Deliverable 2.4

Overall System Architecture Update (Final)

Technical References

Document version	:	1.0
Submission Date	:	28/02/2022
Dissemination Level	:	Public
Contribution to	:	WP2 – Architecture, Requirements and Use Case Definition
Document Owner	:	UNINOVA
File Name	:	BIECO_D2.4_28.02.2022_V1.0
Revision	:	2.0

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	31/10/2021	UNI	Initial document structure
0.2	18/11/2021	UNI	Initial content and figures for sections 2, 4.1, 4.2.1, 4.2.2, 4.2.2.*.
0.3	15/12/2021	UNI	Content for Section 4.2, 4.2.1, 4.2.2.*.
0.3	21/01/2021	IESE	Content for section 4.4.2.2
0.4	26/01/2022	CNR	Revised Figure 3 and provided content of Sections....
0.5	28/01/2022	CNR	Rearrangement of section 4,5,6 and runtime contents editing
0.6	03/02/2022	IESE	Content to section 6.3.1.1 and 6.3.1.2
0.7	04/02/2022	UNI	General formatting of the document. Added content to section 2, 3, 5, 6.
1.0	14/02/2022	UNI	Added content to Section 1 and 7. Preparation of the version 1.0 for internal review.
1.1	24/02/2022	UNI, UTC, IESE	Modifications based on the internal review. Preparation of the final version for submission.
1.2	25.02. 2022	UNI	Final Revision and correction
2.0	28.03.2022	UNI	Finalizing deliverable and submission by coordinator

List of Contributors

Deliverable Creator(s): Ricardo Peres (UNI), Eda Marchetti (CNR), Antonello Callabro (CNR), Emilia Cioroaica (IESE), Ioannis Sorokos (IESE), Enrico Schiavone (RES), Sara Nieves Matheu (UMU), Javier Martínez (GRAD), Said Daoudagh (CNR).

Reviewer(s): Emilia Cioroaica (IESE), Ovidiu Cosma (UTC), Sanaz Nikghadam-Hojjati (UNI), José 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
CE	Controlled Environment
CEP	Complex Event Processing
CPS	Cyber-Physical System
DCT	Data Collection Tool
DT	Digital Twin
FR	Functional Requirement
IACS	Industrial Automation and Control System
ICT	Information and Communication Technologies
IEC	International Electrotechnical Commission
IoT	Internet of Things
ISO	International Organization for Standardization
MUD	Manufacturer Usage Description
NFR	Non-Functional Requirement
NIST	National Institute of Standards and Technology
SUA	System Under Audit
SUT	System Under Test
UI	User Interface
UML	Unified Modelling Language
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

The highly connected nature of modern ICT supply chains, greatly boosted in part by the paradigm shift of Industry 4.0, have brought about several benefits, with opportunities for added value to be generated by the latest advances in Artificial Intelligence, Cyber-Physical Systems and Internet of Things technologies, among others.

These complex, multidimensional systems of systems encompassing varied actors and heterogeneous components also require, however, an ever-growing need for cybersecurity and trust assurance mechanisms to be adopted, requiring continuous monitoring, assessment and improvement across the different phases of the supply chain's lifecycle to ensure their trustworthiness and integrity.

To this end, 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.

In line with this, the present deliverable builds on top of the results from the first half of the project, being particularly a more matured and follow-up version of Deliverable 2.3, which presented a first draft of the overall BIECO framework. To this extend, this deliverable further formalizes the final BIECO architecture, drilling down into its building blocks for both the design and runtime phases.

Furthermore, it details the foreseen interactions and main event flows across the lifecycle, along with a description of possible alternative usage patterns for the BIECO solution.

This specification artifact corresponds to one of the main outcomes of WP2, being the main contribution of Task 2.3 and marking the midpoint of the project. Consequently, it will serve as a guide for the future development and integration efforts, culminating in its instantiation for each of the BIECO use cases.

Project Summary

Nowadays, most of the ICT solutions developed by companies require integration or collaboration with other ICT components, typically developed by third parties. Even though these kinds of procedures are essential to maintain productivity and competitiveness, the fragmentation of the supply chain can pose a high-risk regarding security, as in most 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 in their mindset, assuming an "untrusted by default" position. According to a recent study, only 29% of IT businesses know that their ecosystem partners are compliant and resilient concerning security. However, cybersecurity attacks have a high economic impact, and it is not enough to rely only on trust. ICT components need to provide verifiable guarantees regarding their security and privacy properties. It is also imperative to detect vulnerabilities from ICT components more accurately and understand how they can propagate over the supply chain and impact 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 needed to expand the horizons of the current risk assessment and auditing processes, considering a much broader threat landscape. BIECO is a holistic framework that will provide these mechanisms to help companies understand and manage the cybersecurity risks and threats they are subject to when they become part of the ICT supply chain. The framework, composed of tools and methodologies, will address the challenges related to vulnerability management, resilience, and auditing of complex systems.

Partners


Disclaimer

Table of Contents

List of Figures

List of Tables

1. Introduction

This first section is aimed at establishing the context for the present deliverable, focused on the specification of BIECO's overall architecture, including its components and the main interactions among them.

To this end, Section 1.1 starts by providing the background for this specification, positioning the present deliverable within the overarching activities of BIECO. Then, a brief discussion of related work is presented, with particular emphasis on existing standards and guidelines that are relevant reference points for the design and implementation of the overall BIECO framework.

1.1. Background

The advent of Industry 4.0 and the growing trend of digitalization have marked a shifting point in our societies, with the physical and digital worlds being more connected than ever before thanks to concepts such as the Internet of Things (IoT) and Cyber-Physical Systems (CPS).

This added level of connectivity between people, machines and systems have facilitated the emergence of new business models and services, with a clear move towards more autonomous and increasingly intelligent solutions.

Consequently, the cybersecurity and trust challenges have also been quickly increasing, with potential impact including not only the jeopardized safety of people and equipment and intellectual property, but also other economic impact such as lower quality or quantities of production, financial and legal implications as well as environmental damage or destruction.

In regard to the aspect of trust, this highly connected and collaborative environments of complex systems across entire supply chains have made it so that such ecosystems rely on the assumption that all of their components operate as expected, with a level of trust having to be established among them as a consequence.

BIECO aims to provide mechanisms to ensure that the behaviour exposed by an ecosystem participant (SW component, System) within a collaboration remains trustworthy in case of failures and remains robust and safe in the face of possible attacks or exploitations of vulnerabilities. This makes it possible to empower the resilience of systems that are part of an ecosystem against malicious attacks, displaying a trustworthy behaviour to the user (be it an interacting service or a human).

Since the malicious intent of potential attacks may be hidden in the smart agents and behaviours that comprise modern complex systems of systems, the assessment of the trustworthiness of a given ecosystem participant requires new platforms that cover multiple phases of the lifecycle, from design to runtime, with this being one of the main roles that BIECO intends to fulfil.

Therefore, the present document is a direct follow-up to Deliverable 2.3 [1], which provided 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. This follow-up consists in a maturation of the aforementioned concepts over the course of the following 12 months, which culminated in the finalized version of the architecture, detailed in terms of its components and respective

interactions during both the design and runtime phases. This specification represents the main artifact and contribution that is produced by Task 2.3., marking the midpoint of the project.

A general depiction of the positioning and role of this deliverable within the context of BIECO is provided in Figure 1.

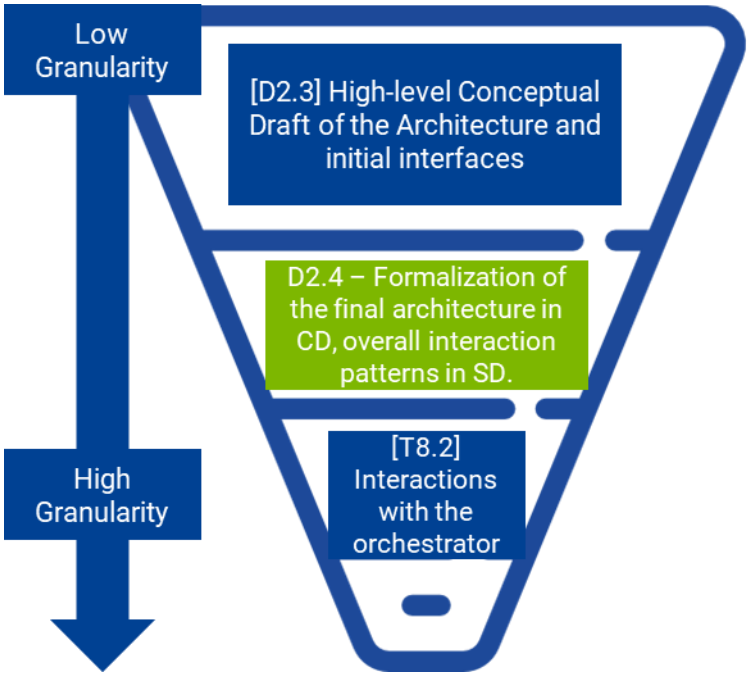

Figure 1 - Positioning of WP2's outputs in terms of the implementation level of granularity

As previously mentioned, the high-level conceptual draft of the architecture provided in D2.3 consisted in the initial guideline for the early stages of BIECO, still at a low level of granularity. From there, this deliverable presents a more formalized and mature specification of the architecture, encompassing both lifecycle phases contemplated in the project, design and runtime, detailed in the form of component and sequence diagrams to guide the implementation and integration efforts for the second half of the project. Finally, in Work Package (WP) 8 the actual implementation of the BIECO platform and its integration with the remaining components takes place, representing the higher level of granularity among the activities planned for the project.

The remainder of this document is structured as follows: Section 1.2 presents a brief overview of related work, particularly in regard to existing standards and guidelines of relevance. Then, Section 2 summarizes the high-level conceptual view of the BIECO framework, followed by a recap of the goals and requirements identified within WP2. Afterwards, Section 3 overviews the BIECO architecture, with its specification being broken down into the Design Phase in Section 4 and Runtime Phase in Section 5. Lastly, a reference architecture instantiation example is provided in Section 6 based on the M18 pre-demonstration use case, finalizing with the conclusions and closing remarks in Section 7.

1.2. Related Work

While in the past the focus of cybersecurity was centred mostly on the defence of organizational perimeters, such as the protection against unauthorized access to private

computer networks (e.g., firewalls, malware protection, etc). Nowadays the increased connectivity brought about by the Industry 4.0 paradigm has forced organizations to rethink their cybersecurity strategy [2].

Advances in Information and Communication Technologies (ICT) have made it possible for Industry 4.0 systems and systems of systems to be highly connected and distributed, with a close link between the physical and the digital world. This makes it so that the topic of cybersecurity is now more critical than ever, with related technologies evolving at a rapid pace to match the increasing risk and threat levels of these systems, including for instance encryption and artificial intelligence-based approaches.

While the literature and regulatory bodies are not consensual in terms of a "one-size-fits all" architecture or solution that completely addresses all cybersecurity challenges, considerable effort has been made over the last years by official institutions to propose recommendations and best practices by using standards as references [3]. These include the International Organization for Standardization (ISO), the International Electrotechnical Commission (IEC) and the National Institute of Standards and Technology (NIST), some of which will be discussing in the coming subsections.

1.2.1. NIST Framework

The NIST has proposed a multi-platform framework for improving critical infrastructure cybersecurity [4], aimed at assisting organizations to manage and reduce cybersecurity risks. The framework core presents a set of functions that provide a strategic overview of the lifecycle for cybersecurity risk management, also discussed in the literature as the pillars of ICT cybersecurity [5], as depicted in Figure 2.

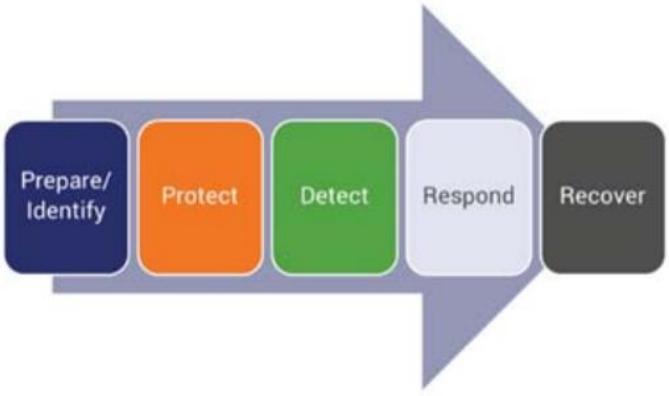

Figure 2 - The five pillars of ICT cybersecurity according to [5]

As per the framework, these functions are not intended to be a static, sequential methodology or lead to a static desired end state, but instead should be executed in parallel and continuously across the lifecycle of a system, in a way that is capable of addressing the dynamic cybersecurity risk. These five pillars or functions are adopted as guidelines for the BIECO solution, and can be defined as follows:

- **Identify** – Involves the understanding of the business context to manage cybersecurity risk in terms of systems, people, assets, data and capabilities. This relates to the activities developed particularly in WP3, WP6 and WP7, concerning vulnerability management, risk identification and the security context and claims.
- **Protect** – Entails the development of mitigation actions to limit or contain the impact of potential cybersecurity events. This aspect is covered in particular by

WP4 and WP6, regarding the study and development of resilience mechanisms and mitigation actions.

- **Detect** – Implement specific activities to identify the occurrence of relevant cybersecurity events. Here the main activities are encompassed within WP3 regarding vulnerability management and particularly WP5, addressing runtime auditing of ICT ecosystems.
- **Respond** – Implement measures to take action upon the detection of cybersecurity events.
- **Recover** – Ensure that appropriate measures are in place to maintain the resilience of the system and restore safe operational conditions, capabilities or services. Once again these last two points relate to WP5, as the Auditing Framework [6] enables the notification of alarms or triggering of mitigation actions according to the results from the complex event processing and the conformity monitoring supported by the predictive simulation

In addition to this, other guidelines and recommendations have been put forth by different organizational bodies, some of which will be discussed in the upcoming subsections.

1.2.2. ISA/IEC 62443

The ISA/IEC 62443 is an international series of standards which define guidelines for the security of an Industrial Automation and Control System (IACS). As mentioned in D7.1 [7], it broadly describes the Security Life Cycle of the IACS as being composed of three main phases: 1) *Assessment*, which includes activities pertaining to the identification of high-level risks, as well as to analyse vulnerabilities and low-level risks and to allocate the minimum security requirements for each component of the system; 2) *Implementation*, which encompasses the activities needed to identify IT risks and define the associated mitigation actions comprised in the security strategy; 3) *Maintenance*, referring to the actions that constitute the process of continuous monitoring of the security level of components.

Going further, it also specifies the IACS automation solution security lifecycle as shown in Figure 3.

Figure 3 – ISA/IEC 62443 IACS Automation Solution Security Lifecycle (adapted from [8])

The different phases can be defined as follows:

- Specification – As mentioned before, it includes the identification of the system under consideration and the initial high-level cybersecurity risk assessment. The result is the specification of the target security levels used for the design phase.
- Design – This phase entails the detailed design of the system, including technical security measures based on the security level and the related organizational security measures.
- Implementation – At this stage the technical security measures specified in the cybersecurity requirements are implemented in the solution. The organizational security measures are developed so that they are available during the verification and & validation phase.

- Verification & Validation – During this part of the lifecycle the solution is tested to ensure the technical and organizational security measures meet the specified security and safety requirements. Some examples include vulnerability scans, intrusion detection tests and access control tests.
- Operation – The operation phase refers to placing the solution into service, and executing different security measures, which should be periodically reviewed and updated.
- Maintenance – Relates to the continuous monitoring of security threats and vulnerabilities during operation. Addressing such threats may require changes to the organizational or technical security measures of the IACS.
- Decommissioning – The decommissioning phase can be triggered by a maintenance activity (e.g., replaced a given hardware component) or by a major upgrade to the system. Regardless, it should be done in a way that the on-going operations are not compromised.

This is of particular relevance to BIECO, as it highlights the importance of addressing the entire lifecycle of a system or component, from its specification and design to its runtime operation with continuous monitoring until the eventual decommissioning.

2. BIECO Conceptual Framework Towards Security and Trust in ICT Ecosystems

The rationale behind BIECO's concept is to deliver a framework for improving trust and overall security claims 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 and its driving goals are depicted in Figure 4.

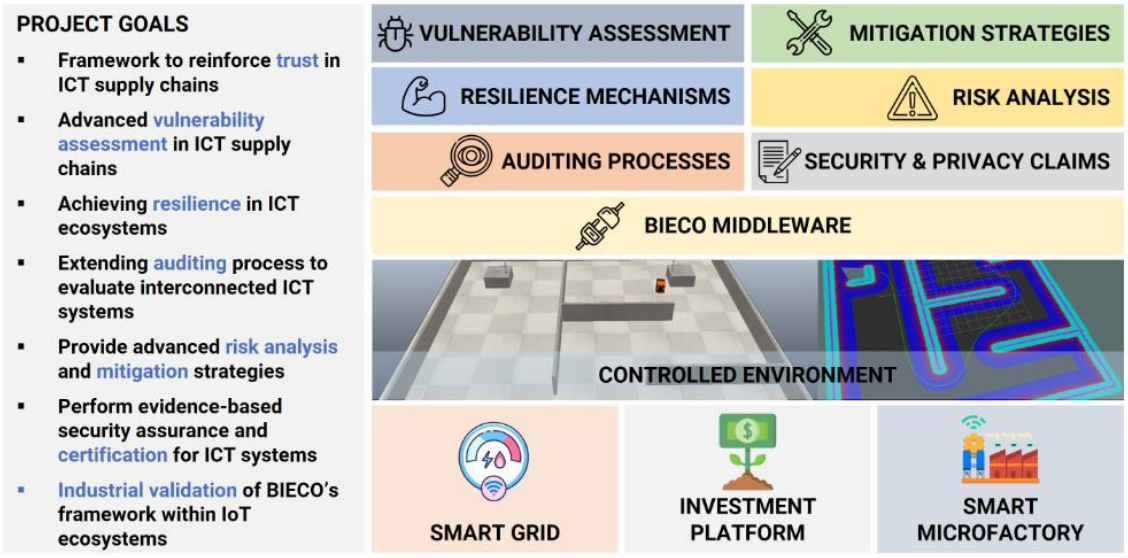

Figure 4 - Overall concept of the BIECO Framework to foster security and trust in ICT ecosystems [9].

BIECO covers the different parts of the product lifecycle, from the design phase with its security assessment methodology, to the runtime with its auditing system. To facilitate the cooperation of these different components at each phase, the BIECO middleware acts as both a common channel for communication and its orchestrator, enabling the interoperability of the BIECO solution via the designed common interfaces. Dedicated communication channels can be setup with the initial assistance of the orchestrator, such as the case for the Auditing Framework and its probes, as explained later in this document, which also mitigates the risk of message overload. Finally, BIECO envisions its validation across three industrial use cases, one addressing a smart grid environment, another focused on an AI-based investment platform and the other dealing with a smart micro factory in a manufacturing setting. A smaller pre-demonstration scenario is also considered, as discussed in Section 6, which addresses an intralogistics use case with mobile robots and acts as a test bed and showcase of the early-stage developments of the project.

In an effort to keep the specification of the BIECO framework self-contained, this section briefly recaps the requirements specification outputs from D2.1[10] and D2.2 [11], framing their relevance and mapping in the context guiding and constraining the specification of the overall BIECO framework presented in this document.

During the initial definition of BIECO's project requirements, documented in D2.1 [10], the following set of project-level goals were identified, taking into account the challenges typically associated with such complex ICT systems of systems:

- **G1** – Providing a framework that will allow the reinforcement of trust in ICT supply chains
- **G2** – Performing advanced vulnerability assessment over ICT supply chains
- **G3** – Achieving resilience in ecosystems formed by unreliable components
- **G4** – Extending auditing process to evaluate interconnected ICT systems
- **G5** – Providing advanced risk analysis and mitigation strategies that support a view of the complete ICT supply-chain
- **G6** – Performing evidence-based security assurance and a harmonized certification for ICT systems
- **G7** – Industrial validation of BIECO's framework within IoT ecosystems

While this deliverable is particularly related with G1, as it entails the specification of the overall BIECO Framework, it is ultimately tied with all of the listed goals since the framework should be defined in a way that facilitates the realization of the project's aims. From these goals, a set of Functional Requirements (FR) were derived, as summarized in Table 1.

Table 1 - BIECO's Functional Requirements at project-level from D2.1 [10]

Requirement	Rationale	Goals	WPs
FR1 – Real-time Monitoring	BIECO should be capable of performing real-time monitoring/auditing of the underlying systems or devices to detect deviations from the expected behaviour.	G1, G3, G4	WP5
FR2 – Adaptation	BIECO should be able to adapt the underlying system/component/device at runtime based on adequate mitigation strategies	G1, G3, G5	WP5, WP6
FR3 – Vulnerability Analysis	BIECO should enable the identification and/or forecasting of vulnerabilities in ICT systems through advance data analytics.	G1, G2, G3	WP3
FR4 – Simulation	BIECO should be capable of simulating the behaviour of underlying systems or components to self-check future failures or vulnerabilities.	G1, G2, G3	WP4
FR5 – Security evaluation	BIECO should be able capable to measure the security of a system in an objective way using empirical tools such as testing.	G1, G5, G6	WP6, WP7
FR6 – Security certification	BIECO should be able to generate a visual and dynamic security label as a result of the security certification process.	G1, G5, G6	WP7

FR7 – Security baseline	BIECO should base the security evaluation on standards and best practices, taking into account also the relevant regulation.	G1, G5, G6	WP7
FR8 – Behavioural profiles	BIECO should design a security behavioural profile as a result of the certification process.	G1, G5, G6	WP7, WP6

These FRs are tied to a specific component or set of components encompassed within the BIECO framework, each tasked with realizing a specific function within the overall scope of the project. For this purpose, Sections 4 and 5 will detail the components contemplated within each of the lifecycle phases addressed by BIECO (design and runtime, respectively), linking each the presented components back to the FRs presented in Table 1.

Consequently, it is of particular importance to also account for the defined Non-Functional Requirements (NFR), as these deal directly not with the functionalities of the system, but instead with how these functionalities should be carried out. These are vital to the specification of the BIECO framework, since such specification must be developed in a way that accounts for such constraints by design. This is evidenced by the fact that every single one of these requirements is associated with G1, which is directly related to the framework. Once more, the list of non-functional requirements is presented in Table 2.

Table 2 – BIECO's Non-Functional Requirements from D2.1

Requirement	Rationale	Goals	WPs
NFR1 – Interoperability	Heterogeneous components of the BIECO ecosystem should be capable of cooperating and exchanging data using common representations and interfaces	G1	WP3, WP4, WP5, WP7
NFR2 – Scalability	BIECO solutions should be agile and dynamic, being as automated as possible.	G1	WP6, WP7
NFR3 – Modularity	BIECO solutions should be loosely coupled, allowing stakeholders to mix and match functionalities of the framework as needed.	G1	WP2, Wp3, WP4, Wp5, WP7
NFR4 – Privacy-Preserving	Measures should be taken to ensure that BIECO's tools preserve the privacy of sensitive data (e.g., source code) of stakeholders.	G1, G2	WP3
NFR5 – Standardization	BIECO solutions should be based as much as possible on current standards.	G1, G5, G6	WP7

Starting from NFR1, within BIECO the interoperability aspect is crucial since the large set of envision FRs will be fulfilled by several heterogeneous tools. These tools must not

only be capable of exchanging data amongst themselves in a way that can be interpretable by all, but also ensuring that the interactions occur in the proper sequence and in a way that can later be extended to accommodate additional tools beyond the scope of the project. For this reason, the BIECO framework includes a key component, the BIECO Middleware, which will simultaneously act as the shared channel for communication and as its orchestrator, providing a common interface for BIECO's components to interoperate.

Regarding NFR2, following the same principle the orchestrator will facilitate the inclusion, setup and automation of additional tools using pre-defined flows for each of the lifecycle phases addressed in BIECO.

Concerning NFR3 (modularity), BIECO presents a loosely coupled architecture, allowing stakeholders to choose which functionalities of BIECO should be deployed, enabling the adoption of either the full solution, or partial subsets of its functionalities. Such considerations are discussed in Sections 4.3.4 and 5.3.2.

In terms of NFR4 and the exchange of sensitive data, BIECO's framework envisions the creation of a Data Management and Storage component, which will be divided into public and private access parts following adequate data management practices. Sensitive data will be stored privately with controlled access (secret key for each use case), being used mostly for the connection between the two contemplated lifecycle phases, as discussed in Section 3. Thus, sensitive data will be linked to the use cases and not made available for the general public. In cases where the data should not be stored due to privacy or intellectual property concerns, BIECO will enable the user to upload the data only for processing via the User Interface (UI) (e.g., such as the case for the vulnerability assessment), without it being persisted anywhere in the platform.

Finally, in relation to NFR5, for the both the design of the BIECO architecture and the implementation of its components, existing standards and guidelines are being taken into account such as those described in Section 1.2, D2.2 [11] and D7.1 [7].

3. The BIECO Architecture

This section details the formalization of the overall BIECO architecture. It provides an overview of the BIECO components comprised along the contemplated lifecycle phases, namely design time and runtime. In addition to this, the bridge between the two phases is discussed, with reference to the components that enable such a link. Then, varied usage patterns for the BIECO framework are presented, differentiating between mode of operation and respective functionalities/limitations based on which components are deployed.

An overview of the architecture encompassing both the design time and runtime phases is provided in Figure 5.

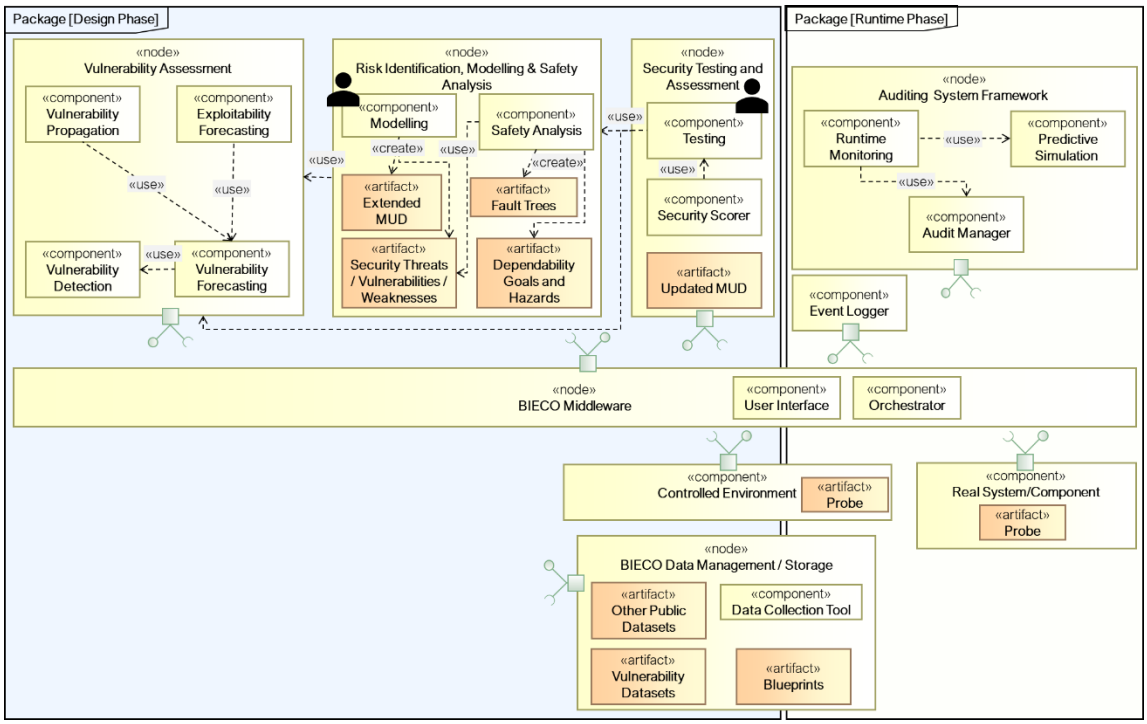

Figure 5 – Overall Architecture of BIECO, encompassing components in both the design and runtime phases. The depiction of the interfaces was simplified for readability. All components communicate through the BIECO Orchestrator.

The architecture is depicted using component diagrams, loosely based on the Unified Modelling Language (UML) notation. Such diagrams are "architectural" in nature, forming the model of architecture space, incorporating the needs and limitations of the organization placed on the system [12]. To facilitate the understanding of these diagrams, some key definitions of their elements are provided:

- **Component**: An entity required to execute a stereotype function. A component provides and consumes behaviour through interfaces, as well as through other components.
- **Node**: Represents hardware or software objects, which are of a higher level than components.
- **Port**: Specifies a separate interaction point between the component and the environment. Represented with a filled square symbol.

- **Package**: Groups together multiple elements of the system. Just as file folders group together multiple sheets, packages can be drawn around several components.
- **Usage Dependency**: A usage dependency is relationship which one element requires another element for its full implementation. It is shown as a dashed arrow with a *<<use>>* keyword. The arrowhead points from the dependent component to the one of which it is dependent.
- **Required Interface**: Represented by a straight line from the component with a half circle. These symbols represent the interfaces where a component requires information in order to properly perform its role.
- **Provided Interface**: Depicted as a straight line from the component with a circle. This symbol represents the interfaces where a component produces information used by the required interface of another component.

As depicted in Figure 5, the BIECO follows a loosely coupled and modular design, with each main component interfacing with the overall BIECO framework through a common interface to the orchestrator.

On the left side we have the design phase package, comprising the vulnerability assessment and risk/security assessment nodes, which are explored further in Section 4. On the right side the runtime package can be found, encompassing the runtime monitoring, predictive simulation and audit system manager nodes. Each of these is further detailed in Section 5.

Furthermore, to facilitate the connection between the two lifecycle phases, some of the components can be shared among them. First and foremost, this naturally includes the BIECO orchestrator, as this component is responsible for managing the communications and the respective flow between the different components.

However, the orchestrator acts as a conduit for the data, without accounting for persistence. For this purpose, the Data Management node, encompassing the Data Collection Tool, also exists in both phases. This node acts as a common data storage, accessible through the orchestrator, in which data that needs to be persisted and shared between components either within the same phase, or across phases can be stored and accessed.

Regarding the design phase, examples include known vulnerability datasets or other public sources of risk and vulnerability data. Another would be the results from the Testing component within the Security Assessment node, which are later used by the Security Scorer to formalize the security assessment of a component or System Under Test (SUT).

Data and results from the design phase processes that are pertinent to the runtime phase can also be stored as blueprints, which effectively allow the passage of relevant data between the two lifecycle phases.

While the BIECO framework has been designed with the full range of functionalities envisioned with the project's concept in mind, it can still support a partial instantiation of dedicated tools, albeit with possible limitations in the provided functionalities and trust assessment. For this purpose, different foreseen usage patterns are discussed in this document. For clarity, these alternative flow patterns are described after the flows for each phase are introduced in their respective sections. Please refer to Sections 4.3 for further details.

4. BIECO for Design Time

4.1. Design Phase

The first stage of the lifecycle contemplated in BIECO is the design phase, during which the validation of the product's security and trust is addressed. This process is achieved by adapting well-known standards and approaches including the methodology from the ARMOUR project, as well as ETSI EG 203 25 and ISO 27001, to the needs of the software supply chain and the industrial requirements identified and analysed in T2.1 and T2.2.

To this end, during the design phase the following elements are considered:

- **Context establishment**: As a starting point, BIECO will consider the best practices, regulation, recommendations and datasets of existing risks and vulnerabilities to create a security profile against which the product should be validated.
- **Vulnerability Assessment**: Taking into account the existing vulnerabilities from the established context, BIECO aims to identify known vulnerabilities and risks in the product's source code, as well as to forecast future vulnerabilities that can be exploited and their propagation paths. With this, it is possible to analyse the impact these vulnerabilities may have on the software or related modules.
- **Risk/Security Assessment**: From behavioural profile modelling and design, implementation and execution of security tests, BIECO will carry out an assessment to score the overall security level of the product.
- **Data Management/Storage**: Beyond serving as a repository for the data shared between the design phase components, the data management of BIECO will act as a bridge between the lifecycle phases, enabling the results from the design phase to be used as an input to the components at runtime.

The coming sections describe the design phase in further detail, starting with the overall specification of its components, followed by the concrete definition of the event sequence and interaction patterns between its actors.

4.2. Design Phase Components

As previously stated, the components involved in the design phase of BIECO can be subdivided into three main nodes, namely those related with Vulnerability Assessment, then Risk/Security Assessment and finally Data Management. Additionally, these components depend on the BIECO Orchestrator to provide a means for communication among them, as well as to ensure that the correct sequence of actions is triggered based on the pre-defined usage patterns supported at design time. Lastly, some of these components, particularly regarding test design, implementation and execution, require the SUT to be running and reachable in a Controlled Environment. Consequently, these last two components, the orchestrator and the controlled environment, are present in

both phases of the lifecycle, albeit with minor differences concerning the latter, which will be further addressed in Section 5.

A component diagram depicting the Design Phase elements and their interdependencies is provided in Figure 6.

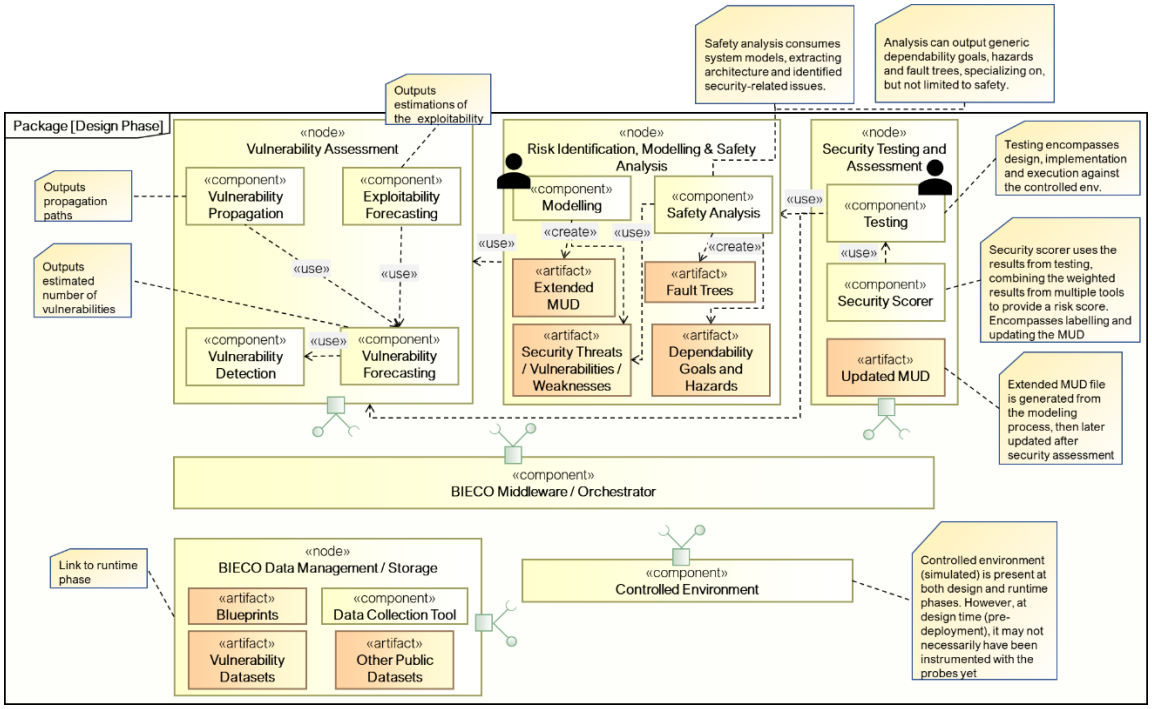

Figure 6 – Design Phase Component Diagram. Interfaces have been simplified at node level for readability. Each component will communicate through the BIECO Orchestrator.

Looking at the Vulnerability Assessment node, the Vulnerability Detection component provides the starting point upon which the remaining components can perform their tasks. From there, Vulnerability Forecasting provides an estimated number of potential vulnerabilities, with the Vulnerability Propagation Component and Exploitability Forecasting outputting the propagation paths and estimations of the exploitability for each of the identified vulnerabilities.

Using the outputs from this first node, BIECO's Risk Identification, Modelling and Safety Analysis node is designed to, as the name suggests, enable the user to model the system, along with its complex structure and interfaces. On the one hand, this allows the user to identify, since the early prototyping stage, which are the weakest components and to further analyse the possible attack paths and interactions that can be exploited. On the other hand, it also serves to extend the initial Manufacturer Usage Description (MUD) file provided by the manufacturer with pertinent information resulting from the modelling stage.

Finally, building on this information, within the Security Testing and Assessment node a comprehensive test suite can be designed and implemented by an expert through the Testing component, which is then executed against the Controlled Environment. The role of the Security Scorer component is then to aggregate the weighted results from the testing phase to provide a risk score. This component is also responsible for labelling the SUT, meaning that the results of the evaluation are communication in a visual and simple way to non-expert consumers (to facilitate comparison of similar products), as well as to update the extended MUD file. As a result of the evaluation, BIECO generates

a behavioural profile containing a set of security policies that the SUT should follow to guarantee secure and trusted operation.

Therefore, in addition to the results from the Vulnerability Assessment, this behavioural profile and the security label constitute a set of artifacts that represent a link between the design and runtime phases, which can be saved in the Data Collection Tool to be later made available to runtime components to ensure the trustworthiness of the SUT.

4.3. Design Phase Flow

In order to provide a clearer description of the sequences of interactions between all of the aforementioned components that play an active role during the design phase, the present subsection formalizes the foreseen interactions in Sequence Diagrams.

Much like the components themselves, these interactions can be divided into three clear stages. The design phase is kicked off by the initial vulnerability assessment and risk identification, based on various information sources such as the provided security context and publicly available databases of risks and vulnerabilities. From there, the next stage is the security testing, which involves the design, implementation and execution of these tests against the controlled environment. Lastly, with the results from the previous phases the security assessment can take place, resulting in the security labelling of the SUT and the updated MUD file.

4.3.1. Vulnerability Assessment and Risk Identification

The first subprocess contemplated in the design phase is the vulnerability assessment. Vulnerabilities can be associated to software, hardware, policies or even the users' behaviour, both intended and unintended. Their assessment is a crucial step in security and trust management, given that it enables the detection and analysis of possible security flaws or bugs of the system under test at an early stage. While such vulnerabilities could be addressed at any stage of the system's lifecycle, an early identification and assessment mitigates not only the associated risks, but also the costs and likelihood of exploitation by attackers.

Within the scope of BIECO, the vulnerability assessment subprocess is focused specifically on the identification of vulnerabilities present in the source code of the system under test at design time. Such vulnerabilities could later end up impacting the confidentiality, integrity and availability of the system. On top of this, this subprocess also aims to analyse the possible long-term effects that the identified vulnerabilities could have on the system, taking into account aspects such as the period of time under which they might be exploited, or how they could propagate to other components downstream in the software supply chain.

A depiction of the event sequence for this subprocess is provided in Figure 7.

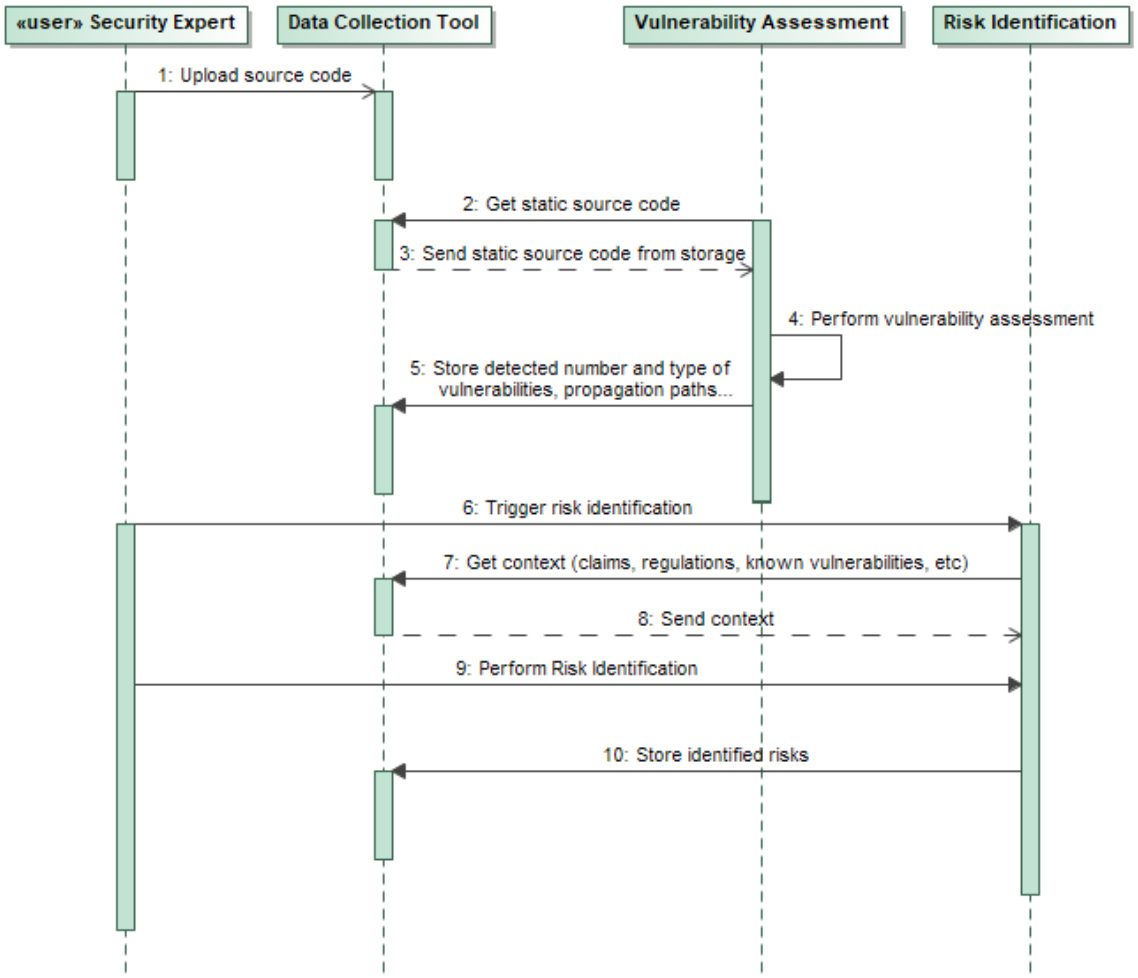

Figure 7 – Vulnerability assessment and risk identification flows in the initial stages of the design phase. Communication between components is assumed to go through the BIECO orchestrator.

This phase starts off with a trigger from the user, who uploads the source code to the BIECO platform. By design, this is expected to be stored in the Data Collection Tool for later processing. However, in the event that for any reason (e.g., intellectual property rights) the code should not be stored, it can be sent directly to the Vulnerability Assessment tool chain. From there, the vulnerability assessment is carried out, with the resulting number and type of detected vulnerabilities, as well as foreseen propagation paths are stored in the Data Collection Tool (DCT).

Once this step is concluded, the system waits for the user to trigger the risk identification process. Once the trigger is sent (via the BIECO orchestrator), the Risk Identification tool, which in BIECO's case is ResilBlockly [13], retrieves the security context from the DCT, then through the user's intervention the initial risk identification is carried out, with the results being once more stored in the DCT for the downstream stages. The user's intervention at this step consists in the modelling and early prototyping of the system performed with ResilBlockly. The latter is a Model-Driven Engineering tool within BIECO that, among other features, enables the association of vulnerabilities and weaknesses to the modelled assets, and allows to identify risks connected to them. The resulting information, vulnerabilities, weaknesses, their risks, and other data introduced by the user through a dedicated GUI, can be included in the extended MUD; then, ResilBlockly delivers the extended MUD to the Data Collection Tool in order to be stored.

4.3.2. Security Testing

Vulnerability Assessment is followed up by the Security Testing. This step is meant to support the security assessment of the SUT with empirical data resulting from the test execution. For this purpose, the tests should be carefully designed with guidance from the results of the previous step, namely the vulnerability assessment in addition to the security context. BIECO follows a model-based testing approach, in which the system and tests are designed at a high-level of abstraction and simulated to verify the compliance of the SUT with a specific behavioural profile.

The sequence of steps comprised in the security testing are represented in Figure 8.

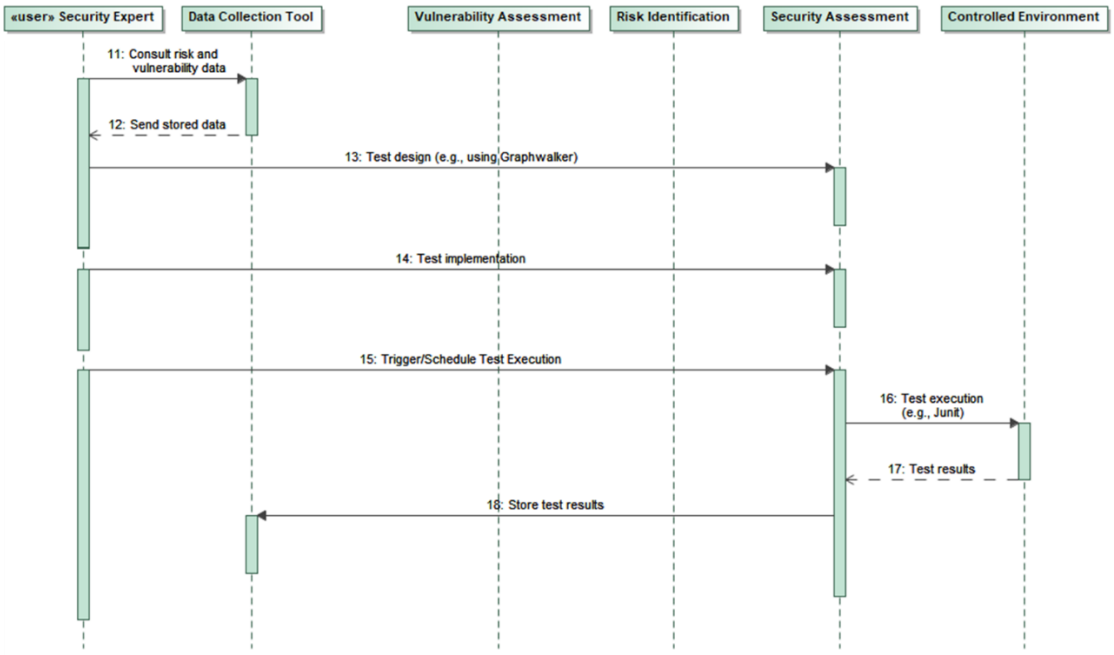

Figure 8 - Security testing sequence of events, following the vulnerability assessment stage during the design phase. Communication between components is assumed to go through the BIECO orchestrator.

As a starting point for this stage, the user (i.e., security expert) consults the results from the previous step made available through the DCT. With this information the test design can then be initiated.

It is worth noting that, within BIECO's execution at design time, it is expected that the overall execution can be stopped and resumed as needed, based on the different execution times of certain jobs with dependencies downstream, or due to asynchronous interactions with the user.

Such an example can be observed in messages 13, 14 and 15, dealing with the test design, implementation and execution, respectively. At each of these stages, the user can perform part of the work offline and then upload it to the BIECO platform once it is ready, thus the involved tools should support this pattern by design.

Consequently, once the tests have been designed and implemented, the system waits for the user to trigger their execution against the Controlled Environment (CE). Once this process concludes, the results are once again stored in the DCT for usage downstream.

4.3.3. Security Assessment

The final step of the design phase is the Security Assessment, which essentially consolidates the results from the design phase into a concrete security score for the SUT. This is a crucial step of BIECO's security certification methodology, as it provides both the security labelling as well as the profile that should be then continuously verified at runtime to ensure that the SUT's behaviour remains secure and trustworthy throughout its execution.

The sequence of events encompassed in this stage are represented in Figure 9.

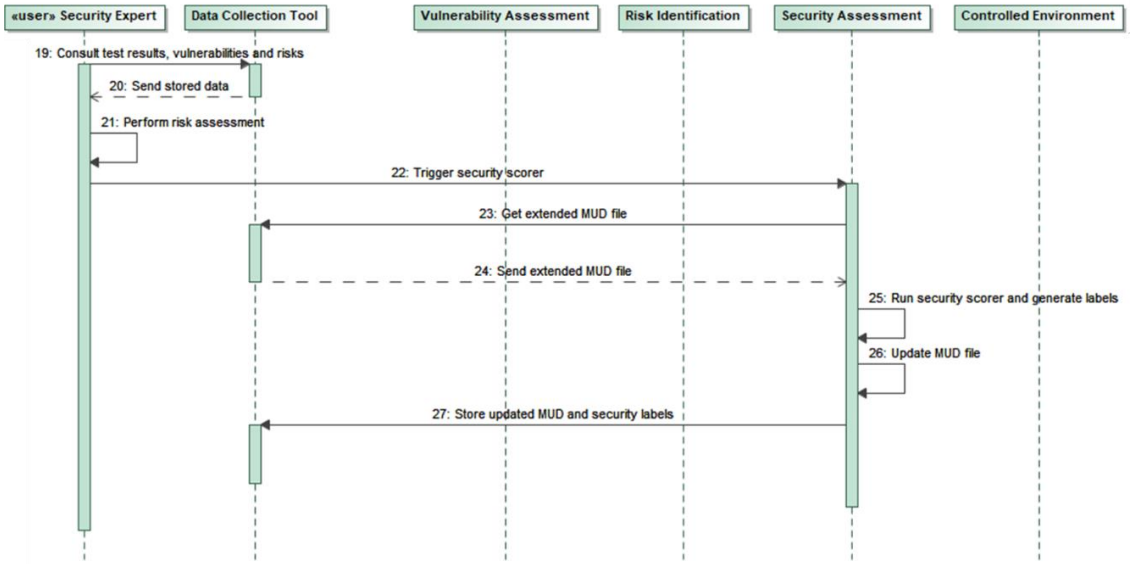

Figure 9 - Sequence of events for the security assessment, following the execution of the tests agains the Controlled Environment during the design phase. Communication between components is assumed to go through the BIECO orchestrator.

Once more the initial trigger for this stage is the user, who consults the data resulting from the stages upstream. With this information, an initial risk assessment can be manually performed, after which the trigger is given by the user (through the orchestrator) to initialize the security scorer tool. This tool retrieves the extended MUD file (resulting from the modelling process) from the DCT and computes the corresponding security score from the weighted integration of the different tools and results from the testing phase.

From the results of the security assessment, a security label and updated MUD files can be generated, which are then stored in the DCT to be made available at runtime. This allows BIECO to verify if the conditions assessed at design time are maintained during the SUT's execution at runtime, effectively enabling the assurance of trust and security of the system.

4.3.4. Alternative Usage Patterns for Design Time

This section describes alternative usage patterns for BIECO's design phase. These pertain to the case where the user may not desire or need to instantiate all BIECO's design phase components, resulting in only a partial set of BIECO's capabilities being available. Optional components within such patterns are listed in *italics*.

4.3.4.1. BIECO for Testing and Security Assessment

Available Components: Testing, Security Scorer, Controlled Environment, *BIECO Middleware (orchestrator)*, *Data Management / Storage*. Regarding the optional components, in the limit this setup could work with manual inputs from the user.

In this setup, the testing component can be run by the user in isolation, as it does not depend on other BIECO components. The user can interact with this tool to model the system, generate for instance the skeleton of the tests and the adapter to link these tests with the real or simulated system within the Controlled Environment.

More specifically, within the scope of BIECO one of the tools realizing this is Graphwalker, which consists in an open-source solution for model-based testing (MBT). The general idea is to model an application as a graph of calls and verifications which in turn can be employed for extensive and automated testing. It provides a GraphWalker Studio, an editor in which models can be created and edited. Studio also has a feature to run test path generation to verify if the models are correct. Moreover, GraphWalker provides command line tools for generating paths, which can be integrated as a maven project. It requires only an implementation of vertices and edges, after which the tests can run automatically.

For the Security Assessment portion, the input from at least one testing tool (e.g, Graphwalker) is required to generate the evaluation results. This tool aggregates the outputs of BIECO's testing tools to evaluate the security of the system following the security evaluation methodology defined in WP7. The result of the evaluation is visually represented as a security label (spider chart) though the BIECO GUI, and optionally an updated extended MUD could be generated from the tests results.

It also requires additional inputs manually introduced by the user or generated from other tools (parameters such as impact, component's sensitivity, the system's components or the tolerance profiles). To generate the updated MUD, the scorer needs as input the extended MUD generated manually or by the modelling tool (i.e., ResilBlockly), which should be retrievable from BIECO's data storage (i.e., the Data Collection Tool).

4.3.4.2. Design Time for Modelling and Risk Identification

Available Components: Resilblockly, Orchestrator, Data Collection Tool

This alternative usage pattern for design time involves only three components: the DCT, ResilBlockly and the Orchestrator. The pattern consists in the import of an original MUD file, initially stored in the DCT, into ResilBlockly; the interaction between the two tools is not happening directly but is realized through the intervention of the Orchestrator. Then, the end user, after having realized the model of the system within ResilBlockly can connect it to the MUD file. ResilBlockly allows to identify and associate vulnerabilities and weaknesses to the modelled system, and to determine the potential risk connected to them; this information, together with security-related data as cryptographic keys, and application protocol, is included in the MUD, generating a so-called extended MUD. The latter is then stored into the DCT, again through the intervention of the Orchestrator.

5. BIECO for Runtime

5.1. Runtime Phase

This section summarizes the BIECO Runtime conceptual vision already introduced in D5.1 [3]. To facilitate the conceptualization, the interactions with the BIECO middleware are omitted until the follow-up sections which specify the components and their respective interaction sequences in further detail. With that being said, Figure 10 depicts a high-level conceptual view of the runtime phase, encompassing the parallel execution of the Event Logger and the Auditing System Framework [6] that includes the predictive simulation, the runtime monitoring and the controlled environment, and Data Storage.

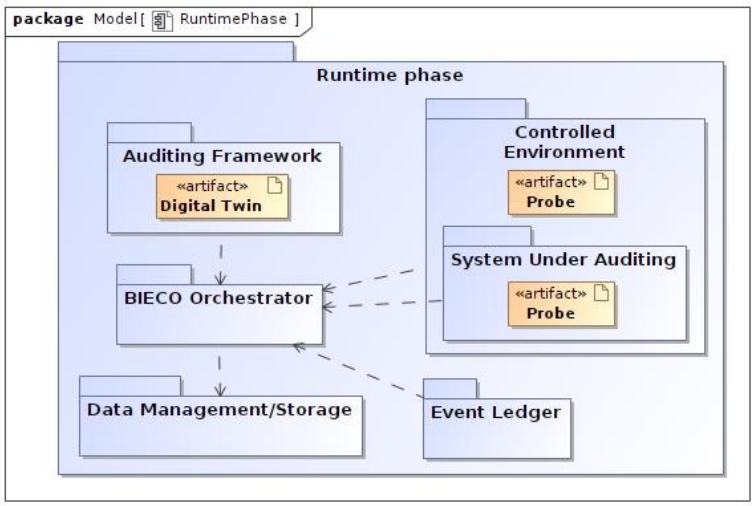

Figure 10 - High-level view of the runtime phase (adapted from [14])

Within the scope of the BIECO project, the runtime phase entails the auditing of a component or system, called here after System Under Auditing (SUA) in execution within a controlled environment (simulated or real). The purpose is the assessment of specific prediction and detailed functional and non-functional properties during the SUA execution. Precondition of the runtime phase is the SUA testing and verification during the design phase. Thus, specific security conditions have been already verified and established.

As detailed in D5.1 section 1.2 [3] the auditing activity focuses on SUA interaction with the Controlled Environment. As detailed D5.1 for assessing the SUA behavior, two different parallel executions will be performed:

- On the right side, the SUA BIECO Controlled Environment is shown.
- On the left side, the execution of the Digital Twin within a Simulation Environment (SE) fed with real-time data can be found. In this case, the DT representation of the device as presented is independently derived from the component specification. DTs are abstract, trusted representations of components that can be executed in a simulation environment.

Thus, during the runtime phase, the Predictive Simulation and the Runtime Monitoring work in synergy, according to a standard and predictive mode, continuously receiving Controlled Environment events. Additionally, the event logger listens to event's passing through the middleware and logs them using a blockchain-based mechanism, aimed at ensuring their non-repudiation.

In the standard mode, the Runtime Monitoring uses the events for matching a predefined set of rules about functional and non-functional properties that the Controlled Environment and the SUA should satisfy. In this case monitoring activity does not rely on Predictive Simulation data. In case of violation an alarm will be risen.

In the predictive mode, the Runtime Monitoring component in parallel assesses the predefined set of rules as described in the standard mode and collaborates with the Predictive Simulation for defining new ones focused on new device or component behaviour predictions.

This effectively represents a key innovation point in BIECO's value proposition, leveraging the monitoring of data from both simulated and real sources to enable the detection of malicious behaviour and empower stakeholders to take timely action.

The upcoming sections explore in further detail the component structure of the runtime phase architecture, followed by the complete specification of the interaction sequences among these components.

5.2. Runtime Phase Components

As done for the design phase, a component diagram depicting the Runtime Phase elements and their interdependencies is provided in Figure 11.

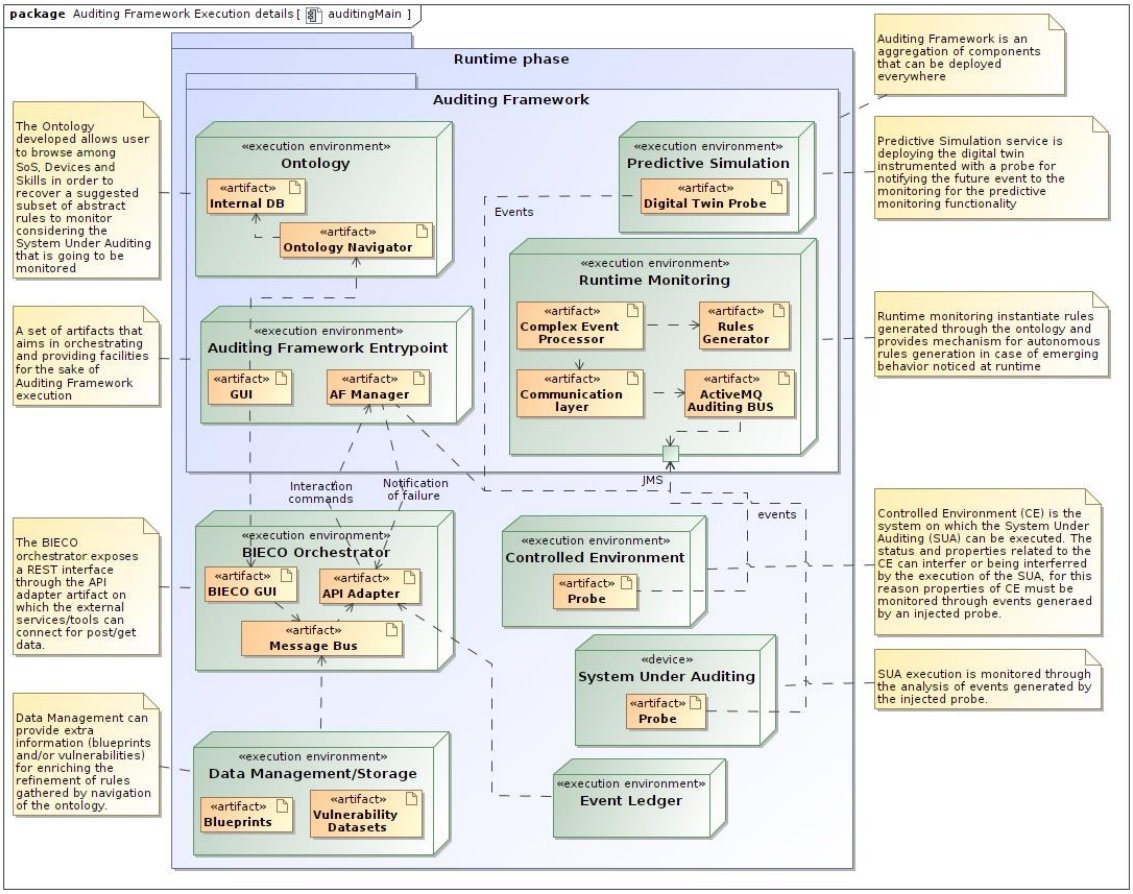

Figure 11 - Runtime Phase Component Diagram. Interfaces have been simplified at node level for readability.

In tandem to what happens at design time, the components encompassed in the runtime phase of BIECO can be separated into two key groups, those that are common to both phases contemplated within the lifecycle, and those that pertain only to the runtime phase. Within the former we can find the Controlled Environment, the Data Management / Storage, the BIECO Middleware (i.e., the orchestrator) and the Graphical User Interface. Concerning the latter the Event Logger and the Auditing System Framework [6] with its runtime monitoring, predictive simulation and Audit System Manager are considered.

As previously mentioned, while the controlled environment is deployed in both phases of the BIECO lifecycle, at runtime it should be instrumented with the probes, which represent artifacts that enable the capturing of real-time events relevant for the auditing package (i.e., runtime monitoring and predictive simulation). Also, at this stage the Data Collection Tool should not only facilitate the access to publicly available datasets of known security risks and vulnerabilities, but also contain within it the blueprints from design time, effectively acting as the bridge between pre-deployment / design time components and the runtime ones. This is also key during the setup of the runtime phase, as it allows the user to model some of the rules for monitoring using this information.

For completeness, this section shortly summarizes the architectural detail of the Auditing framework system extensively described in section 6 of D5.1 [6]. As in D5.1, the Auditing System Framework includes Predictive Simulation Component (Section 6.1 of D5.1) and then Runtime Monitoring one (Section 6.2 of D5.1). Here below an extract taken form D5.1 is reported.

General overview of the Predictive Simulation: Within a digital ecosystem, a system receives a new software smart agent which interacts with other software smart agents, systems and system components within the ecosystem. The software smart agent is typically received as a black box, and it executes on one platform within the ecosystem. Building trust in this black box requires reputation from a trusted source. For building trust, the Predictive Simulation approach follows a set of steps.

1. First of all, the software smart agent is received by a system together with its corresponding DT. The DT are executable descriptions of the algorithm that can be controlled in a simulated environment. Complementary to the algorithm, the DT defines an acceptable behaviour range for the combination of input and output values and the internal state of the algorithm.
2. Then, the Predictive Simulation validates both the correctness and the trustworthiness of the smart agent by evaluating its DT behaviour in the context of a simulation. The DTs execution shows a projection of the behaviour of the smart agent's control algorithm in all situations. This projection yields an abstracted behaviour that reflects the control algorithm's behaviour with bounded accuracy. In this way, the process of building trust in the smart agent does not require software execution on a system, but merely evaluation of the behaviour of the DT in a secured virtual environment.

General overview of Runtime Monitoring: This component is in charge of setting up and managing monitoring activity both in the standard and predictive mode. The Runtime Monitoring is based on event messages. In particular, it enables the collection of specific events that flows during controlled environment, real execution, and Predictive Simulation among the different virtual and real entities (e.g., DT, sensors and ecosystem

components) and infers one or more complex events about the runtime execution (e.g., Complex Event Processing (CEP)).

Complex events inference is based on a set of derived rules, using a ”if-then-else” grammar structure, that define sequences of attended or unattended event patterns.

Details about the general structure of this component, the events, the probes that feed the Complex Event Processor and examples of rules for enact monitoring activities will be provided in Section 5.3.

Referring to D5.1 [14] for an extensive description of the Auditing framework both from an architecture, behavioural and implementation point of view in the section the additional Runtime components and their interaction with the Auditing system is described.

5.3. Runtime Phase Flow

BIECO’s Runtime phase includes two different core steps: the Runtime Setup and the Runtime Execution.

The Runtime Setup includes the configuration of the Auditing System Framework, the Controlled Environment, and the Event Logger. After this setup phase concludes, the Runtime Execution can properly start, marking the beginning of the runtime auditing to ensure the safety and trustworthy behaviour of the SUA.

The main involved flows are reported in the section below, followed by a discussion of usage patterns possible for the runtime phase, accounting for the deployment of only a subset of its components.

5.3.1. BIECO for Runtime Auditing

During the Runtime Setup the initialization of all the involved components is performed.

Once the Design Phase concludes and BIECO’s runtime phase is due to start, a notification is sent by the user to trigger the pre-setup of the following phase. At this stage, the refinement of the initial auditing rules, as well as the retrieval of the domain specific language for the specification of the digital twins take place. After this, several steps can occur offline, which include the creation of the digital twins and the instrumentation of the CE/SUA with the probes. For this reason, BIECO needs to support the freezing/resuming of the ongoing setup session.

Finally, the components which will be running during the Runtime Execution are activated via the orchestrator and the runtime phase can begin. A sequence diagram detailing the involved interactions is presented in Figure 12.

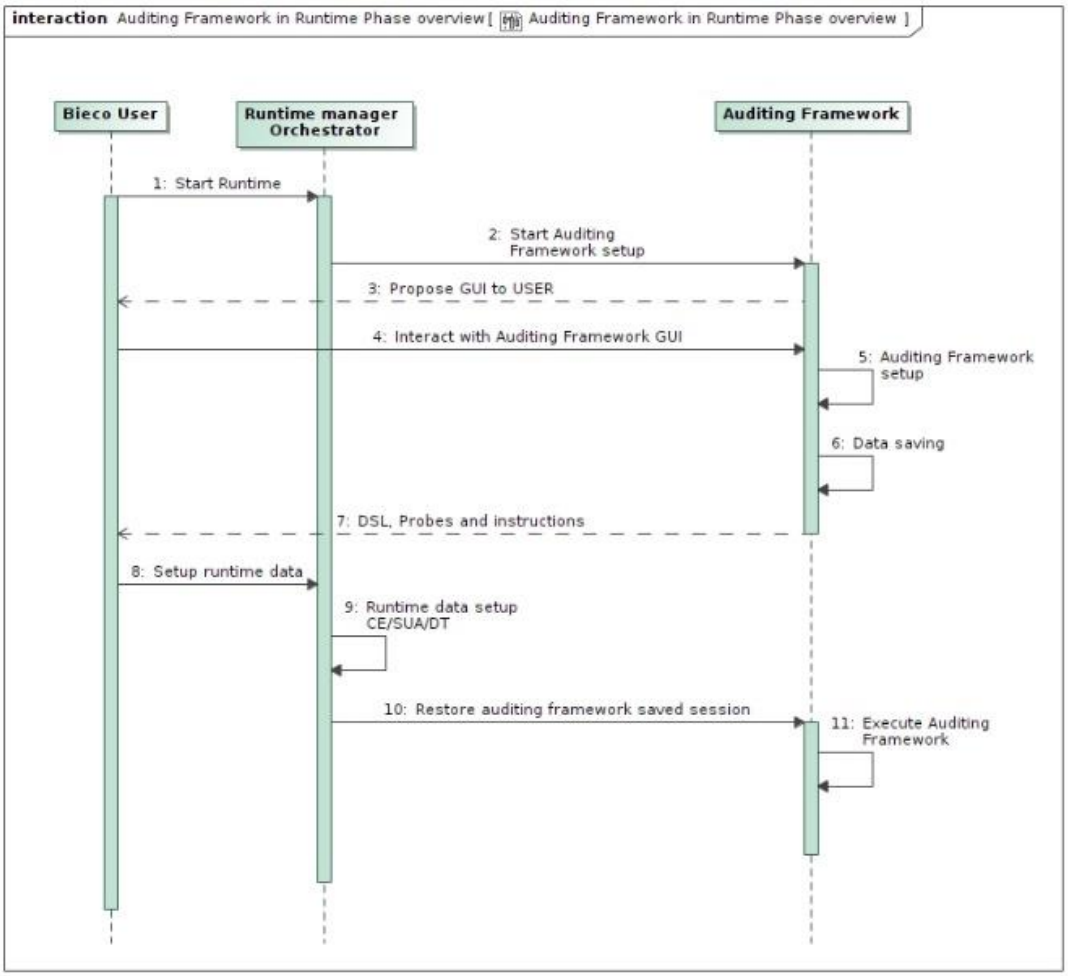

Figure 12 - Overview of the Auditing Framework in the runtime phase, from its setup to execution.

As shown, after the activation of the Runtime Phase by the user (message 1), if the selected execution pattern will require it, the Auditing Framework Setup will be enacted (message 2).

Through a GUI exposed within the BIECO Orchestrator, the user may execute the operations needed to setup the Auditing Framework: browsing the ontology for getting the desired rules subset to monitor, getting probes information or artifacts for instrumenting CE, SUA and DT and get the DSL related to the Digital Twin that he/she needs to instantiate.

More details about those processes and data are described in deliverable D5.1 [14]. The information acquired and managed during the "*Auditing Framework Setup*" phase (message 5), can be saved (message 6) for being recovered after executing offline the operations related to the instrumentation with probes of the CE/SUA and the setup of the DT through the DSL development.

With message 8, the user can set data related to the CE/SUA and DT on the Orchestrator and restore the previously saved session (message 10) in order to modify or confirm the subset of rules selected for the monitoring procedures.

Once confirmed or updated, those rules can be executed by the monitoring platform component of the Auditing Framework (message 11).

In particular, as described in D5.1 the selected properties are translated into executable monitoring rules.

As mentioned, beyond the Auditing Framework, at this stage BIECO also requires the orchestrator to facilitate and coordinate the data exchange between the different components, as well as the presence of the Data Management / Storage component. The latter is responsible for providing access to the blueprints from the design phase and static information required for the runtime setup phase, which will enable for instance the refinement of the auditing system's rules.

During the Runtime Execution step, the Real System is executed into the Controlled Environment. In parallel inside the Auditing System framework:

1) the Predictive Simulation component sends the DT predictions to the Runtime Monitoring for the definition of the prediction rules.

2) Events related to the execution of the Real System and Controlled Environment are captured by the probes and sent through the BIECO middleware/orchestrator to the Auditing System for the rules evaluation.

The Runtime monitor component of the Auditing System Framework compares the prediction with the received events and detect the possible violations. In case of rules violation, the corresponding alarm notification is sent to the BIECO middleware/orchestrator for its management.

5.3.1.1. Predictive Simulation

The subprocess for the predictive simulation component is described herein. As a starting point, a sequence diagram specifying its key interactions and events is presented in Figure 13.

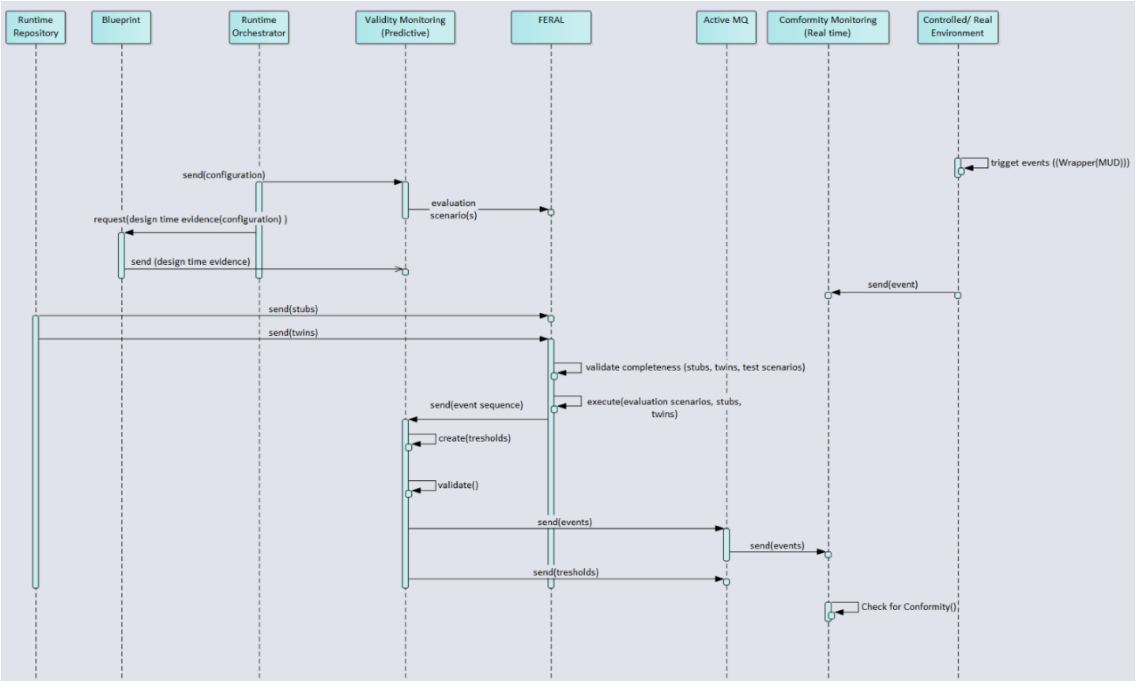

Figure 13 - Sequence Diagram focused on the execution of the Predictive Simulation

As depicted in Figure 13, for a system operating in open context, the runtime orchestrator sends a configuration that needs to be validated. This configuration contains models that enable creation of internal & external context awareness. The validity Monitoring then sends test scenarios to the simulation execution engine. Afterwards, the runtime Orchestrator requests design time evidence for the configuration that is virtually evaluated. And in case there is evidence of this particular scenario, the Blueprint provides the evidence to the Validity Monitoring. Then, the runtime repository sends the stubs and the Twins to the Simulation Engine. Here, an initial sanity check is performed in terms of completeness, then the execution of the evaluation scenarios is performed by exercising the twins and the stubs. The sequence of events is sent to the Validity Monitoring, which creates a threshold of events and values against which the behaviour in the real world is compared.

For enabling provision of artefacts to the interconnected components of the runtime framework, the domain specific language needs to guide the explicit declaration of events that are triggered during an execution along with their types. In this way, the execution of the digital twins will provide a precisely formulated sequence of events that enables the runtime prediction to output artefacts that can be monitored on the system. By instrumenting the definition of the software behaviour of the managed system in a way that it exposes observable artefacts, trusted behaviour signatures can be derived. Then the monitoring component can check the conformity between the real-world execution of the software component and the virtually trusted valid synchronous behaviour and detect deviations. These deviations are indications of a change in the internal and/or the external environmental conditions. In case of unwanted deviations, a reactive feedback loop can be triggered on a single system.

Then the Validity Monitoring performs an internal check, and in case the behaviour is valid, it further on sends these events and the threshold to the Conformity Monitoring via ActiveMQ.

5.3.1.2. Triggering Fail-Over Behaviour

Emerging highly automated autonomous systems are creating a large amount of additional complexity, particularly related to perception, to reasoning and behavioural planning. The emerging complexity that needs to autonomously operate in open context is difficult to formalize. Consequently, the current engineering approaches are missing high levels of confidence for the correct and safe functioning of the systems under all circumstances. One envisioned solution to this situation is through establishment of a redundant parallel channel, that can take the shape of a simplex or supervisor architecture supported by a monitor. In some cases, it is possible to safeguard complex functions by rather simple ones that have the safety responsibility. Then, a redundant safety system can monitor the current risks, and in case of highly critical situations trigger a safety operation.

Within this context, the predictive simulation can enable a dynamical execution of models for looking in the near future. Risk identification can then be extended with a risk mitigation strategy characterized by applying different types of adaptation techniques: parameters adaptation, structure adaptation, or a combination of both. In this way, failures of sensors (e.g., omissions, absence of signals), can be handled by redundant components available in structured adaptation at runtime. For environment

context, parameter adaptation could be triggered through hard breaking, for example. Figure 14 illustrates how this workflow can be envisioned.

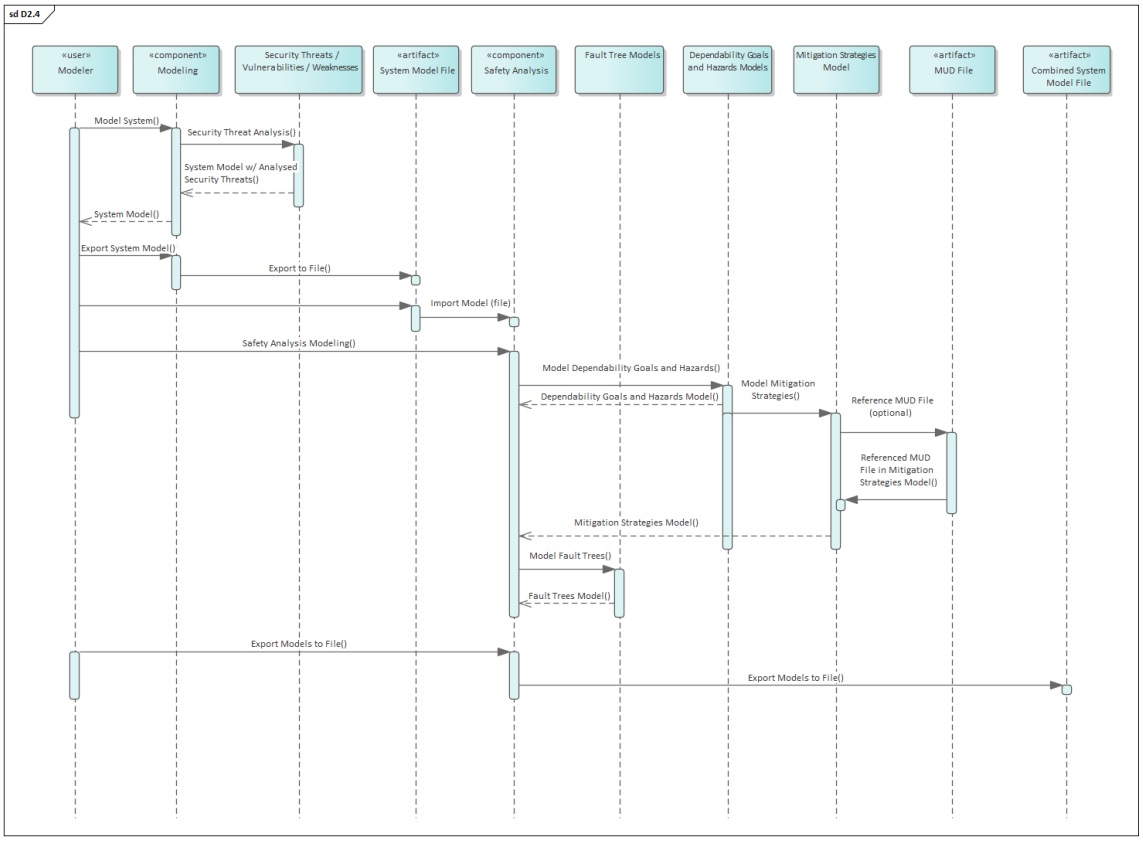

Figure 14 - Simplified Sequence Diagram of Modelling Flow from generic modelling to Safety Analysis to generated Safety-Security Artifacts

The modeler uses the existing BIECO modelling tools to create their models as usual. Security threat analysis can be incorporated to enhance the model with additional information. Once complete, the model can be exported in file format. At a later point in time, the modeler may import the model into the safety analysis toolchain, perform additional safety-specific modelling (e.g., dependability hazard analysis, fault tree analysis, goal specification, or mitigation strategies models). When complete, the updated model can be exported once again in file format, for further processing.

5.3.2. Alternative Usage Patterns for BIECO's Runtime

This section describes alternative usage patterns for BIECO at runtime. These pertain to the case where the user may not desire or need to instantiate all BIECO's runtime components, resulting in only a partial set of BIECO's capabilities being available. Optional components within such patterns are listed in *italics*.

5.3.2.1. BIECO for Runtime Monitoring

Available Components: BIECO Auditing System Framework BIECO Data Management/storage, Controlled Environment, *Real System/Component* BIECO *middleware/orchestrator, BIECO GUI*.

Limited Functionalities: In the BIECO Auditing System the Predictive Simulation component is not involved in the Auditing Set-Up.

During the Runtime Set-up the initialization of all the involved components is performed.

Concerning the Set-up of the Auditing Framework component, As detailed in e D5.1 [14] the Predictive Simulation component is not enacted. Consequently, DT are not instantiated, and the predictive events are not generated. The activity of the Auditing System Framework is limited to the evaluation of functional and non-functional properties defined during the Auditing Set-Up step.

During the Runtime Set up for what concern the Auditing System framework, the selected properties are translated into executable monitoring rules:

During Runtime Set up the probes are instantiated into the Real System/Component and Controlled Environment.

During the Runtime Execution step, the Real System is executed into the Controlled Environment and events related to the execution are captured by the probes and sent through the BIECO middleware/orchestrator to the Auditing System Framework for the rules evaluation. In case of rules violation, the corresponding alarm notification is sent to the BIECO middleware/orchestrator for its management.

6. Architecture Instantiation Example – Autonomous Navigation in Intralogistics

In an effort to provide a reference point for the later instantiations of the architecture to each of BIECO's use cases, a pre-demonstration intended for M18 was planned, focusing on an Industry 4.0 Intralogistics scenario using an autonomous navigation robot system from UNINOVA.

The goals, functional and non-functional requirements pertaining to this pre-demonstration use case are detailed in D2.2 [11]. Nevertheless, the overarching technical goal is to enable the demonstration of the BIECO framework at the project's midpoint, serving as a compass for the initial implementation efforts of both the platform and the ecosystem of tools available at this point.

6.1. Pre-Demonstration Design Phase

Looking first at the design phase, there are five main steps of the BIECO Methodology considered for the pre-demonstration, as illustrated in Figure 15:

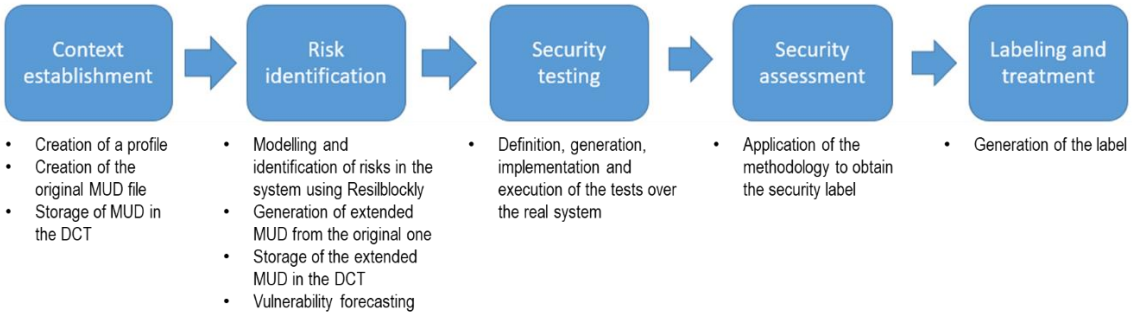

Figure 15 - Pre-demonstration steps for the Design Phase

It starts from the context establishment, which accounts for the creation of a system profile, along with the generation of the initial MUD file and its persistence in the BIECO platform to serve as an input in later stages. Then, the Risk Identification and modelling of the autonomous navigation system take place, enabling the extension of the original MUD file.

From this, the security testing step begins, which consists in the definition, generation of the tests that will be executed against the CE. In the pre-demonstration, at least seven distinct tests are considered, namely:

- **Test1 – Confidentiality1**: Create an item and tasks for the Navigator, generating the plan and velocity commands needed to reach a new position for the robot. Send correct command and analyse if communications are ciphered between the different components.
- **Test2 – Confidentiality2 (depends on test1)**: Create an item and tasks for the Navigator, generating the plan and velocity commands needed to reach a new position for the robot. Send correct command and analyse if ciphering used is strong enough.
- **Test3 – Availability1**: Create an item and tasks for the Navigator, generating the plan and velocity commands needed to reach a new position for the robot. Send

non valid command and analyse if system continues working and manages properly the error.

- **Test4 – Integrity1**: Create an item and tasks for the Navigator, generating the plan and velocity commands needed to reach a new position for the robot. Send modified command and analyse if the system is capable of detecting the modification (MITM).
- **Test5 – Availability2**: DDoS attack based on send/request new velocity commands to the robot. Calculate how many simultaneous requests is capable to process before crashing.
- **Test6 - Confidentiality3**: Update LocalPlanner component in order to check if updates are encrypted or not.
- **Test7 – Confidentiality4 (depends on Test6)**: Update LocalPlanner component in order to check if encryption used in updates is strong enough.

Using the available test results, the Security Assessment step computes the security level and facilitates the last step of labelling and treatment, which outputs the security label of the system to the BIECO platform so that it can be consulted by the user.

The involved BIECO components, their respective instantiated tools, as well as corresponding relevant input and output artifacts are illustrated in Figure 16.

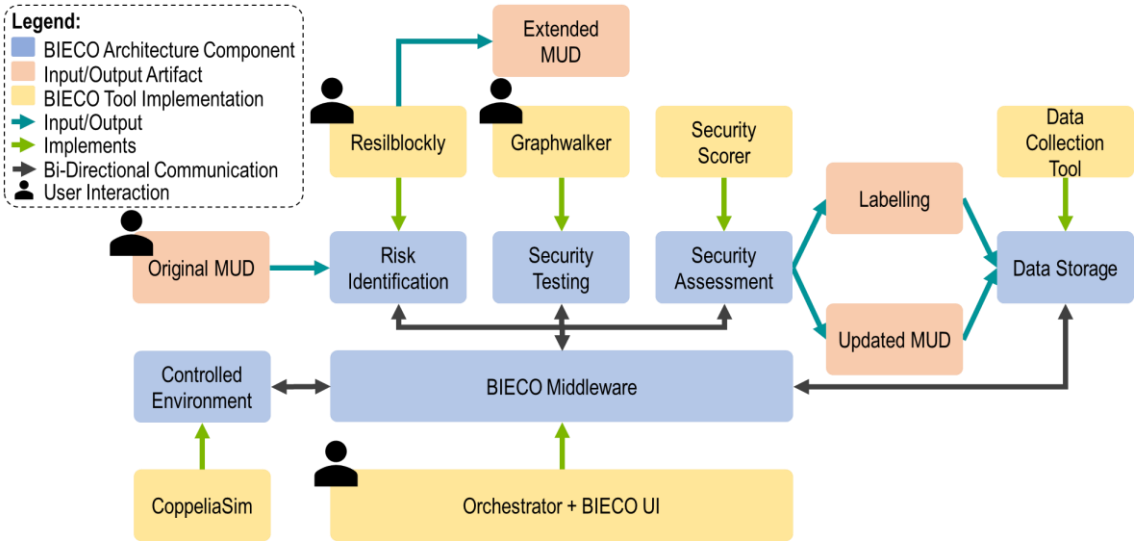

Figure 16 - Architecture Instantiation for the Design Phase

Starting from the CE, in this case it is instantiated as a CoppeliaSim simulation encompassing the intralogistics environment, including the autonomous robots, their navigation system and the specific component under test, which in this case is the local planner of one of the robots.

The CE is made available to the platform through the BIECO middleware, realized by the orchestrator and a graphical user interface being developed within the scope of WP8. The Data Management and Storage component is taken care by the Data Collection Tool developed in WP3, which has been adapted to provide both public and private storage elements, as necessary to match the requirements derived in WP2. Interaction with the user is provided via an iFrame , enabling the different data types to be consulted, including for instance system profile information (e.g., components and dependencies) and software logs.

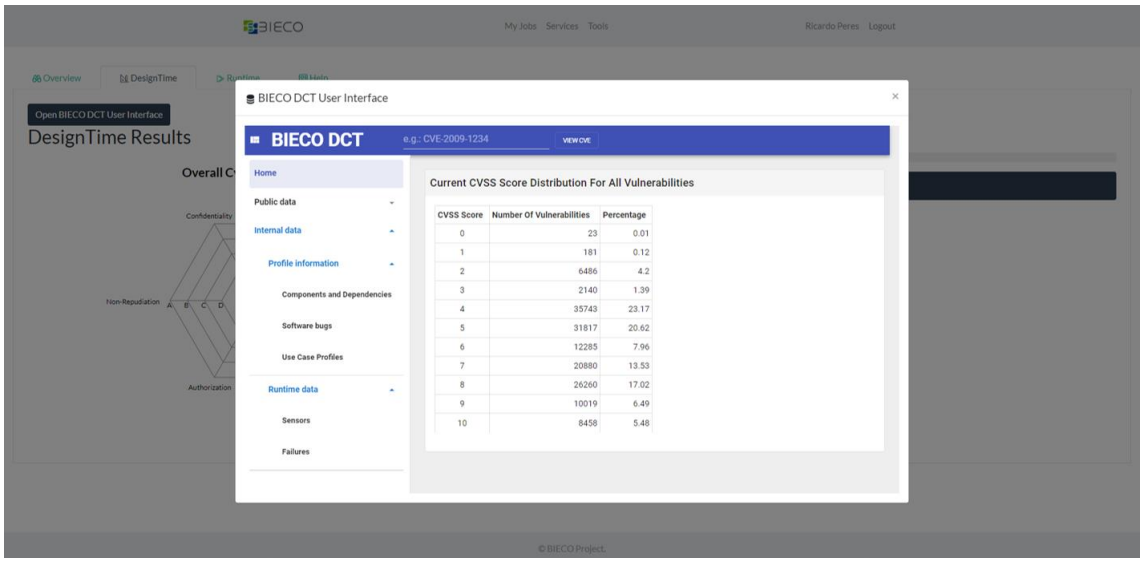

Figure 17 - BIECO GUI for the Data Collection Tool within the platform

The Risk Identification, Security Testing and Security Assessment components are instantiated by Resilblockly, Graphwalker and the Security Scorer Tool, respectively. After the tests are run against the CoppeliaSim CE, the resulting security label is shown to the user as illustrated in Figure 18.

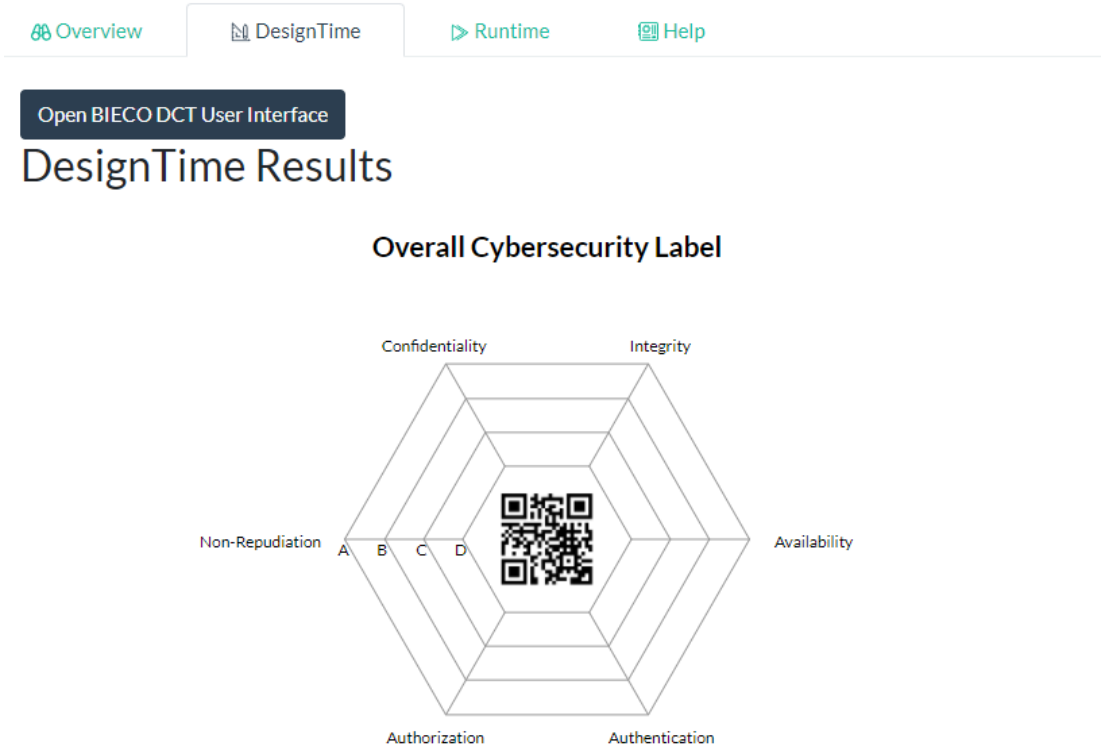

Figure 18 - Example of the security label shown in the BIECO GUI

After these steps conclude, the runtime phase of the pre-demonstration can start.

6.2. Runtime Phase Pre-Demonstration

For the pre-demonstration the components involved at runtime consist in the common set of data storage (i.e., the DCT), middleware (i.e., orchestrator and GUI) and CE (CoppeliaSim), in addition to the runtime specific monitoring and predictive simulation components. These last two are instantiated by the Auditing System Framework, as represented in Figure 19.

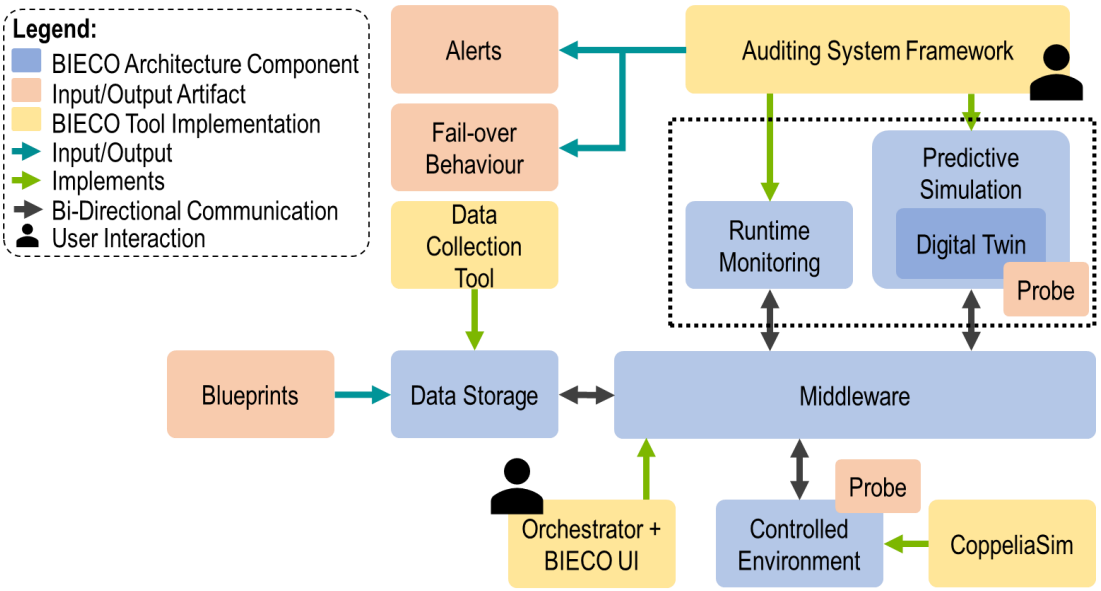

Figure 19 - Architecture Instantiation for the Runtime Phase

After the trigger to start the runtime phase, the required pre-setup can be performed in order to prepare the Auditing System Framework for the execution phase. These steps are facilitated by the BIECO GUI, which once again provides the user with a way to interact and configure the tools in the BIECO platform, this time for the runtime phase, similar to the illustration in Figure 20:

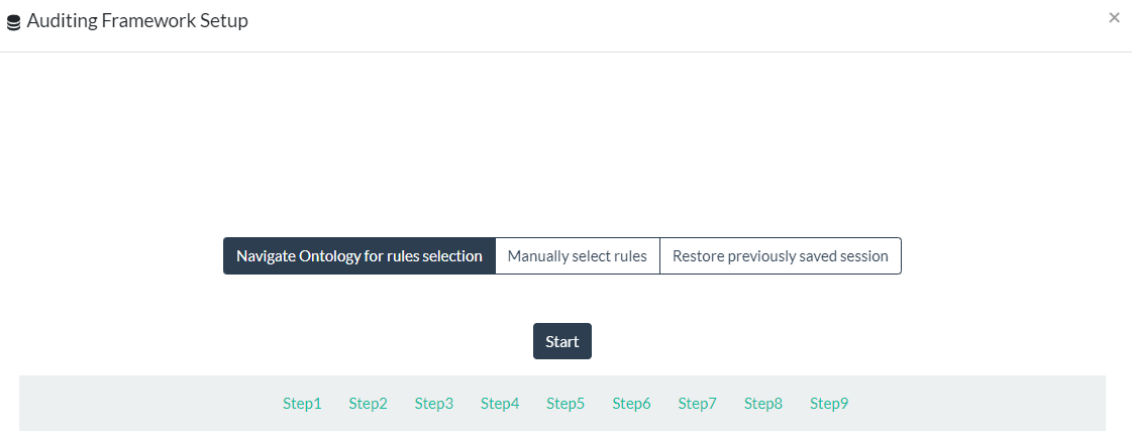

Figure 20 - BIECO GUI for pre-setup of the Runtime Phase in the pre-demonstration use case

After this pre-setup and with the CoppeliaSim CE instrumented with the probes required to monitor the events from the SUA (particularly concerning the navigation system's local planner), the proper execution of the runtime phase begins.

At this stage, event messages related with the robot navigation in the intralogistics scenario are sent to BIECO via the orchestrator and a single endpoint of communication between the CE and the platform, enabling the Auditing System Framework to perform the runtime monitoring of these events as detailed in D5.1, along with the conformity monitoring based on the forecasted events generated by the digital twin.

In case deviations or malicious behaviour is detected, such as the case where one of the robots deviates too far from the expected path, the Auditing System Framework raises an alarm to the user, on top of triggering a fail-over behaviour to bring the system back to a safe and trusted state. For the pre-demonstration, this will be represented by a dynamic reconfiguration of the local planner's parameters, forcing the navigation settings such as velocity and acceleration settings for the robot to be constrained to safety-compliant intervals.

With this, the full lifecycle of BIECO is represented during the pre-demonstration use case, form the early design phase to the runtime auditing and dynamic adaptation of the SUA to ensure its trusted and safe behaviour.

Due to its similarities to the other BIECO use cases, this pre-demonstration will serve support the later instantiation of the architecture to the remaining use cases during the second half of the project.

7. Conclusion

This deliverable presented the final version of the BIECO architecture, being a direct follow-up to D2.3 which put forth the first draft of the design for the overall BIECO framework. The present document formalized the architecture, with its constituent components and their respective interactions, as a result of the maturation stemming for collaborative efforts of the BIECO consortium in the developments and activities of the project from M6 to M18.

A recap and contextualization of the BIECO conceptual framework were provided, along with their alignment with the requirements, both functional and non-functional, resulting from the elicitation process of the earlier activities of WP2.

One of the main contributions of WP2 and the core artifact of T2.3 in particular is the formalization of the BIECO architecture, which was broken down in the two main lifecycle phases contemplated in the project, namely the design and runtime phases. Each was detailed in terms of its components, with their interactions being divided into the main flows that comprise each phase of the lifecycle.

Furthermore, due to the modular and loosely coupled nature of the BIECO architecture, alternative usage patterns are envisioned beyond the deployment of the entire solution, as required on a use case by use case basis. As such, partial deployments of the BIECO solution for each phase are also discussed, including the dependencies between components and the potential resulting functionalities that can be made available to the user in such scenarios.

Finally, in line with the overarching goal of WP2 which is to serve the general guideline for the development activities of BIECO, an example of a possible instantiation of the BIECO architecture was also presented, using as a basis the pre-demonstration case of M18. The objective here was two-fold, on the one hand it was aimed to provide a reference point to guide the instantiation of the BIECO architecture for the other use cases, while on the other hand the aim was more practical. Given the timing of T2.3 in relation to the remaining activities and the overall duration of BIECO (being that it terminates at the projects midpoint), while it was not initially planned to have such a pre-demonstration, this exercise served as the perfect opportunity to iron out some of the unforeseen technical difficulties and kickstart the integration process between the different components earlier, which the consortium hopes will ultimately lead to an easier and more successful deployment in the projects main use cases and beyond.

8. References

[1] B. Consortium, “Deliverable 2.3 - Overall Framework Architecture Design (1st Draft),” 2021.

[2] G. Culot, F. Fattori, M. Podrecca, and M. Sartor, “Addressing Industry 4.0 Cybersecurity Challenges,” *IEEE Eng. Manag. Rev.*, vol. 47, no. 3, pp. 79–86, 2019, doi: 10.1109/EMR.2019.2927559.

[3] V. Mullet, P. Sondi, and E. Ramat, “A Review of Cybersecurity Guidelines for Manufacturing Factories in Industry 4.0,” *IEEE Access*, vol. 9, pp. 23235–23263, 2021, doi: 10.1109/ACCESS.2021.3056650.

[4] National Institute of Standards and Technology, “Framework for improving critical infrastructure cybersecurity,” *Proc. Annu. ISA Anal. Div. Symp.*, vol. 535, pp. 9–25, 2018.

[5] X. Masip-Bruin *et al.*, “Cybersecurity in ict supply chains: Key challenges and a relevant architecture,” *Sensors*, vol. 21, no. 18, 2021, doi: 10.3390/s21186057.

[6] E. Calabro, Antonello and Cioroaica, Emilia and Daoudagh, Said and Marchetti, “BIECO Runtime Auditing Framework,” in *Computational Intelligence in Security for Information Systems Conference*, 2021, pp. 181–191.

[7] B. Consortium, “Deliverable 7.1 - Report on the Identified Security and Privacy Metrics and Security Claims to Evaluate the Security of a System,” 2021.

[8] ISAGCA - ISA Global Cybersecurity Alliance, “Security Lifecycles in the ISA / IEC 62443 Series. Security of Industrial Automation and Control Systems,” no. October, pp. 1–18, 2020.

[9] R. S. Peres *et al.*, “The BIECO Conceptual Framework Towards Security and Trust in ICT Ecosystems,” in *The 33rd IFIP International Conference on Testing Software and Systems*, 2021.

[10] B. Consortium, “Deliverable 2.1 - Project Requirements,” 2021.

[11] B. Consortium, “Deliverable 2.2 - Use Case Definition,” 2021.

[12] B. Unhelkar, *Software engineering with UML*. CRC Press, 2017.

[13] B. Consortium, “Deliverable 6.1 - Blockly4SoS Model and Simulator,” 2021.

[14] B. Consortium, “Deliverable 5.1 - Definition of the Simulation Model and Monitoring Methodologies,” 2021.