagram in order to keep it simple.



	Responsible
STAMP	
STAMP Tools and Services	
STAMP Early Prototype (M1)	
STAMP Initial Prototype (M14) - D12,...	
STAMP Enhanced Prototype (M24) - D13,...	
STAMP Final Tools (M34) - D14, D24, D34	
STAMP Final Services (M36) - D44	
STAMP Validation - GA T5.3-T5.7	
Validation Phase1 - (M6-M18)	AEon
Pilot Trial	AEon
User Survey	Atos
Focus Group	xWIKI
D5.5	AEon
Validation Phase2 - (M19 - M30)	OW2
Pilot Trial	TellU
User Survey	Atos
Focus Group	xWIKI
In-Lab Controlled Experiment	xWIKI
D5.6	OW2
Validation Phase 3 - (M31-M36)	Atos
Pilot Trial	xWIKI
Open Field Trial	OW2
In-Lab Controlled Experiment	Atos
D5.7	Atos

Figure 1 STAMP Validation Timeline (Gantt Diagram) – Phase I



	Responsible
STAMP	
STAMP Tools and Services	
STAMP Early Prototype (M1)	
STAMP Initial Prototype (M14) - D12,...	
STAMP Enhanced Prototype (M24) - D13,...	
STAMP Final Tools (M34) - D14, D24, D34	
STAMP Final Services (M36) - D44	
STAMP Validation - GA T5.3-T5.7	
Validation Phase1 - (M6-M18)	AEon
Pilot Trial	AEon
User Survey	Atos
Focus Group	xWIKI
D5.5	AEon
Validation Phase2 - (M19 - M30)	OW2
Pilot Trial	TellU
User Survey	Atos
Focus Group	xWIKI
In-Lab Controlled Experiment	xWIKI
D5.6	OW2
Validation Phase 3 - (M31-M36)	Atos
Pilot Trial	xWIKI
Open Field Trial	OW2
In-Lab Controlled Experiment	Atos
D5.7	Atos

Figure 2 STAMP Validation Timeline (Gantt Diagram) – Phase II and III

The remaining of this section provides details about the each validation phase. See section for a description of the elements of the evaluation framework mentioned in the tables. See Appendix B for a description of the elements of the evaluation method mentioned in the tables.

8.2.1. Phase 1

Table 5 Phase 1 description

Phase	Phase 1
Timeline Frame	M3-M18
Validation Objectives	● Provide an early assessment on the functionality and usability of the STAMP techniques and the toolset; ● Provide an initial empirical assessment on the practical utilization of the early and initial prototypes of the STAMP toolset; ● Involve end-users in the development process of the STAMP toolset, aiming at aligning the toolset features and usage to their industrial needs.
Validation Activity Types	● User surveys; ● Pilot trials. ● Focus group In the period M3-M9, before the STAMP validation framework is reported in this document, we are conducting informal, not systematic (e.g. not methodological and framework supported) validation activities, consisting of using existing early STAMP prototypes of the toolset for test amplification on the different industrial use cases. Findings (e.g. metrics reported in D5.1 and D5.3 about test coverage), usage issues, bugs, required features, etc. are being reported through different channels, including informal discussion threads on chats or by email, and formal discussion in trackers (GitHub). Although informal discussion (e.g. using thread, chats, virtual meetings) is permitted, it is encouraged the formalization of the feedback provided to the tool developers through the use of the issue tracker associated to each tool repository in GitHub (see Table T1). Additionally, a discussion forum among developers and industrial evaluators have been established in the STAMP GitHub portal: https://github.com/STAMP-project/docs-forum After M9, a systematic validation, adopting this framework and following this

	proposed roadmap, will be conducted.
Validation Expected Outcome	● Early feedback about toolset features and usability for developers on technical WP1-WP4: • Communication of additional required features and/or refinement/improvement of existing ones. The internal elicitation of new/refined features for STAMP tools in the scope of the industrial use cases will be conducted in the pilot trial activity of this phase (and also in phase II). • Reporting on toolset utilization issues (e.g. bugs, usage limitations, configuration, etc.); ● Industrial scoped recommendations for toolset improvement.
Reporting	D5.5 UC validation report V1, M18

8.2.2. Phase 2

Table 6 Phase 2 description

Phase	Phase 2
Timeline Frame	M18-M24
Validation Objectives	● Provide an empirical assessment on the practical utilization of the stable prototypes of the STAMP toolset; ● Involve end-users in the development process of the STAMP toolset, aiming at aligning the toolset features and usage to their industrial needs; ● Assess the benefits of adopting the STAMP toolset on industrial cases compared to those gained using third-party solutions (see D6.3 [14] for an market analysis that collect these competitive solutions: see also section 10.7.3 for a list of these solutions associated to proposed comparative experiments).
Validation Activity Types	● Pilot trials; ● User surveys; ● Focus group; ● Comparative experiments. In both validation activities, we conduct a systematic, validation framework assisted assessment of the STAMP toolset adoption on real industrial cases, in Lab controlled testing environments.

Validation Expected Outcome	● Feedback about toolset utilization for developers on technical WP1-WP4: • Reporting on issues (e.g. bugs, usage limitations, configuration, integration, etc.); ● Final refinement of proposed KPIs and industrial requirements; ● Reports on KPI assessment and hypothesis fulfillment: • KPI metric figures for pilot trials and comparative experiments; ● Industrial scoped recommendations for toolset improvement.
Reporting	D5.6 UC validation report V2, M24

8.2.3. Phase 3

Table 7 Phase 3 description

Phase	Phase 3
Timeline Frame	M25 - M36
Validation Objectives	● Provide an empirical assessment on the practical utilization of the final stable prototypes of the STAMP toolset; ● Assess the benefits of adopting the STAMP toolset on industrial cases compared to those gained using third-party solutions; ● Promote the adoption of STAMP techniques and methods in open communities of developers; ● Collect feedback from open communities of developers on their usage of the STAMP toolset in real development situations.
Validation Activity Types	● Pilot trials; ● Open Field Trials; ● Comparative experiments.
Validation Expected Outcome	● KPI hypothesis assessment: • Compared KPI metric figures, adopting STAMP toolset and competitive solutions; ● Final assessment conclusions; ● Industrial partner recommendations for future work.
Reporting Date	D5.6 UC validation report V2, M36

9. Conclusion

This document describes the method, framework and roadmap for the industrial-driven evaluation of the STAMP toolset and services. The method defines the process to conduct the evaluation activities. The framework provides the tools for supporting the evaluation activities. The roadmap defines the timeline of the activities, in agreement with the development roadmap. These elements, working together, drives the STAMP evaluation and the reporting of resulting findings and recommendations, aiming at improving the effectiveness, efficiency and user-satisfaction of industrial adopters on their real-life testing situations that require test amplification.

This document provides the final specification of the method, framework and roadmap, which have been improved and extended from the initial version reported in D5.2 (M9). This version has largely been refined in reaction to the lessons learnt during its utilization in the first evaluation period (M9-M20), paving the way for the remaining evaluation until the end of the project.

In particular, in this version of the roadmap we have anticipated the second evaluation period to M24 in order to have a evaluation checkpoint in the second year of the project, as recommended during the interim project review. The STAMP toolset target of the industrial evaluation has been clearly identified (together with their user interfaces). Moreover, the industrial stakeholders' expectations and objectives that drives the evaluation process have been stated in the context of the five use cases.

On the context of the framework, the product quality model that drives the qualitative evaluation have been identified, based on industrial standards. A concrete set of comparative studies that will drive the in-lab controlled validation activities have been proposed for each STAMP tool in order to compare them to the current industrial state of the practice.

Finally, a beta campaign has been proposed in order to support the engagement of software testers of open source communities of developers in the evaluation of the STAMP tools, to provide us feedback and help us to build an open community for STAMP.

10. Appendix A: Elements of the Evaluation Framework

In the following subsections, the concepts and constituents of the validation framework are introduced in detail.

10.1. Quality Model, Requirements and Metrics UPDATED

The overall goal of validation is to make the software usable in a broad sense i.e. “the user can do what he wants to do the way he expects to be able to do it, without hindrance, hesitation, or questions” [3]. However, a broad range of characteristics can influence the quality of system, as defined in industry standards such as the ISO/IEC 25010:2011[10]. This standard is part of a suite known as SQuare (ISO/IEC System and Software product Quality Requirements and Evolution). It defines a hierarchical **quality model** [6], a model that defines a set of quality entities and their relationships, which allows the formal specification of quality requirements and their usage on the evaluation of software quality. Leaf quality entities in this hierarchical tree model are measurable, that is, they are metrics, which enable the quality assessment of software.

From the broad range, taking into account the purpose of the usage of the software product, a narrow set of characteristics are chosen, influenced from ISO 25010 product quality model, to drive the evaluation process. These characteristics have been selected based on the common categories of requirements related to the software that are considered in the STAMP validation process, to specify detailed objectives of user acceptance validation. These categories are as follows:

- Functionality
- Compatibility
- Performance
- Usability
- Reliability
- Portability

These categories are described below in Table 8 in more detail. For each characteristic within the category, its description, examples of validation questions and metrics that will be considered during validation are presented.

[10] ISO/IEC 25010:2011, “Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models”, 2011 http://iso25000.com/index.php/en/iso-25000-standards/iso-25010

Table 8 Product Quality model for validation (subset of ISO/IEC 25010)

Characteristic	Functional Suitability		
Functional Completeness	Definition	degree to which the set of functions covers all the specified tasks and user objectives.	
	Questions	● Does STAMP generate test cases that detect regressions?	
	Metrics	Qualitative	Qualitative subjective user's perception, expressed using a 5-level Likert scale
		Quantitative	● Functional metrics defined in T5.1 (reported in D5.1 and D5.3) for identified KPIs; ● Number of new/refined elicited functional requirements, collected in trackers;
Functional Correctness	Definition	degree to which the functions provides the correct results with the needed degree of precision.	
	Questions	● Does STAMP improve the test coverage for Java based projects? ● Does STAMP generate valid test cases?	
	Metrics	Qualitative	Qualitative subjective user's perception, expressed using a 5-level Likert scale
		Quantitative	● Functional metrics defined in T5.1 (reported in D5.1 and D5.3) for identified KPIs; ● Measured results of comparative experiments.
Functional Appropriateness	Definition	degree to which the functions facilitate the accomplishment of specified tasks and objectives.	
	Questions	● Does STAMP help to find optimal deployment configurations that maximizes the performance of services? ● Does STAMP help to generate test cases that reproduces	

		exceptions shown in runtime logs?	
	Metrics	Qualitative	Qualitative subjective user's perception, expressed using a 5-level Likert scale
		Quantitative	● Functional metrics defined in T5.1 (reported in D5.1 and D5.3) for identified KPIs; ● Measured results of comparative experiments.
Characteristic	Compatibility		
Coexistence	Definition	degree to which a product can perform its required functions efficiently while sharing a common environment and resources with other products, without detrimental impact on any other product.	
	Questions	● Does STAMP tools integrate seamlessly with CI/CD tools and pipelines? ● Does STAMP tools integrate seamlessly with IDEs and project management tools such as Maven or Gradle?	
	Metrics	Qualitative	Qualitative subjective user's perception, expressed using a 5-level Likert scale
		Quantitative	● Functional metrics defined in T5.1 (reported in D5.1 and D5.3) for identified KPIs; ● Measured results of comparative experiments.
Characteristic	Performance - Efficiency		
Time-behavior	Definition	degree to which the response and processing times and throughput rates of a product or system, when performing its functions, meet requirements.	
	Questions	● How does STAMP toolset and services performs in terms of efficiency attending to the scale of the input source code and test suites?	
	Metrics	Qualitative	Qualitative subjective user's perception, expressed using a 5-level Likert scale

		Quantitative	● Functional metrics defined in T5.1 (reported in D5.1 and D5.3) for identified KPIs; ● Measured results of comparative experiments.
Characteristic	Usability		
Appropriateness recognisability	Definition	degree to which users can recognize whether a product or system is appropriate for their needs.	
	Questions	● Do the users understand the purpose and the usage of STAMP tools? Do they miss any information? ● What are the benefits users perceive from using the STAMP toolset?	
	Metrics	Qualitative	Qualitative subjective user's perception, expressed using a 5-level Likert scale
		Quantitative	● Number of questions posted by users to the developers in the STAMP trackers/forums/mailing lists. Percentage of them correctly answered (e.g. voted as positive by users); ● Number of validation tasks that could not be completed during the comparative studies (over the total) because of wrong usage of the tools;
Learnability	Definition	degree to which a product or system enables the user to learn how to use it with effectiveness, efficiency in emergency situations.	
	Questions	● Are the toolset interfaces intuitive enough, easy to understand, learn and use by average users, depending on adopted role? ● Do they find information support for using the tools in their daily test amplification tasks? ● Are the users blocked because they can't find the procedures to use the tools in concrete tasks?	
	Metrics	Qualitative	Qualitative subjective user's perception, expressed using a 5-level Likert scale
		Quantitative	● Number of questions posted by users to the developers (for instance in a private (i.e. for STAMP consortium) or public (i.e. for external communities) forums/mailing lists). Percentage of

			them correctly answered (e.g. voted as positive by users); ● Number of validation tasks that could not be completed during the comparative studies (over the total) because of wrong usage of the tools;
Operability	Definition	degree to which a product or system is easy to operate, control and appropriate to use.	
	Questions	● Are the toolset interfaces intuitive enough, easy to understand, learn and use by average users, depending on adopted role? ● Do the users feel comfortable using this interface or do they require changes on it?	
	Metrics	Qualitative	Qualitative subjective user's perception, expressed using a 5-level Likert scale
		Quantitative	● Number of reported usability issues; ● Number of validation tasks that could not be completed during the comparative studies (over the total) because of wrong usage of the tools; ● Time required to complete validation tasks (computing only time required to manage the interface);
Characteristic	Reliability		
Maturity	Definition	degree to which a system, product or component meets needs for reliability under normal operation.	
	Questions		
	Metrics	Qualitative	Qualitative subjective user's perception, expressed using a 5-level Likert scale
		Quantitative	● Number of issues/bugs reported and collected in trackers related to the STAMP functionality: percentage of issues fixed; ● Number of validation tasks that could not be completed during the comparative studies (over the total) because of bugs in tools;

Characteristic	Portability		
Installability	Definition	degree of effectiveness and efficiency in which a product or system can be successfully installed and/or uninstalled in a specified environment.	
	Questions	- Can be STAMP tools used in different OSs, including at least Windows 7/8/10, Linux, MacOS?	
	Metrics	Qualitative	Qualitative subjective user's perception, expressed using a 5-level Likert scale
		Quantitative	● Number of issues reported and collected in trackers related to the installation of STAMP tools on different OSs: percentage of issues fixed; ● Number of validation tasks that could not be completed during the comparative studies (over the total) because of the incompatibility of STAMP tools with user's OS/environment.

The assessment of above validation objectives will be verified by computing the metrics associated to each requirement type (e.g. aspect) in above classification. In general terms, these metrics can be classified into:

Qualitative metrics: these metrics are suitable to qualify subjective opinions from STAMP users. They will be measured (from individual users) using:

- Textual feedback, when the subjective opinion cannot be foreseen beforehand. Results are processed manually;
- 5-Likert gauges [15], when the feedback type is identified beforehand and included in the evaluation questionnaires. Collected results can be processed statistically, showing relevance depending on the number of responses.

Quantitative metrics: these metrics are suitable to quantify objective facts that characterize diverse aspects on the usage of the STAMP toolset and services on controlled experiments. They should not depend on experimenters' experience and background (e.g. unbiased results). In this group we include:

- Metrics defined in D5.1 and D5.3 for selected KPIs;
- Productivity gains (for instance in productivity) measured in lab controlled experiments (see section 10.7.3) that show the benefits of adopting STAMP in real industrial scenarios, as

expressed by computed KPIs. Eventually, in those industrial scenarios that previously adopted some state of the practice techniques that target some STAMP functional aspects (e.g. better test coverage, regression bug detecting, etc.), the measured productivity gains will be also compared to these state of practice baseline;

- Number of reported issues or bugs (and the percentage that are addressed or fixed), which can be easily computed for the Github trackers. Available tools, such as Cucumber (https://github.com/cucumber/github-issue-stats) could help to compute this and similar Github metrics;
- Number of requested features (and the percentage of them, that having being accepted in the scope of the STAMP project, are implemented), also computable from the tracker;
- Number of detected toolset usage errors during controlled experimentation, etc.

10.2. Validation Target Groups

STAMP methods and techniques, implemented as a toolset and public services, are intended to be adopted by different types of users when performing common test amplification tasks in different phases of the software development life-cycle. Therefore, one of the elements of the validation framework is the identification of these roles, among the potential adopters of the STAMP technologies, together with their needs and expectations, by surveying the widest possible number of communities involved in software development. These roles are identified by collecting the demographic information of the participants in the different validation activities. When possible they will be associated to the different test amplification tasks and supporting tools, determining, through the responses to questionnaires (see Appendix C) to what extend the tools satisfy their needs and expectations.

A number of relevant roles have been already identified by the consortium:

- Software developer: functional development of applications, service and their features;
- Test developer: development of tests cases, including unitary, integration, system tests, etc.;
- Software tester: execution and analysis of test cases; quality assurance;
- DevOps integrator: integration and deployment of software application and services. Optimal deployment configuration for selected non-functional indicators;
- Software maintainer: evolution of software application and services with new features. Maintenance of software releases.

As part of the collected demographic profile (see appendix C) for each participant in the evaluation activities, the following information will be requested:

- Technological expertise, focusing on:
    - Programming languages;

    - Testing technologies and methodologies;
    - Software management and DevOps technologies.
- Industrial background, focusing on:
    - Academic studies;
    - Years of experience;
    - Operational domain (e.g. target market) of the company;
    - Adopted roles within the software engineering lifecycle

Demographic profiles will be taken in consideration when preparing the validation groups (i.e. for instance adopting random sampling strategies) in order to reduce the bias possibly introduced in the validation results because of an unbalanced selection of these target groups.

10.3. Validation roles

Different roles are involved in the validation activities:

- **participants** are directly involved in the assessment activities and constitute the validation target groups (see section 10.1). They conduct the experimentation by performing proposed validation tasks on concrete target systems, adopting a treatment under assessment (e.g. the STAMP toolset);
- **facilitators** are involved in some concrete validation activity types (e.g. comparative studies, see section 10.7) organizing the validation target groups, coordinating the validation tasks and recording some metrics, but they are not part of the validation target groups (e.g. the validation subjects).

10.4. Supporting material UPDATED

Validation roles are supported before and during the experimentation with some materials required to perform the validation activities. Each activity type will require different kinds of materials, which will be make available (or linked) in the STAMP website[11]:

- **Documentation**
    - STAMP project presentations (slides): introductory presentation to STAMP project, concepts, methods and tools. Slides are available at the Slideshare: https://www.slideshare.net/stamp-project/;
    - Toolset technical documentation: getting-started, user-manual documents and tutorial for dummies accompanying the STAMP toolset, are available in GitHub: https://github.com/STAMP-project;

[11] The private documentation for consortium usage will be restricted from public access

- Walk-through user guides: step-by-step tutorial explaining the usage of STAMP toolset in concrete application scenarios;
- Demos (videos): available from STAMP website or YouTube channel, they may introduce the STAMP project in a nutshell and/or a concrete method, technique or tool;
- Webinars: an informative online presentation of the STAMP project and/or a specific aspect. It can be supported by slides and recorded as video.

● **Questionnaires/Forms**
  - Demographic profile: metadata characterizing a concrete participant or a group of them participating in a validation activity, particularly focusing on expertise and background related to the subject of validation. See demographic profile included in questionnaire in Appendix C;
  - Feedback-collection questionnaires: online questionnaires intended to collect qualitative feedback from participants of subjective validation concerns. See questionnaire in Appendix C;
  - Experiment/tasks descriptions: detailed description of the tasks (and their steps) to be conducted during some validation experiments (e.g. comparative studies). See section 10.7.3;
  - Metric collection forms: intended to collect measures of quantitative metrics associated to KPIs assessed in the validation activities. See D5.3;
  - Requirements elicitation forms (e.g. issue trackers): STAMP uses the GitHub infrastructure (https://github.com/STAMP-project) that includes an issue tracker to collect functional requests. See Table 9 for the form template to report new requirements and issues;
  - Issue/bug tracking tools: using the same tracker mentioned above.

● **Experimentation environment**
  - STAMP Toolset VM: preconfigured (standalone) STAMP toolset installed in a virtual environment, ready for experimentation. The STAMP VM, which contains a snapshot of all up-to-date STAMP tools is maintained by WP4 (see D4.2 [16]) and it is available for download at: https://nextcloud.ow2.org/index.php/s/JBfypUlhqYEmwNk[12];
  - Experimental source code projects: pre-existing source code projects intended to be used for test amplification during some validation activities, such as comparative studies (see DHell testing project available at GitHub: https://github.com/STAMP-project/dhell). Additionally, some industrial use case providers have public codebase repositories in Github (see section 7)

12 This link is provisional

- **Reporting forms**: agreed template for reporting validation results analysis and findings.

Some materials will be authored by the validation work-package (WP5), such as demos, reporting forms, etc. and others by the technical work-packages (WP1-WP4)[13], such as technical documentation, webinars, experimental environment, etc.

As a result of the validation activities, feedback will be duly reported to the STAMP toolset and services development teams[14]. There are different goals for providing this feedback:

- Provide early feedback to developers aiming at supporting them to improve the functionality, usability, efficiency, effectiveness and other aspects of the STAMP toolset and services and cover a wider application range on industrial real situations:
  - New functional (and nonfunctional) requirements and features;
  - Detected issues (and possible bugs) encountered during the utilization of the tools and services on industrial situations requiring test amplification;
  - Identified usability issues that makes difficult or hampers the utilization of the tools under the situations required by adopters;
- Report usage experiences adopting the toolset and services on concrete industrial scenarios of code testing;
- Assess the fulfillment of the hypothesis associated to the baseline KPIs defined in the GA (see D5.1 and D5.3) and provide quantitative estimations of the benefits of adopting STAMP technologies in industrial real applications;
- Assess the efficacy and efficiency of the STAMP methods and their toolset implementation (eventually compared to the equivalent results formerly obtained by adopting state of the practice techniques, when they were available for some industrial cases).

10.5. Reporting

Reports will be structured according to common templates depending on the type of reporting findings. These templates describe the required information and the format adopted to structure it:

- Requirements (new, amendments) will be reported according to the standard issue schema adopted by the STAMP project issue tracker tool in Github to report features (epics or user stories) [10]. See table 9 below that describes issue template for reporting new features;
- Results of comparative experiments with assessment of KPI hypothesis (see section 10.7.2.3) will be reported adopting proposed standards of scientific reporting guidelines for controlled experiments in software engineering [7].

[13] WP1-WP3 will contribute to the technical aspects of the documentation, while WP4 will contribute to the usability aspects, including users' manuals for available tool chains and tool clients, etc.

[14] WP1-WP3 development teams will take care of the feedback related to the core functionality and implementation of the STAMP tools, while WP4 will take care of the feedback related to the tool clients, services, interoperability aspects, usability, etc.

- Datasets resulting of above experiments (but also of any experimentation with the STAMP tools in the context of the pilot trials) will be provided to tool developers using tool-specific repositories created in the Github STAMP portal:
    - DSpot industrial use case experiments: https://github.com/STAMP-project/dspot-usecases-output
    - Descartes industrial use case experiments: https://github.com/STAMP-project/descartes-usecases-output
    - CAMP industrial use case experiments: N/A
    - Evocrash industrial use case experiments: https://github.com/STAMP-project/evocrash-usecases-output
- Issues/bugs will be reported according to an agreed issue schema adopted by the STAMP project (see Table 9) issue tracker tools to report issues or bugs, based on common tracker schema such as Jira, Bugzilla, MantisBT, etc[15]:
- Other discussions among use case industrial evaluators and tool developers are supported through discussion forums enabled in the STAMP GitHub portal:
    - https://github.com/STAMP-project/docs-forum

Next table 9 describes the issue template defined by STAMP to report new requirements and bug issues.

Table 9 Template for issue reporting: new features and bugs

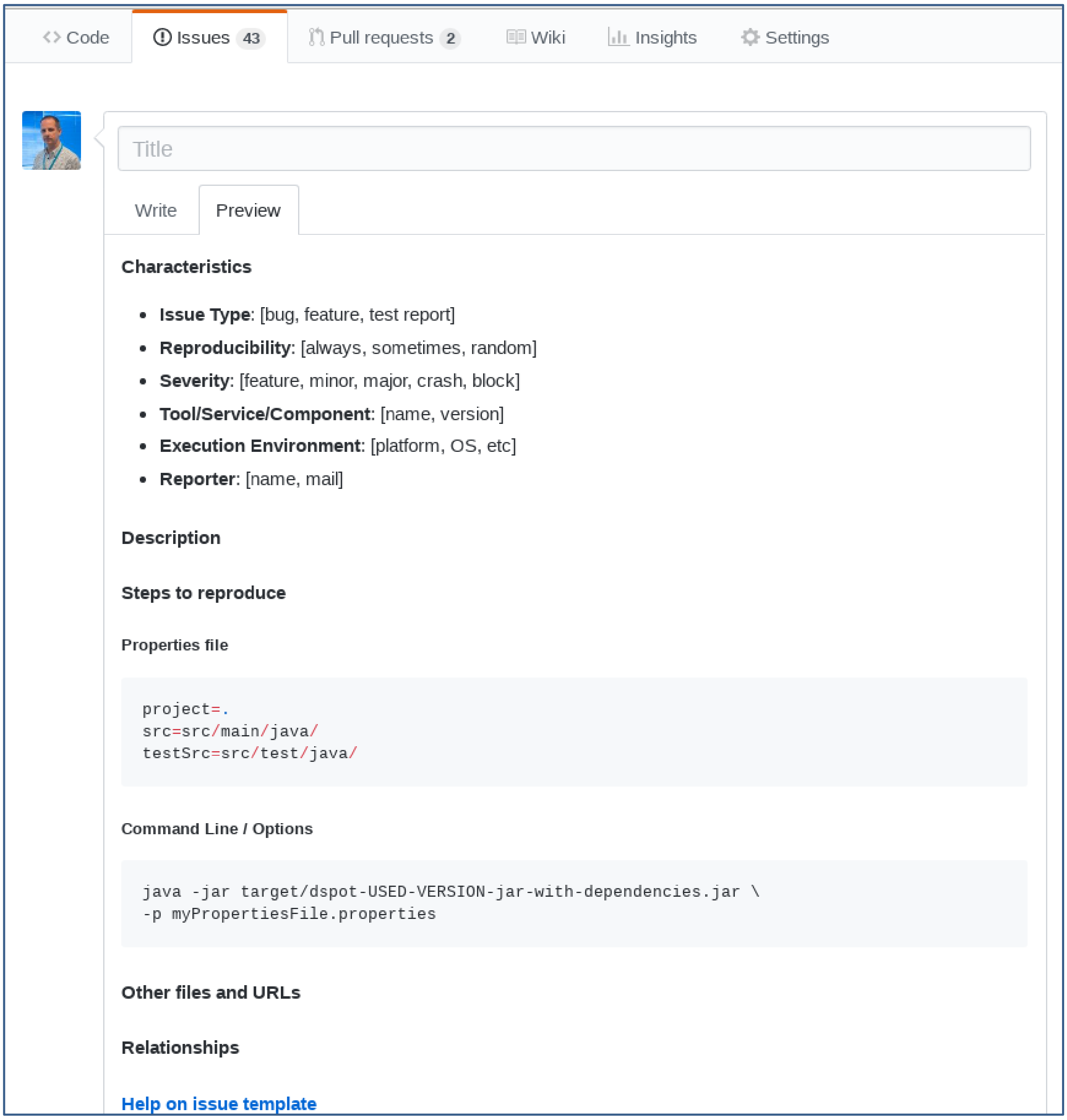
Issue <Number>	Issue Id, provided by the tracker
Key	A brief one-line summary of the issue
Type	Type of reported issue. It could be: ● **Bug**: issue perceived by the reporter as a potential bug, which has to be confirmed by assignee; ● **Feature**: issue describing a new requested functionality or a non-functional property to be supported.
Tags	Above issue type can be further refined, in the case of bugs, by adopting a number of predefined tags (taken from a proposed STAMP tag cloud), including[16] REGRESSION, CONFIGURATION, PERFORMANCE
Description	A detailed description of the issue.

[15] https://marker.io/blog/bug-report-template/
[16] It is not an exhaustive tag list at the time of releasing this document.

	For features, this section should provide a functional description of the required functionality. When describing features formally as user stories, the description can include this formal syntax: *As a <**role**>, I can <**activity**> so that <**business value**>* With this form, all the stakeholders involved in the requirement analysis can understand both the role of the user and the business benefit that the new functionality provides. For bugs, this section should describe as well: ● the observed execution behavior and obtained results; ● the expected execution behavior and results.
Reproducibility *<For bugs only>*	The occurrence of the issue, that is, the likelihood to be reproduced. Possible values: always, sometimes, random
Severity *<For bugs only>*	The degree of impact (as perceived by the reporter) the bug has on the operation of the tool/service being used in a concrete industrial situation. Possible values are: critical, major, minor, trivial
Tool/Service/Component **Version**	A reference name of the tool/service/component and the version the reported issue is related to (or affecting to)
Execution Environment **Steps to reproduce** **Snapshots** **Other files and URLs** *<For bugs only>*	A detailed description, step-by-step of the procedure followed by the reporter to reproduce the bug reported. This section should also include a description of the execution environment (platform, OS, etc.), including information about the version of the executed STAMP tools/services and their local dependencies (in case of standalone execution). Additional visual proofs, such as snapshots, providing additional visual information of the bug can be included, as well as input files required for reproducing the bug or URLs pointed to the sources of such inputs.
Relationships	A list of relationships to other issues. In case of features, these relationships can be used to structure them, grouping related features. Possible relationships: ● Child of / Parent of ● Related to ● Depends on

Reporter: **Name, email**	Reference information of the reporter, so the assignee can contact back for further issue refinement, if needed. A reference to the reporter could be automatically added by the tracker.

This template has been implemented into the STAMP Github issue tracker. Depending on the tool, some specializations of the template have been introduced, as shown in Figure 3 for DSpot issue tracker editor.



Characteristics

- **Issue Type**: [bug, feature, test report]
- **Reproducibility**: [always, sometimes, random]
- **Severity**: [feature, minor, major, crash, block]
- **Tool/Service/Component**: [name, version]
- **Execution Environment**: [platform, OS, etc]
- **Reporter**: [name, mail]

Description

Steps to reproduce

Properties file

```
project=.
src=src/main/java/
testSrc=src/test/java/
```

Command Line / Options

```
java -jar target/dspot-USED-VERSION-jar-with-dependencies.jar \
-p myPropertiesFile.properties
```

Other files and URLs

Relationships

Help on issue template

Figure 3 Issue reporting template for DSpot tool

- KPI metrics will be reported according to the information schema adopted in T5.1 and reported in D5.1 and D5.3
- Qualitative (and subjective) findings, based on user feedback collected in different validation activities will be reported by adopting the information schema depicted in the following Table 10. Evaluation report resulting of phase 1 did not embrace this reporting template, therefore, it will be encouraged for reports of phases II and II at M24 and M36.

Finding <Number>	Finding descriptive title
Rationale	Reasoned description of the finding
Aspect	List of categories (see section 10.1) the finding belongs to
Related to	List of STAMP toolset components, services or functional aspects (e.g. unit test amplification, etc.)
Priority	Importance given to the finding by stakeholders or evaluators
Reported on	Evaluation activity (phase, type) from where the finding was inferred.

Table 10 Reporting schema for validation findings

The project adopts an iterative and frequent reporting style, fostering an agile communication between UC evaluators (in WP5) and technical developers in WP1-WP4, who provide first[17] (WP4) and second[18] (WP1-WP3) level support, through:

- On continuous basis internal reporting: duly providing feedback, typically toolset usage experiences, bug/issues or requirements as a result of continuous evaluation (i.e. focus groups, pilot trials) by UC partners involved in the project, using indirect communication channels, mainly the Github issue tracker. Other direct communication channels (e.g. email, chats, etc) are discouraged because these discussion threads are neither public nor traceable;
- Discrete official reporting: at fixed milestones, through official WP5 deliverables (D5.5-D5.7). These reports will include detailed reporting of the validation activities conducted during the validation period (including both continuous and discrete activities), description of collected data, their analysis and results.

[17] First level support on tool client usage
[18] Second level support on tool internal operation

10.6. Validation Environments

The validation activities are conducted under different experimentation environments, characterized by their context and their available control mechanisms to drive the assessment activities, namely:

Lab controlled environment

In a Lab controlled environment there is a precise management of the different factors or conditions that can influence the execution of the assessment activities and their results. All these factors are controlled by the validation facilitators, who take care of preparing the assessment setup to concrete values. In particular, a controlled assessment environment includes, among others, some of these features:

- Concrete evaluation goals and associated verifiable hypotheses
- Specific setup of control and treatment methods and techniques, which constitutes the toolset target of the assessment
- Proper selection of experimenters' groups, suitably chosen to reduce assessment bias
- Similarly, a proper experimentation plan, which combines experimentation tasks and conducting groups also aiming to reduce bias on the assessment findings
- Specific measurement frameworks, collecting different metrics during the execution of the experimentation tasks
- Observed experimentation (by facilitators) who take care of record additional experimentation metrics
- Formal analysis of experimentation results
- Formal reporting of validation findings

Lab controlled validation activities can be conducted on continuous basis, but typically are also conducted in discrete events, such as workshops.

Online field environment

Unlike Lab controlled environment, online field environment shows little control over the experimentation context under which groups of evaluators are conducting assessment experiments, typically because:

- Assessment activities are conducted on the real-world context of the evaluators themselves, which are unknown and/or out of control for facilitators of the validation

- The evaluation activities are conducted by open communities of potential adopters, having neither control of the participants and its number nor of their demographic profile

The precise experimentation conditions are not held in the online field environment.

However, both approaches are complementary. Under lab controlled environment, evaluation hypothesis can be accurately verified scientifically, resulting in concluding findings. However, these concluding facts are only true under very concrete experimental conditions that rarely resemble practical situations. The online field environment enables the assessment of STAMP toolset under real and practical situations, not constrained to predefined conditions, and assessed by a larger group of practitioners. However, the accuracy of findings cannot be determined unless by the large statistical significance of the sampling.

10.7. Validation Activity Types

A number of different validation activities have been identified and designed with the purpose of validating the STAMP toolset and services addressing different validation goals. These types can be roughly classified according to previous taxonomy of validation environments into online field and lab controlled. See section 11 for a description of the common tasks mentioned in the validation method row.

10.7.1. Online/Field

10.7.1.1. User surveys

Table 11 User Survey description

Validation Activity Type	User Survey
Timeline Frame	**Phase**: Phase 1 and 2 **Event type**: Discrete/Continuous Event
Target User Group	**Participant type**: ● UC partners' members not involved in the project (2-4 participants per partner); ● Members of external organizations; ● No previous knowledge about STAMP project, aiming at reducing bias on results. **Number of participants**: 10-20 participants

Activity Goal	Collect participants' feedback about the STAMP approach, concept, toolset functionality, and the baseline of KPIs proposed in the GA. Collected feedback can be structured as toolset new requirements or refinements of existing ones.
Supporting materials	Participants are introduced on the STAMP project objectives and toolset features, using different materials: ● Slide-based presentation; ● Usage scenarios: they describe the adoption of STAMP concepts, methods and techniques in concrete and real situations, which exemplifies the practical usage of the STAMP toolset for potential adopters; ● "STAMP for dummies" manuals.
Validation method	1. Establish the goal of evaluation; 2. Identify tool and services target of evaluation; 3. Plan evaluation activities; Dedicated session (1-2 hours) attended by external participants and STAMP consortium representatives: ● External participants are introduced to the STAMP project concepts, methods, toolset and usage scenarios (1 hour); ● External participants are prompted to provide feedback about the project (1 hour or less), including: o Demographic information; o Feedback about the STAMP project from industrial adoption point of view, by answering an online (i.e. Google Forms) questionnaire that includes: ▪ General questions about the project; ▪ Usage scenario specific questions; ● In the online questionnaire, participants can also express new functional requirements and KPIs, amend or refine the functionality already described for the STAMP toolset and/or rank them; ● The questionnaire can capture subjective responses with 5-Likert gauges and text boxes for comments; ● Assess the evaluation results; ● Author the evaluation report;

10.7.1.2. *Open field trial* ^{UPDATED}

The Open Field Trial (OFT) is a significant extension to the validation exercise provided by the use case providers. While the use cases are provided by project partners and more specifically by IT professionals in close contacts with the STAMP technology core developers, the OFT is about submitting the STAMP tools to users way beyond the scope of the project partners. We know there is a gap between developers and users, even between research teams and early adopters in the market, and the challenge to bridge this gap is what OW2 calls the Delivery Challenge. It's the challenge to make brilliant ideas, algorithms and code available to real life users. STAMP must be able to reach out to potential users and partners if it aims to have a future on the market. The OFT will help us address this challenge and make STAMP tools market ready.

The OFT will be supported by the launch of a full fledged beta-testing campaign with download instructions, testing tutorial, online survey to collect beta testers feedbacks and a prize to motivate them. A beta testing campaign is actually a technico-marketing initiative: it is technical because its content is focused on the technology, but it is also marketing because its purpose is to reach-out to potential beta testers. Logically the beta testing campaign is thus structured in two principal corresponding parts:

- a technical part with a focus on productizing STAMP,
- and a marketing part with a focus on recruiting beta testers.

The technical part will be executed in the framework of WP4 in collaboration with WP5. Its role is to complement the STAMP toolset with product attributes such as proper packaging, documentation, tutorials and licensing so as to facilitate its discovery by potential beta testers. Table 1 collected the main STAMP products target of this OFT evaluation: a set of tools and user interfaces (CLI, Maven, Gradle, Eclipse, SaaS/PaaS, Docker). Stable releases of the STAMP tools will made available to the public using different channels (depending on the UI):

- CLI: Core STAMP tools (they include the CLI) are available for download from the corresponding repositories within the STAMP Github portal: https://github.com/STAMP-project. Accompanying documentation is available in the docs folder of each repository and on the readme.md
- Maven/Gradle: these plugins for STAMP tools will be available in the Maven Central repository: https://search.maven.org/
- Eclipse: these plugins for STAMP tools will be available in the Eclipse marketplace (https://marketplace.eclipse.org/) and in a update site available within the STAMP project site: https://www.stamp-project.eu. Accompanying documentation will be available within the Eclipse IDE documentation.
- SaaS/PaaS: STAMP tools will be exposed as SaaS/PaaS in a virtualized infrastructure provided by WP4.

- Docker: image containers for STAMP tools (CLI) will be available at the DockerHub: https://hub.docker.com/

Additional courseware produced by WP4 (tool documentation, video tutorials) will be available within the STAMP portal and Github repository. Video tutorials will be available as well in the STAMP channel in YouTube. STAMP tools presentations will be available the STAMP site in Slideshare: https://www.slideshare.net/stamp-project

Feedback will be recorded from participants in the beta program using different mechanisms, mainly through the GitHub issue trackers available for each tool, and through a number of questionnaires available within the STAMP site. Some STAMP tool interfaces (such as Eclipse) can implement an inline form-base feedback mechanism.

The marketing part will be executed in the framework of WP6. IIts role is to convince potential beta testers to spend time, energy and skills on STAMP. Here there are also two parts, one is to develop the marketing approach and the other one is to actually reach-out to potential beta testers.

The marketing approach of the beta testing campaign will comprise the following key elements:

- A call for beta testers, sent as a newsletter sent to the target communities
- Beta-testing webpage with links to key resources, code, tutorials, etc. to facilitate participation
- Beta-testing online survey: a short questionnaire aimed at collecting feedback
- Beta-testing prize: an incentive for beta testers who could win the prize through a lottery.

It will be possible to recruit beta testers by targeting open source and Java communities. Open source communities include: Apache, Eclipse, OpenStack, OW2 and Apache. We will reach out to them via well-attended open source events such as EclipseCon, FISL, FOSDEM, OpenStack Summit, OSCON, OW2con and Paris Open Source Summit. Java developers will be reached through specific Java-oriented events such as Devoxx, Voxxed Days and local Java User Groups (JUGs). The beta testing campaign will become the main driver of the dissemination strategy: the objective is to seed market adoption by recruiting 20 beta testers.

In summary, the beta testing campaign is a key milestone in positioning STAMP in its market. From the technical perspective feedback from beta testers will help improve the tools and from the marketing one the beta-testing will help demonstrate the tools are fit to be used by software developers. It is the first significant step to build a community of developers and testers in general who will start using the tools, report issues, share experiences and contribute to the open source project.

Table 12 Open field trial description

Validation Activity Type	Open Field Trial
Timeline Frame	**Phase**: Phase 3 **Event type**: Continuous Event
Target User Group	**Participant type**: Developers of open-source communities, such as OW2, FIWARE or Eclipse communities, and also the academic community. **Number of participants**: Unbound
Activity Goal	Evaluate the STAMP concepts, methods and toolset in daily practical testing situations dealt with by developers engaged in open-source development communities
Supporting materials	Participants engaged in this validation activity are supported by different materials publicly available online in the STAMP website: ● Stable STAMP toolset and services: • Downloadable standalone toolset, for supported platforms; • Online accessible STAMP services; ● Public documentation (with links accessible from STAMP website): • Webinars, demos, video channel in Youtube, presentations in Slideshare, public repository readmes and getting started documents; • Scientific articles describing the underpinnings of each tool; ● Means to collect feedback from STAMP adopters: • Online questionnaires (i.e. Google Forms) ■ Links to these questionnaires will be advertised in STAMP website, public documentation, mailings, newsletters, etc.; • A public mailing-list letting the participants to get support in their process of adopting the STAMP toolset and services; • Feedback forms embedded in the STAMP toolset and/or public services; • Public toolset tracker in Github; • Web analytics (i.e. Google Analytics).
Validation method	There are several features that characterize the validation method adopted by this validation activity type. The tasks for validation method described in section 11 are not applicable here, as the validation objectives are determined by the participants

	themselves, outside of the STAMP consortium control: ● Free experimentation: participants are free to use the STAMP toolset and services as they need for different personal purposes, according to their own circumstances and adopting their own experimentation method, not being possible nor desirable to impose a concrete one from the STAMP project; ● Toolset and services usages is guided by public documentation; ● Usage feedback is collected through online questionnaires and feedback forms; ● Other possible feedbacks are collected by the toolset trackers, as reported issues or bugs, as well as requests for additional functionalities; ● Metrics will be reported accounting for qualitative findings and qualitative number of reported issues.

10.7.2. Lab controlled

10.7.2.1. Focus groups

Table 13 Focus group description

Validation Activity Type	Focus Groups
Timeline Frame	**Phase**: Phase 1 and 2 **Event type**: Continuous or discrete (e.g. workshop based)
Target User Group	**Participant type**: ● UC partners' members involved and external to the project (2-3 participants per partner); ● Participants must adopt (in real situations) roles that match the STAMP target roles. **Number of participants**: 10 - 15
Activity Goal	The refinement, formalization and prioritization of STAMP requirements and KPI: ● Existing requirements and KPI defined in the GA; ● New requirements and KPI collected during the user survey.
Supporting materials	● Usage scenarios provided as supporting material for use survey

	● Requirements and KPI collected during the user survey
Validation method	1. Establish the goal of evaluation; 2. Identify tool and services target of evaluation 3. Plan evaluation activities: ● Collect current industrial testing practices without adopting STAMP methods and techniques; ● Introduce STAMP concepts, methods, techniques and scenarios of usage; ● Systematic requirement and KPI refinement, prioritization and association of metrics to KPIs: • Discussion will be recorded for further reference and analysis. 4. Refine the quality model for evaluation; 5. Assess the evaluation results; 6. Author the evaluation report.

10.7.2.2. Pilot trials

Table 14 Pilot trial description

Validation Activity Type	Pilot Trial
Timeline Frame	**Phase**: Phases 1, 2 and 3 **Event type**: Continuous event
Target User Group	**Participant type**: ● Members of the industrial UC partners; ● Active member participation in STAMP project, hands on the toolset. **Number of participants**: 10 - 15
Activity Goal	● Reporting usage experiences of toolset/service application on concrete industrial scenarios of code testing; ● Reporting issues and potential bugs experienced during toolset/service utilization; ● Assessment of concrete requirements and KPI, collecting and reporting associated metrics.
Supporting materials	● Available prototypes of STAMP toolset and services;

	● Toolset/service developers support (e.g. installation, configuration, usage); ● Toolset/service internal documentation: wiki; ● Framework for metric measurement and reporting; ● Framework for feedback reporting: tracker.
Validation method	1. Establish the goal of evaluation; 2. Identify tool and services target of evaluation; 3. Refine/Specify the quality model (optional); 4. Specify quality requirements; 5. Select the quality metrics; 6. Define criteria for assessment; 7. Plan evaluation activities: ● Define common test amplification tasks, based on aforementioned usage scenarios; ● Specialize or customize above tasks for each industrial UC; ● Define additional UC specific amplification tasks if needed; ● Continuous-base experimentation, adopting STAMP methods and techniques, using STAMP toolset and services, performing common and specific above tasks; 8. Assess the evaluation results; 9. Author the evaluation report: ● Report experimentation findings and issues, through identified communication channels back to WP1-WP3 developers: • Daily usage reporting; • Reporting on official deliverables.

10.7.2.3. In-Lab controlled experiments

Table 15 In-Lab Controlled experiment description

Validation Activity Type	In-Lab Controlled Experiment
Timeline Frame	**Phase**: Phases 2 and 3 **Event type**: Discrete (e.g. workshop) event
Target User Group	**Participant type**: ● Industrial UC partners;

	● No previous experience in STAMP project; ● Background on software development and testing, adopting roles targeted by STAMP toolset. **Number of participants**: 15 - 30
Activity Goal	Assess the efficacy and efficiency of the STAMP methods and their toolset implementation in test amplification tasks conducted in common industrial scenarios, confirming the hypothesis associated to the baseline of KPIs proposed in the GA. Eventually, comparative analysis of KPI results can be conducted for those industrial cases that previously adopted some of the state of practice techniques associated to the STAMP functional objectives.
Supporting materials	Participants engaged in this validation activity are supported by different materials publicly available online in the STAMP website: ● Stable STAMP toolset and services: • Pre-built, pre-configured, toolset experimentation environment, packaged for being unzipped or directly executed (e.g. a Virtual Machine, Docker container): ■ It includes all artifacts required to conduct the experimentation tasks, such as source code and development projects and other required tool dependencies; • Online accessible STAMP services; ● Training documentation (with links accessible from STAMP website or included within the package experimentation environment): • They describe the usage of STAMP toolset and services; ● Documentation describing the experimentation tasks to be conducted during the workshop; ● A metric measurement framework, also included within the package experimentation environment; ● Online questionnaire or forms for reporting experimentation results.
Validation method	1. Establish the goal of evaluation; 2. Identify tool and services target of evaluation; 3. Refine/Specify the quality model (optional); 4. Specify quality requirements; 5. Select the quality metrics; 6. Define criteria for assessment; 7. Plan evaluation activities: The following bullets depict the validation method adopting in comparative

experiments. New release of this deliverable will further formalize this method.

- Two groups of experimenters (G1, G2) will be formed, randomly chosen to avoid any bias on the selection of groups based on the participants' expertise and background;
- Both groups will be trained on the purpose of the experiments and the technologies under scrutiny:
    - Introduction to testing and test amplification;
    - Training on experimentation treatment:
        - Introduction to STAMP test amplification tools:
- Concrete experiments will be designed, combining the groups of experimenters with a concrete combination of treatment and control tools for conducting the experimentation tasks:
    - In treatment tasks, STAMP toolset will be used;
    - In control tasks, current state of practice techniques for solving the proposed task will be used[19];
- A set of common test amplification tasks will be proposed:
    - These tasks will be customized to different UCs showcased during the experimentation;
    - Proposed tasks are simple enough (in terms of complexity) as to be successfully completed by experimenters within a reasonable timeframe, and similarly affordable by adopting both the treatment and the control toolset;
- Following the experiment design, both groups (G1, G2) will perform these tasks adopting the treatment and the control toolset;
- During experimentation, experiment facilitators (e.g. members of STAMP consortium) will record the experiments, collecting different data, manually or assisted by a measurement framework:
    - Number of complete tasks: effectiveness;
    - Issue preventing the completion of tasks;
    - Issues faced by experimenters during the task execution;
    - Task completion time: productivity, efficiency;
    - For selected KPIs, control and treatment metrics;
- After experimentation, experiment facilitators will request participants to provide post-experimentation feedback, by filling in a questionnaire;

8. Assess the evaluation results:

[19] It only applies to those validation tasks conducted in the context of an industrial case where an alternative state of practice was previously adopted. If there are not such practices, the treatment tasks will not be conducted, and no comparative analysis of hypothesis will be performed.

	● Analysis of collected data; ● Assessing experimental hypothesis; 9. Author the evaluation report: ● Reporting in deliverables: • KPI metrics measurements; • Participant feedback; • Hypothesis assessment; • Key findings; • Discussion about results.

10.7.3. Comparative experiments NEW

This section specifies the comparative experiments that are proposed to be conducted, for every STAMP tool, in the Lab-controlled validation activities described in the previous sections.

Tool	**DSpot**
Objective	assess the utility of DSpot to **increase the number of system-under-test (SUT) behaviours reproduced and tested by the existing test suites**
Description	Compare the utility of **DSpot** with the existing **industrial state of practice** to increase the number of SUT behaviours reproduced and tested by the existing test suites. Test coverage of new SUT behaviours could request the amplification of test cases in terms of adding new test cases and/or assertions. According to D6.3 but also to the opinions expressed by STAMP tool developers and use case providers, there are not COTS[20] competitive tools offering features similar to those offered by the STAMP DSpot tool. Therefore, comparative experiments will be conducted comparing DSpot with the current industrial state of practice methodology adopted by the industrial partners.
Competitive Tool/Method	manual addition of new test cases and/or assertions by developers
KPIs	K01

20 Comercial out of the shell

Experiments	**Baseline**	use case focused codebase with an targeted test suite	
	Participants	**Group A**	manual extension of the targeted test suite, either adding new assertions to existing test cases or adding new test cases
		Group B	using **DSpot** to amplify the existing test suite
	Method	*Input*: (both groups) ● same code base and test suites ● same selected target code base package for test amplification *Experimentation time*: same time for both groups. *Objective*: amplify the test suite for given input target code base *Group A*: add manually new assertions to existing test cases or adding new test cases *Group B*: use DSpot to amplify the existing test suite *Data comparison*: compare the amplified test suite for group A and B	
	Measurements	● **K01** (code coverage) using Clover (or similar). See D5.3 for ruther details. ● Number of new generated test cases and/or assertions	
	Hypothesis	DSpot contributes to increase the code coverage	

Table 16 Comparative experiment for DSpot

Table 17 Comparative experiment for Descartes

Tool	**Descartes**
Objective	Objective: assess the utility of Descartes to improve the quality of existing test suites to: 1. **increase number of system-under-test (SUT) behaviours reproduced and tested by the existing test suites**. (see above description for DSpot) **2. improve the test suite quality by increasing the mutation score**

Description	Compare the utility of **Descartes** with the existing industrial state of practice, based on PIT, to improve the quality of a targeted SUT test suite, w.r.t. the ability of this test suite to: ● increase the number of SUT behaviours reproduced and tested by the existing test suites (see above description for DSpot). Using Descartes report, users are encouraged to increase the number of SUT behaviours detected by the target test suite. ● improve the mutation score of the target test suite. Using Descartes report, users are encouraged to reduce the number of pseudo/partially tested codebase methods and to increase the mutation score.		
Competitive Tool/Method	PIT[21]		
KPIs	● K01, for objective 1 ● K10, for objective 2		
Experiments	**Baseline**	use case focused codebase with a targeted test suite	
	Participants	**Group A**	participants use PIT to analyse the target SUT codebase and test suite. Using PIT report, participants are requested to improve the target test suite, either adding new method invocations and/or assertions in existing test cases or generating new test cases, aiming at increasing the code coverage and mutation score.
		Group B	participants use Descartes to analyse the target SUT codebase and test suite. Using Descartes report, participants are requested to improve the target test suite, either adding new method invocations and/or assertions in existing test cases or generating new test cases, aiming at increasing the code coverage and mutation score. Descartes report also includes the mutants that haven't been covered. Since Descartes mutates methods, this

21 http://pitest.org/

			report could be as a proxy for a method-level coverage. Then, by addressing in the test cases the uncovered mutants, the coverage can be improved.
		Group C	participants get access only to the coverage report, instead of to the mutation report. This group constitutes a baseline (or control group), playing the regular scenario where testers only have the coverage information. Using it, participants improve existing test cases to increase the code coverage.
	Method	*Input* (both groups): ● same code base and test suites for both groups. ● same target code base (e.g. package, classes) for applying test improvements *Experimentation time*: same time for both groups. *Objective*: improve test cases to kill live mutants, increase the mutation score and code coverage. *Group A*: ● compute code coverage (see K01) ● run PIT and get report (mutation score, code coverage, live mutants): baseline report ● improve test cases aiming at killing mutants within the target code base ● run PIT after test improvement, get final report. *Group B*: same as for group A, but using Descartes instead of PIT Grout C: same as for group A, but with access only to the coverage report *Data comparison*: compare delta between final and baseline report.	
	Measurements	● **K01** (code coverage) using Clover (or similar). See D5.3 for further details. ● **K10** (mutation score) using PIT. ● Number of test issues reported that are fixed/number of modifications introduced in the test suite. This metric gives an estimation of the easiness to understand Descartes mutation report compared to the similar one provided by PIT.	

	Hypothesis	● Descartes report is easier to understand by testers and therefore more useful for fixing testing issues ● The number of mutation issues fixed by testers using Descartes report is higher than the number of issues fixed using PIT reports. ● Descartes contributes to improve the code coverage ● Descartes contributes to improve the mutation score

Table 18 Comparative experiment for CAMP

Tool	**CAMP**
Objective	assess the utility of CAMP to evaluate the **behaviour of the SUT** under **different environment deployment configurations**
Description	Compare the utility of **CAMP** with the existing **industrial state of practice** to automate the assessment of SUT behaviours under multiple environment deployment configurations. According to D6.3 but also to the opinions expressed by STAMP tool developers and use case providers, there are not COTS[22] competitive tools offering similar end-to-end configuration amplification, SUT instantiation and test execution features similar to those offered by the STAMP DSpot tool. There are competitive industrial tools which can be adopted in different tasks within this complete process, for SUT instantiation, including Docker Compose[23], Docker Swarm[24], Kubernates[25] and others. Therefore, comparative experiments will be conducted comparing CAMP with these methods and tools within the current industrial state of practice methodology adopted by the industrial partners.
Competitive Tool/Method	● Manual generation of deployment/environment configurations ● Semi-automated instantiation of the SUT under generated configurations, using Docker Compose, Swarm, Kubernates or similar tools. ● Semi-automated execution of test sutes targeting the instantiated SUT ● Manual collection, aggregation and consolidation of test results

22 Commercial off the shelf

23 https://docs.docker.com/compose/

24 https://docs.docker.com/engine/swarm/

25 https://kubernetes.io/

KPIs	● K03 ● K04 ● K05		
Experiments	**Baseline**	use case SUT with a given input deployment/environment configuration and with an targeted test suite	
	Participants	**Group A**	use case participants (test developers, system administrators) adopt their current state of practice method, which is tool semi-automatically assisted, to conduct the test suite against multiple deployed instances of the SUT, which are configured with deployment/environment configuration they generate. Testing results are collected and aggregated manually. Tasks execution times are recorded. Proposed KPIs are also collected.
		Group B	use case participants (test developers, system administrators) conduct the same experiment than group A, the end-to-end process is fully automated by CAMP tool. Same KPI measurements are collected.
	Method	*Input*: (both groups), same: ● sistem under test: codebase and Docker configuration ● test suite *Experimentation time*: variable for each group. *Objective*: launch the test suite on an number of different SUT configurations *Group A*:manual generation of configurations, semi-manual SUT instantiation and automatic test execution *Group B*: use CAMP to complete the end to end process Data comparison: ● both groups to compute the end-to-end and tasks specific experimentation time. Compare results for both groups ● compare consolidated test results, organized by configuration., for both groups	
	Measurement	● **K03** (end-to-end test execution time). See D5.3 for further details.	

	s	● **K04** (number of inter-service execution traces). See D5.3 for further details. ● **K05** (number of detected system specific bugs). See D5.3 for further details.
	Hypothesis	● CAMP increases the number of tested configurations ● CAMP reduces the end-to-end test execution time over all the targeted configurations ● CAMP increases the number of inter-service execution traces ● CAMP increases the number of detected system specific bugs

Table 19 Comparative experiment for Evocrash

Tool	**Evocrash**
Objective	assess the utility of Evocrash to **generate test cases that replicate SUT runtime crashes**
Description	Compare the utility of **Evocrash** with the existing **industrial state of practice** to automate the generation of crash replicating test cases. According to D6.3, but also in agreement with the opinions expressed by Evocrash tool developers and use case providers, there are no COTS competitive tools offering a similar feature for generating test cases that replicate runtime crashes. Indeed, there are similar competitive research initiatives described in D6.3 which cannot be adopted in this industrial comparative evaluation due to the maturity distance between these research prototypes (if available), the technical support they may offer and what is required to conduct this comparative assessment. Moreover, it is not the intention of this activity to conduct a scientific evaluation of different proposed research lines. Therefore, comparative experiments will be conducted comparing Evocrash with a manual generation by test developers of test cases that replicate selected runtime crashes.
Competitive Tool/Method	● Manual selection of a runtime test crash and its target stack frame ● Manual generation of a test case that aims to reproduce the same stack trace than the target, starting from the selected frame ● Execution of the generated test ● Confirmation of stack trace reproduction

KPIs	● K08		
Experiments	**Baseline**	SUT codebase and runtime logs. Target runtime crash selected (including target frame)	
	Participants	**Group A**	use case participants (test developers, system administrators) adopt their current state of practice manual method to generate a test case that reproduces the target crash from selected frame.
		Group B	use case participants (test developers, system administrators) use Evocrash to generate test cases that reproduce the target crash from selected frame.
	Method	*Input*: (both groups): ● code base ● execution logs (including a number of target crash stack trace) *Experimentation time*; fixed same time for both groups *Objective*: generate replicating test cases of all selected target crash stack traces for selected frames. *Group A*: participants use their current state of practice manual method to generate a test case that reproduces the target crash from selected frame. *Group B*: participants use Evocrash for the same objective. *Data comparison*: compare the number of valid (they fulfil the experiment objective) test cases generated by both groups.	
	Measurements	**K08** (number of test cases reproducing runtime crashes). In particular, we measure whether or not, Group A and B are capable to reproduce the crash, from the target frame, using the generated test case. See D5.3 for further details.	
	Hypothesis	Evocrash increases the number of test cases that reproduces runtime crash stacktraces	

11. Appendix B: Evaluation Method UPDATED

The evaluation of the STAMP toolset and services show multiple facets: it involves different target stakeholders (see section 10.1), it pursues different evaluation objectives (see section 7.1) and it conducts several evaluation activities (see section 10.7). Therefore, adopting a single evaluation method, by customizing one or several existing standards, could not be possible. On the contrary, it may require the specification of tailored methods for each validation activity, according to the different pursued validation objectives. The approach adopted here consists in collecting a set of standard validation tasks that suit the different validation needs. For each validation activity, a number of selected tasks are aggregated to create a tailored validation method. These tasks constitute process fragments that, when tiled together, constitute a complete customized process. This approach based on process fragments has been adopted, for instance, in the customization of business processes [9]. In the following, the initial set of validation tasks adopted in the validation activities outlined in section 10.7 is described. These validation tasks are inspired by existing standards, such as the ISO/IEC 25040:2011[26], or proposed reporting guidelines for controlled experiments in software engineering [6].

11.1. Establish the goal of evaluation

The purpose of the evaluation should be established in clear terms by defining the evaluation goals. They should be aligned to the overall objectives of the STAMP project as they were stated in the GA. These goals are identified and agreed by the stakeholders involved in the evaluation activities, including industrial adopters (either within the consortium UC partners or within external members of the open communities of developers) and members of the STAMP toolset development teams. Concretely, evaluation goals are determined within each evaluation activity (see section 10.7) at the beginning of the activity. The stakeholders that agree on the evaluation goals are WP5 industrial evaluators, WP1-WP3 technical providers and WP4 industrialization providers.

The overall industrial evaluation goals for the entire validation process have been declared in section 7. The specific evaluation goals for the first phase were declared by each industrial use case provider in D5.5. Concrete evaluation goals for the second and third phases will be stated by each use case in D5.6 and D5.7, respectively.

[26] ISO/IEC 25040:2011, "Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Evaluation process", 2011: https://www.iso.org/standard/35765.html

11.2. Identify tools and services target of evaluation

The specific STAMP toolset parts or services that will be the target object of a concrete evaluation activity should be identified among the complete set of STAMP tools available at that moment, together with the specification of the concrete release that will be evaluated. The STAMP tools target of these industrial evaluation, namely: Descartes PITest, DSpot (WP1), Configuration Test Amplification Tool (CAMP) (WP2), Evocrash (WP3) and their corresponding additional clients, including CLI, Maven/Gradle, Eclipse, SaaS/Paas (WP4) are collected in table 1 in section 7.

Before starting each phase and each individual validation activity, the concrete tool(s) target must be specified. The selection of target tools are influenced not only by the validation objectives for that phase/task, but also by other factors such as the maturity of the tool(s), their suitability for the validation task, the suitability of available tool clients, their integration with the CI/CD pipelines, etc.

11.3. Refine/Specify the quality model for evaluation

Depending on the goals of the validation activities and the maturity level of the STAMP toolset and services (target object of the evaluation) and the scope and context of the validation activity in specific industrial use cases or scenarios, the quality model predefined in section 10.1 may require some refinement. It can be needed to refine existing quality attributes, remove some or add others, in order to accommodate the adopted quality model. For example, depending on the toolset maturity, quality attributes associated to functional aspects, usability, performance, etc., should be reformulated in the quality model as the toolset could not be mature enough as to use strict formulations of these attributes. Finally, the concrete quality model instance adopted in each evaluation activity should be specified.

In the first phase validation, reported in D5.5, the quality model was refined from the the initial version introduced in D5.2. As a result, a subset of the ISO 25010 product quality model was adopted and reported in this document in section 10.1.

11.4. Specify the quality requirements

Once the quality model has been selected, a concrete set of leaf quality attributes or measureable requirements are selected from the model, to be used during the evaluation activity as the focused quality target. Additionally other external quality requirements (e.g. not belonging to the quality model but collected from the stakeholders in the set of elicited requirements) can be included. Different factors influence this selection, including the evaluation goals, evaluated toolset parts, specific industrial use case stakeholders requirements, etc. For instance, each UC partner can select a concrete set of relevant KPIs as the quality model target to be evaluated in their UC during an evaluation activity.

As an example, a quality model instance used by Atos in the report of the first evaluation period (D5.5) focused on the concrete leaf characteristics of the ISO 25010 product quality model, but it also borrowed some characteristics from the quality in-use model.

11.5. Select the quality metrics

The restricted set of quality attributes, selected in the previous step, should be measurable, that is, they have associated metrics. In this step, concrete metrics for each attribute (or requirement) are selected, together with their bound measurement function. This mathematical function provides a quantitative value for the metrics, calculated from collected input data. Additionally, a measurement tool can be chosen from those associated to each metric to collect the input data.

The main set of quantitative metrics are those associated to the KPIs defined in D5.3. The qualitative metric adopted for measuring the degree of agreement (or satisfaction) of evaluators with the propose quality model is the 5-Likert gauge. Other quantitative metrics can be adopted, in particular some have been associated to the comparative experiments defined in section 10.7.3.

11.6. Define criteria for assessment

The assessment of the quality attributes or requirements, using the measurable metrics, requires the specification of a concrete criteria, for each metric, based on rating scales consisting of ranges defined by thresholds, which represents quality levels. For example, KPIs defined in the GA establish binary scales, defined by a single threshold, so the fulfillment of the KPI can be determined by the achievement of metric measurement over such a threshold. Nonetheless, multi-range scales can also be adopted, such as the Quper model for product management decision [11].

11.7. Plan evaluation activities

This plan specifies the timeline of sub-activities (which also defines) to carry out during the validation activity. For instance, in some evaluation activity types, the quality is assessed by observing representative users carrying out representative task in a realistic context of use. For these kind of activities, such as comparative experiments (see section 10.7.3), a plan of evaluation sub-activities includes (see section 10.7):

- Design experimentation task and identify context of use;
- Select evaluation participants;
- Specify data collection methods;
- Define evaluation timeline and milestones

11.8. Assess the evaluation results

Collected data is processed (e.g. using statistical mathematical methods), analyzed and compared against the criteria defined for quality assessment of each quality attribute (and metric) included as target quality model. Results are interpreted by the stakeholders according to the industrial scope and context of evaluation, constituting an overall assessment.

11.9. Author the evaluation report

The evaluation results are formally reported and notified to the interested parties, through established communication means (i.e. published documentation) and channels (i.e. WIKIs, forums, trackers, etc.), with the purpose of contributing to the fulfillment of the ultimate objectives of the evaluation. For instance, feedback such as recommendations, lessons learnt, findings, etc. on toolset usage can be communicated back to the STAMP development teams aiming to help them to improve, in next releases, those quality aspects analyzed during the evaluation activities.

12. Appendix C: User Survey Questionnaire ^{NEW}

This questionnaire has been designed to collect information about users' previous experience with testing methods, techniques and tools, and in particular with their experience in test amplification techniques.

This questionnaire is available at Google Forms:
https://docs.google.com/forms/d/e/1FAIpQLSfeayzbVgJTsPLOX5WL0wp8mbH-FuQDsJXpZDRsEk8GS4RXiw/viewform?usp=sf_link

STAMP User Survey Questionnaire

This questionnaire aims at collecting the feedback about developers and testers practices in the area of DevOps and test automation.

This feedback aims at adjusting the research and development activities of the STAMP project:

https://www.stamp-project.eu

To learn more about the project and automatic testing concepts, please visit this page:

https://www.stamp-project.eu/view/main/concepts

Filling this questionnaire should not take you more than 5 minutes. Thank you so much for you collaboration.

Please, within next section, provide us you email contact in case you are interested on receiving the

results of this survey. Alternatively, visit STAMP project portal, where survey results will be published.

*Required

Demographics

Please, provide some information about you, your background and expertise on software testing. No

personal information will be reported. Please, report your contact information is you want to collaborate with us for further elaboration on your responses to this questionnaire

1. **First name, last name**

____

2. **email**

____

3. **How many years of experience on software test-driven development and/or delivery do you**

have? *

Mark only one oval.

◯ Less than 5

◯ Between 5 and 10

◯ More than 10

4. **Which is your role in your current organization?** *

Tick all that apply.

☐ Test Engineer: develop test cases

☐ QA Tester: executes the test cases

☐ Test Manager: manages the QA processes within your organization

☐ Test/QA Researcher

☐ System Integrator/Delivery Manager

☐ System Maintainer

☐ Other: ____

5. **Which of the following testing methodologies are adopted in your organization?** *

Tick all that apply.

☐ Testing Driven Development (TDD)

☐ Incremental Test Last Development (ITLD)

☐ Testing integrated with CI/CD

☐ Testing is not adopted in your development life-cycle

☐ Other: ____

General Questions

General questions about STAMP objectives and approach

6. **Is QA included in your software development and delivery procedures?**

Mark only one oval.

◯ Yes, QA is a key aspect that rules the entire development and delivery cycle and mandatory according to the company policies

◯ Yes, QA is included in the development and delivery procedures, but it is not imposed by the delivery policies

◯ Maybe, QA is occasionally used during the development and delivery procedures

◯ No, we don't care about QA

◯ Other: ____

7. **Is your QA process concerned with detecting (and capable to do so) software bugs of any of**

the following kinds?

Tick all that apply.

☐ Regression bugs

☐ Configuration bugs

☐ Operation bugs

☐ Other: ____

8. **Is your QA process facing any of the following situations?**

Tick all that apply.

☐ Costly and time consuming test case authoring

☐ Low test coverage

☐ Time consuming test case execution

☐ Lack of support for test execution in multiple configurations

☐ Lack of support for re-injecting online test cases in production environments

Other: ____

Unit Test Amplification related questions

Specific questions about STAMP Unit Test Amplification

9. **What type of test cases do you include in your CI process?**

Tick all that apply.

☐ Unit

☐ System

☐ Integration

☐ UI

☐ Other: ____

10. **What are the main issues you detect in continuous testing?**

Tick all that apply.

☐ Flaky tests

☐ Test suite execution

☐ Test code coverage

☐ Other:

11. **Are you using any test coverage software to measure the efficiency of your test cases?**

Tick all that apply.

☐ Clover

☐ Jacoco

☐ Other:

12. **If you are adopting mutation techniques for amplifying unit tests and improving the test efficiency, what are those techniques and what are the purpose you used them for?**

____

____

____

____

____

Configuration Amplification related questions

Specific questions about STAMP Configuration Amplification

13. **Do you adopt DevOps Continuous Integration (CI) and Continuous Delivery (CD) tools for your**

software development and delivery lifecycle?

Tick all that apply.

- ☐ Maven
- ☐ Gradle
- ☐ Jenkins
- ☐ TravisCI
- ☐ Docker
- ☐ Vagrant
- ☐ Other: ____

14. **Explain how do you test your software under different deployment configurations (e.g. development, pre-production, production, etc)?**

____

____

____

____

____

15. **Before delivering your software, do you need to test it under different execution configurations that manage?**

Tick all that apply.

☐ Different software dependencies and versions (e.g. RDBMS: Oracle, MySQL, etc, Application Containers, etc)

☐ Different parameterizable configurations that accounts for NFRs (e.g. performance, high availability, high concurrency, etc)

☐ Different resources (e.g. CPU cores, maximal memory, etc.)

☐ Different architectures at runtime (e.g., scaling out some components, reorganizing the topology)

☐ Other: ____

16. **How many configurations would like to test under different circumstances?**

Mark only one oval per row.

	Only one representative configuration	A few (2-5) fixed but well-defined representative configurations	A few (1-5) parameterized configurations for specific purpose	A number of (5+) configurations to cover the configuration space
After each build	◯	◯	◯	◯

Before major release	◯	◯	◯	◯
After software updates for supporting new configurations	◯	◯	◯	◯

17. **If you do test, or could test, your software under different configurations, what would be the**

motivation or benefit?

Tick all that apply.

☐ Make sure the software works for all the customers

☐ Reveal the bugs that may expose more easily under particular configurations

☐ Evaluate the performance of different configurations, and find the optimized ones

☐ Reproduce and debug the failures reported by users under specific configurations

☐ Other: ____

Runtime Amplification related questions

Runtime test amplification is focused on using production logs to automatically create new test cases.

For instance, generating a test case that reproduces a crash that happens in production.

18. **Do your applications produce logs?**

Mark only one oval.

- ◯ Yes, but we do not use them
- ◯ Yes, and we use them
- ◯ No
- ◯ I do not know

19. **Do the developers have access to production logs?**

Mark only one oval.

- ◯ Yes, all of them
- ◯ Yes, some/part of them (for failure diagnosis for instance)
- ◯ No
- ◯ I do not know

20. **Those logs are used for (more than one answer possible)**:

Tick all that apply.

- ☐ application monitoring purpose (liveliness/throughput)
- ☐ application failure detection (failures are detected and reported)
- ☐ application failure diagnosis (failure reports are used by developers for debug)
- ☐ anomaly detection (abnormal behaviour of the application/user are detected and reported)
- ☐ Other: ________________________________

21. **Is there a common logging policy defined for the application development in your department/organization?**

Mark only one oval.

- ◯ Yes, logging policy is described (in a document) and is enforced during application quality assessment
- ◯ Yes, logging policy is described (in a document) but is never checked
- ◯ Yes, logging policy is a common knowledge shared amongst developers (between senior and junior developers for instance)
- ◯ No, there is no common logging policy defined for our developments

22. Do you have log anonymization concerns (for instance, removing user personal or sensible

information from logs before using them)?

Mark only one oval.

- ◯ Yes
- ◯ No.
- ◯ I do not know

23. Do you have any other logging concerns that you would like to see addressed (for instance,

scaling, level of detail, inconsistencies in the log entries, etc.)?

____
____
____
____
____

Thank you!