ENERGY SHIELD

Integrated Cybersecurity Solution

for the Vulnerability Assessment, Monitoring and Protection of

Critical Energy Infrastructures

INNOVATION ACTION

H2020 Grant Agreement Number: 832907

WP5 TOOLKIT INTEGRATION

D5.1 INTEGRATION AND TEST PLAN

(updated)

	Document info
Contractual delivery	**30/06/2020**
Actual delivery	**30/06/2020 / 30/11/2020**
Responsible Beneficiary	**SIMAVI**
Contributing beneficiaries	**PSI, SIGA, FOR, L7D, TEC, KT, CITY, KTH, NTUA**
Version	**1.1**

This project has received funding from the European Union's H2020 research and innovation programme under the Grant Agreement No. 832907

DOCUMENT INFO

Document ID:	D5.1 (updated)
Version date:	30/06/2020 /30/112020
Total number of pages:	39
Abstract:	This task will plan the activities in tasks 5.2, 5.3 and task 5.4 and will create the corresponding deliverable (Integration and Testing Plan). The plan will be based on the architectural document since it relies on dependencies among components. The test plan will contain the testing strategy, testing setup and test cases (preconditions, test execution and expected results). The test cases will be marked passed or failed during task 5.4 and acceptance criteria will be set based on priority and percentage of passed test cases. This testing specification documentation will also aim to stress out platform capabilities (functional and non-functional) in relation with all the defined use cases.
Keywords	cybersecurity, integration, toolkit, testing, common platform

AUTHORS

Name	Organisation	Role
Iacob Crucianu	SIMAVI	Overall Editor
Lavinia Dinca	SIMAVI	Editor
Ana-Maria Dumitrescu	SIMAVI	Editor
Per Eliasson	FOR	Contributor
Hagai Galili	SIGA	Contributor
Rosana Babagiannou	KT	Contributor
Yisrael Gross	L7D	Contributor
Spiros Mouzakitis	NTUA	Contributor

REVIEWERS

Name	Organisation	Role
Valentin Kolev	CoTTP	Overall Reviewer
Christian Hain, Matthias Rohr	PSI	QA Reviewer

VERSION HISTORY

V0.1	13/05/2020	ToC and analytical framework, testing methodology
V0.2	22/05/2020	Integration and deployment approaches
V0.3	15/06/2020	Toolkit – demonstrator preview
V0.4	17/06/2020	Version ready for internal review
V0.5	24/06/2020	Version integrating overall and quality review suggestions
v1.0	30/06/2020	Final version, released to the EC
V1.1.	30/11/2020	Updated version following recommendations from the first review

EXECUTIVE SUMARY

The current report is a framework document defining the approach and planning of the integration and testing activities alongside with the corresponding deliverables within the integration work package.

The integration plan relies on the architectural design document and introduces the integration, deployment and testing strategy for EnergyShield tools.

Integration & deployment approaches are described from logical, development and deployment perspectives accompanied by examples. The detailed plan of activities and phases of building the EnergyShield toolkit are also part of this report.

Testing strategy refers to testing setup and test cases (preconditions, test execution and expected results). Each technology provider will perform the testing of the tools deployed in EnergyShield and provide a test report accompanied by a user and installation manual.

A preview of the **integration concept** of the EnergyShield toolkit is included in the current report as a baseline reference for technology providers and pilot leaders. Both conceptual and logical insights are described as reference guidelines for the forthcoming integration deliverables (reports and demonstrators).

This version includes the updates the evaluators requested during the first EnergyShield project evaluation in July 2020. The recommendations and the updates address the sections referring to requirements, mainly *“D5.1 The OT and smart grid aspect is not evident at the presented testing plan. It would be beneficial to consider at least one case per addressed domain of the smart grid to consider.”*

OT and smart grid aspects are addressed alongside with examples.

All the new content is marked in green.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

ACRONYM	DESCRIPTION
AD	Anomaly Detection
API	Application Programming Interface
CI	Continuous Integration
D	Deliverable
DDoSM	Distributed Denial of Service Module
DSO	Distribution System Operator
EPES	Electrical Power and Energy System
FDD	Feature Driven Development
FR	Functional Requirement
HW	Hardware
IAM	Identity and Access Management).
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of Things
JEE	Java Enterprise Edition
ISO	International Organization for Standardization
KB	Knowledge Base
MQTT	Message Queuing Telemetry Transport
NoSQL	Not only Structured Query Language
OS	Operating System
OT	Operational technology
RDBMS	Relational Database Management System
REST	Representational state transfer
RBT	Risk Based approach to Testting
SBA	Security Behaviour Analysis
SGAM	Smart Grid Architecture Model
SIEM	Security Information and Event Management
T	Task
TC	Test Case
TRL	Technology Readiness Level
TSO	Transmission System Operator
VA	Vulnerability Assessment
VM	Virtual Machine
WP	Work Package

1. INTRODUCTION

1.1. SCOPE AND OBJECTIVES

This work package focuses on the integrations of the tools developed in WP2, WP3 and WP4, and the common platform (included software development kit) is required to run and deploy the tools for the field trials.

Testing and quality assurance activities are also part of the current report and specific activities are detailed. The testing specification documentation will both stress out platform capabilities (functional and non-functional) in relation with all the defined use cases and provide details about using and installing each tool.

1.2. STRUCTURE OF THE REPORT

The report is structured into four main sections as follows:

Section 1 describes the integration and testing approached proposed for EnergyShield including the timeline of activities

Section 2 provides details about the integration and deployment activities starting from the architecture design, continuing with the description of the logical view and the technology support. Additionally, some integration prospects are approached from development and deployment perspectives.

Section 3 describes how the testing and quality assurance activities will be performed for both tools and integrated toolkit.

Section 4 provides a preview of the EnergyShield toolkit in terms of design and the technology proposed to support the implementation of the integration activities.

The last section of the report concludes the integration plan and summarized the following steps and the forthcoming deliveries related to integration.

1.3. TASK DEPENDENCIES

This task takes-over the outcomes form Deliverable 1.1 (D1.1) technical requirements specification [**ESH19**] and from Deliverable 1.4 (D1.4) Architectural design [**ESH20**] which include details about the proposed use cases, tools integration and deployment possibilities.

This report will feed-in the following integration deliverables (demonstrators: D5.2 Common software platform release, incl. user and developer documentation, D5.3 System release v1, D5.4 System release v2, D5.5System release v3, D5.7 Common software platform release, incl. user and developer documentation - final version and reports: D5.6 Test/QA report) and will also guide the elaboration of the tools documentation.

2. INTEGRATION AND TESTING APROACHES

This section presents the integration approach specific to EnergyShield project, based on the technical requirements elicited in Deliverable 1.1 (D1.1) Technical requirements specification [**ESH19**] and on the architectural design and technological insights gathered in the previous Deliverable 1.4 (D1.4) System Architecture [**ESH20**].

2.1. INTEGRATION PERSPECTIVE

Different possibilities of integration are investigated in this section and different perspectives are considered to identify the most feasible solutions for EnergyShield. Based on the outcomes of Deliverable 1.4 (D1.4) System Architecture [**ESH20**] the EnergyShield integration concept is addressed from a logical view and by considering the available technologies.

A series of specific activities needs to be detailed and planned throughout project implementation. Figure 1, below, presents the major steps to be taken.

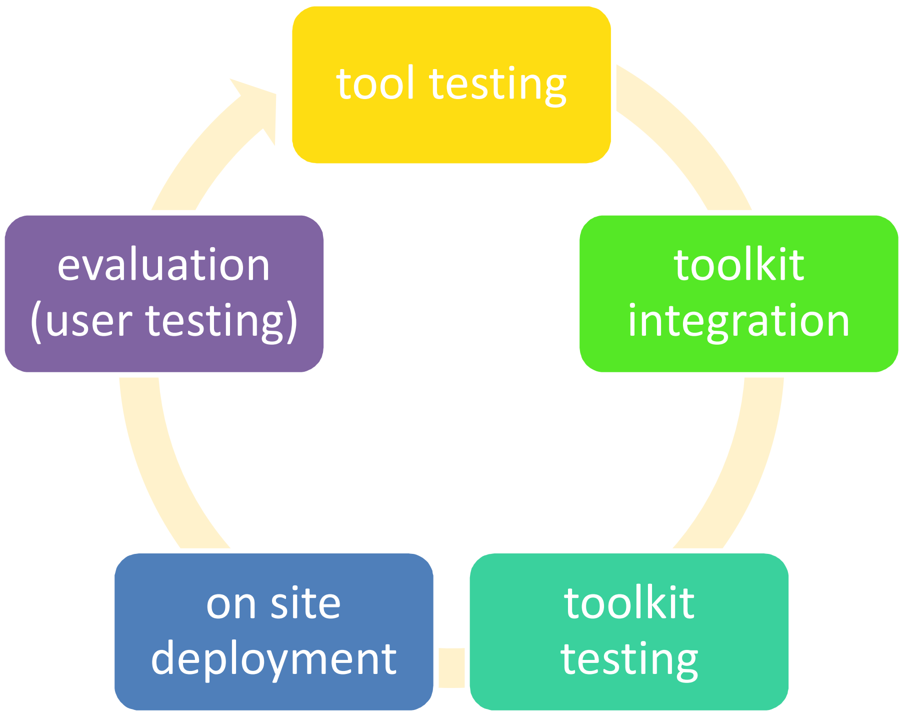

Figure 1. EnergyShield integration, deployment and testing activities

Tools testing. Following the first release of EnergyShield tools (M12) the available features and capabilities are tested and mapped against the needs of the pilot cases provisioned in EnergyShield project.

Toolkit integration. The integrated design proposed for EnergyShield is multi-layered covering operating systems, middleware, database and IoT harvesting methods.

Toolkit testing refers to different testing from unit testing individual modules, integration testing an entire system to specialized forms of testing such as security and performance.

On site deployment includes all the operations to prepare EnergyShield system for assembly and transfer to the computer system(s) on which it will be run in production.

2.2. TIMELINE OF ACTIVITES

The integration plan has been organised in 7 phases (i.e., phase 0 to 6) as shown in Table 1, below, in which the development and integration will follow a stepwise plan to ensure that the components are individually developed and ready to be deployed it the common environment.

The timeline of tools release is synchronized with the planned release of the integrated demonstrator. With every release the integrated demonstrator progresses towards a demonstrator that can be deployed on pilot site for field trials.

While the first four phases of integration accommodates EnergyShield tools with part of the functionalities releases, the last three phases refine the toolkit with all the functionalities available, via testing and quality control to ensure a smooth and effective on-site deployment.

Table 1. EnergyShield integration phases and corresponding reports

Phase	Name	D	M12	M15	M19	M20	M26	M30	M34
	1st release of tools		*x*						
0	Integration plan	D5.1	x						
1	Toolkit concept	D5.2		x					
	2nd release of tools				*X*				
2	Toolkit v1	D5.3				x			
3	Toolkit v2	D5.4					x		
	Final release of tools							*x*	
4	Toolkit v3	D5.5						x	
5	Testing & QA	D5.6						x	
6	Toolkit – final version	D5.7							x

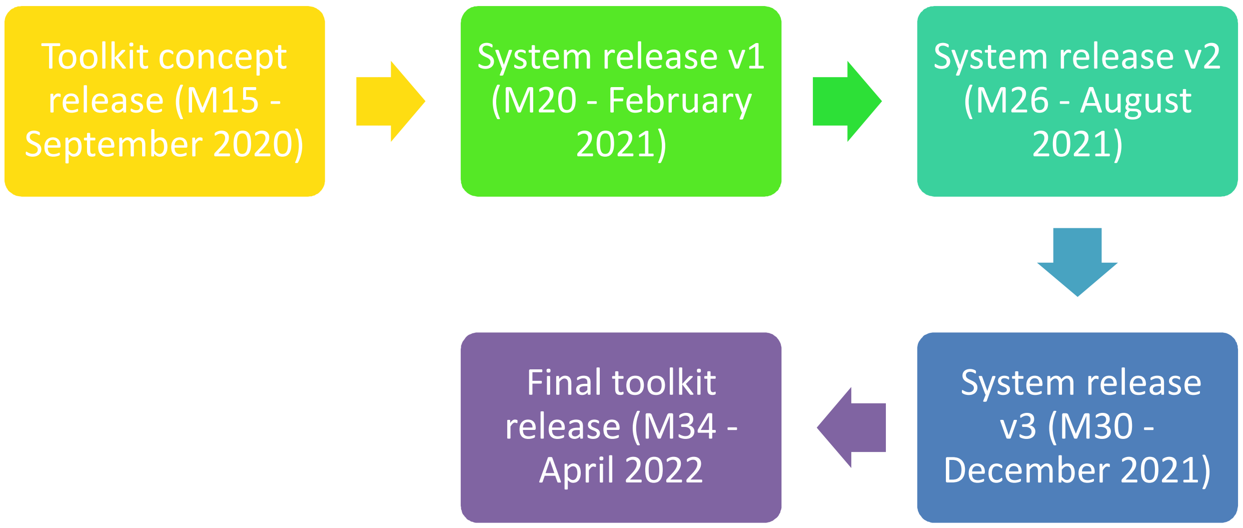
Toolkit concept release (M15 - September 2020)
System release v1 (M20 - February 2021)
System release v2 (M26 - August 2021)
System release v3 (M30 - December 2021)
Final toolkit release (M34 - April 2022

Figure 2. Toolkit demonstrator release timeline

Phase 0 – represents the integration stage at the moment of submitting this report, i.e. the design of the integrated prototype is ready and the tools have been released with the first set of functionalities and the integration activates are detailed and planned in accordance with the planned deliveries.

Phase 1 – builds the EnergyShield integration concept based on the 1st set of tools supporting documents (user manual and installation manual).

Phase 2 – represents the moment of releasing the first integrated EnergyShield toolkit including the first release of tools functionalities

Phase 3 – represents the second release of the integrated toolkit with the second set of functionalities released by technology providers

Phase 4 – is the third release of the toolkit including the final version of tools

Phase 5 – refers to performing the tests and quality assurance

Phase 6 – releases the final version of the toolkit, ready for field trials.

Each phase presented above exploits the outcomes of the previous phase and integrates further components and capabilities as they are developed and released.

Every release of the tool will be functionally tested by the technology provider, while the system releases will be tested and evaluated by Practioners. Within WP6 Filed trials - that starts in M15 - the evaluation methodology will be defined and planned in three evaluation cycles considering the already planned tools and system releases.

3. INTEGRATION & DEPLOYMENT ACTIVITIES

3.1. SHORT OVERVIEW OF THE PROPOSED ARCHITECTURE

In the Task 1.4 System Architecture document we proposed a hybrid project concept that integrates both state of the art IT and energy sector architectures.

The integrated view of the proposed EnergyShield architecture is based on two reference architectural perspectives, 4+x and SGAM iconic for their respective fields, IT and energy sector.

Figure 3 depicts the architectural perspectives proposed in close connection with SGAM layer. Each architectural view is detailed in the Deliverable 1.4 (D1.4) System architecture document [**ESH20**].

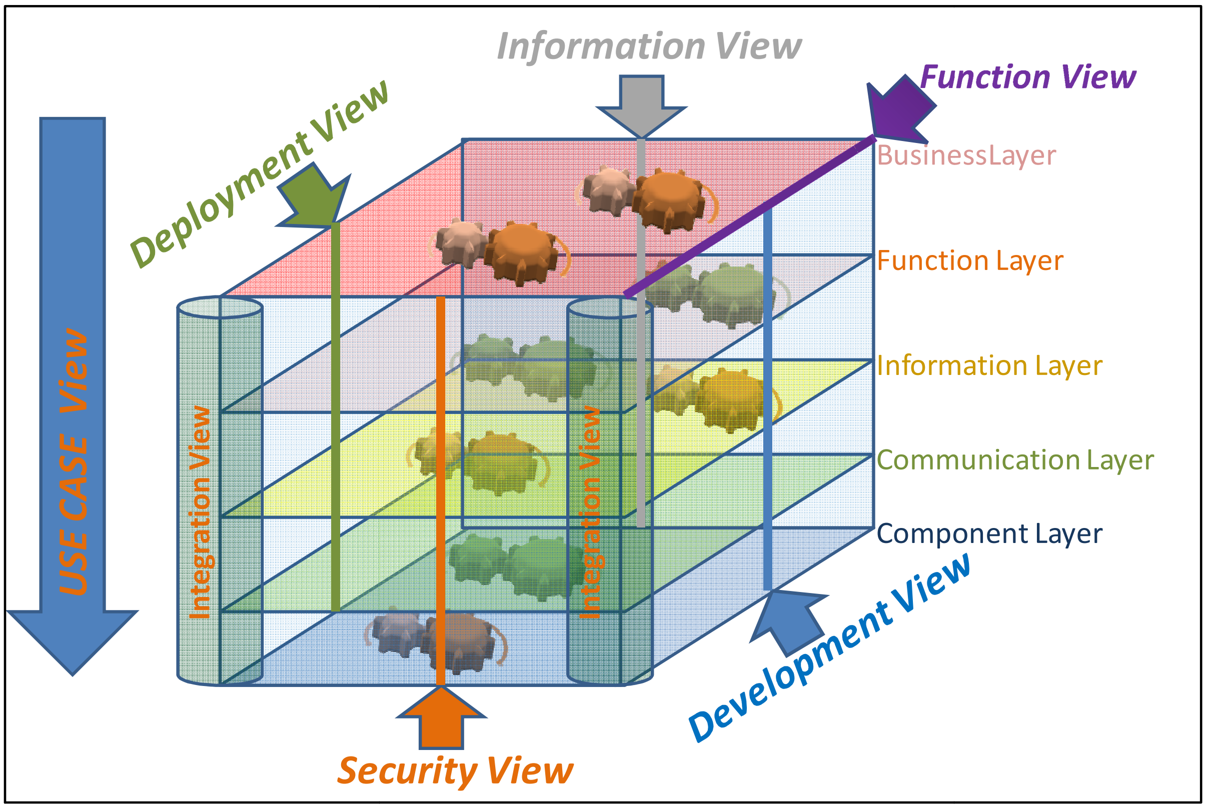

Figure 3. EnergyShield architecture design

The EnergyShield project aims at developing an integrated toolkit covering the complete EPES value chain (generation, TSO, DSO, consumer). The toolkit combines novel security tools from leading European technology vendors and will be validated in large-scale demonstrations by end-users.

The EnergyShield toolkit will be organized in several “shelves” or “drawers” and contains hardware components, software components, and communication ports. It is protected by an authentication mechanism.

This section presents the significant aspects concerning the integration and deployment approaches specific to Energy Shield project, based on the architectural design and technological insights gathered in the previous work packages: WP1 System Specifications & Architecture and from WP2, 3 & 4 that cover the tools development tasks.

The EnergyShield environment is expected to be a system which will be cloned (replicated) for each pilot, and used in the location of that pilot. The pilots cover a wide area of functionality, and this is why we consider the toolkit applicable in many places.

3.2. LOGICAL VIEW

The global logical view of the integrated system is that of a toolkit, carefully placed inside a toolbox. We may compare it with the content of a toolbox, like that presented in Figure 4, below.

Figure 4. Toolkit inside a toolbox | Source: https://www.stanleytools.com/

Placing the toolkit inside a toolbox means the organization of the tools on different layers, specific to the domain in which they are used. For our domain, the reference is the SGAM model technical reference architecture.

3.3. TECHNOLOGY SUPPORT

To accommodate an important number of components developed in different technologies, and having very different characteristics (from standalone applications to light micro-services), we consider the following technologies used to support this complex system.

A. Containers, and containerized deployment of components, using Docker
B. Virtual machines used to support components which are not containerized, like native specific Windows applications

Based on these technologies we are defining a continuous integration machine, like one presented in the figure below, where the components are:

1. Independent modules repository. It is a place (file based repository) where the installation kits for standalone applications are placed. From this place, the specific components are taken using a script, then the user defined installations are run for each specific pilot, where such an application is necessary.
2. User defined Installations are application kits having all necessary parts used to install and configure an application on a specific pilot and environment (for example .zip files, or .msi files, or tar.gz files). These installations are targeted to virtual machines.
3. Component repository. Is the place where common components are placed. It is a maven or nexus repository of components. The components are taken by a buiding and deployment tool and placed in the right destination system.
4. Script runner for component integration. Is a tool used to get the necessary components, then build and integrate them in a destination system. It is a bash or bat file, calling maven, ant or another building tool.
5. Code repository for applications developed as part of this project. Is the place where programmers are placing their code. (SVN, Git, MS Teams).
6. Build tool. Is a system able to build modules from source code. It depends on the programming language and development environment used by developers when using component 5. The most used components are Maven, Gradle, make.
7. Code inspector. Is a tool used to check the quality of code. The tool used is SonarQube.
8. Unit tests, is the component used to run the unit tests. It is Junit for Java code, and Nunit for .NET code.
9. Jira is the bug tracking system used to:
    a. Define the backlog for SCRUM managed development teams
    b. Enter issues as the result of testing the system

   Deployment Server is the main part of the whole continuous development and integration system. It is the hosting place (toolbox) for the whole toolkit. It will host all the tools, organized in layers. The main layers are: Virtualization and Operating System,Basic components (Docker Engine, Message Brocker, Databases, App servers), Specific components and containers (VA, AS, DDM, SBA, SIEM), Authorization and Authentication, Connetions. It will be used to be replicated as content, or to offer services for the pilots. The replication process will include Virtualization and Operating System, Basic components (Docker Engine, Message Brocker, Databases, App servers), Authorization and Authentication, and Connetions, for all pilots, plus specific tools needed by each pilot, from the components and containers layer.
10. Automated integration test tools are used to run integration tests on each pilot. Important to mention that integration tests are specific to pilots. They include testing scripts (.bash, .bat)

11. Here there are the pilots. For each particular one, there will be a specific deployment, which extracts the right components from all available. This is the final place for the continuous development and integration process. Each time a component is changed, at any one on the nodes marked from 1 to 8, the Deployment server (10) start preparing a new delivery. This delivery will influence one or more pilots. Each pilot will have all the time the last valid software version. During the delivery process, pilots will sequentially be updated stating with the simples one. If for one piot the deployment fails, the process will be restarted from the first pilot influenced by the current deployment.
12. Virtual Machines. They are used to host native applications which can not be containerized. Our target is to limit the usage of these components, but as there are some already valuable developed systems basically on Windows OS, with rich user interface and deep usage of OS API, the proposed approach is to host them on virtual machines.
    - The virtual machines will be placed in the same environment, and will be configured to be accessed and to have access with all the components they have to interact. The hosts will be protected by firewalls.
    - The continuous deployment tools will be aware on the component version, and call build or installation scripts when necessary.
    - For processes which cannot be automated (manually managed), the continuous integrating tool will be notified via a script about the version deployed and used.

3.4. INTEGRATION PROSPECTS

Considering the integration of IT solutions, two major prospects are considered:

- **The development side**. This approach is focused on how to develop the different parts of the system (as a framework of tools) in a orchestrated way, taking into consideration that different technologies (tools) are delivered by different technology providers.
- **The deployment side**. This approach is focused on how the final (integrated) system/solution will look like and how its specific parts will exchange and handle data and will trigger processes.

3.4.1. THE DEVELOPMENT SIDE

The main idea is to create a framework where all tools from the toolkit are organized inside a toolbox. They will be replicated on each pilot site, and also they will offer services for all pilots.

We are organizing the development and integration framework in a way that:

A. There is only one place where all components are managed

B. Each time a change is done in a component, the change will trigger an action which will ask the system to recreate all deployments influenced by this change

C. Each pilot will have to work with ONLY the software specific and necessary.

Even if the framework is designed to allow the usage of all components delivered by technology providers, the final real implementation of a pilot will use a specific configuration, which not necessarily contains all the components present in the framework.

For the better understanding of integration requirements we are classifying the components to be integrated in:

1. Software projects (1.1-1.6) in Figure 5, below
2. Components offering full functionality, developed in different technologies, which are:

- Standalone Applications like: Window Applications (.exe files, .dll Files), Linux Applications: Libraries (.lib, .so, .dll) embeddable in other products

3. Hardware components
4. Services exposed by the toolkit and called by pilots.

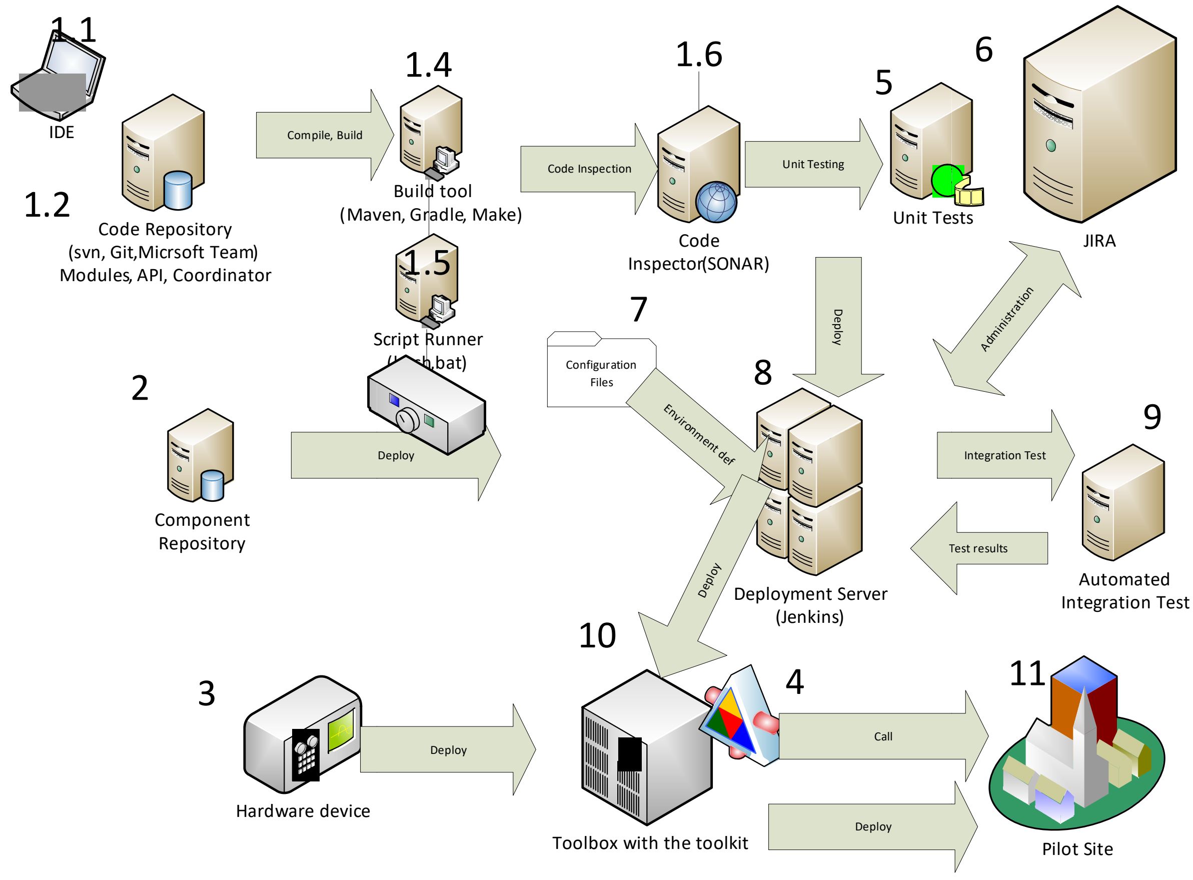

Figure 5. Development process

3.4.2. THE DEPLOYMENT SIDE

To create this integration environment, there is a sequence of steps, as described bellow:

1. Document the pilot data models
2. Define the Knowledge Base (KB) data model
3. Define the meta model
4. Define the integration environment
5. Setup the integration environment. Define VM, Create DBs, Install all components (1 to 10)
6. Define the path of integration of the components developed as part of EnergyShield project. That means 1,2,3,4 from the figure. Install and setup components 1,2,3,4.
7. Install and setup of deployment server (component 8)
8. Link components 1-4 to component 8
9. Place the toolkit inside the toolbox (10).
10. Run the deployment for pilot having only applications developed as part of this project (Bulgarian or Italian pilot)
11. Run tests on pilot

3.4.3. EXAMPLES

- Example A (changes in source code):
  - A programmer is changing something in the source code an API request call for VA
  - The change is reflected in component 1.2, Vulnerability Analysis library, committed on repository
  - The component 1.4 responsible with CI, based on the defined pipeline, will detect that new versions are placed in repository will call Maven to build the system
  - Component 1.6 (SONAR Qube) will inspect the code. If OK, then
  - Component 5 (JUnit) will run unit tests. If OK then
  - Component 8 (Jenkins) will build the whole system
  - Component 9 (Integration scripts) will run the application
  - The component is placed on the toolbox (10 EnergyShield toolkit)
  - The toolbox will be replicated ONLY to affected pilots (11 Bulgarian Pilot, Italian Pilot)
- Example B. (Components changed)
  - A technology provider is changing a component by deploying a new version.
  - Component 2 (docker image of the component) is affected. The repository is updated
  - Component 7 (docker compose) will be activated to make component setups
  - Component 8 (Jenkins) will Build the whole system

    - Component 9 (Integration scripts) will run the application
    - The component is placed on the toolbox (10 EnergyShield toolkit)
    - The toolbox will be replicated ONLY to affected pilots (11 Bulgarian Pilot, Italian Pilot)

- Example C. Hardware tool
    - A technology provider is changing a component by deploying a new version. This is not automatically detected
    - Component 3 (Hardware device) is affected. The repository is updated manually
    - The component is placed manually in the toolbox (10 EnergyShield toolkit)
    - The toolbox will be replicated ONLY to affected pilots (11 Bulgarian Pilot, Italian Pilot)

4. TEST PLAN

This chapter addresses the relevant aspects concerning the testing stage in EnergyShield: testing strategy, test objectives, specific testing approach, testing procedure (guidelines), applicable standards, testing methodology, test case (TC) design and specifications, testing plan, testing deliverables and test results.

The most suitable approach concerning the specific levels of testing in EnergyShield was investigated to address the needs of the pilots.

4.1. APROACH

The testing methodologies to be applied at both tool and pilot solution level are unit testing, integration testing and performance testing.

- Functional testing (Black Box Testing) is the software testing method in which the internal structure/design of the tested item is not known to the testers; applicable to functional testing, scenario testing and simulations.
- Unit testing (White Box Testing) refers to the testing of the software solution's internal structure, design and coding and is applicable to unit level and integration level (e.g. web services).
- Integration Testing tests how parts of the system work are similar to some extent to unit testing. The major difference between those two testing approaches refers to the fact unit tests are isolated from other components, while integration tests are not.
- Performance Testing tests how the software application performs given the expected workload.

Testing strategy covers the major aspects of testing stage (concepts, principles, deliverables and work-products) in the context of developing a typical software system.

Testing is an overall context and shows the dependencies between the different tasks, deliverables and work-products, as follows:

- Defining the goals and objectives of testing stage
- Drafting the testing procedure / process
- Master and detailed test plan.
- Test specification (Test Cases and test scripts)

The testing strategy provides an overview of the scope of testing and includes details about the testing activities together with the corresponding methodology. Most of the testing principles and practices will be applied in order to mitigate the project tasks, testing risks and identify their exposure degree. Various methods and best practices were explored to provide a common approach and terminology when addressing all levels of testing.

The testing phase is important in the development of software or product since it allows the development organization to build quality into the developed system during the life cycle and validate that the quality is achieved. More specifically, a well-defined testing process can have a number of different objectives during the different levels of the development of software or product:

- Find defects or bugs at the developed code, either at unit or integration level
- Provide information regarding the level of quality of the developed software or product
- Ensure that the end result meets the defined business/project requirements
- Ensure that the end result satisfies the system requirement specifications
- Provide a quality product to the end user

4.1.1. TESTING MEHODOLOGY REVIEW

According to the testing strategy described above, the most significant aspects of our testing approach are related to the testing methods and principles constituting the testing procedure, also in line with the available standards. A number of methods and best practices were explored to provide a common approach and terminology when addressing all levels of testing. The addressed aspects are in line with the development of specifications for the EnergyShield solution.

One of the most challenging parts of a testing approach is to find the most suitable testing methodology that meets optimally the performance demands required for a developed software component / application.

As far as the development methodology is concerned, three models have been investigated to determine whether they are in line with the EnergyShield solution development methodology: Waterfall, V-Model and Agile Model.

The principles and practices specific to Agile and V-Model fit to the needs of testing in EnergyShield considers the complexity of the technological context of implementation of the EnergyShield integrated solution. This complexity lies in the difficulty of a unitary approach for the interoperability of the various tools with different maturity levels (TRL) based on diverse technologies and complex architectural solutions.

4.1.2. APPLICABLE STANDARDS

- IEEE 829 – 2008 – IEEE Standard for Software Test Documentation [IEE08]
- ISO-IEC-IEEE 29119 – Software and system engineering – Software testing [**IEE16**]

Another demanding part of the testing approach is to address the existing standards, and more particularly by matching the methodology and concepts of the testing procedure with the rules and stipulations of the standards.

Two IEEE sets of standards are applicable for the testing stage in EnergyShield: IEEE 829-2008 – IEEE Standard for Software and System Test Documentation [**IEE08**] and ISO-IEC-IEEE 29119 – Software and system engineering – Software testing [**IEE16**].

The first applicable standard, IEEE 829 – 2008 [**IEE08**] is a comprehensive guide that specifies a common framework for planning the testing activities [**IEE08**], focusing on software test documentation. Most of the rules and stipulations of this standard were taken over by the new ISO-IEC-IEEE 29119 standard, in Part 3: Test Documentation.

The new international software testing standard ISO-IEC-IEEE 29119, with its latest part published in 2016, aims at replacing the existing standards for software testing: IEEE 829 for test documentation, IEEE 1008 for unit testing, BS 7925-1 for vocabulary of terms in software testing and BS 7925-2 for software component testing [**IEE16**].

4.1.3. TESTING MODEL

Compatibility with the above-mentioned standards was the basis for choosing the two conceptual development models applicable in EnergyShield – V- model and Agile.

The first model (V-Model) provides an applicable option regarding how the testing levels are organized in the standard IEEE 829 – 2008 (Unit (component) testing, Integration testing, System testing and Acceptance testing) and is compliant with our testing approach. The second model (Agile model) is an applicable option due to the agile iterative cycle of development-testing style applied within EnergyShield, and it meets the requirement of continuous improvement of software based on a rapid feedback from testing. [**IEE08**]

4.1.4. SPECIFIC SMART GRID AND OT CONSIDERATIONS

Traditional power grids distributed and managed the power from a central location, but with growing demands for energy as well as reliability and operations, an interconnected dynamic model was developed known as a Smart Grid (SG). Thus, testing on SG becomes very complex due to the complex and multi layer system that need to function together in order for the SG to work. In the context of SG each layer needs to be tested separately as a single entity and also as a whole system [**CHR18**]. Because SG is a cyber system that involves both hardware and software devices, modelling and simulations must be used to discover integration issues [**PAL14**],[**KHA15**]

In [**SCH18**] a brief testing summary as defined in literature is defined, please see table 2.

Table 2. Summary of SG testing from literature, as defined in [SCH18]

Literature reference	Testing area	*How to test*
Kok et al. [KOK11]	Power flow	*Real using 1:1, scaled or simulated data.*
Kok et al. [KOK11]	Data flows	*Power grid only, information grid only, and combined scenarios.*
Kok et al. [KOK11], **Karnouskos and Holanda** [KAR09]	Interaction capture	*Large data volume information capture*
Karnouskos and Holanda [KAR09], **Wang et al.** [WAN10]	Topological changes	*Capture state before test, during test, and after test*
Karnouskos and Holanda [KAR09]	Multi-agent systems	Test one entity by breaking it down into components
Karnouskos and Holanda [KAR09]	Simulator integration	Through APIs
Karnouskos and Holanda [KAR09], **Hahn et al.** [HAH13]	Entity classification	Classify each entity like: prosumer, consumer, transporter, network intruder, SCADA.
Hahn et al. [HAH13]	Network requirements	Network analysis, packet injection, simulate intruders
Wang et al. [WAN10]	Topology generation	Automatic, must test if the model autoscales
Wang et al. [WAN10]	Testing platform	The platform should support different SG topologies.

When testing SGs the information exchange testing process is usually very complex because there are multiple devices types as detailed in table

Table 3. SG devices types

Device	Device short description	*Device details*
A	Not connected to the grid, exchanging data	*These type of devices are part of the SG and sending/receiving data to other parts of the grid or external entities, These devices can be both software only and hardware, eg. weather station. These types of devices are control systems SCADA.*

B	Connected to the grid, not exchanging data	*These type of devices are the necessary grid hardware like power lines and transformers that don't have any network reporting capabilities, There are very important for the SG because they represent the physical SG topology and are required when all SG components must be tested.*
C	Connected to the grid, exchanging data.	*These type of devices are a combination of type A and Type b. When testing a simulation environment must be used to simulate both the power grid and information exchange,*
D	Intruders (both virtual and physical)	*These type of devices are used to simulate, study and test cyber attacks on the SG network. Type D devices are mainly concerned with simulation and study of cyber-physical attacks on the SG.*

As shown in the subchapter before, there are many testing methodologies, but the one chosen by us ISO/IEC/IEEE 29119 [**IEE16**] standard is the one that should be used to test SGs [**SCH18**]. By applying this standard a risk based approach will be chosen [**FEL14**]. The integrated testing process is shown below.

4.1.5. TESTING PROCESS

The appropriate agile testing techniques are specific to the following Agile testing areas:

- A1: Technology-facing tests that support the team
- A2: Business-facing tests that support the team
- A3: Business-facing tests that critique the product (focused on the acceptance of the product)
- A4: Technology-facing tests that critique the product (focused on the performance criteria of the product).

Analysing the specific features of each testing area, two of the agile testing areas were mainly addressed within, and tailored to this particular stage of EnergyShield project implementation:

The particularity of this type of testing is given by the conceptual focusing of testing on feature – *feature driven testing*, applying the same approach as for the feature driven development (FDD) [**FDD19**].

The testing process proposed for EnergyShield, based on agile methodology, and specifically, feature driven testing, takes into account the specific methodological requirements of the project and the conceptual architectural design of the EnergyShield toolkit.

The outcomes of the testing process include documents referring to:

- What was tested (using test cases) and test failures (bugs)
- Which bugs caused failures and when they can be addressed

Specific tasks for each team are registered in Jira - a specific application for issue tracking and general project management features.

4.1.6. TEST PLAN

A test plan provides the necessary planning information for a development organization to get ready for testing by allocating the appropriate resources early on at the project. By providing detailed requirements for hardware, software and people as early in the lifecycle as possible, the quality of the end product is ensured minimizing the possibility of delays, additional costs and additional effort at later stages of the project.

The goal of a master test plan is to:

- Provide an overview of the testing activities at a detailed level, in compliance with the testing strategy
- Define the test environment requirements
- Define each level of testing, focus areas and testing techniques to be applied
- Define in detail the features / functionalities and test cases to be tested
- Define the testing tasks to be executed, and assign responsibilities
- Define the business and technical risks that can be addressed through each level of testing.

As mentioned earlier a risk based approach (RBT) is considered for testing. RBT is a specialized testing mantra that prioritizes software tests based on their risk of failure calculated as an average between likelihood and impact. In theory one can define an infinite number of tests, RTB makes sure that the features with the highest risk of failure and most impact to the organisation are tested first and that accurate tests are done specifically for those features.

The structure of the test plan is in accordance with the guidelines set by the IEEE 829-2008 [**IEE08**].

4.1.7. TEST CASE DESIGN SPECIFICATIONS

Test design specification addresses the features to be tested. A standardized form of designing test specifications undertakes the definition of high level test cases that fulfil the defined business requirements and ensure traceability.

A Test Case Specification refers to the features tested within an appropriate testing scenario and requires test scripts or a procedure. A Test Case is defined by the following items, pursuant to the guidelines of ISO/IEC/IEEE 29119 Software Testing:

- Objective
- Test actions (flow / path of steps)
- Expected results
- Preconditions for execution

These guidelines can be formulated in five steps describing the process to be followed for extracting representative test cases:

- Review of the project defined requirements and use cases (as presented in Deliverable 1.1 (D1.1) [**ESH19**]) as well as the integration methods of the different components
- Define the features to be tested and classify them accordingly to test groups
- Form detailed test cases for the validation of named features
- Execute the test cases
- Document the test protocols

Using a standard form in designing the test cases defines the basis of a common terminology and a consistent definition of processes, actions, requirements and results. Thus, the tester will easily implement the specific tasks in accordance with the organizational test process as described in the ISO/IEC/IEEE 29119 standard. The proposed form for the test case design is compliant with the above-mentioned structure and proposed methods and is presented in Table 4.

Table 4. Test case design

TC	***<ID>***	***<Title of the use case>***
UC	*<related use cases*	
FRs	*<related functional requirements>*	
Precondition	*<the setup needed for a test case to be executed successfully>*	
Test environment (optional)	*<hardware, software and network configuration needed>*	
TC Step (actions)	**Obtained result**	**Verdict**
1. <steps to be executed>	*<expected results>*	*<pass>, <fail>*

4.1.8. PHASE APPROACH

Considering the particularities of the system, a phase test is proposed to demonstrate how the tools integrated in the EnergyShield toolkit are placed in the Operational technology (OT) environment, and how they can influence OT behaviour, especially in relation with Smart Grid type operations.

The proposed phase test approach follows three main stages:

1. Testing the tools individually in a laboratory environment. The expected outcome of these tests would be demonstrating the functionalities of the individual tools
2. Testing the tools placed in the toolkit in a laboratory environment. At this stage, the way the tools are interfaced and are communicating is tested by means of input and output information received from consumption points.
3. Testing the tools integrated in the toolkit and placed in the OT environment. This is the most extensive test and tests the functionality as designed and implemented in the final system.

The proposed approach considers each tool's unique functionalities and particularities. Thus, the next sub-chapters are presenting in detail how each phase will be applied to each tool, and finally for the integrated toolkit.

4.1.8.1. PHASE 1 TESTING

For this phase there is NO interaction with OT. The tools are producing information to be used by other tools, but the focus will be just on individual tool functionality.

Vulnerability Assessment (VA) tool. The main output considered here is the model created. Using this model different simulation will be made. The model will be tested against testing data provided by the developer of the tool.

Security Behaviour Analysis (SBA). This tool will be tested individually by users by deploying the tool on a isolated test environment. A set of test cases will be executed where actual and expected output will be compared.

Anomaly Detection (AD) tool. This is a hardware tool and will be tested in laboratory conditions against test data.

DDoS Mitigation (DDM) tool. This tool will be installed in laboratory conditions, and simulated attacks will be sent. The results will be tested against test defined data.

SIEM Tool. This tool will be installed in laboratory conditions with different systems. The test will be performed based on predefined test data and test results.

4.1.8.2. PHASE 2 TESTING

For this phase there is NO interaction with OT. The tests target the interaction between tools, and less considering individual tools functionality.

To run this testing phase, the message broker mechanism should be up and running. The dialog between tools is considered to be part of the message exchange governed by the message broker.

Each tool will produce messages for some specific topics, and will consume messages from other topics.

- Vulnerability Assessment (VA) tool.

The outputs are for DDM, SIEM

It consumes information from SBA, AD, SIEM.

To test the outputs, simulations will be run on the model. The outputs, placed on output message queues (topics) will be tested against test data.

To test the inputs, test data will be placed in the input message queues (topics). The model will be updated based on that data. Then the simulation will be run, and again, the results placed in the output topics will be tested against test data.

- Security Behaviour Analysis (SBA).

The outputs are for VA .

SBA shall only indirectly interact with SIEM.

To test the outputs, the tool will collect standard (test) data from OT users. Then the data placed in the output topics will be tested against test data..

- Anomaly Detection (AD) tool**.** This is a hardware tool.

The outputs are for VA, DDM, SIEM

It consumes information from SIEM

To test the outputs, the tool will be connected to a laboratory device. Then the data placed in the output topics will be tested against test data.

To test the inputs, test data will be placed in the input message queues (topics). The toll will be run and outputs will be compared with expected test data.

- DDoS Mitigation (DDM) tool.

The outputs are for VA, AD, SIEM

Inputs are from AD, VA, SIEM

To test the outputs simulated attacks will be sent. Messages placed in output queues will be tested against test defined data.

To test the inputs, test data will be placed in the input message queues (topics). The tool will be run and outputs will be compared with expected test data.

- **SIEM Tool**.

The outputs are for VA, AD, DDM

Inputs are from VA, AD, DDoSM

To test the outputs, the application will be run with the user entering test data. Then the output topics will be read and data compared with test data.

To test the inputs, test data will be placed in the input message queues (topics). The tool will be run and outputs will be compared with expected test data.

Test the full path.

This will follow several full paths to be tested. Our proposal is:

1. AD->VA->SIEM->AD
2. AD->VA->DDM->SIEM->DDM
3. SBA->VA->SIEM
4. VA->DDM->AD->SIEM->VA

4.1.8.3. PHASE 3 TESTING

For this phase **there IS interaction with OT**. The tests are focused on the interaction between tools, and in relation with OT.

To run this testing phase the message broker mechanism should be up and running. The dialog between tools is considered to be part of the message exchange governed by the message broker.

Each tool will produce messages for some specific topics, and will consume messages from other topics.

The installation in the OT will be performed so that this will not degrade the functionality.

This is the most extensive test phase and includes:

a) Testing the placement of the tools in OT and how they influence OT normal functionality
b) Testing the tools against the expected results when placed in OT
c) Registering OT behaviour when tested tools are working.
d) Testing the integrated toolkit placed in OT, for the way the tools are interacting inside toolkit and between toolkit and OT.
e) Testing how smart grid components are influenced by the tool

The points from a) to e) are explained for each tool and for the integrated toolkit in the following.

- **Vulnerability Assessment (VA) tool**

a) The tool will be installed on o computer inside OT. Considering the specification of the tool it will not interact directly with OT component.
b) The tests will be the same as that for the Phase 2.
c) It is expected that the OT will be not directly influenced by the presence of the tool alone. But when integrated it is expected to indirectly influence the OT via DDoSM or SIEM.
d) The interaction between tools will be similar with that defined in Phase 2.
e) The smart grid is expected not to be influenced by the functionality of this tool alone. But when integrated is expected to indirectly influence the smart grid via DDoSM or SIEM.

The outputs are for DDM, SIEM

It consumes information from SBA, AD, SIEM.

- **Security Behaviour Analysis (SBA)**

a) The tool will be installed on o computer inside OT. Considering the specification of the tool it will not interact directly with OT component.
b) The tests will be the same as that for the Phase 2.
c) It is expected that the OT will be not directly influenced by the presence of the tool alone. But when integrated is expected to indirectly influence the VA.
d) The interaction between tools will be similar with that defined in Phase 2.
e) The smart grid is expected not to be influenced by the functionality of this tool alone. But when integrated is expected to indirectly influence the smart grid via VA.

The outputs are for VA and eventually for SIEM (via VA)..

- **Anomaly Detection (AD) tool.** This is a hardware tool.

a) The tool will be installed inside OT. Considering the specification of the tool it will interact directly with OT component. After installing the tool, the OT components considered will be tested and compared with their functionality without AD.
b) The tests will be the same as that for the Phase 2.
c) It is expected that the OT will be not influenced by the presence of the tool alone. But when integrated is expected to indirectly influence the OT via VA, DDoSM or SIEM.
d) The interaction between tools will be similar with that defined in Phase 2.
e) The smart grid is expected not to be influenced by the functionality of this tool alone. But when integrated is expected to indirectly influence the smart grid via VA, DDoSM or SIEM.

The outputs are for VA, DDM, SIEM

It consumes information from SIEM

- **DDoS Mitigation (DDoSM) tool**

a) The tool will be installed on a VM thant can receive the traffic, before reaching the OT. Considering the specification of the tool it will interact directly with the IT infrastructure of the OT After installing the tool, the OT components considered will be tested and compared with their functionality without DDM tool present. No differences should be found.
b) An attack against the OT will be simulated. As the attacks comes from the external IT, this simulation is to be performed from the external IT. It is expected that the tool will stop the attack, the external attacker will be isolated, and the OT will continue to function normally.
c) It is expected that the OT will be influenced in the way that the attack will be stopped,meaning that it will not reach the OT, or if reaching at the beginning, it will be stoped before producing effects. As a DDoS attack can have cascading

effects, it is expected that OT will react even by stoping the information from smart meters.

d) The interaction between tools will be similar with that defined in Phase 2.

e) The smart grid is expected to be influenced by the functionality of this tool in the sense that, according with DDoS definitions, some functionality will be changed to avoid the attack.

The outputs are for VA, AD, SIEM

Inputs are from AD, VA, SIEM

- **SIEM Tool**.

a) The tool will be installed on o computer inside OT. Considering the specification of the tool it will interact directly with OT component. After installing the tool, the OT components considered will be tested and compared with their functionality without AD

b) Different commands will be issued from SIEM tool, according with the test data defined. The behaviour of OT will be registered and compared with functionality without SIEM tool.

c) It is expected that the OT will be influenced in the way that the OT will behave according with the commands sent..

d) The interaction between tools will be similar with that defined in Phase 2.

e) The smart grid is expected to be influenced by the functionality of this tool in the sense that, according with SIEM definitions, some functionality will be changed.

The outputs are for VA, AD, DDM

Inputs are from VA, AD, DDoSM

- **Test the full path.**

When testing the full path, in OT environment the assumption is that all the tools are already installed, and the behaviour of the OT with each individual tool functioning alone is known.

This will follow several full paths to be tested. Our proposal is:

Protect the OT - Command smart grid when an anomaly is detected

1. AD->VA->SIEM->AD->OT

This test will simulate an anomaly. The AD tool should detect, send a message to VA and SIEM, and SIEM should act to command the smart grid accordingly.

Protect the OT - Avoid a DDoS Attack, and add elements to white list

2. DDM->SIEM->DDM-OT

This test will simulate an attack. The DDoSM tool should detect, send a message to SIEM, and SIEM should will consider low risk, and tell this to DDoSM. Then another attack will be simulated, a high risk one. This time DDoSM is expected to command the smart grid accordingly.

Improve the system - Adjust VA parameters based on SBA

3. SBA->VA->SIEM ->OT

This test will conduct a behaviour analysis. Based on the result the VA model will be updated, and the result will be sent to SIEM. SIEM will assess the informationand ask the administrator to adjust OT parameters.

Improve the system - Adjust DDM based on Vulnerability Analysis

4. VA->DDM->AD->SIEM->VA->OT

This test will obtain data from VA tool to update DDoSM definitions which will be sent to AD tool which - based on the definitions - will detect new anomalies and send them to SIEM to be validated. Finally, the information will be sent again to VA which is expected to command the Smart Grid accordingly.

4.1.9. EXPECTED RESULTS

Expected testing results are very significant items within the testing process. The monitoring activity of testing results is retrieved in the test plan and is materialized by the elaboration of testing reports.

A standardized structure for each type of report will be defined taking into consideration the rules and specifications of the IEEE Standard 29119, section IEEE 29119-3 [**IEE16**]:

A template for the results documentation (outcomes will be drafted and included in the updated version of this report. This should include at least details about:

- Test cases for each tool and the test execution report;
- User and installation manuals for each tool
- Integration test cases for the integrated pilot solution and the execution report.

The proposed approach for testing should be considered as a guideline to deliver a complete and stable software application for the field trials. If constraints apply and the proposed methodology cannot be followed providing a piece of evidence of testing by other means also covers the testing requirements.

5. ENERGYSHIELD TOOLKIT – DEMONSTRATOR PREVIEW

This section presents a preview of the EnergyShield integrated toolkit. Integration and deployment possibilities have been addressed in D1.4 Architecture design [**ESH20**] where EnergyShield tools have been evaluated considering different views: development, deployment, information, function, use case and security. This assessment provided extended information about the integration and deployment possibilities that should be considered when design and building the integrated toolkit.

Figure 6 describes the steps followed in building the EnergyShield toolkit.

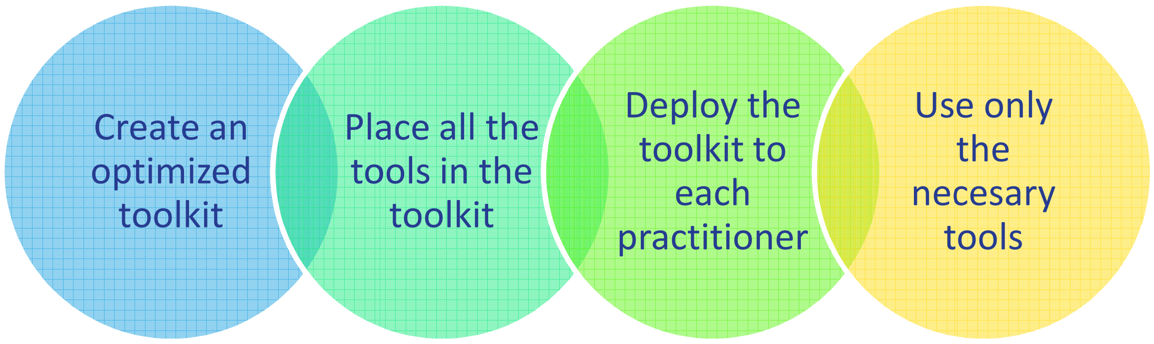
Create an optimized toolkit
Place all the tools in the toolkit
Deploy the toolkit to each practitioner
Use only the necesary tools

Figure 6. EnergyShield toolkit integration and deployment flow of activities

Starting from the necessity of having an assembly of tools easy to deploy on different sites, creating a toolkit was considered.

- **Create an optimized toolkit.** The set of building units was designed and configured for optimal performance
- **Place all the tools in the toolkit.** The EnergyShield tools have been surveyed considering the deployment capabilities.
- **Get the toolkit to each practitioner.** A dedicated instance will be defined for each Energy Shield pilot to be installed on-site.
- **Use only the necessary tools**. The toolkit deployed on-site will include only the necessary tools and dedicated user interfaces.

Open standards and protocols are used for all components placed in the toolkit both in terms of development and ease of use.

The integration characteristics of the system are presented using the following structure:

- Tools used to connect components on the same layer
- Tools used to connect components between layers.

Example of tools used to connect components on the functional layer

- API used between backend and frontend
- Data messaging tools to connect different IoTs in component layer.

Example of tools used to connect components between layers

- Communication gateway used to connect component layer and communication layer
- MQTT Broker used to connect component layer and communication layer
- Kafka messaging used to connect communication layer, functional layer and business layer

The toolkit is organized in several “shelves” or “drawers” and contains hardware components, software components, and communication ports. It is protected by an authentication mechanism (Identity and Access Management).

The representation of the toolkit with the corresponding shelves is depicted in Figure 7, below.

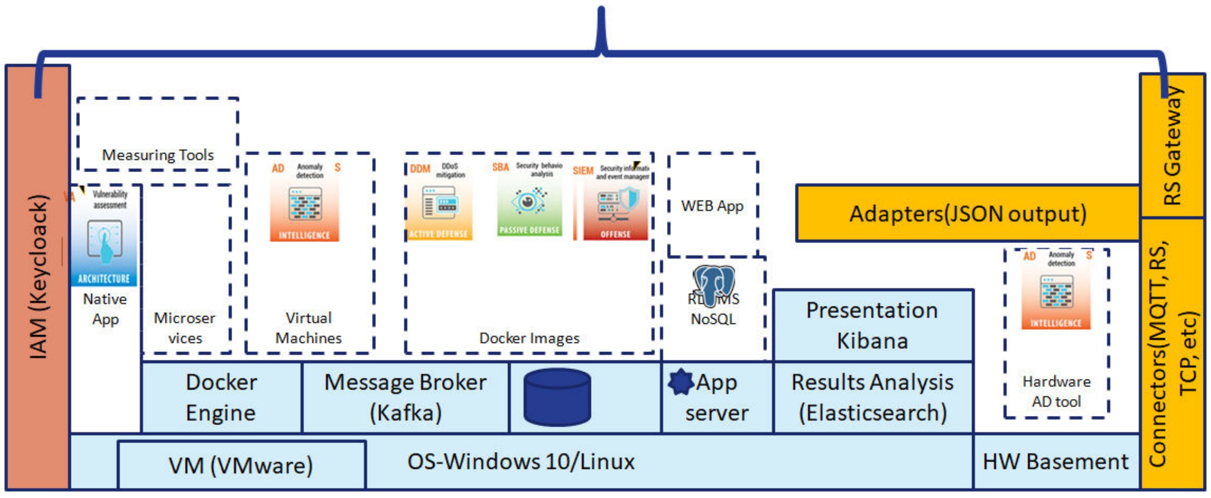

Figure 7. EnergyShield toolkit organization

The shelves considered are grouped in

- **General** common components that include the framework structure tools installed and configured to accommodate all EnergyShield tools.
- **Specific** components including the applications and their corresponding functionalities

The General common component refers to:

- **Basement hardware shelf (HW Basement).** Contains power supplies, cables and measurement units.
- **Basement software shelf:** It contains the operating system, Java Runtime and virtualization mechanism, serving all the software components. In our case it is Linux running VMWare system and running Java 1.8 and JEE. Also, Windows 10 is accepted by the system

- **Container Manager:** It runs a container engine (Docker), where all Docker images can be run
- **Communication HUB**: It runs a message broker (Kafka) and communication bus for REST services.
- **Persistence**: There are software components used for data persistence. They can be RDBMS (PostgreSQL) databases and NoSQL (Cassandra)
- **Application servers.** Software components (e.g., Apache Tomcat) that host web application server
- **Presentation tools**. Software tools used to display data in a format required by end users, such as Kibana.

The **specific components** are covering the functionalities of VA, AD, SBA, DDoSM, SIEM and from the toolkit point of view are:

- Virtual machines (software components)
- Docker images (software components)
- Standalone applications (software components)
- Message producers (communication ports)
- Message consumers (communication ports)
- REST Clients (communication ports)
- REST APIs (communication ports)
- Hardware components used for AD (hardware components)

Considering the particularities of the proposed architecture alongside with the deployed tools and technologies, the EnergyShield integration approach builds upon technology-oriented pillars supporting architectural layers.

6. CONCLUSION

The present document presents the approach considering the integration activities planned within EnergyShield project. It is a framework document including the proposed analytical framework, the detailed activities and the expected outcomes of the integration work package.

Based on the previously achieved results in terms of technical requirements, use cases descriptions, the architecture design and the first release of the EnergyShield tools, the integration plan provides relevant details about the toolkit integration concept from logical, development and deployment perspectives. Moreover, a preview of the integrated concept is provided.

This report will be the main input for the forthcoming demonstrators and the tools delivery timeline together with their corresponding documentation (user manual, installation manual, test report).

The most suitable approach concerning the specific levels of **testing** in EnergyShield was investigated to address the needs of the toolkit. A reference testing model was drafted and illustrates the steps to be taken by each tool provider to align the testing activities and deliver the results in a standardized manner. The proposed approach for testing should be considered as a guideline to deliver a complete and stable software application for the field trials. If constraints apply and the proposed methodology cannot be followed providing a piece of evidence of testing by other means also covers the testing requirements.

Overall, this report provides the framework activities planned for a successful delivery of the EnergyShield toolkit for field trials.

REFERENCES

[CHR18] Chren, S.; Rossi, B.; Bühnova, B.; Pitner, T. Reliability data for smart grids: Where the real data can be found. In Proceedings of the 2018 Smart City Symposium Prague (SCSP), Prague, Czech Republic, 24–25 May 2018.

[ESH19] Energy Shield Consortium (2019) D1.1 Technical requirements specification

[ESH20] EnergyShield Consortium (2020) D1.4 System Architecture

[FDD19] Agile Modeling (2019), http://agilemodeling.com/essays/fdd.htm, accessed in March 2019

[FEL14] Felderer, M.; Ramler, R. Integrating risk-based testing in industrial test processes. Softw. Qual. J. 2014, 22, 543–575.

[GIN20] Mirko Ginocchi, Amir Ahmadifar, Ferdinanda Ponci & Antonello Monti (2020) Application of a Smart Grid Interoperability Testing Methodology in a Real-Time Hardware-In-The-Loop Testing Environment, Energies 2020, 13(7), 1648; https://doi.org/10.3390/en13071648

[IEE08] IEEE 829-2008 - IEEE Standard for Software and System Test Documentation, https://standards.ieee.org/standard/829-2008.html

[IEE16] IEEE 29119-5-2016 - ISO/IEC/IEEE International Standard - Software and systems engineering - Software testing -- Part 5: Keyword-Driven Testing , https://standards.ieee.org/standard/29119-5-2016.html,

[HAH13] Hahn, A.; Ashok, A.; Sridhar, S.; Govindarasu, M. Cyber-Physical Security Testbeds: Architecture, Application, and Evaluation for Smart Grid. IEEE Trans. Smart Grid 2013, 4, 847–855

[KAR09] Karnouskos, S.; Holanda, T.N.D. Simulation of a Smart Grid City with Software Agents. In Proceedings of the 2009 Third UKSim European Symposium on Computer Modeling and Simulation, Athens, Greece, 25–27 November 2009; pp. 424–429.

[KHA15] Khaitan, S.K.; McCalley, J.D. Design techniques and applications of cyberphysical systems: A survey. IEEE Syst. J. 2015, 9, 350–365.

[KOK11] Kok, K.; Karnouskos, S.; Ringelstein, J.; Dimeas, A.;Weidlich, A.;Warmer, C.; Drenkard, S.; Hatziargyriou, N.; Lioliou, V. Field-testing smart houses for a smart grid. In Proceedings of the 21st International Conference and Exhibition on Electricity Distribution (CIRED 2011), Frankfurt, Germany, 6–9 June 2011.

[PAL14] Palensky, P.; Widl, E.; Elsheikh, A. Simulating cyber-physical energy systems: Challenges, tools and methods. IEEE Trans. Syst. Man Cybern. Syst. 2014, 44, 318–326.

[SCH18] Martin Schvarcbacher, Katarína Hrabovská, Bruno Rossi & Tomáš Pitner (2018) Smart Grid Testing Management Platform (SGTMP) Appl. Sci. 2018, 8, 2278, DOI: 10.3390/app8112278

[WAN10] Wang, Z.; Scaglione, A.; Thomas, R.J. Generating Statistically Correct Random Topologies for Testing Smart Grid Communication and Control Networks. IEEE Trans. Smart Grid 2010, 1, 28–39.

DEVELOPING THE CYBER-TOOLKIT THAT PROTECTS YOUR ENERGY GRID

www.energy-shield.eu