Deliverable 2.3

Overall Framework Architecture Design (1st Draft)

Technical References

Document version	:	1.0
Submission Date	:	28/02/2021
Dissemination Level	:	Public
Contribution to	:	WP2
Document Owner	:	UNI
File Name	:	Overall Framework Architecture Design (1st Draft)
Revision	:	1.1

Project Acronym	:	BIECO
Project Title	:	Building Trust in Ecosystem and Ecosystem Components
Grant Agreement n.	:	952702
Call	:	H2020-SU-ICT-2018-2020
Project Duration	:	36 months, from 01/09/2020 to 31/08/2023
Website	:	https://www.bieco.org

This project has received funding from the European Union´s Horizon 2020 Research and Innovation Programme under Grand agreement No. 952702.

Revision History

REVISION	DATE	INVOLVED PARTNERS	DESCRIPTION
0.1	07/01/2021	UNI	Document structure. Initial contribution to sections 2, 3.1, 3.2, 3.3, 3.4, 3.5. 4.3
0.1	12/01/2021	UMU	Initial version of 4.3, contribution on section 3.4 and comments
0.1	13/01/2021	CNR	Initial version of 3.3
0.1	14/01/2021	IESE	Review, comment and suggest edits on section 3.3, edit Table 4
0.1	17/01/2020	IESE	Initial version of 3.2.1, added an initial description to the figure in section 2.
0.1	19/01/2021	UMU	Initial version of 4.1
0.2	19/01/2021	IESE	Added content on section 1, added in the table of Glossary, edit table from section 3.2.3, updated the picture in section 3.3.3.
0.2	19/01/2021	RES	Modifications in Table 2 (T4.1). Initial version of 3.4 and insertion of a Figure. Modifications and comments in Table 4
0.2	19/01/2021	UNI	Modifications to section 4.3 and the common vision diagram. Initial runtime view figure.
0.2	19/01/2021	CNR	Review of section 3.3
0.3	20/01/2021	GRAD	Contributions to section 3.1
0.3	21/01/2021	UTC	Contributions to section 3.1. and 3.2
0.3	26/01/2021	IESE	Text to 3.4.1.2
0.3	26/01/2021	RES	Modifications to section 3.4.1.1 and replacement of Figure in the section 3.4
0.3	26/01/2021	GRAD	Contributions to section 4.1 and 4.3
0.4	28/01/2021	UMU	External Review, resolution of comments related with MUD, figure 6 and modifications to section 4.
1.0	02/02/2021	UNI	Text to Section 4, 4.2 and conclusions. Revision of the overall document to prepare v1.0.
1.0	05/02/2021	RES	Inputs to sections 3.2.4 and 3.4.7 regarding interfaces
1.0	05/02/2021	UNI	Text for Sections 1, 2 and 3, the executive summary, glossary and acronyms. Revision, formatting and harmonization of the whole document.
1.0	08/02/2021	7B	Inputs to section 3.4.7 regarding ATB's interface.
1.1	27/02/2021	UNI	Finalization of the document based on the project coordinator review

List of Contributors

Deliverable Creator(s): Ricardo Peres (UNI), Emilia Cioroaica (IESE), Sara Matheu (UMU), Enrico Schiavone (RES), Gabriele Morgante (RES), Rosaria Esposito (RES), Lorenzo Falai (RES), Eda Marchetti (CNR), Antonello Calabrò (CNR), Lilian Adkinson (GRAD), Ovidiu Cosma (UTC), Cosmin Sabo (UTC), Radosław Piliszek (7B).

Reviewer: Andrea Ceccarelli (RES), Ricardo Peres (UNI), Felicita Di Giandomenico (CNR), Sanaz Nikghadam-Hojjati (UNI), Jose Barata (UNI)

BIECO project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No **952702**.

Acronyms

Term	Definition
API	Application Programming Interface
CEP	Complex Event Processing
CPE	Common Platform Enumeration
CVE	Common Vulnerabilities and Exposures
DSL	Domain Specific Language
DT	Digital Twin
FMI	Functional Mock-up Interface
HW	Hardware
ICT	Information and Communication Technology
IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
JSON	JavaScript Object Notation
MOCC	Models for calculation and communication
MUD	Manufacturer Usage Description
RAMI	Reference Architectural Model for Industrie 4.0
REST	Representational State Transfer
SoS	System-of-Systems
SW	Software
TCP	Transmission Control Protocol
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
WP	Work Package

Glossary

Term	Definition
Actor	An Actor represents a non-cyber-physical party of the ecosystem, such as a specific person, company, or some other legal entity that interacts with systems and digital assets, such as software components.
Controlled environment	This is a controlled setup of software and hardware components (or alternatively their stubs or mocks), network configurations and necessary settings useful for the execution of the software/system in a real or realistic context. It enables the execution of validation and verification activities and the collection of results/events in a context in which the system can be stressed in a safety way. To this purpose, the controlled environment and/or its components (mocks stubs, real devices and so on) can be equipped with probes.
Design Time	It is the software lifecycle phase in which the product is designed, developed, implemented, verified and even certified, before its release to the market. At the end of these processes, the product is intended to be ready for its usage and validated in terms of functionality and security.
Digital Ecosystem	A structural and behavioural construct that forms around digital products, which dynamically interact. These products can be software components or cyber physical systems.
Digital Twin	This is a simulation model fed with real time or predicted data.
Execution Time	The time when a system/system component executes within a real (at runtime) or a virtual environment (at design time).
Framework	Composition of tools that communicate over well specified interfaces. It enables implementation of methods.
ICT	Information and Communication Technology - it indicates the domain of telematics, computer science, multimedia and internet.
Middleware	Acts as an integration layer to facilitate the interoperability amongst the components of BIECO's ecosystem. In this context, it supports communications in two key schemes, one being a publish and subscribe pattern for time critical communications, the other a service-oriented pattern for remote execution/access. For the latter, the middleware contemplates two main supporting functionalities, one being a yellow-pages directory facilitator for service discovery/registration, the other a service orchestration mechanism for complex management of service interactions and composition.
Mock	This is an object that emulate the behaviour of a real object
Predictive Simulation	Simulation based on a set of well-defined situations that evaluate DT behaviour in a virtual environment
Predictive Virtual Evaluation	Execution of system/system behaviour in a simulated environment that takes place before the actual behaviour is executed in the real world.
Probe	A piece of code injected in the system/component/ able to notify the occurrence of an event

Risk assessment	The process of identifying, prioritizing, and estimating risks
Runtime	The time when system or system component executes in the real world (for example, a car driving on the streets)
Security Certification	Comprehensive evaluation of an information system component that establishes the extent to which a particular design and implementation meets a set of specified security requirements
Security Testing	The process to determine that an information system protects data and maintains functionality as intended
Software Smart Agent	An intelligent software component involved in the automation of processes within a system, system component or ecosystem.
Stub	A piece of code simulating a method/object interaction and response
Validation	A set of activities intended to ensure that a system or system component meets the operational needs of the user. The user in this sense can be an actor within the ecosystem, or another system or system components that receives its services.
Verification	A set of activities that checks whether a system or a system component meets its specifications.
Vulnerability	A weakness an adversary could take advantage of to compromise the confidentiality, availability, or integrity of a resource.
Weakness	Implementation flaws or security implications due to design choices.

Executive Summary

Modern ICT supply chains are complex, multidimensional and heterogeneous by nature, encompassing varied technologies, actors and interconnected resources. This makes it so that cybersecurity has become a major concern for such ecosystems, particularly given the tremendous velocity cybersecurity threats evolve requiring continuous monitoring, assessment and improvement of these ecosystems to assure their integrity and security.

In this regard, BIECO aims to deliver a holistic approach to building and validating methodologies and technologies tailored to foster security and trust within ICT ecosystems across their entire lifecycle, from design to runtime phases.

As such, the present deliverable provides the first draft of the overall framework architecture of BIECO. For this purpose, an initial description of each work package and their respective tools is provided, along with the expected actors, inputs, outputs, interactions and respective interfaces. This acts as the foundation for the first draft of the architecture, which is divided into two main views for different phases of the lifecycle, namely design time and runtime. This specification corresponds to one of the main outcomes of BIECO's first agile cycle. It is expected that this specification will be revised, extended and then finalized as the project evolves until M18 (using D2.3 as a reference point). At that time, the final version of the architecture will be documented in D2.4.

Project Summary

Nowadays most of the ICT solutions developed by companies require the integration or collaboration with other ICT components, which are typically developed by third parties. Even though this kind of procedures are key in order to maintain productivity and competitiveness, the fragmentation of the supply chain can pose a high risk regarding security, as in most of the cases there is no way to verify if these other solutions have vulnerabilities or if they have been built taking into account the best security practices.

In order to deal with these issues, it is important that companies make a change on their mindset, assuming an "untrusted by default" position. According to a recent study only 29% of IT business know that their ecosystem partners are compliant and resilient with regard to security. However, cybersecurity attacks have a high economic impact and it is not enough to rely only on trust. ICT components need to be able to provide verifiable guarantees regarding their security and privacy properties. It is also imperative to detect more accurately vulnerabilities from ICT components and understand how they can propagate over the supply chain and impact on ICT ecosystems. However, it is well known that most of the vulnerabilities can remain undetected for years, so it is necessary to provide advanced tools for guaranteeing resilience and also better mitigation strategies, as cybersecurity incidents will happen. Finally, it is necessary to expand the horizons of the current risk assessment and auditing processes, taking into account a much wider threat landscape. BIECO is a holistic framework that will provide these mechanisms in order to help companies to understand and manage the cybersecurity risks and threats they are subject to when they become part of the ICT supply chain. The framework, composed by a set of tools and methodologies, will address the challenges related to vulnerability management, resilience, and auditing of complex systems.

Partners

Disclaimer

The publication reflects only the author´s view and the European Commission is not responsible for any use that may be made of the information it contains.

Table of Contents

List of Figures

List of Tables

1. Introduction

With progressing digitalization and the trend towards autonomous computing, systems tend to form digital ecosystems, where each participant (system or actor) implements its own operational goals. Systems operating within ecosystems can deploy smart agents in the form of software applications, which would enable cooperative behaviour with other ecosystem participants, and achievement of common tactical and strategic goals. Effective collaboration within these emerging digital ecosystems strongly relies on the assumption that all components of the ecosystem operate as expected, and a level of trust among them is established based on that. In BIECO we design mechanisms that ensure the collaboration between ecosystem participants remains trustworthy in case of failures. By making systems resilient in face of malicious attacks, a trustworthy behaviour is always displayed to the user (which can be an interacting service or a human user).

The key difference between digital ecosystems and systems of systems is that digital ecosystems involve actors with goals, which significantly influences the dynamics within an ecosystem. In cooperation, the actors might have not only collaborative goals, but also competitive goals, which may influence the health of the ecosystem. In digital ecosystems, where hardware and software components of cyber-physical systems are provided by different actors, malicious behaviour can be introduced along with software components by actors who join a smart ecosystem based on declared collaborative goals, but who are actually acting in competition.

Typically, the admission to a digital ecosystem has been based on the actors' commitment to published roadmaps organized and provided by an ecosystem orchestrator for the long term. Emerging digital ecosystems, however, are particularly faced with the challenge of intended malicious behaviour which may be hidden in the smart agents. As a consequence, besides being functionally correct, a trusted digital ecosystem also needs to assess the participants' trustworthiness before granting them admission. Assessing the trustworthiness of ecosystem participants requires new platforms that enable behaviour evaluation at runtime, with this being one of the main goals of BIECO.

Thus, this deliverable aims to provide the initial set of guidelines and specifications concerning the design and implementation of the BIECO solution for improving the resilience and trustworthiness of digital ecosystems, building on the requirements derived in D2.1 and the ongoing work of T2.2 concerning the use case specification. These will be provided as the first draft of the BIECO overall framework architecture, which will be later revised, completed and finalized in D2.4, due in M18.

The remainder of this document is structured as follows: Section 2 presents a recap of the BIECO concept, serving as the conceptual context for this deliverable. Following this, each of the framework's components (organized by WP) is detailed in Section 3, including an initial description of their role and internal architecture, along with its main interactions and planned interfaces. This acts as the foundation for the first draft of the BIECO architecture, which is presented in Section 4. This specification encompasses different phases of the ICT lifecycle, from design to runtime, with each being depicted in its own view. Finally, Section 5 concludes the document with a brief summary of the main outcomes and future outlook.

2. BIECO Concept

The rationale behind BIECO's concept is to deliver a framework for improving trust and security within ICT supply chains. These are complex ecosystems comprising several heterogeneous technologies, processes, actors (e.g., end-users, software or hardware providers and organizations) and resources, all of which generate or exchange data forming extremely complex information management systems.

Due to this, cybersecurity and integrity are particularly important aspects to take into account in this context, which need to be addressed with an integrative approach that contemplates the entire chain, as opposed to restraining it only to the individual components.

In this direction, BIECO aims to deliver a holistic approach to building and validating methodologies and technologies tailored to foster security and trust within ICT ecosystems. The general concept of BIECO's framework is depicted in Figure 1.

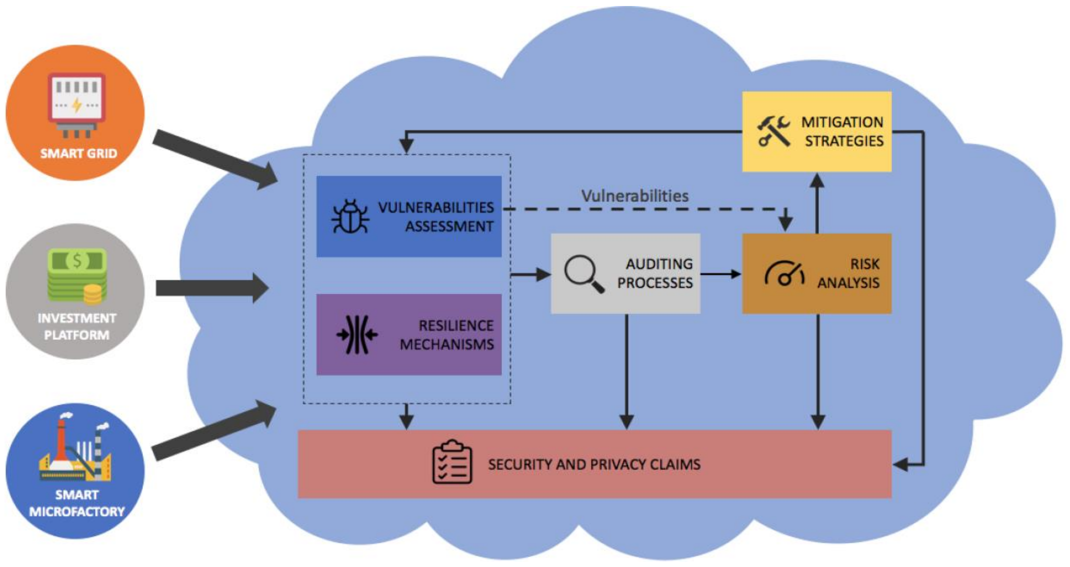


Figure 1 - Overall concept of the BIECO Framework to foster security and trust in ICT ecosystems.

The goal is to instantiate the framework iteratively in order to enable a continuous assessment and improvement of ICT supply chain's security, given the speed at which the cybersecurity landscape evolves with new threats emerging every day. As shown, the methodologies and tools developed or adapted in this context will be evaluated in three use cases from different sectors, namely smart grid / energy, financial and manufacturing industry sectors.

To better illustrate how BIECO intends to address these challenges along the entire lifecycle of the ICT supply chain, Figure 2 shows broadly the interaction flow between the different phases of the lifecycle, as well as the core functionalities involved.

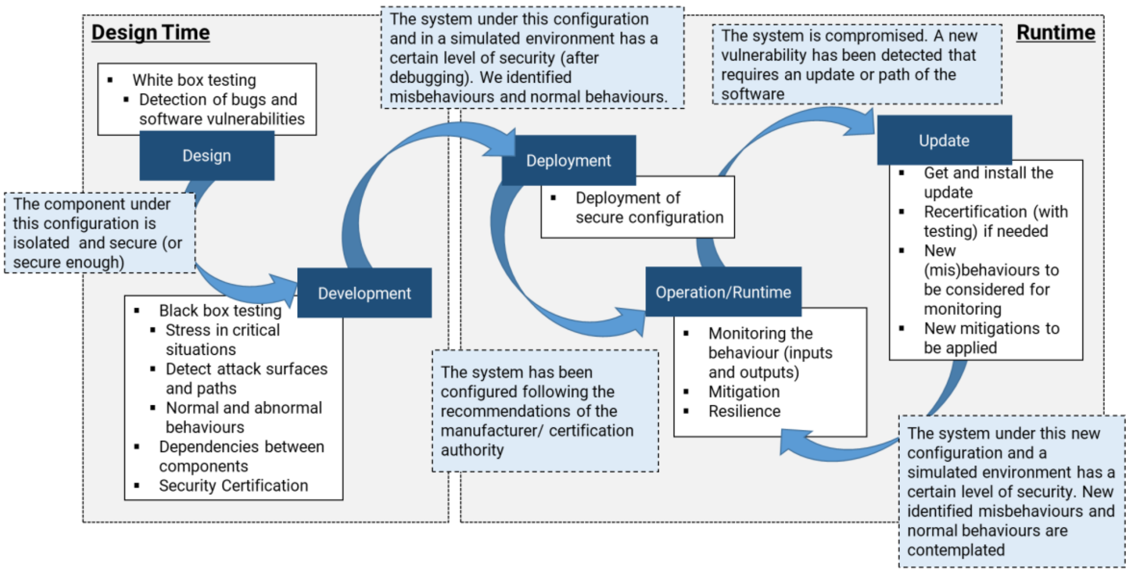

Design Time
White box testing
Detection of bugs and software vulnerabilities
Design
The component under this configuration is isolated and secure (or secure enough)
Development
Black box testing
Stress in critical situations
Detect attack surfaces and paths
Normal and abnormal behaviours
Dependencies between components
Security Certification
The system under this configuration and in a simulated environment has a certain level of security (after debugging). We identified misbehaviours and normal behaviours.
Deployment
Deployment of secure configuration
The system has been configured following the recommendations of the manufacturer/ certification authority
The system is compromised. A new vulnerability has been detected that requires an update or path of the software
Runtime
Update
Get and install the update
Recertification (with testing) if needed
New (mis)behaviours to be considered for monitoring
New mitigations to be applied
Operation/Runtime
Monitoring the behaviour (inputs and outputs)
Mitigation
Resilience
The system under this new configuration and a simulated environment has a certain level of security. New identified misbehaviours and normal behaviours are contemplated

Figure 2 – General interaction flow across the phases of the lifecycle

To realize this vision, BIECO's architecture will thus contain a set of interoperable tools and methodologies capable of ultimately ensuring the trustworthy execution of systems and system components within complex digital ecosystems. From design to runtime, vulnerabilities and failures are detected, evaluated, and mitigated together with prompt reactions that ensure the ultimate trustworthy execution of systems and system components. In order to open the path towards future development and for enabling the possibility to keep up to agile technological progress supported by runtime updates of systems (including safety-critical systems), we further on design the BIECO architecture with expandability in mind. One possibility in this direction will be that based on detected deviations, runtime updates of systems can be accommodated through a natural extension of the BIECO framework, including the feedback of information to the design time for continuous improvement.

3. Components of the BIECO Framework

This section details each of the core functionalities contemplated in the BIECO framework, along with their respective tools divided by each of the WPs. The main purpose is to provide a foundation for the specification of the architecture and later on serve as a reference point for the development efforts within each WP.

3.1. Vulnerability Management (WP3)

Vulnerability assessment is an essential step of security management that allows to detect and analyse on an early stage the possible security flaws or bugs of a system. Even though the vulnerabilities can be addressed at any stage of a system lifecycle, their early identification helps to reduce the risks in general, minimizing the costs as well as the probability of a later exploitation by an attacker.

In general terms, vulnerabilities can be associated to software, hardware, policies or even to the behaviour of users (intended or unintended). In particular, in BIECO the vulnerability assessment process is focused on the identification of vulnerabilities that appear in the source code during the design time of a software, and that could end having an impact on the confidentiality, integrity or availability of the system. This vulnerability assessment is not only focused on the detection of software vulnerabilities, but it aims also at analysing the possible long-term impact that the identified vulnerabilities could have, taking into consideration aspects such as the period of time under which they might be exploited, or how they could propagate to other components of the software supply chain. It is important to highlight that the vulnerability assessment performed in WP3 is focused only on design time, taking into account a static view of the source code, and it does not analyse other aspects associated to the execution or behaviour of the software during runtime, which are considered within other BIECO work packages (WP4, WP5, WP6).

3.1.1. General Description

The tools of WP3 will be developed mainly as REST services and deployed as part of the BIECO platform. These tools will rely mainly on the use of Machine Learning techniques and their purpose is to support different aspects of the assessment process of a software vulnerability, including its identification in a piece of source code, the evolution of its exploitability over time and its possible impact on other parts of the software (or even other systems that integrate the software under analysis, as part of a supply chain).

The tools that will be developed in the work package are:

- Data collection and pre-processing: REST service that collects data directly from BIECO pilots and from public data sources. The resulting datasets are delivered to the internal stakeholders using a REST API. The specifications and examples are delivered using the Postman application.
- Vulnerability detection: a tool that enables the identification of software vulnerabilities in source code. The tool, based on Machine Learning and data mining techniques, considers the use of privacy preserving techniques such as Federated Learning, in order to improve its accuracy and ensure confidentiality for the non-public training datasets (e.g., the source code from use cases).
- Forecasting: it will follow two different approaches.

- Exploitability forecasting: the tool allows, taking as an input a vulnerability detected in a certain piece of source code, to predict the time frame in which the vulnerability could be exploited.
- Vulnerability forecasting: RESTful web service that, based on the dataset delivered by the API developed in T3.2, allows to forecast the number of vulnerabilities that will appear in a time frame.

- Vulnerability propagation: tool developed also as a REST service that provides a visual representation on how a certain vulnerability can propagate across the source code of one or more software modules. The tool provides a graph-based representation, as an intuitive manner of visualizing the different propagation paths of the vulnerabilities.

As far as possible, the tools will be deployed by means of containers technologies such as Docker[1]. This technology allows easily to create, test, deploy and scale applications. The containers include all the libraries and dependencies that the tools require in order to be executed, and they avoid issues with specific dependencies of the versions of operating systems.

3.1.2. Actors

The actors of WP3 tools will be in principle software developers/owners (e.g., the use cases providers) who are interested in determining if their source code contains any vulnerabilities and, in case they exist, understand to which extent they could have an impact on their systems, and even business processes. In order to ease the interaction with the tools, a GUI (Graphic User Interface) will be provided to the actors.

3.1.3. Interactions

The Figure 3 presents the interactions of the tools provided by WP3 with the rest of tasks and work packages in BIECO.

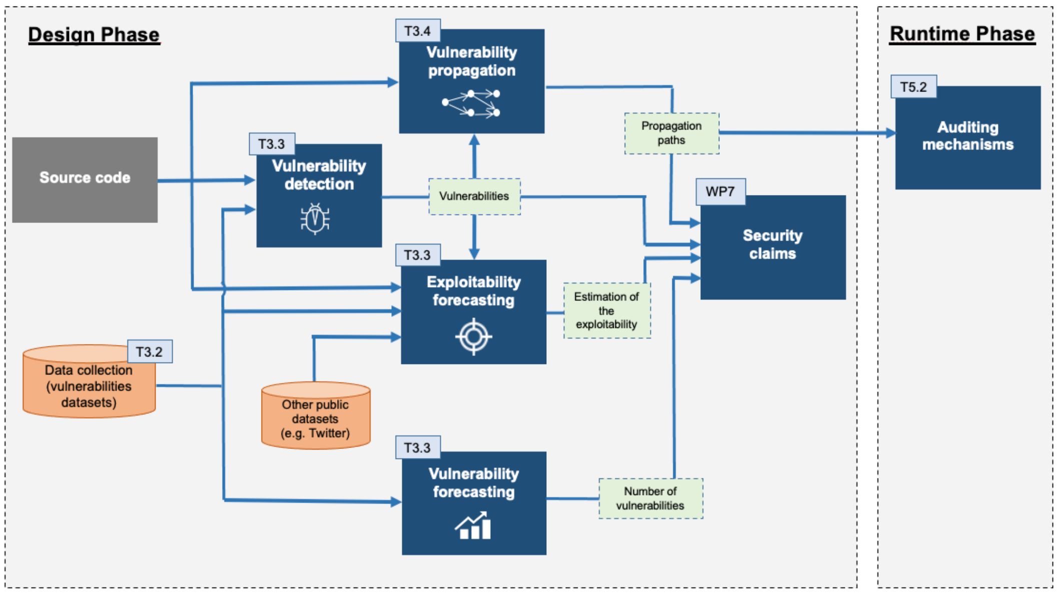

Design Phase
Runtime Phase
T3.4
Vulnerability propagation
Source code
T3.3
Vulnerability detection
Propagation paths
T5.2
Auditing mechanisms
Vulnerabilities
WP7
Security claims
T3.3
Exploitability forecasting
Estimation of the exploitability
T3.2
Data collection (vulnerabilities datasets)
Other public datasets (e.g. Twitter)
T3.3
Vulnerability forecasting
Number of vulnerabilities

Figure 3 - Interrelations among WP3 tools and other BIECO elements.

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

The vulnerability assessment process can begin through the instantiation of two alternative components: i) the vulnerability forecasting tool from T3.3 that allows to forecast the number of vulnerabilities that will appear in a certain period of time, taking into account just information from vulnerability datasets; or ii) the vulnerability detection tool developed also in T3.3, which takes as an input the source code to analyse and public information from the vulnerabilities (such as its type or the CVE). The vulnerabilities detected in this step can be further assessed by means of two additional tools developed within the work package: the exploitability forecasting tool from T3.3 that allows to determine in which period of time the detected vulnerability might be exploited, and the vulnerability propagation tool from T3.4 that allows to model how the detected vulnerability can impact other dependant software modules.

The following table presents a summary of the tools provided by WP3, including their inputs, outputs, and the expected users.

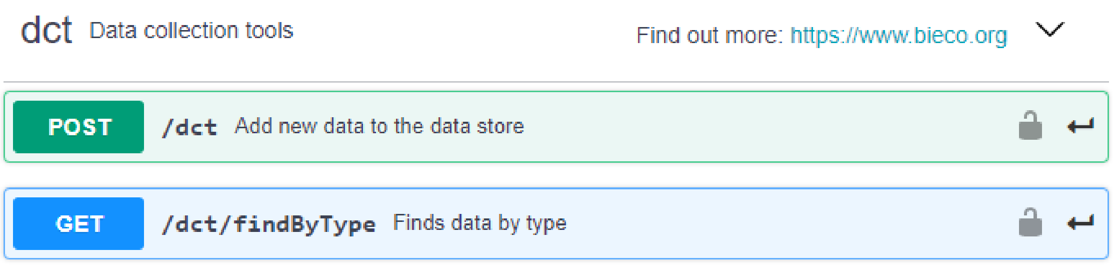
Table 1 – Interaction mapping for WP3

Tool	Functionality	Lifecycle	Inputs	Outputs	User
Data collection and pre-processing [T3.2 UTC]	Data set of vulnerabilities with filtering capabilities	Design time	• Public datasets • Internal datasets	JSON files	User interface
Vulnerability detection [T3.3, GRAD]	Machine learning and data mining-based tool that allows to identify software vulnerabilities in source code	Design time	• Source code, • Vulnerabilities datasets	• List of the vulnerabilities identified in the input source code	• Use cases. • Vulnerability propagation tool • Exploitability forecasting tool • WP7
Exploitability forecasting [T3.3, GRAD]	Tool for analysing the exploitability of a software vulnerability in a future time frame.	Design time	• Vulnerability to be analysed. • Source code • Vulnerability datasets • Other public datasets (e.g., Twitter) • Time window (6/12 months)	Probability of a certain vulnerability to be exploited in an indicated period of time	• Use cases. • WP7
Vulnerability forecasting [T3.3, UTC]	Forecast for the number of vulnerabilities	Design time	• Data set from T3.2	• Number of vulnerabilities that will occur in a certain time interval.	• Use cases. • WP7
Vulnerability propagation [T3.4, GRAD]	Tool for modelling how a vulnerability in a piece of source code (e.g., a library) can propagate across one or more systems.	Design time	• Source code • List of the vulnerabilities to be analysed	• A structured file (e.g., JSON) containing information of the propagation path of a certain vulnerability.	• Use cases. • T5.2 • WP7

It is important to note that since WP3 requires source code as an input, privacy-preserving mechanisms will be explored to protect sensitive data and intellectual property. This could include the use of federated learning, differential privacy, or secure multi-party computation schemes.

3.1.4. Planned Interfaces

This section details the planned interfaces for the five tools that will be provided by WP3. Each of these tools will be developed as a REST service. The concrete details of their interfaces, request parameters and results are presented below.


dct Data collection tools

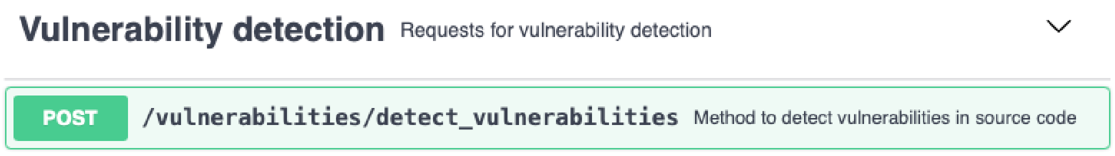
Find out more: https://www.bieco.org

POST /dct Add new data to the data store

GET /dct/findByType Finds data by type

Figure 4– Data Collection Interface

The data collection tool will gather information mainly from the National Vulnerability Dataset[2] and from the use case providers. The tool will be implemented as a REST service and it will provide information through its API. The API will support the following requests: GET vulnerability data and POST vulnerability data. The POST vulnerability data request will be used by the use case providers to store data in the tool database. The details of the vulnerabilities will be provided in JSON format. The GET vulnerability data request will provide information about the relevant resources gathered from the dataset (for example the CPE URI, the attack vector, the time interval, and the required fields) and the service will provide the result in JSON format.


Vulnerability detection Requests for vulnerability detection

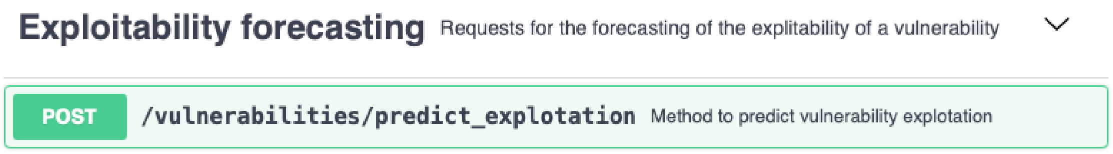
POST /vulnerabilities/detect_vulnerabilities Method to detect vulnerabilities in source code

Figure 5– Vulnerability Detection Interface

The vulnerability detection tool will provide a single method through its API, *detect_vulnerabilities*. The method will support POST requests including two parameters: a zip file with the source code to be analysed, and a string value detailing the type of programming language in which the source code was developed.

The result of the request will be a file in a structured format, such as JSON, providing at least, for each of the identified vulnerabilities, the name of the file that contains the vulnerability, its approximated location (i.e., the line number) and its type, as well as other possible additional parameters.

[2] https://nvd.nist.gov

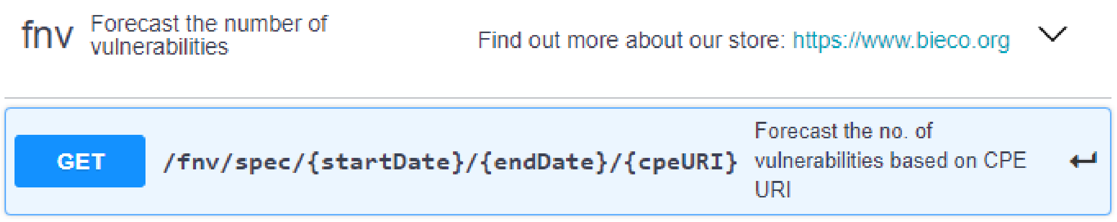

Exploitability forecasting Requests for the forecasting of the explitability of a vulnerability

POST /vulnerabilities/predict_explotation Method to predict vulnerability explotation

Figure 6– Exploitability Forecasting Interface

The tool that allows to predict if a certain vulnerability will be exploited within the next months, will expose a single method through its interface, *predict_exploitation*. This method will require as an input four parameters: one of the vulnerabilities objects returned by the method *detect_vulnerabilities*, a zip file with the source code in which the vulnerability appears, a string value indicating the programming language, and the time window (6 or 12 months) in which the vulnerability exploitability will be taken into consideration.

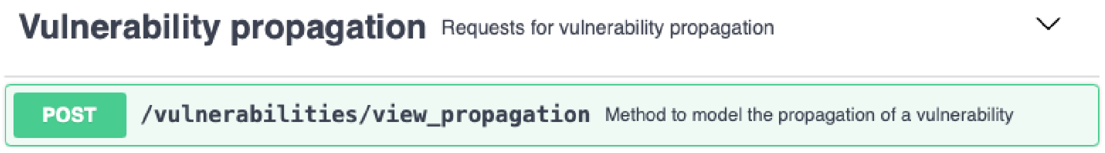
The result of calling this method will be a value indicating the probability of the vulnerability to be exploited within the time period indicated in the input request.


fnv Forecast the number of vulnerabilities Find out more about our store: https://www.bieco.org

GET /fnv/spec/{startDate}/{endDate}/{cpeURI} Forecast the no. of vulnerabilities based on CPE URI

Figure 7– Vulnerability Forecasting Interface

The tool that will forecast the number of vulnerabilities will take input from the T3.2 dataset, which will contain vulnerability information from the National Vulnerability Database and the use case providers. The API of the tool will support a GET forecast request. The parameters of the request will be the CPE URI of the investigated component and the time interval. The result will be the number of vulnerabilities that will be reported for that component within the specified time frame.


Vulnerability propagation Requests for vulnerability propagation

POST /vulnerabilities/view_propagation Method to model the propagation of a vulnerability

Figure 8 – Vulnerability propagation Interface

The propagation tool will provide a method, *view_propagation*, that allows to analyse how a certain vulnerability can affect other dependent software modules. The request will receive a zip file containing the source code, a string value indicating its programming language, an object (returned by the method *detect_vulnerabilities*) containing information of the concrete vulnerability under analysis, as well as other optional parameters, in order to configure the propagation graph.

The result of the request will be a structured file, following a format such as JSON, that will include information of the propagation graph of the vulnerability.

3.2. Development of Resilient Systems (WP4)

In face of security attacks that exploit runtime residual vulnerabilities, mechanisms for assuring resilience of systems in case of runtime failures are necessary for two reasons. Firstly, collaboration within a digital ecosystem is necessary in order to assure an ultimate trustworthy behaviour of systems and system components. Secondarily, system resilience in case of failures is necessary for gaining the ultimate user trust and hence, improve the acceptability rate of systems and services.

3.2.1. General Description

Resilient systems are those systems capable to react and recover very fast from disturbances. Disturbances during operation are typically generated by undiscovered faults that make their way from design time to runtime. In terms of security, attackers can exploit such undiscovered vulnerabilities to attack the system operation at runtime. Within complex dynamic ecosystems in particular, systems, system components and a variety of actors interact with each other in new contextual situations that may not be completely foreseen during design time. This complex dynamism opens the path to a multitude of vulnerability exploitation with an end effect of reaching untrusted runtime execution. In particular, the need for speed on which many organizations operate towards deploying systems may inadvertently allow vulnerabilities exploitable by attackers to be present in deployed systems. For example, a software component deployed on a system with the goal of enhancing an existing functionality can contain intended malicious faults that express in malicious behaviour in key situations when the target impact is likely to be achieved. In order to assure the ultimate trustworthiness of a system as well as a trusted interaction with other users, mechanisms for making the system resilient in face of runtime failures are developed within WP4. In particular:

1) **Methods and tools for self-checking of vulnerabilities and failures** with the focus on supporting software in the capabilities to perform self-checks on the target system on which it is executed. In this way, vulnerabilities that have not been discovered during design time are discovered during runtime execution through the execution of scanners. Besides residual vulnerabilities, software failures as well as random hardware failures are detected by performing periodic checks and on-demand checks that rely on self-test libraries and software safety mechanisms that perform tests on the instructions of microcontrollers. Then,
2) **Methods and tools development for failure prediction** are developed for enabling the forecast of failures within a system, with a specific focus on software components which are the ecosystem entities that can contain malicious behaviour. Within digital ecosystems, systems and system components engage in collaborations according to a set of demands and guarantees that specify the functional and the non-functional behaviour that collaborators can expect from each other.

   Building on development from the two above mentioned activities,
3) **Methods for assuring system resilience** are developed in order to bring a system into a safe-operational state, in case of malicious attacks that make their way through the runtime. In this regard, strategies for making a system adaptive to predicted failures are designed accounting for redundancy in operation. For example, for assuring system's safety in case of predicted failures caused by security attacks,

simplex architectures designed around controlling a complex function through a redundant, simpler channel are explored.

3.2.2. Actors

Actors who interact with the tools developed and extended within WP4 include:

1) engineers involved in the process of system development.

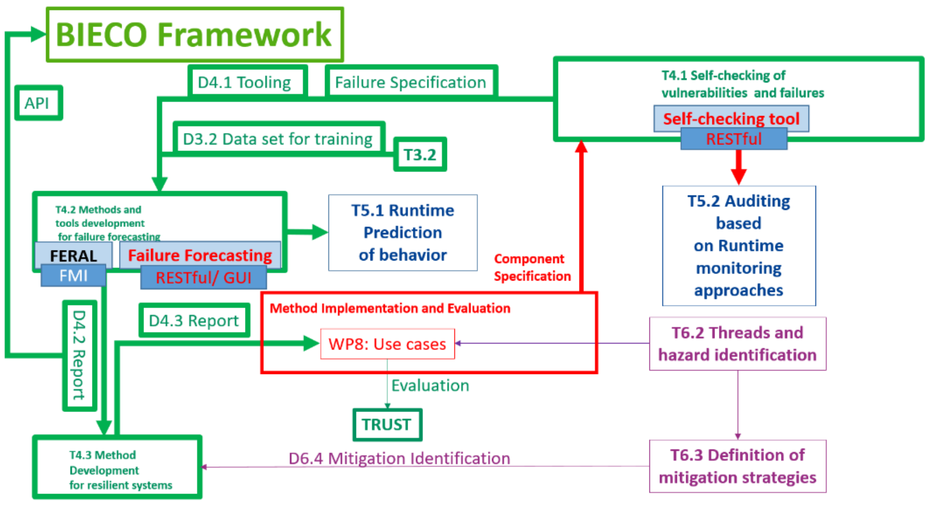
2) The actors of the failures forecasting tool developed in T4.2 are users and developers.

3) developers of software applications and manufacturers of hardware resources.

Actors who interact with digital assets which are systems under evaluation for the tool. They will be able to estimate the failure rates of certain components. For easy operation, a GUI will be provided for the tool.

3.2.3. Interactions

An overview of the interactions between the tools provided by WP4 and the remaining WPs of BIECO is presented in Figure 9.



BIECO Framework
API
D4.1 Tooling
Failure Specification
T4.1 Self-checking of vulnerabilities and failures
Self-checking tool
RESTful
D3.2 Data set for training
T3.2
T4.2 Methods and tools development for failure forecasting
FERAL
FMI
Failure Forecasting
RESTful/ GUI
T5.1 Runtime Prediction of behavior
Component Specification
T5.2 Auditing based on Runtime monitoring approaches
D4.2 Report
D4.3 Report
Method Implementation and Evaluation
WP8: Use cases
T6.2 Threads and hazard identification
Evaluation
TRUST
T4.3 Method Development for resilient systems
D6.4 Mitigation Identification
T6.3 Definition of mitigation strategies

Figure 9 - Overview of the interaction between WP4 and the remaining WPs

A detailed description for each of the tools encompassed in WP4, including their inputs, outputs and intended users is provided in Table 2.

Table 2 - Interaction mapping for WP4

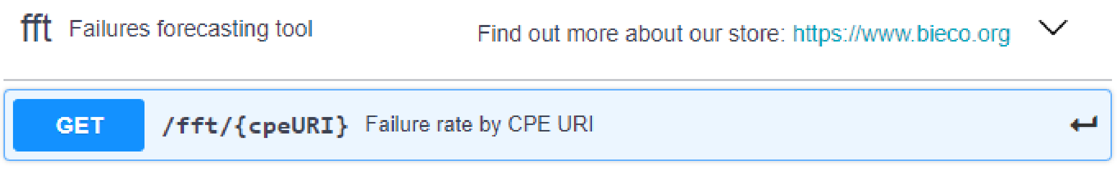
Tool	Functionality	Lifecycle	Inputs	Outputs	User
Methodologies (and later software solutions) for periodic self-checking of HW/SW failures [T4.1, RES]	Periodic software tests on HW/SW components. Check for HW failures (e.g., with Self-Test Libraries), and SW	Runtime	• (for SW failures) Data stream to be monitored, signature of the SW execution.	• (for both HW/SW failures) Boolean output (on the correct functioning of HW features / correctness of the SW control flow)	T4.2 T5.2

Tool	Functionality	Lifecycle	Inputs	Outputs	User
	failures (e.g., with control flow monitoring)				
Co-Simulation tool [T4.2, T4.3, IESE]	Behaviour specification of systems and system components can be integrated into a co-simulation framework that enables their execution.	Execution time	• Sim Results from WP5 • T4.2, T5.1, T6.3 • Specifications from WP5 + WP6 based on WP7 – MUD Files Behavioural profile • Architectural structure of the use case • Interactions (MUD-compliant) between WP5-WP6 and WP4	• Activation of fail-over behaviour to return the system to operational state when faults are predicted. • Functional mock-up units for interconnections with other (simulation) tools	• Use cases (functional mock-up units) • T5.1
Forecasting systems failures [T4.2, UTC]	Tool for predictive virtual evaluation of software components that enter the ecosystem	Runtime	• D4.1 Self checking tool • D3.2 Dataset	• Forecast for the failure rates of components	• Use cases • T4.3 • T5.1

3.2.4. Planned Interfaces

1) Active MQ is one of the most popular Java-based multi-protocol communication protocol, built on top of JMS (Java Messaging Service). In order to enable communication between components developed in possibly different multiple languages, ActiveMQ will be used as a communication bridge.

2) The FMI (Functional Mock-up Interface) standard will be used. This interface allows the construction of complex co-simulation environments on a functional level. Using FMI, models are coupled by means of numerical solution methods in order to realize a cross-domain system simulation. This usually includes functional simulation models that represent the behaviour of components, physics and environment of a system, but not platform properties. The reason for this lies in the calculation and communication models (MOCC) used for the simulation. These break down the overall simulation into calculation steps and control the exchange of data between the simulation models. Only after a simulation tool has completed a calculation step are data and events exchanged.


fft Failures forecasting tool

Find out more about our store: https://www.bieco.org

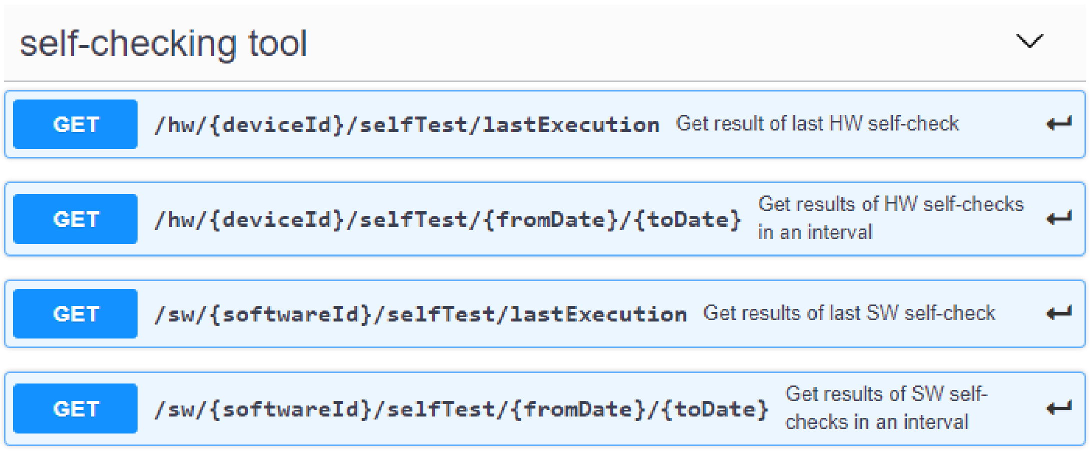
GET /fft/{cpeURI} Failure rate by CPE URI

Figure 10– Failures Forecasting Tool

The failures T4.2 forecasting tool will take input from the T4.1 self-checking module and from the T3.2 dataset. The API of the tool will support a Get forecast request. The parameter of the request will be the CPE URI of the investigated component. The result will be the estimated failure rate of the component.

The Self-checking software tool devised in the context of T4.1 will periodically communicate the results of the application of methodologies for self-checking of HW/SW failures through REST API. The output in case of self-tests for detecting HW failures will be a Boolean result (true or false value according to the result of detection) while the checks for SW failures will take in input the data flow and a signature of SW execution from the use cases and will produce a JSON with the result which integrates mechanisms for data acceptance.

A list of REST services for the self-checking tool is given in the Figure 11.



self-checking tool

GET /hw/{deviceId}/selfTest/lastExecution Get result of last HW self-check

GET /hw/{deviceId}/selfTest/{fromDate}/{toDate} Get results of HW self-checks in an interval

GET /sw/{softwareId}/selfTest/lastExecution Get results of last SW self-check

GET /sw/{softwareId}/selfTest/{fromDate}/{toDate} Get results of SW self-checks in an interval

Figure 11– List of Rest Service for the Self- Checking Tool

3.3. Methods and Tools for Auditing ICT Ecosystems (WP5)

The methods and tool developed for auditing the ICT ecosystems focus on the evaluation of interactions within an ecosystem by means of simulation and runtime monitoring facilities. The auditing emulates and/or retrieves field usage data and provides useful feedback about intended behaviour from predictive simulation (T5.1) and real behaviour from a controlled environment or the real world (WP8). Features of this work package include:

- Definition of the executable simulation models based on digital twins (DT) technology according to use case specifications.
- Definition of the predictive simulations environment able to exploit the current state of systems and system components (with particular focus on software components) within the ecosystems so as to predict their behaviour in the future based on digital twin execution. This includes the computation of the parameters against which the behaviour of the ecosystem participants (systems or system components) and their interacting entities within an ecosystem will be judged as being trustworthy or not.
- Set up of the Definition of monitoring methodologies and tools detecting malicious behaviours of ICT components within the ecosystems and assessing the validity of the simulation models.

- Set up of the runtime monitoring tool to identify which simulation parameters can have a high and critical impact on the security properties of the (simulated) ecosystem components. In particular the monitoring activity will focus on:
    - Detection of suspicious interactions between ecosystem components, such as hardware/software components of the ecosystems.
    - Detection of behavioural changes as a response of changes in the environmental and operational conditions.
    - Identification of suspicious/malicious behaviours.

3.3.1. General Description

The Auditing component includes tree main sub-components as reported in Figure 12.

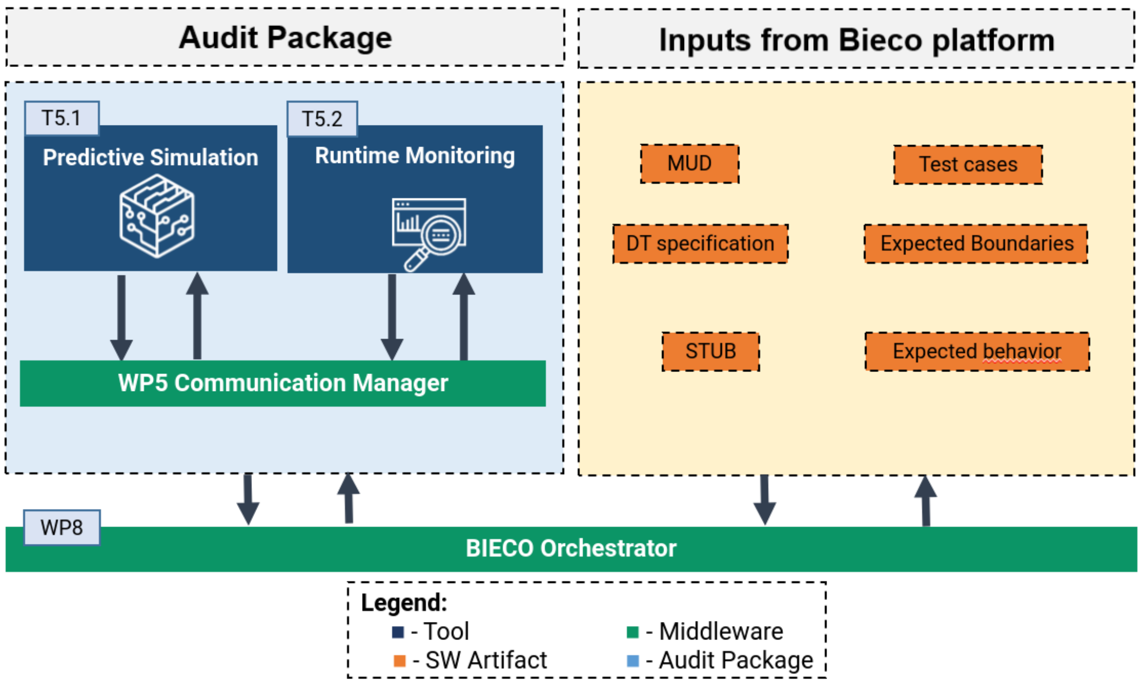


Figure 12 – Audit package description

A general description of the components is provided in the following subsections.

3.3.1.1. Predictive Simulation

This is in charge of setting up the predictive simulation environment based on digital twins (DT) technology. The digital twins are abstract models representing the executable abstractions of the ecosystem components (ICT systems, ICT system components such as software components and actors) and their interactions. Linked predictive simulations are used for the evaluation of the DT' behaviour. In the linked predictive simulation, the current state of the system is used to predict behaviour in the future.

For enabling detection of malicious behaviour hidden within software components, a Domain Specific Language (DSL) that enables definition of control functions will be developed.

The predictive simulation works in collaboration with the Monitoring Engine component. In particular monitoring data are used for sensitive analysis useful to identify which simulation parameters have a high and critical impact on the security properties as well as comparison between predicted behaviour and actual behaviour.

3.3.1.2. Runtime Monitoring

This is in charge of setting up and managing monitoring component. The Runtime Monitoring is based on event messages.

In particular it enables the collection of specific events that flows during controlled environment or real execution among the different entities (i.e., DT, sensors, ecosystem components and so on) and infers one or more complex events about the runtime execution (Complex Event Processing - CEP).

Complex events inference is based on a set of derived rules, i.e., "if-then-else" grammar expressions that define sequences of attended or un-attended events patterns.

Thus, Runtime Monitoring includes a set of generic rules templates (a meta-rules) that can be instantiated at runtime according to the scenario to be observed.

The events that trigger the execution of a rule are generated by a probe, i.e., a piece of code injected in the entities to be observed during the runtime execution able to notify the occurrence of the events to the Monitor Engine.

Specifically, the Runtime Monitoring is based on ActiveMQ messaging protocol[3] , but also exposes a REST interface. In this case REST messages are translated by the Monitoring Engine into ActiveMQ messages.

3.3.1.3. WP5 Communication Manager

Communication Manager is the entity in charge of orchestrating communication within the Auditing component. It is able to route the messages to/from the internal Auditing component. It exposes all the interfaces of the Auditing component to the BIECO Framework.

3.3.2. Actors

According to the Actor definition reported in this deliverable in WP5 there are no actors interacting with the WP5 components.

[3] https://activemq.apache.org/

3.3.3. Interactions

An overview of the interactions between the tools provided by WP5 and the remaining WPs of BIECO is presented in Figure 13.

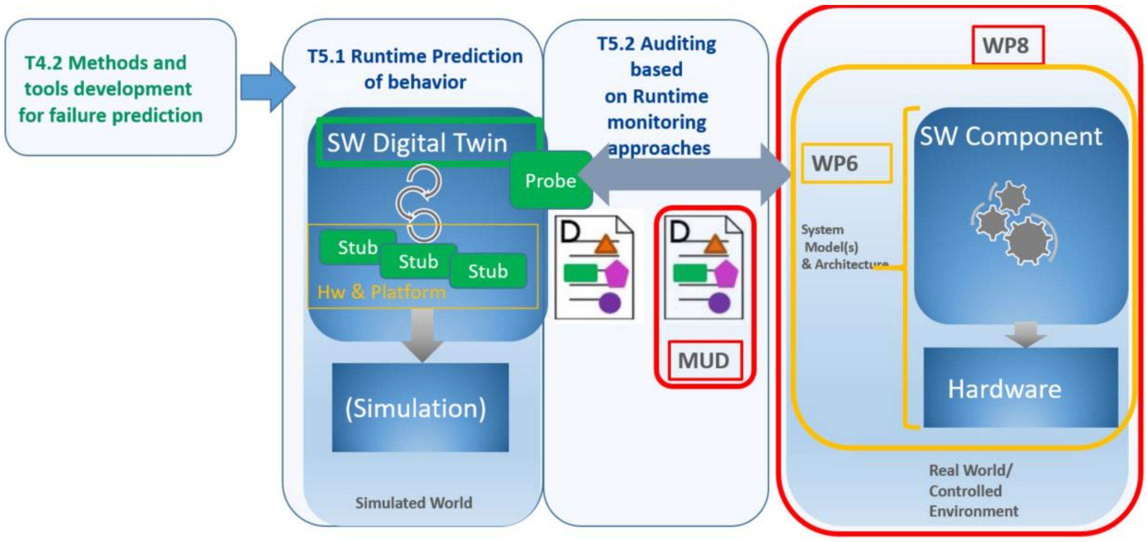

T4.2 Methods and tools development for failure prediction
T5.1 Runtime Prediction of behavior
SW Digital Twin
Probe
Stub
Stub
Stub
Hw & Platform
(Simulation)
Simulated World
T5.2 Auditing based on Runtime monitoring approaches
MUD
WP8
WP6
System Model(s) & Architecture
SW Component
Hardware
Real World/ Controlled Environment

Figure 13 - Overview of the interactions between WP5 and the remaining WPs.

A detailed description for each of the tools encompassed in WP5, including their inputs, outputs and intended users is provided in Table 3.

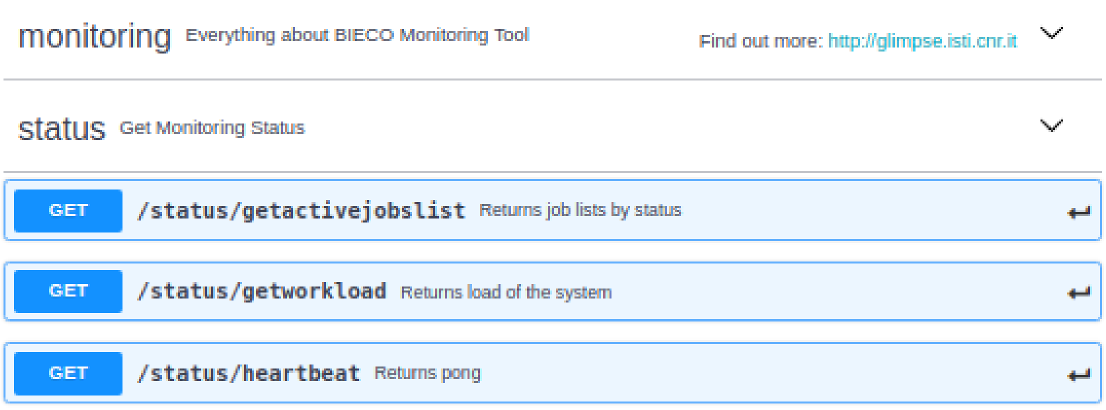
Table 3 - Interaction mapping for WP5

Tool	Functionality	Lifecycle	Inputs	Outputs	User
Predictive simulation [T5.1]	Perform predictive simulation though execution of Digital Twins. expressed in a DSL	Runtime	• Systems and system components specification Model of the system or components. • Outputs from the tools that can be simulated to test (e.g., data to simulate an attack, stress, boundary condition) (Source WP6/WP8) • Simulation tools from WP4, WP6 • MUD files as input for specifying allowed interactions	• Sequence of events	T5.2, WP4, WP8
Monitoring Tool [T5.2, CNR]	1.Data logging 2) Complex Event processing (CEP) 2.1) Functional and non-functional properties evaluation 2.2) Rule's violation notification	Runtime: 1. predictive simulation 2. controlled environment	• Model of the expected behaviour (BPMN,PetriNet or equivalent model): the sequence of events expected or possible paths in terms of sequences of events. Functional and non- functional properties to be monitored associated to the expected behavioural path on which they should be respected. Can be expressed using SLA. • Boundaries/ thresholds for events	• Monitoring results • Possible link to T 6.4 to record the logs to the Blockchain	WP4, 6

Tool	Functionality	Lifecycle	Inputs	Outputs	User
			• MUD files as input for specifying allowed interactions		

3.3.4. Planned Interfaces

The Runtime Monitoring will provide REST interfaces for enabling interaction within the BIECO platform. In the figures below first version of the exposed REST interfaces is proposed:



monitoring Everything about BIECO Monitoring Tool Find out more: http://glimpse.isti.cnr.it

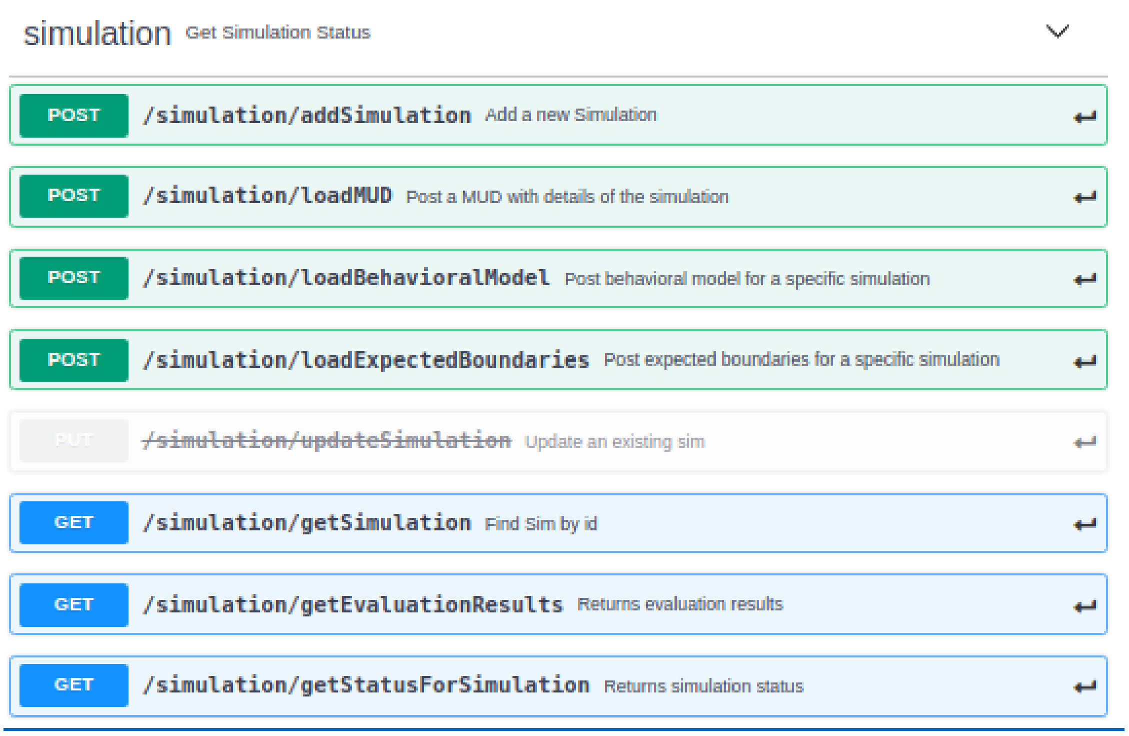
status Get Monitoring Status

GET /status/getactivejobslist Returns job lists by status

GET /status/getworkload Returns load of the system

GET /status/heartbeat Returns pong

Figure 14- Runtime Monitoring Interfaces (a)



simulation Get Simulation Status

POST /simulation/addSimulation Add a new Simulation

POST /simulation/loadMUD Post a MUD with details of the simulation

POST /simulation/loadBehavioralModel Post behavioral model for a specific simulation

POST /simulation/loadExpectedBoundaries Post expected boundaries for a specific simulation

PUT ~~/simulation/updateSimulation~~ Update an existing sim

GET /simulation/getSimulation Find Sim by id

GET /simulation/getEvaluationResults Returns evaluation results

GET /simulation/getStatusForSimulation Returns simulation status

Figure 15-Runtime Monitoring Interface(b)

3.4. Risk Analysis and Mitigation Strategies (WP6)

The methodology and supporting tools devised for analysing the risks in the ICT supply chain are focused on identifying, assessing, and mitigating threats since the early prototyping of a system.

The activities of this WP include:

1. Modelling of an ICT system and its architecture, starting from the identification of its main assets, components, and interfaces.
2. Identification of threats, attack patterns, weaknesses and vulnerabilities that apply to each asset and interface, including not only cyber threats, but also physical and cyber-physical ones that could emerge due to the interaction of the system with its environment.
3. Computation of severity of impact of the threats.
4. Determination of a likelihood of occurrence of the threats.
5. Derivation of a resulting risk for each threat.
6. Determination of attack paths by linking one or more related threats.
7. Simulation of behaviour and interactions between components when attacks are exploited.
8. Definition of countermeasures and controls to mitigate the identified risks.
9. Design of security, privacy and accountability measures.

3.4.1. General Description

The methods leverage three main tools, which are described in the following subsections.

3.4.1.1. Blockly4SoS

Blockly4SoS, output of the AMADEOS project (FP7-ICT-610535)[1], is a tool for modelling, validating and simulating Cyber-Physical System-of-Systems that leverages Google Blockly library. With the tool, the behaviour of each block can be modelled and simulated in a python environment, following the interactions described in a sequence diagram. Blockly4SoS, by integrating a System-of-System (SoS) SysMLprofile is aimed to provide a simple and intuitive interface to model a SoS with minimal training to the designer.

In the context of BIECO, and in particular within Task T6.1, the tool is first being refactored in order to allow the user to create new meta-models, that might better fit on their interests and on the application domain of a specific ICT system. The meta-models are abstractions that can be instantiated in multiple models and that are continuously validated by construction.

The tool will then be extended with new features that will enable threat modelling, risk assessment, and visual representation of attack paths. In addition, it will be investigated the possibility to generate with the tool an extended version of the MUD file for specification of behaviour of components.

Task T6.2 will then apply the extended functionalities of Blockly4SoS to model the use case systems, and in particular their threats, enabling the analysis of attack paths and the rating of risks according to a risk assessment process derived from the reference standards.

3.4.1.2. SafeTbox

SafeTBox [2] is a tool that supports engineers in the assessment process of a given system's safety. During the development of safety-critical systems it is essential to guarantee the functional safety. In the process of assuring safety, different analyses and development artifacts must be created according to various standard specifications such as: IEC 61508, ISO 26262, ISO 13849. As system complexity evolves, important system properties such as maintainability and traceability need to be guaranteed in order to avoid problems linked to efficiency and quality, which in the best case only costs money, but in the worst case can cost human lives and an ultimate loss of trust in the systems and their provided services.

Through model-based systems and safety engineering techniques, the common activities necessary in the context of systematic safety engineering, such as hazard analysis and risk assessment, safety analysis and development of safety concepts including mitigation strategies and the synthesis of a safety case are integrated with systems engineering techniques. For this, SafeTbox offers a modelling technique for the specification of the system architecture that permits assigning failure models directly to system artifacts, guaranteeing traceability. In addition to linking failure models and architecture, SafeTbox permits the creation of dynamic links between all development artifacts. These links make it possible to easily find referenced elements as well as to navigate to these. The modelling techniques integrated in SafeTbox have been developed in accordance with the concept of modularization (Component Fault Trees, system components and system functions) in order to support easy replacement of components, increase of maintainability, and efficient reuse of systems or system components in new contexts.

As part of the work provided over the course of BIECO, IESE will enhance safeTbox's interface, to support increased interoperability with the other BIECO tools. The enhancement will be centred around the provision of web-based APIs to other tools, enabling access to the tool's models.

3.4.1.3. Blockchain-based Accountability

Blockchain-based Accountability will be the new tool dedicated to achieving non-reputability of audit logs. Due to the use of blockchain technology, it would be possible to detect changes in audit logs which happened after the original log had been stored and secured with blockchain hash. Detailed architecture and used technology will be prepared within task T6.4. Initially, we assume usage of the Ethereum-based hash codes. After calculating hash code using SHA-2 or SHA-3 algorithm for a given log file revision, the hash code will be stored in Ethereum, with the given date and timestamp. Thanks to that it will be possible to detect any changes in the log file which happened after generating and storing the hash in the Ethereum ledger. The hash code stored in the Ethereum ledger should be the same as the one calculated on the actual file. Differences mean that a log file has been tampered with after storing the hash code in the Ethereum ledger.

The tool will work according to the flow presented below:

1. The audit file is generated and stored (e.g., on the local file system).
2. The Blockchain-based Accountability calculates the hash code for this file.
3. The hash code is sent to Ethereum network with a given date and timestamp.

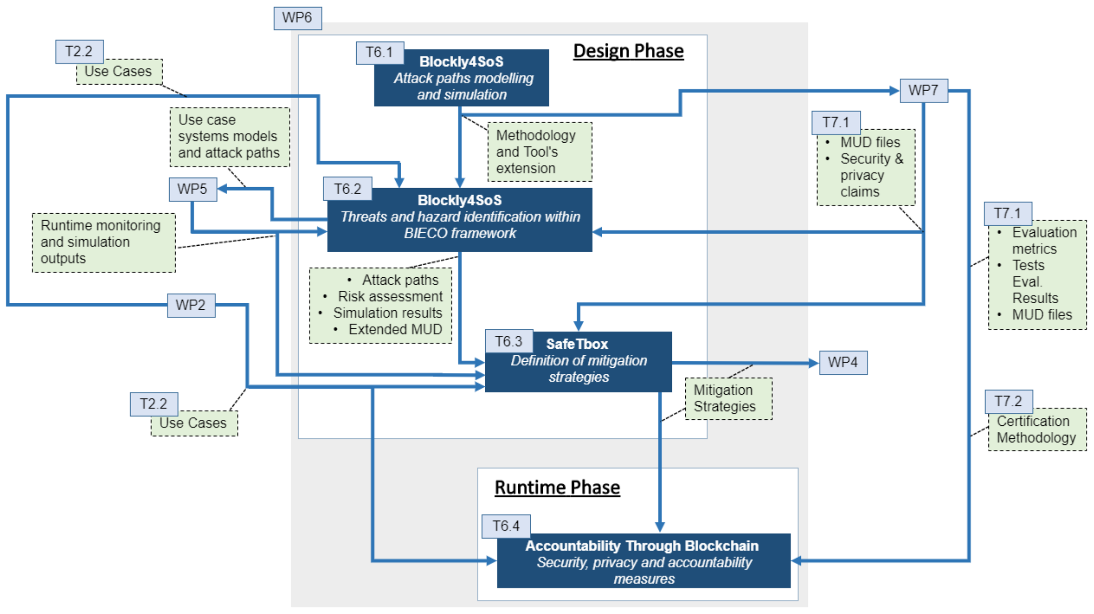
Thanks to that, even in the case of an attacker obtaining the super user privileges on the given server, it would not be possible to modify log files, e.g., by removing some operations in an unnoticeable way. The change in the log file will change the hash of the file and cause a difference between the actual hash and the hash stored in the Ethereum ledger.

3.4.2. Actors

Possible actors are companies and personnel which, adopting the methodologies and interacting with the software tools object of this WP, may perform risk analysis of their own system or of a system owned by a third party (i.e., a company providing assessment services).

3.4.3. Interactions

In Figure 16, a summary of the main interactions between WP6 and the remaining WPs of BIECO is provided.



WP6
Design Phase
T6.1
Blockly4SoS
Attack paths modelling and simulation
T2.2
Use Cases
Use case systems models and attack paths
WP5
Runtime monitoring and simulation outputs
T6.2
Blockly4SoS
Threats and hazard identification within BIECO framework
Methodology and Tool's extension
WP7
T7.1
MUD files
Security & privacy claims
T7.1
Evaluation metrics
Tests
Eval. Results
MUD files
Attack paths
Risk assessment
Simulation results
Extended MUD
WP2
T6.3
SafeTbox
Definition of mitigation strategies
WP4
Mitigation Strategies
T2.2
Use Cases
T7.2
Certification Methodology
Runtime Phase
T6.4
Accountability Through Blockchain
Security, privacy and accountability measures

Figure 16 - Overview of the interactions between WP6 and the remaining WPs.

A detailed description for each of the tools encompassed in WP6, including their inputs, outputs and intended users is provided in Table 4.

Table 4 - Interaction mapping for WP6

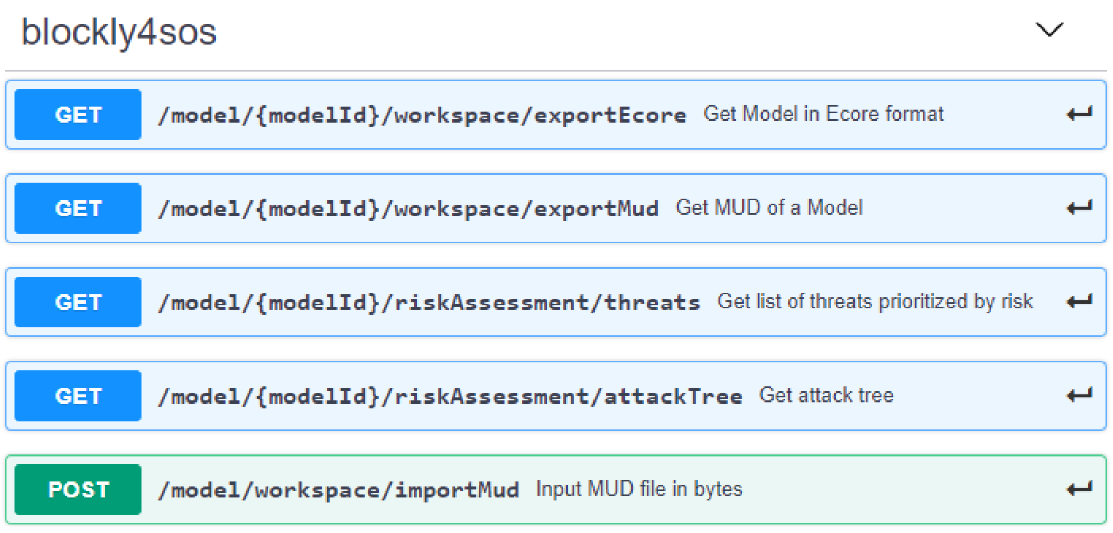
Tool	Functionality	Lifecycle	Inputs	Outputs	User
Blockly4SoS [T6.1, T6.2 RES]	Model-based threat and risk analysis of systems.	Design time	• WP2 Architecture of the system within the use cases • Original MUD files • WP5 runtime monitoring and simulations outputs	• Model of a system in a standard format (e.g., Ecore) • Prioritized list of threats (by risk) • Attack-Tree like representation • Extended MUD file	WP5, WP7, Use cases

Tool	Functionality	Lifecycle	Inputs	Outputs	User
safeTBox (T6.3, IESE)	Analysis of the architecture of the systems	Design time	• WP2 Architecture of the system within the use case(s) • WP5 runtime monitoring and simulations outputs	• Reports with Diagrams of the systems analysed • CVS files and other formats upon request, diagrams etc	WP4, WP5
Accountability through Blockchain (ATB) [T6.4, 7B]	Traceability / integrity / trust of auditing logs	Runtime	• Results of the evaluation from WP7 • Audit log files defined from use case applications.	Non repudiable blockchain (Ethereum) based registry with the hashes of audit entries.	Use cases, WP8

3.4.4. Planned Interfaces

3.4.7.1. Blockly4SoS

The Blockly4SoS tool will be able to interact with the BIECO architecture through REST APIs, providing a well-defined interface to be called by the BIECO middleware or in general by other tools. In addition, the Blockly4SoS tool will be able to call specific RESTful Web Services exposed by the BIECO middleware/tools. An initial list of the planned REST services is given below (Figure 17).


blockly4sos

GET /model/{modelId}/workspace/exportEcore Get Model in Ecore format

GET /model/{modelId}/workspace/exportMud Get MUD of a Model

GET /model/{modelId}/riskAssessment/threats Get list of threats prioritized by risk

GET /model/{modelId}/riskAssessment/attackTree Get attack tree

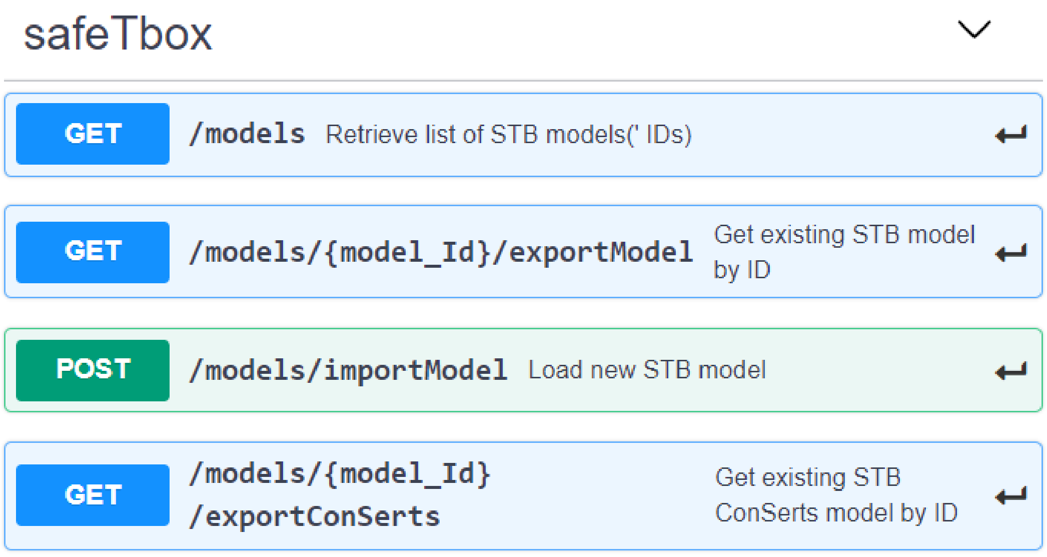
POST /model/workspace/importMud Input MUD file in bytes

Figure 17 - Blockly4SoS Planned Interfaces

3.4.7.2. safeTBox

The safeTbox tool will provide access to its system, safety, and security analysis models both during system development, as well as during runtime, via a RESTful API, optionally accessible via HTTPS. The mentioned API will be developed as part of BIECO work contributed by IESE. Mitigation strategy models will be available in the form of system (safety and/or security) requirements during development. At runtime, mitigation strategies will be available in the form of executable models, referred to as Conditional Safety Certificates (ConSerts)[3], which enable dynamic reconfiguration of the system, based on monitoring evidence (collected in part via predictive simulation as described in

WP5), as well as contract-based service information collected from collaborating systems. The following options will be initially available as described in Figure 18:



safeTbox

GET /models Retrieve list of STB models(' IDs)

GET /models/{model_Id}/exportModel Get existing STB model by ID

POST /models/importModel Load new STB model

GET /models/{model_Id}/exportConSerts Get existing STB ConSerts model by ID

Figure 18 - safeTBox Planned Interfaces

3.4.7.3. Accountability Through Blockchain

The Accountability Through Blockchain (ATB) tool will offer an API available over a secured HTTP channel. The security of the channel will be implemented using TLS with mutual authentication (also known as mutual TLS – mTLS). This will ensure that clients are able to verify that the hashes they store, and retrieve are handled by the expected server and the server will be able to verify the origin of the requests (especially for purposes of storing). The ATB functionality will be available via two methods (names and details are subject to change):

1. RegisterFileHash – which receives the filename and the hash and stores the tuple, along with the client's identity, securely in the blockchain.
2. RetrieveFileHash – which receives the filename and desired identity, and returns the hash stored in the blockchain for that tuple.

The first method will be available for clients identified as hosts maintaining logs, the second by clients identified as audit sources, possibly being other parts of the BIECO framework but not limited to them.

4. The BIECO Architecture (1st Draft)

This section addresses the first draft of the design and specification of the overall BIECO architecture. To this end, the architecture is logically split into two main phases of the ICT lifecycle, namely regarding design time and runtime.

This division is largely inspired by the Reference Architectural Model for Industry 4.0 (RAMI 4.0) supported by the European Commission[4].

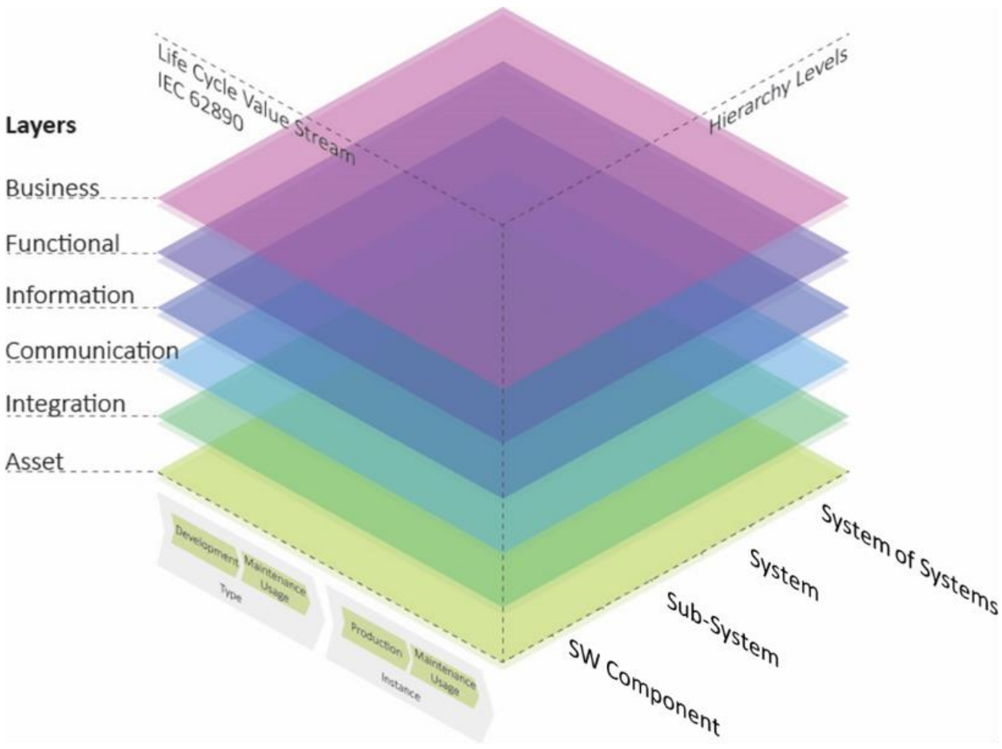


Figure 19 – Adaptation of RAMI4.0 to the BIECO context

In this sense, an initial effort to bridge the concepts of BIECO's architecture to the RAMI reference model [5] is carried out here, which will be later revised and further detailed in the final version of the architecture due in M18.

The *business layer* ensures the integrity of the value stream, including for instance legal and regulatory conditions or requirements. In BIECO's case these are driven by the use cases and security certification bodies. The *functional layer* embodies the formal description of functions and horizontal integration platforms, which in this case consist in each of the tools depicted in the application layer later discussed in the upcoming subsections, as well as BIECO's digital integration platform represented as the interoperability middleware. The *information layer* further ensures this interoperability, addressing the common representation of data and its persistence, represented in BIECO by the Data Management component. Data exchange is accounted for in *communication layer*, following the commonly adopted format and protocols. This will be addressed in BIECO within the middleware in the form of different interaction schemes, one following a publish/subscribe pattern (e.g., MQTT), the other a service-oriented approach (e.g., REST) through well-defined interfaces. The *integration layer* deals with digitalization and provision of information on the assets, which in this case could be represented by dashboards and monitoring probes. Finally, the *asset layer* represents physical components, documents, code, or even human stakeholders. For BIECO, this can be regarded as for instance the MUD files, source code to be assessed and other elements within the controlled environment.

Similarly to RAMI 4.0, BIECO's architecture contemplates the full ICT software lifecycle for consistent data management, security assurance and continuous improvement across the different phases, with each being thoroughly described in Sections 4.1 and 4.2. Following this, Section 4.3 covers the link between these two phases along with their interdependencies within the context of the BIECO framework.

4.1. Design Time View

Design time is understood in the frame of BIECO project as the software lifecycle phase in which the product is designed, developed, implemented, verified, and even certified, before its release to the market. At the end of these processes, the product is intended to be ready for its usage and validated in terms of functionality and security.

The architecture for the design phase is provided in Figure 20. It encompasses the application layer, where BIECO's main services are included, the data management layer for persistence of historical data, the integration middleware and the controlled environment being assessed.

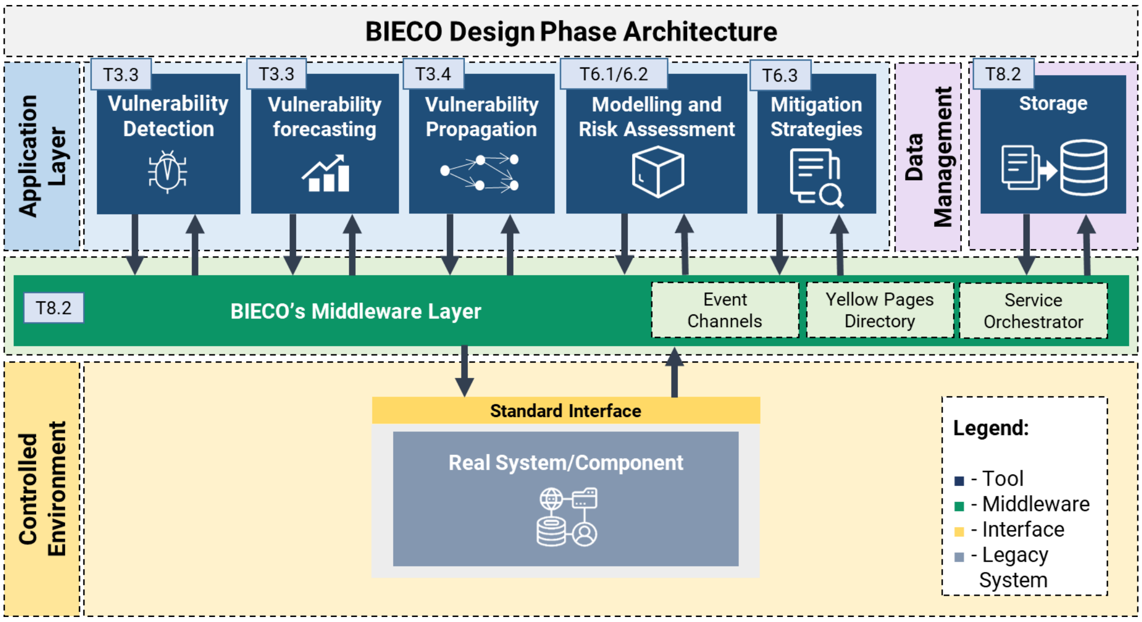


Figure 20 - BIECO's Design Phase Architecture

BIECO project will address the validation of the product security adapting well known standards and approaches such as ARMOUR project[6], ETSI EG 203 25 or ISO 27001, to the needs identified within the software supply chain. In particular, BIECO will consider the following processes:

- Context establishment: As a starting point, BIECO will consider the best practices, regulation, recommendations, and existing vulnerabilities to create a security profile against which the product should be validated.
- Vulnerability assessment: Taking into account the existing vulnerabilities from the context establishment, it will aim at identifying known vulnerabilities in the source code and analysing their possible impact in the own software or other related modules.

- Modelling: It serves to model the system, its complex structure, and interfaces, with the aim of identifying since the early prototyping stage, which are the weakest components, and of representing the paths and possible interactions when attacks are exploited.
- Security testing: The security evaluation is meant to be objective, based on empirical data coming from testing. In this sense, the previous steps of context establishment and vulnerability assessment will guide the definition of the tests. For automation purposes, BIECO will follow a Model Based Testing (MBT) approach from the modelling phase, in which the system and tests are designed at a high level and simulated to verify the compliance of the system with respect to the profile.
- Security risk assessment: The outputs of the modelling and testing processes will be used to measure the overall security level of the product.
- Labelling: The results of the evaluation will be communicated in a visual and simple way to non-expert consumers, so it can be used to compare the security of similar products.
- Treatment: As a result of the evaluation, and dealing with the security problems encountered, BIECO will generate a behavioural profile. This profile will contain a set of security policies that the product should follow to guarantee a secure functioning.

In this sense, the behavioural profile and the label represents the link between the design phase and the runtime phase, in addition to the vulnerability paths encountered during the vulnerability assessment.

4.2. Runtime View

As the name entails, the runtime phase refers to the stage after launch/deployment of a product which has been previously validated and certified. Within the context of BIECO its purpose is to ensure that the product remains secure and within its expected functional boundaries during usage by leveraging the different runtime tools developed in WPs 3-6, deployed within BIECO's digital platform implemented within WP8. An overview of BIECO's runtime architecture can be seen in Figure 21.

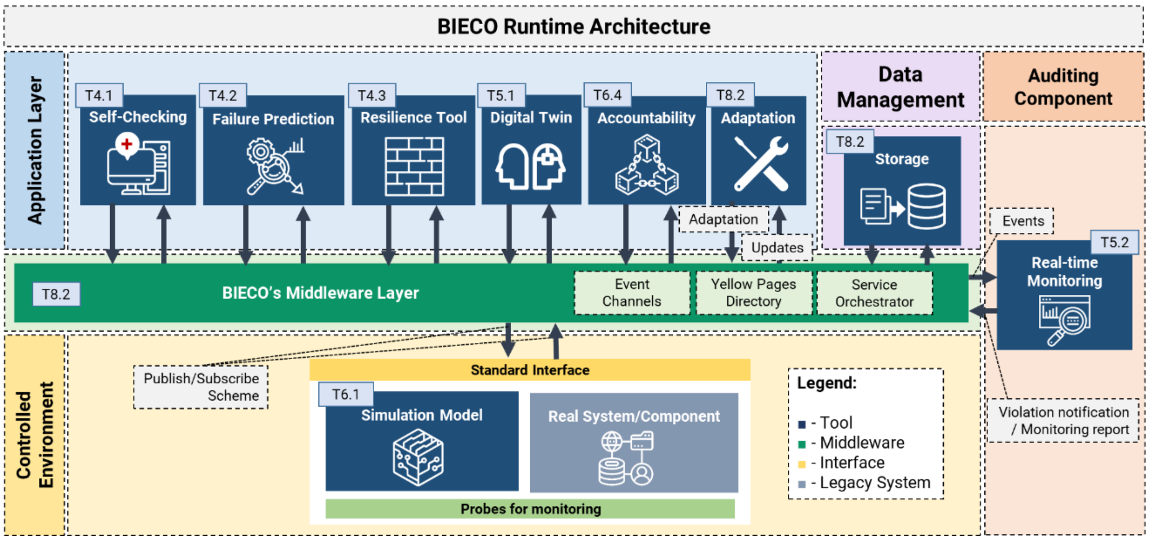


Figure 21 - BIECO's Runtime Architecture

Firstly, the deployed product is represented as the controlled environment. This can be either abstracted as a simulation model or the actual real system or component, upon which probes are injected for monitoring. Hence, this distinction is transparent for the rest of the ecosystem, with communication happening in a commonly agreed format (i.e., standard interface) that instrumentalizes the controlled environment following a publish and subscribe scheme.

This instrumentalization is achieved partially through the real-time monitoring tool, which constantly monitors incoming events from this environment and provides reports on any violations or deviations from the expected behaviour. Collected data can then be made available to the rest of the BIECO ecosystem through the middleware, represented in green in Figure 21. This layer acts as the main driver behind the interoperability, integration and orchestration of BIECO's components.

In order to support a wide array of use cases with different requirements, the middleware is designed to support both event-driven messaging (for communications with runtime constraints) and service-oriented approaches. For the former, appropriate data channels can be made available to which the different system actors can publish or subscribe to. For the latter, the middleware will take care of the complex service orchestration, while providing a yellow-pages directory facilitator in order to make services discoverable within the ecosystem. This can be useful for instance in case specific tools need to access historical data or static files persisted in the data management component.

Finally, the application layer encompasses all of the tools that comprise BIECO's runtime environment and functionalities, as described in Section 3. These include self-checking capabilities of SW and HW components, the digital twin, resilience mechanisms and the adaptation of the controlled environment. Further detail regarding the information flow and interactions between these components is provided in Section 4.3.

4.3. Bridging Design/Runtime

During the design phase, the software is designed, the functional and security requirements are analysed, the software is implemented, tested, and it can even be certified. However, all this valuable information is lost when the product is released into the market if we do not establish a link between the design phase and the execution phase. In this sense, BIECO establishes this link through not only the security certificate (label) that is generated as a result of the security evaluation, but also through a behavioural profile (see Figure 22).

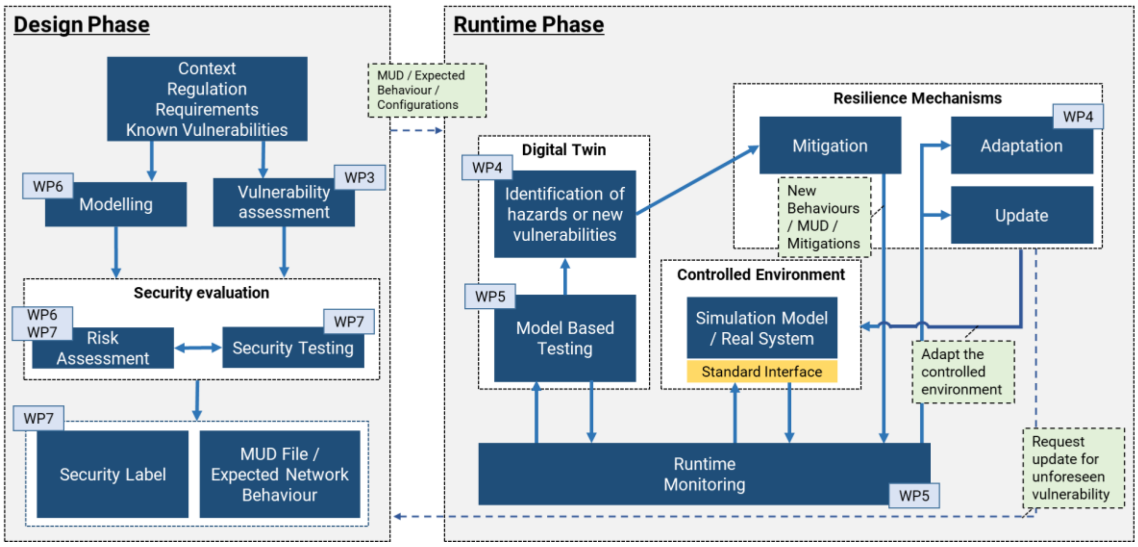


Figure 22 – Connection between the different ICT lifecycle phases addressed in BIECO.

This behavioural profile contains what the software is expected to do during its normal operation; with which devices will it be able to talk to or not, what ports it will use, what protocols, etc. In itself, the behavioural profile is a kind of datasheet of its expected behaviour.

BIECO will base the definition of the behavioural profile on the recently standardized Manufacturer Usage Description (MUD). This standard defines a common structured format (the MUD file) to define such behaviours in form of Access Control Lists (ACLs), and it gives certain indications on how to manage the obtaining of MUD files when the device is installed on the network where it will operate. The MUD uses high-level terms that allow defining several behaviours in a compact way. This way, the MUD abstracts from all the information that depends on the domain in which the device will be installed, such as IP addresses. However, the expressiveness of the MUD model is limited to certain network aspects (ports, Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) and network access control), and therefore, more fine-grained security aspects or related to other protocol stack's layers cannot be described. BIECO will extend the MUD model to allow the definition of additional behaviours beyond firewall-like behaviours, providing extra relevant information from the design phase to the runtime.

The extended MUD profile will be generated from the security evaluation process that takes part during the design phase, combining the information coming from the original MUD file with the results of the testing and the risk assessment processes (Figure 23). As a result, the extended MUD file will be published in the manufacturer's server, so it can be requested during the runtime.

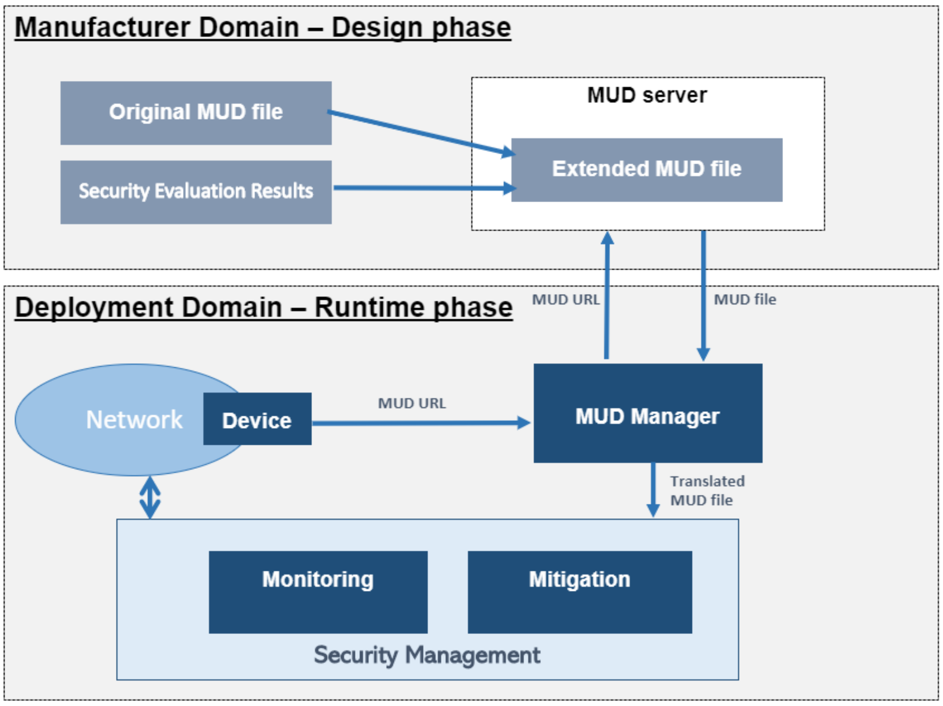


Figure 23 - Representation of the MUD perspective

During runtime, the MUD file can be obtained by the MUD manager from the manufacturer's server using the MUD URL provided by the device, as specified in the MUD standard. BIECO will use the defined behaviours inside the MUD to monitor and detect suspicious behaviours that can leverage to a hazard situation. If a suspicious behaviour is detected that will lead to an attack or a malfunction of the system, a mitigation will be applied, which may be based on the MUD specifications. If there is no known way to mitigate this situation, the system could stop and require an update.

A possible future extension of this behaviour (beyond the initial scope of BIECO) would be to link back to the design phase, enabling the system to autonomously request an update to deal with the unforeseen hazard or vulnerability (represented by the flow in the dashed line of Figure 22).

Finally, if the applied mitigation is inconsistent with the MUD specifications, it will mean that the behavioural profile will need to be adapted, which can lead to a possible update of the MUD by the manufacturer, closing the interaction cycle between the design phase and runtime phase.

5. Conclusion

This deliverable presented the first draft of the design for the overall framework architecture of BIECO, being one of the main artefacts of BIECO's first agile cycle. It represents the results that culminated from the discussions and work carried out during the first six months of the project, serving as the guideline and reference point of alignment for the developments and discussions within and between the various work packages during this period.

Thus, a recap of the BIECO concept was provided, along with a full initial description of the main development work packages, namely regarding WP3 through WP6. For this purpose, a general description was presented for each WP, along with its interactions within the BIECO ecosystem, main actors and planned interfaces. The remaining WPs, WP7 and WP8, are not included as only the ecosystem's tools are considered in this scope, not methodologies (WP7) or integration/implementation efforts specific to the use cases (WP8).

Using this as the foundation, the first draft of the architecture was defined, drawing inspiration and an initial parallel to the RAMI 4.0. Consequently, the architecture was designed to encompass the full ICT lifecycle from the design to the runtime phases, with each being depicted with its own architectural view and each element mapped to the respective work packages.

As future work, it is foreseen that the architecture will mature and be adapted as the development of the various work packages progresses, which will be later documented in the final deliverable of the architecture due on M18. Hence, the artefact (architecture specifications) will be continuously updated and integrated following the agile methodology. It is expected that the final deliverable will include the full specification of the concrete interaction patterns for cybersecurity in ICT ecosystems contemplated within the scope of BIECO, as well as the mapping for the instantiation to the use cases.

6. Reference

[1] A. Babu, S. Iacob, P. Lollini, and M. Mori, “AMADEOS framework and supporting tools,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2016.

[2] S. Velasco, J. Reich, and M. Tchangou, “Interactive information zoom on component fault trees,” in *Lecture Notes in Informatics (LNI), Proceedings - Series of the Gesellschaft fur Informatik (GI)*, 2018.

[3] D. Schneider and M. Trapp, “Conditional safety certificates in open systems,” in *ACM International Conference Proceeding Series*, 2010.

[4] K. Schweichhart, “Reference Architectural Model Industrie 4.0 (RAMI 4.0) - An Introduction,” *Plattf. Ind. 4.0*, 2016.

[5] Standardization Council Industrie 4.0, “Alignment Report for Reference Architectural Model for Industrie 4 . 0 / Intelligent Manufacturing System Architecture,” *Fed. Minist. Econ. Aff. Energy Ger.*, 2018.

[6] G. Baldini, A. Skarmeta, E. Fourneret, R. Neisse, B. Legeard, and F. Le Gall, “Security certification and labelling in Internet of Things,” in *2016 IEEE 3rd World Forum on Internet of Things, WF-IoT 2016*, 2017.