SOFIE - Secure Open Federation for Internet Everywhere

779984

DELIVERABLE D4.1

Validation and Evaluation Plan

Project title	SOFIE – Secure Open Federation for Internet Everywhere
Contract Number	H2020-IOT-2017-3 – 779984
Duration	1.1.2018 – 31.12.2020
Date of preparation	31.10.2018
Author(s)	Vasilios A. Siris (AUEB-RC), Mikael Jaatinen (LMF), Yiannis Oikonomidis (SYN), Santeri Paavolainen (AALTO), Tommi Elo (AALTO), Nikos Fotiou (AUEB-RC), Spyros Voulgaris (AUEB-RC), Dmitrij Lagutin (AALTO), Ektor Arzoglou (AALTO), Giuseppe Raveduto (ENG), Priit Anton (GT), George C. Polyzos (AUEB-RC), George Xylomenos (AUEB-RC)
Responsible person	Vasilios A. Siris (AUEB-RC), vsiris@aueb.gr
Target Dissemination Level	Public
Status of the Document	Completed
Version	1.0
Project web-site	https://www.sofie-iot.eu/

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 779984.

Table of Contents

1. Introduction

The main objective of the evaluation work package (WP4) is the qualitative and quantitative evaluation of the SOFIE architecture and framework. Key Performance Indicators (KPIs) and assessment criteria defined in WP2 (Federation Architecture & Framework) will be leveraged. Moreover, the results of the validation and evaluation work conducted in WP4 and recommendations based on these results will be fed into the architecture and framework design in WP2 and the business integration in WP3 (Business Platforms Integration). This feedback will be obtained in the two planned cycles of validation and evaluation activities, which correspond to an initial (first) and second report of findings and recommendations for the SOFIE architecture and framework, guiding the following SOFIE design and development activities. The cycles of validation and evaluation will map against the main releases of the SOFIE platform.

The work in WP4 involves two distinct directions: *validation* and *evaluation*.

- Validation will focus on checking whether the SOFIE platform meets the requirements of the stakeholders that use the platform. Validation is distinct from integration and verification, which includes testing the behavior and interoperability of components, undertaken in WP3.
- Evaluation will focus on assessing and measuring the performance of the SOFIE architecture and platform, based on the KPIs and assessment criteria defined in WP2 and other more general systems metrics and goals. The evaluation will include both a *quantitative* and a *qualitative* component.
    - The quantitative evaluation will focus on aspects such as *response time*, *throughput*, *resource utilization*, *scalability*, and *availability*. Some of these metrics can be assessed through measurements from testbed experiments with implementations of the SOFIE framework, while others, such as scalability and availability, will be assessed using analytical models and tools.
    - The qualitative evaluation will focus on architectural aspects, such as the *security* and *privacy* features of the overall architecture and of a subset of the platform components, the ability to *integrate* different IoT platforms and provide *services across domains*, and the ability to support innovative applications.

Although validation and evaluation have different objectives, the same platforms will be used for both validation and evaluation.

This deliverable is structured as follows. In Section 2 we discuss the methodology for the validation and evaluation work, along with the corresponding time-plan. In Section 3 we describe the evaluation platforms, some of which have already been implemented at the time this deliverable was prepared. In Section 4 we discuss the tools that will be used in the validation and evaluation activities, which include both software and analytical tools. In Section 5 we conclude by summarizing the methodology and plan, emphasizing the goals and key directions for the validation and evaluation work.

2. Validation and Evaluation Plan

This section gives a high-level description of our plan and methodology for validating and evaluating the SOFIE project outcomes.

2.1 Validation methodology and plan

The purpose of software (SW) validation is to assure that software does what the customer or end user wants. It is perfectly possible for a piece of software to pass a series of verification test cases, and still not fulfill customer expectations. This usually happens because requirements were not properly communicated or defined to the software developers. On the other hand, it is quite unlikely that software that fails verification testing will meet the customer's requirements. Validation is a wider concept than verification against defined test cases and requirements. In addition to pure verification, validation will involve trial usage of the system with the customers and representative users, to ensure that any missing or misunderstood requirements are revealed and can be incorporated (typically) through a change management process.

Testing (i.e., verification) can be divided as follows:

- **Unit testing:** Verification of the correct behavior of single SW components or units. This phase is either within the WP2 scope or beyond SOFIE’s scope (e.g., for third-party or free and open-source SW).
- **Integration testing:** Verification of the start-up procedures and SW component interfaces. Typically, a major part, or even the full range of these tests are automated in a continuous integration process. Integration testing is within the scope of WP3.
- **Functional testing:** Verification of the correctness of the implementation of functional requirements. This activity is within the scope of WP4 validation work.
- **Interface testing:** Verification of the correctness of interfaces and interoperability between subsystems and systems against designated API specifications. This activity is within the scope of WP4 validation work.
- **System testing:** Verification of system-level functional and non-functional requirements (e.g., realized as KPIs in SOFIE). This activity will take place in WP4 during validation and evaluation, with the majority of the activities will be part of evaluation.
- **Acceptance testing:** Formal process where the complete system is verified against customer/end user defined acceptance requirements. In SOFIE, this phase may not be required, but if it is, it will be shared between WP4 validation and evaluation.

Verification in a SW development project is primarily a risk mitigation function. The purpose of verification is to advise the project on the quality risks in the delivery, seeking to mitigate the greatest risks as early as possible and to provide information on the residual risks such that a rational judgment between cost, time, and quality of the delivery can be taken.

The trial usage part of validation will take place in SOFIE as part of the four pilots.

The SOFIE integration plan, D3.1 [SOFIE D3.1], describes the integration-related parts of the validation plan and the overall integration process. As a continuous integration methodology will be applied in WP3, not all versions of the SOFIE business platforms will be delivered to WP4 and WP5 (Pilots). The following main principles will apply:

- The SOFIE testbed release (version 0) will undergo a partial scope of validation activities, focusing on functional testing, interface testing and SW quality controls.
- The SOFIE main SW releases version 1 and version 2 will undergo the full validation scope.
- For intermediate SOFIE SW releases, WP4 will only validate SOFIE business platform versions with meaningful new content, such as new features, or important fault corrections.

- Within WP4, validation and evaluation will be parallel activities, i.e., there will be no acceptance testing required by evaluation.
- WP5 will only deploy SW that has undergone at least functional testing, interface testing, and partial system testing.
- Acceptance testing and definition of acceptance test cases will be done jointly across WP4 and WP5 pilots.

The time plan for the validation work will proceed with the following milestones:

- CI/CD (Continuous Integration/Continuous Delivery) environment design and setup as per deliverable D3.1 [SOFIE D3.1] (October 2018 – June 2019), aligned to meet the needs of deliverable “D4.3 - First Architecture and System Evaluation Report” due in June 2019.
- First architecture and system validation (June 2019): this will define the test case baseline (passed / not passed) and the feedback from pilots.

  Second architecture and system validation (April 2020): this will define the test case baseline (passed / not passed) and the feedback from pilots.

  Final architecture, system, and pilot validation (December 2020): this will define the final test case baseline, feedback from pilots and the final test report.

2.2 Evaluation methodology and plan

The goal of evaluation is to assess the value of a software concept, system, and subsystems. Evaluation is a crucial stage in the software development process, both for commercial software and for software developed as part of an academic exercise. For commercial software, the customer will want to evaluate the software to determine how well it performs its advertised functions. Evaluation is split into *qualitative* and *quantitative* evaluation.

The qualitative evaluation will focus on the following aspects: security and privacy of the overall architecture and of individual platform components; ability to integrate different IoT platforms and federate services across domains; support for innovative applications built on top of SOFIE’s open federation platform and interworking with applications implemented directly on IoT platforms outside the federation; and deployment complexity.

The quantitative evaluation will include performance evaluation, e.g., system throughput and delay for various workloads and conditions, but also scalability and robustness. Also, individual components of the SOFIE framework can be evaluated from such a perspective and their synthesis can be investigated (and whether properties and performance of the system can be deduced from those of the components). This is considered particularly important in the SOFIE case since the emphasis is on open federation of potentially not fully understood (with respect to their internal structure, behavior and performance) systems.

Two cycles of evaluation activities are planned, corresponding to the ‘cycle 1’ and ‘cycle 2’ implementations of the platform. The majority of the work in this task will focus on the ‘cycle 1’ prototype in order to provide feedback to WP2 and WP3 as early as possible. Following this, the ‘cycle 2’ prototype evaluation will ensure that the design and implementation refinements based on the first cycle feedback have improved the overall performance of the platform and its individual components, without introducing negative impacts or regressions.

The time plan for the evaluation work will proceed with the following milestones:

- Testbed environment design and setup, deliverable D4.2 (February 2019): this will include a detailed description of the testbed topology and the initial experience with its setup depending on the evaluation scenarios and targets, including a justification of the evaluation targets and Key Performance Indicators (KPIs) that are to be investigated.
- First architecture and system evaluation, deliverable D4.3 (June 2019): this will include the initial evaluation scenarios and results of the SOFIE architecture and platform, from both a qualitative and a quantitative perspective.

- Second architecture and system evaluation, deliverable D4.4 (April 2020): this will include the complete evaluation of the SOFIE architecture and platform.
- Final architecture, system, and pilot evaluation, deliverable D4.5 (December 2020): this will include the evaluation of the SOFIE platform, combining results from the architecture evaluation, testbed evaluation, and results from the pilot experiments.

2.3 Joint analysis of testbed and pilot results

The results from the pilots will be analyzed jointly with WP5 and reported in conjunction with the final set of system-level testbed evaluation results from WP5. This joint analysis and comparison will verify the gains of the SOFIE platform in real conditions and validate its advantages. Moreover, it will ensure consistency in reporting and make the final evaluation results available in a single comprehensive document. This will also be useful for the overall evaluation of the SOFIE architecture and its federation framework and business platform.

2.4 Evaluation from a business perspective

Blockchain and interledger technologies can have a significant impact on business models due to the following key features: (i) they can provide integrity and trust while reducing the need for trusted third parties, (ii) they can provide product provenance and immutable tracking, and (iii) they can eliminate the need for intermediaries and can enable flexible peer-to-peer trading in a decentralized environment. These features can collectively enable the development of innovative applications, while reducing transaction costs and increasing efficiency.

These technologies can also lead towards open and decentralized platforms, where anyone can provide services with minimum barriers and even synthesize services from other existing services, perhaps without permission from the providers of the original services. Illustrating these properties and evaluating them from multiple, different perspectives would help their eventual realization and monetization. A Game Theoretic modelling of open platforms will be attempted, e.g., in the spirit of A. Salazar [Sal15], in order to better elucidate the interactions and dynamics among players and the role of open (digital) platforms, in particular compared to the more traditional (closed, owned, or monopolistic) digital platforms.

Related KPIs that have been identified in the Annex of D2.2 [SOFIE D2.2] include the ability to conduct transactions across multiple ledgers, the ability to develop applications on top of multiple ledgers, and the ability of data publishers to dynamically control who has access to their data.

2.5 Evaluation using a system dynamics approach

System dynamics is a field of study that perceives the world as a complex system where “everything is connected to everything else.” This connection is depicted with feedback loops that can be either positive (i.e., reinforcing) or negative (i.e., balancing). The feedback structure of a system is first illustrated with causal loop diagrams (CLDs). Causal loop diagrams capture the basic hypothesis about the causes and effects of dynamics and elicits mental models of researchers. This is the initial step to communicate and assemble important input and feedback from experts or the community. Finalized causal loop diagrams are then modelled into stock and flow diagrams [Ste00].

More generally, *systems thinking* refers to the ability to see the world as a complex system, in which we understand that “you can’t just do one thing” without affecting other things. If people had such a holistic worldview, it is argued, they would then act in consonance with the long-term best interests of the system as a whole, identify the high leverage points in the system, and avoid policy resistance. System dynamics is a method to enhance learning in complex systems. It is, partly, a method for developing simulation models to help us learn about dynamic complexity, understand the sources of policy resistance, and design more effective policies [Ste00].

We are currently using the system dynamics methodology to create causal loop diagrams for the digital business platforms of iPhone and Android devices [AEM+18], with the intention of first understanding the smartphone industry in terms of value creation, market growth, and user's and app developer's incentives for participation. These causal loop diagrams will, in turn, be converted to stock and flow diagrams.

Once we have gained initial understanding of how digital business platforms work in general, through this initial study and perhaps other studies into existing systems, our plan is to adapt and use the results of our research on the SOFIE pilots. So far, there is scarce literature that applies system dynamics to digital business platforms [RCK17], and specifically to case studies concerned with the smartphone [DPR+17] or transportation [VB17] industry.

3. Validation and Evaluation Platforms

This section describes the platforms that will be used for validation and evaluation. These platforms include local testbeds, the interconnection of local testbeds, software platforms designed and implemented, and the pilots.

3.1 Local testbeds

3.1.1 LMF Ericsson validation environment

LMF Ericsson’s testbed for validation of the reference platform functionality, before delivery to evaluation and pilot deployment, logically consists of three different systems:

- Supporting components that are used to manage and monitor the validation environment (logging, monitoring, access control, backups, etc.).
- Development process support, e.g., (DevOps) Continuous Integration (CI) and Continuous Delivery (CD) systems.
- Staging and “production” environments, which consist of necessary pilot component deployments in both staging and “production”[1] and any other static components supporting them, including Distributed Ledger Technology (DLT) setups (i.e., Ethereum, private Ethereum, Hyperledger Fabric etc.).

3.1.2 AUEB testbed

The AUEB local testbed is currently focused on experimentation with IoT device integration, and user authentication and authorization. It consists of the following elements:

- A private Ethereum chain implemented using:
  - Two Virtual Machines (VMs) acting as miners and RPC servers.
  - One VM that implements auxiliary functions (bootstrap node, monitoring tool).
  - One Android smartphone running the Geth[2] Ethereum client.
- An instance of the WS02 identity server[3], which is based on open standards and open source principles and provides identity and access management functions.
- An instance of the OAuth2.0 server[4], which implements an open authorization framework that provides delegated authorization to protected resources.
- A VM running Mozilla’s Thing Gateway[5], which allows monitoring and controlling devices through a single secure web interface.
- A Hyperledger Fabric network.
- Hash-lock based interledger functionality between Ethereum and Hyperledger Fabric.
- Access to Guardtime's KSI Blockchain.

AUEB is additionally planning to provide:

- A Hyperledger Indy deployment.
- A Hyperledger Quilt deployment.

3.1.3 Aalto testbed

Aalto’s testbed is mainly used for research and it may also be used for teaching in the future. It currently consists of:

[1] “Production” in the sense of being in production for pilot testing and evaluation purposes, not in the general sense of being in “production” as generally available to entities outside the project members.

[2] https://geth.ethereum.org/downloads/

[3] https://wso2.com/identity-and-access-management/

[4] https://github.com/bshaffer/oauth2-server-php

[5] https://iot.mozilla.org/gateway/

- An Ethereum miner, which operates in the context of the Nanopool mining pool. Ether gained from mining can be used for smart contract experimentation on the real Ethereum network.
- An archival node on the main Ethereum network for research purposes. An archival node, in addition to storing the blockchain history, also stores additional information such as the blockchain state history information normally retained by full nodes only for a limited number of blocks. Such information is useful for analyzing the Ethereum network.
- Access to Guardtime's KSI Blockchain.

In the future the testbed will be extended with:

- An Ethereum node on a SOFIE private Ethereum network (connected to AUEB and other partners).
- A Hyperledger Fabric network for interledger research and testing. If needed, multiple small-scale networks will be run, and/or connected to other partners' networks.

3.2 Testbed interconnection

Below we describe the planned interconnections among the local SOFIE testbeds.

- Interconnection of the AUEB, Aalto and LMF local testbeds

This will involve the interconnection of the local Ethereum nodes that are part of SOFIE's private Ethereum network. This interconnection will include two options. The first will have miners running in nodes located in different local networks to be part of the same private Ethereum network. The second option will involve interledger operations between nodes located in different local networks, which belong to different private blockchains; these blockchains can be of the same type (e.g., Ethereum) or different types. The second option requires a module (gateway) outside the main nodes that handles the communication among the blockchains that are necessary for running the hash time-locked contracts on the two blockchains.

- Interconnection of the AUEB testbed with the food chain pilot

Synelixis has provided access to its SynField platform, where a device dedicated for experimentation provides various measurements including air temperature, solar radiation level, air humidity, device temperature, and device current. These measurements can also be accessed using a REST Web service. An application running at AUEB's premises periodically uses the REST API provided by SynField and pulls measurements. Then, it records the hash of the measurements in an Ethereum smart contract located on AUEB's private Ethereum network.

- Interconnection of local testbeds with Guardtime's KSI Blockchain

Aalto and AUEB testbed connections to Guardtime's KSI Blockchain will be realized through try-out servers for both Catena and the KSI Gateway. Guardtime will take care of user management and of providing Aalto and AUEB with access credentials. In fact, AUEB has already been given access and has connected its local testbed to the KSI blockchain, using KSI's try-out Catena API. The blockchain is hosted by KSI and it is accessible through a REST-based API.

3.2.1 Description of KSI Blockchain access

KSI signatures are server based, meaning that signing data requires online access to the KSI service. The verification of the signatures can be done both offline and online. There are two options for access to KSI:

- KSI Software Development Kit (SDK).
- Catena middleware.

The KSI SDK provides the lowest level of integration. It enables "full access" to the KSI functions (signing, extending, verifying) and lets the integrator fine-tune all possible options. As a consequence, it leaves many common challenges, such as signature storage and extension, to the integrator to solve.

Catena is middleware that is meant to address common integration challenges, such as asynchronous signing, signature persistence, and automatic extension. It provides the integrators with higher-level functionality, such as annotating signatures and linking signing events (data provenance), in order to reduce the effort for building a complete solution. Catena internally uses the aforementioned SDK to perform low-level KSI operations. The functionality of Catena is grouped and packaged into logical applications (Catena-KSI, Catena-DB, Catena-Prov) so that the integrator can choose which ones to deploy and use.

KSI SDK and Catena are not mutually exclusive: they can be used in combination if needed. This depends on the application type and the requirements for signing and verifying data.

3.2.2 Integration Resources

Depending on which integration option one wants to use, the following is needed to sign data, extend and verify signatures with KSI:

- access to KSI Gateway and KSI SDK to communicate with it; or
- access to Catena middleware and HTTP client library to communicate with it.

3.2.3 Try-out Servers

Catena

For the list of available API endpoints and reference documentation see https://tryout-catena.guardtime.net. It is a Swagger UI that also allows direct execution of requests. This is the quickest option to get started with signing.

KSI Gateway

The KSI Gateway has two endpoints, one for aggregation/signing and one for signature extension. For tryout these are:

- http://tryout.guardtime.net:8080/gt-signingservice
- http://tryout-extender.guardtime.net:8081/gt-extendingservice

The KSI Gateway uses HMAC-based authentication built in the KSI protocol. The KSI Gateway endpoints are expected to return "Bad URI" when just used in web browser.

In addition, the KSI publications file URL is needed for signature extension and verification with the KSI SDK. For Guardtime provided KSI services, this is https://verify.guardtime.com/ksi-publications.bin.

KSI SDK

The SDK is available for Java, .NET and C. The description and documents on how to access are provided to the try-out user account.

3.3 Pilots

All SOFIE pilots will leverage the Business Platforms (BPs) that will be developed based on SOFIE's framework and reference platform. Data collected by the various IoT platforms that lie beneath the BP level will be available via the BP, hence, the availability of this data could serve as a means of validation for the system.

The following sections present the specific platforms developed in the context of each individual pilot.

3.3.1 Food Chain

In this pilot, we currently have three IoT platforms involved in the different segments of the supply chain path. Those platforms are either already commercially exploited by the partners they belong to, or they are extended versions of the outcomes of other H2020 EU projects. More specifically, we have the following IoT platforms (their details can be found in SOFIE Deliverable D5.1 [SOFIE D5.1]):

- **SynField IoT platform**, consisting of hardware components (deployed in the pilot's field) and software components (in the form of a Cloud platform).
- **Aberon platform**, consisting of hardware components (deployed in the pilot's warehouse) and software components (in the form of a Cloud platform).
- **Transportation IoT platform**, consisting of hardware components (deployed on the pilot's transportation elements, e.g., vehicles and smart boxes) and software components (in the form of a Cloud server).

For the purposes of the pilot, the IoT platforms listed above will be enhanced with DLTs, such as Ethereum, Hyperledger Fabric, and KSI, based on their specific requirements. In addition, a Trace-History Marketplaces component will be developed, which will serve as the Business Platform for this specific pilot (as described in the SOFIE Deliverable D2.3 – Federation Framework, 1st version.

User feedback: In this pilot, the main user feedback consists of the user's (i.e., end customer's) reaction when she acquires information about a product. This information comes from the different DLTs via the pilot's Business Platform. Hence, the success or failure of this process could be regarded an end-to-end evaluation of the whole system.

3.3.2 Energy (Estonia)

In this pilot, we have three environments that will be utilized: the smart-meters layer, the Elering Estfeed platform layer, and the smart-meter owners and data exchange (distribution) layer, using the KSI Blockchain API for integrity and security purposes.

The smart-meters layer provides input to the distribution layer and is the core for any data exchange demonstrated by the energy pilot. The smart-meters data input that will be used can be split into two categories: the data controlled by the grid operators (hence the pilot will only receive their input as provided) and the data managed by Guardtime in order to demonstrate the complete data provenance chain.

The Elering Estfeed platform connects all smart-meter data in Estonia and offers platform users the interfaces through which various data sources can be used in the desired applications.

The use of the KSI Blockchain via the respective API will guarantee the integrity of smart-meter data and authentication and support the agreements between parties.

User feedback: The evaluation in the Estonian energy pilot from a user perspective will target the following two aspects (and owner groups). First, given that it is an existing, deployed smart meter platform (Elering in Estonia and Energinet in Denmark) the expectation of the owners for the platform is that access among their systems will have enhanced security because of the SOFIE federated platform. Second, a small smart meter owner group has an interest in the SOFIE platform supporting the smart meter data exchange, including the requirements for integrity, authentication, and security aspects.

3.3.3 Energy (Italy)

The Energy pilot in Italy will leverage smart meters, electric-vehicle (EV) support equipment (e.g., charging stations), and electric vehicles (six of them). It is worth mentioning that this infrastructure is already deployed on the pilot site.

For the purposes of this pilot a software component will be developed for load and prediction forecasting that will utilize historical data collected from the aforementioned infrastructure. Additionally, a Flexibility Marketplace will be developed, which will serve as the Business Platform of this pilot. This marketplace will also leverage a blockchain infrastructure.

User feedback: During this pilot, a marketplace, implementing the SOFIE framework, will offer automated services to the involved actors (distribution system operator (DSO), Fleet manager, EV user). Those services will also leverage the interoperability offered by SOFIE, leading to a more efficient and secure flexibility management process. The “smooth” operation of the marketplace could serve as an indicator of SOFIE’s added value to a system that would otherwise be manually managed. Thus, feedback will be sought from the operator of the marketplace (and the marketplace itself) and the involved actors.

3.3.4 Mobile Gaming

The mobile gaming pilot will make use of an existing smartphone game platform and infrastructure that consists of a game development engine, servers, and services (owned by Rovio).

For the purposes of this pilot, the use of Ethereum will be investigated and particularly its ability to support a very large number of users.

User feedback: In the context of this pilot, a game will be developed. Due to the nature of this pilot, the evaluation of SOFIE could be linked to the feedback of users playing the game. If the game receives positive feedback (from many users, good reviews, etc.), we can use that fact as an indicator of SOFIE’s framework, which would have been leveraged during the development of the game and, hence, would have also played a role in the game’s positive feedback. More direct feedback for aspects related to the use of Ethereum will also be sought for evaluation, mostly through user feedback, such as the level of increased trust, usability etc.

4. Tools

This section describes the tools used for validation and evaluation. The first two sections focus on generic tools for validation and evaluation. Subsequently, the next section focus on tools specific to our testbeds and to our analytic evaluation.

4.1 Validation tools

Note that a detailed description of the integration and validation environment can be found in the WP3 documentation. The subsections below are an overview of *how* the environment will support and will be used for validation.

4.1.1 Source control management

SOFIE will be using Git as the source control management technology. Git has become the de facto control management tool and is supported in all relevant development environments. It is supported also by so-called *hosted repository* services such as GitHub[6] and GitLab[7], which in turn have integrated auxiliary services such as code quality inspection services, continuous integration services etc.

The source code itself is managed by following a development model with a set of core developers who have write access to related SOFIE source code repositories. The detailed workflow of this is detailed in WP3 documentation, but the overview is that this set will follow a specific development methodology, including branches, code reviews, issue tracking, etc. The general community of open source developers can then participate in the development of SOFIE code by issuing pull requests, which are then reviewed and controlled by the core developers before being (or not) accepted into the main repository.

4.1.2 Continuous code quality inspection

The SOFIE project, especially the pilots, contain both proprietary and open source development. Similarly, some of the integration environment information needs to remain out of the public eye (deployment keys, etc.). This implies a two-fold approach to source code management and its associated tools:

- Open source portions of SOFIE development can use free-for-open-source-project services such as Travis CI, etc. See the next section on these.
- Proprietary or confidential source code relies on tools and services that can be operated for non-open-source projects, such as SonarQube Community Edition.

The detailed description of the integration environment and code quality automation is described in WP3 documentation, but the overview of the process is as follows:

1. All code changes are run through a continuous integration process. This process includes operations such as:
    a. Compiling the source code.
    b. Running unit tests.
    c. Performing code quality analysis.
    d. Archiving and marking artifacts of a successful CI run.

2. After a successful CI run, a continuous delivery process is performed. This includes:
    a. Deploying the system into a Cloud or container environment.
    b. Running integration tests against the deployment.

[6] https://github.com/
[7] https://gitlab.com/

c. Marking artifacts successfully completed integration tests.
d. Deploying the system into a staging environment.

4.2 Verification tools

While the code-level unit testing uses tools suitable for the particular language and framework, on the integration level, which is more focused on network APIs and user interfaces, the goals of integration, aka verification testing, is to ensure a deployed *system*, comprising of multiple components, operates as planned.

For integration testing, the suite of relevant tools contains:

- **User interface (UI) testing:** In the case of SOFIE, primarily Web UIs. This uses tools such as Selenium, Robot Framework, and SoapUI.
- **Network API testing:** There are a variety of tools, including SoapUI and various testing frameworks for different languages extending the unit-testing framework towards REST API testing.

Additionally, performance and stress testing can use tools such as JMeter, Siege etc.

Most of these tools would be integrated into the automated CI and CD workflow. Some tools may be run only manually, for example, to evaluate the performance of a deployment with a specific (exceptional) configuration.

4.3 Testbed tools

We now present some tools that can be used for testbed-based validation and evaluation.

4.3.1 Remix IDE

Remix[8] is Ethereum's official IDE. It is Web-based and it can be used for developing and debugging smart contracts. Furthermore, it can execute a smart contract using a JavaScript VM inside the browser, or it can deploy it to a testing network using tools such as MetaMask. It provides performance indicators, including cost and time required for a smart contract call.

4.3.2 MetaMask

MetaMask[9] is an extension for the Chrome browser that acts as an Ethereum wallet. It allows the development of browser-based DApps for the Ethereum blockchain. In particular, it extends Chrome's JVM to support Ethereum-specific operations, enabling Web pages to interact with an Ethereum network. Furthermore, MetaMask can act as a "proxy" and enable Remix to deploy and execute smart contracts to a testing network.

4.3.3 Ethereum Network Stats

Ethereum Network Stats[10] is a network monitoring tool that can be used for monitoring the status of an Ethereum network, even for private deployments. It can be used to measure various network metrics, including block generation time, proof of work difficulty, block propagation time, gas price, and others.

[8] https://remix.ethereum.org
[9] https://metamask.io/
[10] https://github.com/cubedro/eth-netstats

4.3.4 BlockScout

BlockScout[11] is a blockchain inspection tool that can be used with any Ethereum Virtual Machine compatible chain. It can be used for searching transactions, viewing accounts and balances, and verifying smart contracts.

4.3.5 Solium

Solium[12] is an Ethereum smart contract analyzer. It can analyze smart contracts and propose fixes for style and security issues.

4.3.6 OpenZepellin

OpenZepellin[13] is a library that can be used for developing secure smart contracts. It provides implementations of standards like ERC20 and ERC721, as well as Solidity components to build custom contracts and more complex decentralized systems.

4.3.7 Fuzzing tools

Fuzz testing or Fuzzing is a black box software testing technique used for finding implementation bugs, using malformed and semi-malformed data injection in an automated fashion.[14] The project will use tools and libraries, such as Peach,[15] to develop fuzzing tools in order to validate and evaluate the robustness of the developed communication protocols.

4.3.8 Hyperledger Explorer

Hyperledger Explorer[16] acts as a feature-rich blockchain explorer for Hyperledger Fabric. Specifically, it enables inspection of the blockchain at both the block and transaction level, while also allowing users to interact with the blockchain via query and invocations directly through the dashboard. It is a powerful tool that also aggregates and produces useful statistics of the network such as the transaction throughput, transaction throttle and median processing time of a block.

4.3.9 Hyperledger network visualization

While no visual tools have been officially created by Hyperledger, there are ways one can visually represent a Hyperledger Fabric network. The easiest way would be to attach a network visualization program, such as Weave Scope on the Docker layer.[17] Since the topology of a Hyperledger Fabric network can be visualized by observing the interactions that occur between the various Docker containers that run on each node, it is possible to visually represent the network by logging these interactions in a visual format. Another approach that can be used is to build the actual visualization engine by subscribing directly to an operating node in a similar fashion to Hyperledger Explorer. The EventHub channel provided by Hyperledger Fabric allows programs to listen to events from multiple channels and thus could be utilized to achieve the logging level required by a network visualizer.

[11] https://github.com/poanetwork/blockscout
[12] https://github.com/duaraghav8/Solium
[13] https://github.com/OpenZeppelin/openzeppelin-solidity
[14] https://www.owasp.org/index.php/Fuzzing
[15] https://sourceforge.net/projects/peachfuzz/
[16] https://www.hyperledger.org/projects/explorer
[17] https://hub.docker.com/r/weaveworks/scope/

4.3.10 Hyperledger Caliper

Hyperledger Caliper[18] acts as a Command-Line Interface (CLI) tool for programmatically benchmarking the transactional throughput capabilities of a blockchain implementation in Hyperledger Fabric. The performance metrics that are calculated by Hyperledger Caliper include network transactions per second, resource utilization, and transaction latency.

4.3.11 Hyperledger Cello

Hyperledger Cello[19] is a relatively new tool that is geared towards DevOps, enabling developers to frictionlessly manage the full lifecycle of a blockchain at scale. Hyperledger Cello, which is written in Python, is one of the crucial technologies that empower the Blockchain-as-a-Service model of IBM.

4.3.12 Hyperledger Composer

Hyperledger Composer[20] is one of the most resilient and close to production Hyperledger tools. It is a high-level modelling language that enables the visual creation of entities within the Hyperledger Fabric network, enabling users to create complex networks easily without actually writing code. While still at an incubation state, it has seen rigorous development and has been extensively utilized because of its ease of use and developer friendliness.

4.3.13 Tineola

Tineola[21] is an aggressive penetration testing tool for Hyperledger Fabric networks, including custom chaincode (Hyperledger’s term for smart contract like functionality) that can be deployed and further test the security of the network.

4.3.14 Project Things

Project Things[22] is a framework of software and services for connecting Things to the Web in a safe, secure and interoperable way. Project Things is composed of three major components. The first component is the Things Gateway, an open source implementation of a Web of Things (WoT) gateway which helps bridge existing IoT devices to the Web. The second one is the Things Framework, reusable software components to help create IoT devices which directly connect to the WoT. The third is Things Cloud, a collection of Mozilla-hosted Cloud services to help manage a large number of IoT devices over a wide geographic area.

4.4 Analytical evaluation tools

In addition to software evaluation tools, examining and modelling the technical specifications of the involved blockchains and overall systems can prove a valuable analytic tool for the evaluation and assessment of the proposed architecture and framework.

Blockchain properties that should be considered can be grouped into *performance metrics* and *usability metrics*.

The most obvious performance metrics are the blockchain's *throughput* and *latency*. Throughput refers to the number of transactions the blockchain can process per second, while latency refers to the expected time between the submission of a transaction and its inclusion in the blockchain.

[18] https://github.com/hyperledger/caliper
[19] https://www.hyperledger.org/projects/cello
[20] https://hyperledger.github.io/composer/latest/
[21] https://github.com/tineola/tineola
[22] https://iot.mozilla.org

These metrics determine the blockchain's *scalability* with respect to the anticipated transaction load.

A third performance metric related to a blockchain's latency is the time it takes for a transaction to be considered “safe enough,” i.e., the number of blocks that should succeed a given transaction's block to provide sufficient guarantees against the probability of a fork. This criterion is subjective and depends on the level of assurance a user desires for a given transaction.

Another performance metric is the amount of *resources* a blockchain uses to run. This is directly dependent on the consensus algorithm used. It is well known that Proof-of-Work uses significantly more resources than other consensus algorithms, like Proof-of-Stake, Proof-of-Authority, and Proof-of-Elapsed-Time. When IoT devices are at stake, the selection of an energy-friendly algorithm can prove vital to the feasibility of the system. By considering the number of transactions processed per second, we can estimate the *cost per transaction*, in terms of energy and resources.

In addition to the transaction cost in terms of energy, in most blockchains there is also a *monetary cost* associated with each transaction, typically paid in the blockchain's own currency. This cost is typically paid to miners and it serves as an incentive for miners to keep running the system. When the transaction rate exceeds the system's throughput, the monetary cost of each transaction increases as a result of competition among parties to give their transactions higher priority over others.

The second group of properties used to evaluate a blockchain concerns usability metrics. These are qualitative metrics that depict a blockchain's versatility in terms of functionality.

The most significant usability metric for blockchains is the language in which smart contracts are written. The language's expressiveness can range dramatically. For instance, Bitcoin only supports a simple language, called *Script*, which is *not* Turing-complete, and whose scope is limited to validating whether someone has the right to spend a given amount or not. Ethereum's smart-contract language, *Solidity*, is a Turing-complete language, allowing arbitrary computations as well as interaction among smart contracts. Finally, Hyperledger Fabric pushes smart contract flexibility even further, allowing smart contracts to run as fully-fledged VMs written in any language, as long as Hyperledger Fabric provides bindings for that language.

The blockchain's permission model constitutes a qualitative evaluation criterion that depends on which nodes are allowed to participate in a blockchain. Permissionless blockchains (e.g., Ethereum) allow arbitrary nodes to join and contribute to their network, just by downloading and executing their software. Permissioned blockchains (e.g., Hyperledger Fabric) are closed to the public. A node needs to have authorization to join such a blockchain's network.

Last but not least, a qualitative property that can play a significant role in complex systems hosting multiple different blockchains, is the blockchains' ability to interact with each other, such as through the Interledger Protocol (ILP). A blockchains' compliance to ILP is important for ensuring that systems relying on them can interact seamlessly within the reference architecture.

5. Conclusions

The main objective of WP4 is the validation and qualitative and quantitative evaluation of the SOFIE architecture and framework. The results of the validation and evaluation work and the recommendations generated will be fed to the architecture and framework design in WP2 and the business integration in WP3. This feedback will be performed in two planned cycles, which will result in an initial (first) and second report with validation and evaluation results and recommendations for the corresponding SOFIE architecture and framework cycles.

The work in WP4 involves two distinct directions: validation and evaluation.

Validation will focus on checking whether the SOFIE platform meets the requirements of the stakeholders that use the platform. Validation is distinct from verification (verifying that the platform meets design specifications) and from the integration and interoperability testing, which is the focus of WP3.

Evaluation will focus on assessing and measuring the performance of the SOFIE platform, based on the KPIs and assessment criteria defined in WP2, but also more general metrics and goals. The evaluation will include both a quantitative and a qualitative component.

Quantitative evaluation will focus on aspects such as response time, throughput, resource utilization, scalability, and availability. Some of these metrics can be assessed through measurements from testbed experiments with implementations of the SOFIE framework while others, such as scalability and availability, will be assessed using analytical models and tools.

Qualitative evaluation will focus on architectural aspects, such as the security and privacy features of the overall architecture and of a subset of the platform components, the ability to integrate different IoT platforms and provide services across domains, and the ability to support innovative applications.

Although validation and evaluation have different objectives, the same platforms will be used for both validation and evaluation. Many different relevant tools for the validation and evaluation processes have been selected and briefly described in this deliverable. The extent to which they will actually be useful and impactful for specific results and recommendations remains to be seen based on the actual attempts to apply them.

6. References

[AEM+18] E. Arzoglou, T. Elo, J. Mattila and P. Nikander, “Applying System Dynamics to Digital Business Platforms: The Case of iOS vs. Android,” Manuscript, Aalto University, October 2018.

[DPR+17] A. Dutta, A. Puvvala, R. Roy and P. Seetharaman, “Technology Diffusion: Shift Happens—The Case of iOS and Android Handsets”, *Technological Forecasting and Social Change*, 118, pp. 28-43, 2017.[RCK17] S. Ruutu, T. Casey V. and Kotovirta, “Development and Competition of Digital Service Platforms: A System Dynamics Approach,” *Technological Forecasting and Social Change*, 117, pp. 119-130, 2017.[Sal15] A. Salazar, “Platform Competition: A Research Framework and Synthesis of Game-Theoretic Studies”, 2015. Available at SSRN: http://dx.doi.org/10.2139/ssrn.2565337

[SOFIE D2.2] S. Paavolainen et al., “Federation Architecture, 1st version”, SOFIE Deliverable D2.2, August 2018. Available at: https://media.voog.com/0000/0042/0957/files/SOFIE_D2.2-Federation_Architecture_1st_version_v1.00.pdf.

[SOFIE D3.1] M. Jaatinen, “Integration Plan”, SOFIE Deliverable D3.1, June 2018. Available at: https://media.voog.com/0000/0042/0957/files/SOFIE_D3.1-Integration_Plan.pdf.

[SOFIE D5.1] I. Oikonomidis et al., “Baseline System and Measurements”, SOFIE Deliverable D5.1, June 2018. Available at: http://media.voog.com/0000/0042/0957/files/SOFIE_D5.1-Baseline_System_and_Measurements.pdf.

[Ste00] J.D. Sterman, *Business Dynamics: Systems Thinking and Modeling for a Complex World*, McGraw Hill, 2000.

[VB17] M. Von Kutzschenbach and C. Brønn, “Education for Managing Digital Transformation: A Feedback Systems Approach,” *Journal of Systemics Cybernetics Informatics*, 15(2), pp. 14-19, 2017.