FP7-610416-PANOPTESEC

Dynamic Risk Approaches for Automated Cyber Defence

D3.1.2: System High Level Design

Work-Package	WP3	Deliverable	D3.1.2
Due Date	27-03-2015	Submission Date	27-03-2015
Main Author(s)	RHEA		
Contributors	All project participants		
Version	2.0	Status	Final
Dissemination Level	PU	Nature	R
Keywords	Functional/Non-Functional Requirements, High Level Design, Software Architecture, SysML, Architecturally Significant Requirements		

Part of the Seventh

Framework Programme

Funded by the EC - DG Connect

EXECUTIVE SUMMARY

In this document, the High Level Design of the PANOPTESEC System is proposed, along with a design methodology based on Model Based System Engineering techniques.

The PANOPTESEC System Functional and Non-Functional Requirements are analysed in order to extract a High Level Architecture of the System based on the results of the Preliminary High Level Design, by using an iterative Software Architecture methodology.
In [D3.1.1], a first decomposition of the System designed by considering the Functional Architectural Significant Requirements and Architectural Means evaluated with respect of the Non-Functional Architecturally Significant Requirements has been presented.

In the present deliverable, this Preliminary Design has been further analysed in order to consider all Requirements from [D2.2.1] and explore in detail functionalities and behaviour of the PANOPTESEC System. Each of the main components of the PANOPTESEC System depicted in the present document is further detailed in [D4.1.2], [D5.1.2] and [D6.1.2].
The PANOPTESEC High Level Design Project in SysML is presented and it is shown how it maintains consistency across the multiple Designs depicted in [D3.1.1], [D4.1.2], [D5.1.2] and [D6.1.2].

The PANOPTESEC High Level Design is presented by using Views and Viewpoints (Data View, Logic View and Process View) in order to depict a complete high level understanding of the System useful for further analysis in the detailed deliverables. Since in [D3.1.1] the need of an Integration Framework able to fulfil many of the Non-Functional Requirements had been stated, a deep analysis on the Integration Framework architecture within the PANOPTESEC System is presented.

Possible risks related to the current architecture are stated and their possible impact on the Project is evaluated.

HISTORY

Version	Date	Name/Partner	Comment
v0.1	28/08/14	Matteo Merialdo/RHEA	Creation of the document: Proposal of a first Table of Content.
v0.2	15/09/14	Matteo Merialdo/RHEA	High Level Design Block Diagrams added.
v0.3	02/10/14	Matteo Merialdo/RHEA	SysML section added.
v0.4	20/10/14	Matteo Merialdo/RHEA)	High Level Design Sequence Diagrams added.
V0.5	23/10/14	Matteo Merialdo/RHEA	Integration Framework Chapter added with contribution from Gabriela Mihalachi (RHEA).
V0.6	25/10/14	Matteo Merialdo/RHEA	Component View section developed in detail. Various contributions from TUHH, SUPELEC, ALUBL, IMT, EPIST.
V0.7	26/10/14	Matteo Merialdo/RHEA	New Requirements Diagrams added, global restyling.
V0.8	28/10/14	Matteo Merialdo/RHEA	Behavior View completed (Matteo Merialdo), Integration Framework Section improved (by Gabriela Mihalachi - RHEA)
V0.8.1	29/10/14	Matteo Merialdo/RHEA	Minor updates in style, updates in Requirements view section, updates in Visualization Component Views (with contribution of UROME). New Requirements coverage section added.
V0.9	30/10/14	Matteo Merialdo/RHEA	Updates on Requirements Diagrams and Operational Requirements Coverage
V0.9	30/10/14	Gabriela Mihalachi(RHEA)	Final updates on the Integration Framework Chapter, table of images added
V0.9	30/10/14	Douglas Wiemer(RHEA)	TPM comments
V0.9.1	31/10/14	Raniero Rapone(RHEA)	Changes in style and formatting
V0.9.1	31/10/14	Matteo Merialdo/RHEA	Updates on reactive chain management
V1.0	31/10/14	Matteo Merialdo/RHEA	Minor updates and Annexes section added
V1.1	03-03-2015	Matteo Merialdo/RHEA	Initial creation of the document with the new provided template and summary
V1.2	12-03-2015	Matteo Merialdo/RHEA	Chapters 1,2 developed
V1.3	13-03-2015	Matteo Merialdo/RHEA	Chapters 3,4 developed
V1.4	14-03-2015	Matteo Merialdo/RHEA	Chapters 5.1 developed, 5.2 started
V1.5	17-03-2015	Matteo Merialdo/RHEA	QA Review comments included, Section 5.2 concluded
V1.6	21-03-2015	Matteo Merialdo/RHEA	Chapters 5, 6, 7 completed. ANNEXES Completed
V1.7	22-03-2015	Matteo Merialdo/RHEA	Section 4.1 completed, Review conducted
V1.8	25-03-2015	Matteo Merialdo/RHEA	ANNEXES completed
V1.9	26-03-2015	Matteo Merialdo/RHEA	TPM, IMT review comments included. Requirements Coverage Tables added (ANNEX D)
V2.0	27-03-2015	Douglas Wiemer/RHEA	Final edits for grammar checking and formatting.

TABLE OF CONTENTS

TABLE OF FIGURES

LIST OF TABLES

ACRONYMS AND DEFINITIONS

Table 1: Acronym List

Acronym	Meaning
ACEA	ACEA S.p.A.
ALBLF	Alcatel-Lucent Bell Labs France
CIS-UROME	Universita Degli Studi Di Roma La Sapienza
EPIST	Epistematica SRL
IMT	Institut Mines-Telecom
RHEA	RHEA System S.A.
SUPELEC	Ecole Supérieure D'Électricité
SVN	Subversion repository
UoL	Universität zu Lübeck
DRMRS	Dynamic Risk Management Response System
ICS	Industrial Control System
ICT	Information and Communication Technology
ADD	Attribute-Driven Design
ASR	Architecturally Significant Requirement
QA	Quality Assurance
QAM	Quality Assurance Manager
PR	Peer Review
DR	Design Review
DCI	Data Collection Interface
SCADA	Supervisory Control And Data Acquisition
DCC	Data Collection Collector
LLC	Low Level Correlator
RMC	Reachability Matrix Correlator
MIM	Mission Impact Model
AGG	Attack Graph Generator
HOC	High Level Online Correlator
PAI	Potential Attack Identifier
RQ	Risk Quantifier
SRD	Strategic Response Decider
TRD	Tactical Response Decider
PDP	Policy Deployer
ANM	Analysis Module
ASM	Actual Status Monitor
ARM	Attack Response Manager
TPM	Technical Project Manager
WP	Work Package
OIAE	Operational Impact Assessment Evaluator
SIAE	Steady Impact Assessment Evaluator

1 INTRODUCTION

1.1 Context

The PANOPTESEC System is a beyond-state-of-the-art prototype of a cyber-defence decision support system. The aim of the project is to demonstrate a risk based approach to automated cyber defence that take into accounts the dynamic nature of information and communications technologies (ICT), vulnerabilities in *Supervisory Control and Data Acquisition* (SCADA) systems and Industrial Control Systems, and the constantly evolving capabilities of cyber attackers.

1.2 Purpose

The purpose of this deliverable is to describe the evaluation and assessment regarding the PANOPTESEC System Architecture, after the first analysis depicted in [D3.1.1]. It provides a complete decomposition of the PANOPTESEC System, in order to understand the main components of the System and how these components interact in order to fulfil the Operational Requirements (Functional and Non-Functional) described in [D2.2.1].

This document describes the PANOPTESEC System's overall software structure. It also describes the methodology used for depicting the complete System High Level Design and the detailed High Level Design produced within Work Package 4, Work Package 5 and Work Package 6 in deliverables [D4.1.2], [D5.1.2] and [D6.1.2].

1.3 Scope

The scope of this deliverable includes the description of identified architectural constraints that lead the architectural design, due to non-functional Requirements from [D2.2.1], a functional decomposition of the System due to functional Requirements from [D2.2.1] and from the first analysis in [D3.1.1], a summary of the most important architectural decisions taken in order to fulfil architecturally significant requirements.

1.4 Document Structure

This [D3.1.2] deliverable is structured in the following manner:

Section 1 Introduction: describes the context, purpose and scope of the deliverable.

Section 2 Methodology: describes the methodology followed in the development of the deliverable.

Section 3 System Overview: a summary of the PANOPTESEC System, its functionalities and its first architectural decomposition

Section 4 Stakeholders and Architecturally Significant Requirements: Stakeholders for the PANOPTESEC System architecture are identified. ASRs from [D2.2.1] are evaluated and identified as the foundation of the developed architecture.

Section 5 System Architecture: ASRs are analyzed in order to identify functional and non-functional main constraints for the proposed architecture. Results are presented in form of architectural Views in order to depict composition and behaviour of the System.

Section 6 Functional and Non-Functional Requirements Traceability: traceability tables over

requirements from [D2.2.1] are proposed, with rationale.

Section 7 Conclusion: summarizes the findings, results and recommendations.

Section 8 References.

2 METHODOLOGY

2.1 Information collection

The main sources of information for this document are the Project's Description of Work, the Operational Requirements deliverable [D2.2.1], the Deficiency Evaluation deliverable [D2.1.1] and the PANOPTESEC System Preliminary High Level Design [D3.1.1]. These documents highlight the PANOPTESEC System stakeholders' needs, Functional and Non-Functional Requirements of the System and a first decomposition and analysis of the PANOPTESEC System.

PANOPTESEC System architecture depicted in this deliverable is the result of various iterations of the design methodology that has been followed during the development of the overall PANOPTESEC System Design, the Attribute-Driven Design methodology (ADD).

As it is described in [SAiP], ADD is an iterative method that, at each iteration, consists of:

- Choosing a part of the system to design;
- Evaluating and considering every Architecturally Significant Requirement (ASR) related to that part;
- Creating a design for that part and evaluate it over the identified ASRs.

Being an iterative method, it is up to the architects to decide the desired level of detail of the resulting design. Within the PANOPTESEC Project most of the iterations will be carried out by WP3 on in order to define the global architecture depicted in [D3.1.1] and [D3.1.2] (at different levels of details). After the first ADD iterations, a Preliminary High Level Design depicted in [D3.1.1] has been produced. In the current deliverable a further analysis over ASRs and over all Functional and Non-Functional Requirements from [D2.2.1] has been conducted, in order to create a complete High Level Design for the PANOPTESEC System.

Single components of the System will be then designed in details within design iterations carried out by WP4, WP5 and WP6 (with respect of the Specialized Requirements described in [D4.1.1], [D5.1.1] and [D6.1.1]) with the help of WP3.

The result will be a consistent and deep High Level Design of the PANOPTESEC System that will be further detailed during the second year of the Project by single components Low Level Designs.

In order to document the resulting architecture, a careful study of existing standards and software architecture books was made. Among the sources of information: [SAiP] and [DSA].

2.2 Information analysis

The PANOPTESEC High Level Design is developed using various iterations of the design methodology described in Section 2.1. The resulting architecture depicts the PANOPTESEC System at various levels of details by combining different Views of the System.
The foundation of the method is the identification of the ASRs: design iterations can start when a set of ASRs is known.

In addition to ASRs, input the design methodology includes a context description.
These inputs are important because some external constraints can impact the overall architecture. In the PANOPTESEC context, for example, the architecture is developed in order to be able to be feasible not only in the ACEA environment: since it is not known what kind of data input the system will receive, the data collection component will be designed as flexible as possible.

The chosen language for designing the architecture documented in [D3.1.1], [D3.1.2], [D4.1.2], [D5.1.2] and [D6.1.2] is SysML (by using the Papyrus plugin in an Eclipse Environment [Papyrus]).
SysML includes concepts such as an enhanced interface and data flow specifications and integrated Requirements handling. Since an architecture is built over the Functional and Non-Functional Requirements identified for the System (so, over the needs of the stakeholders), SysML's Requirements Diagrams and Matrixes offer a very powerful way to trace Requirements with Design.

The PANOPTESEC Architecture Design is developed with the aim of creating a consistent Design Project covering all PANOPTESEC System components, interfaces and interactions between components at various levels of details.
One of the challenges that the architects of the PANOPTESEC System have to face is the complexity of the Project and the consequent need to have multiple designing teams working on different components. These conditions easily bring to a final lack in consistency that needs to be avoided at any cost. For that reason the PANOPTESEC Consortium put WP3 responsible for maintaining the global consistency across different design iterations in different components designed by different teams.

The developed PANOPTESEC Design Project has a consistent structure composed of various interconnected SysML Projects covering in detail each of the main components of the PANOPTESEC System (there is one SysML Project each for WP4, WP5 and WP6) and a single *PANOPTESEC High Level Design Project* (developed by WP3) covering the overall PANOPTESEC System at a higher level of detail.

The PANOPTESEC High Level Design Project directly derives from the *PANOPTESEC Preliminary High Level Design Project* analysed in [D3.1.1].
Requirements (both Functional and Non-Functional) are modelled in referenced SysML Projects and every Requirement will be traced with the Design, in order to be able to manage changes and updates (and validate these changes) both in the Design or in the Requirements.

The *PANOPTESEC High Level Design Project* is the glue of the overall PANOPTESEC Design environment. Here each main component of the PANOPTESEC System is described (to the level of detail of the sub-components), with the aim to maintain a global consistency across various projects and various designing teams. A PANOPTESEC Design Object Model is then created, with an High Level Data Model that will be common for all the sub-design Projects:

this Design Object Model remains consistent across each single main PANOPTESEC component design (WP4, WP5 and WP6 design). Updates on each single main component Design Project is then validated on the overall PANOPTESEC High Level Design Project and, if needed, is reflected on the other Design Project: every component, even if designed by different teams, remains consistent with the others. The result is a very strong Design Object Model of the PANOPTESEC System that covers various levels of details.

These features add a high level of consistency to the overall Design) and require, of course, a constant work carried out by WP3 in order to ensure interactions and communications between WPs during all the design phase.

In addition, each Design View of a SysML Design Project like the PANOPTESEC High Level Designs project is consistent with every other View (Sequence Diagrams use the same Design Elements defined in Block Diagrams, for example).
Even if the PANOPTESEC Consortium is distributed across many teams, with the PANOPTESEC High Level Design a common language and a common understanding have been reached, with a high level of consistency that will be the foundation for the development phase.
WP3 organized regular meetings (both via video-conference and physical) with all the partners involved in design activities (WP4, WP5 and WP6).

2.3 Synthesis of results

The results of the PANOPTESEC Consortium efforts in the developing of a consistent architecture will allow the Project implementation phase rely on a deep and coherent foundation. The PANOPTESEC High Level Design is depicted in [D3.1.2], [D4.1.2], [D5.1.2] and [D6.1.2], but the SysML Projects will remain the source of information for the entire Consortium, due to the limited graphical expressiveness of the deliverable on dealing with big and complex diagrams.

In order to provide a useful documentation of the architectural efforts and results within the PANOPTESEC Project, WP3 decided to rely on the widely used concept of Architectural Views. As described in [SAiP], *a software architecture is a complex entity that cannot be described in a simple one-dimensional fashion. A view is a representation of a set of elements and relations among them: documenting architecture is a matter of documenting the relevant views and then adding documentation that applies to more than one view.*

In order to establish what the relevant views of the System are it is fundamental to consider stakeholders needs. The set of Views chosen in order to depict the PANOPTESEC System will be then coherent with the desire and the needs of the PANOPTESEC System stakeholders (or at least, of a set of these stakeholders).

2.4 Change Management

2.4.1 Introduction

One of the aims of the WP3 within the PANOPTESEC Consortium is to ensure a valid architecture change management process able to handle and keep track of updates and

changes, from Requirements to the Low Level Design of the components, through the entire life of the Project. SysML language helps the designers with a set of elements that allow the complete tracking of the Requirements with the Design (*Trace links, Satisfy links* and *Verify links*). Change management is dealt by managing the traceability between Requirements (Section 2.4.2), between Requirements and Design Elements (Section 2.4.3), between Design Elements and other Design Elements (Section 2.4.4). All these processes will be managed under the global PANOPTESEC change management process governed by WP1.

2.4.2 Change requests and approval

Changes in the design are likely to occur during the project's lifecycle. A general Change Management procedure has been established in the Project Handbook ([PH15]) and will be applied also to Design change requests. WP3 (and then, WP7) is responsible for managing all Design Projects for the PANOPTESEC System. If a change in the design is needed, WP3/WP7 Leader will present a request for change to the TPM using a Change Request Form.

The TPM reviews the request in collaboration with the user agency. If it is deemed required, beneficial and appropriate, the TPM assesses the impact to the project in collaboration with the WP Leaders. If the TPM and WP Leaders determine it is a minor change and within the scope of the project, meaning it can be implemented within the planned budget, approval is given to the project team without going through the remainder of a formal process for change request approval. Additional approval is not required for each minor change. The request is logged in the Change Request register.

If the TPM and WP Leaders determine it is a major change, meaning it would significantly affect the planned scope, budget, or schedule of the project, and require updates to baselines, the formal change control continues, seeking formal approval by the Steering Committee.

2.4.3 Requirements Diagrams and connections between Requirements

As depicted in [D2.2.1], [D4.1.1], [D5.1.1] and [D.6.1.1], SysML Requirements diagrams are the foundation of the PANOPTESEC Architecture. All Requirements (Functional and Non-Functional) are depicted in Requirements Diagrams and connections between them are designed and can be accessed via Requirements Matrixes. Following that path, if a Requirement needs to change, it is possible to easily find if any other Requirement has a connection or a dependency with it: changes can then be evaluated and tested with respect of the other Requirements. These processes are managed by WP2 and WP3.

2.4.4 Connections between Requirements and Design

Since the global architecture is built over the ASRs (usually Non-Functional Requirements are also ASRs) and then refined in order to cover most (if not all) of the non-ASRs Requirements (usually, most of the Functional Requirements), it is important to link the produced design with the relative Requirements, in order to verify the level of coverage of the design over the Requirements and in order to manage change. If the design changes, it can be validated over the traced Requirements, and vice versa. These processes are managed by WP2 and WP3. *Satisfy* and *Trace* Matrixes are created and updated during the

lifetime of the project. In the Design Deliverables a rationale behind each *Satisfy* link should be provided.

Non-Functional Requirements usually are not easily traced to single design elements, since they regard architectural decisions involving multiple aspects of the design. When needed, a rationale (for example, an Architecture Tactic) is added to the Non-Functional Requirements Matrixes. In Section 6 the traceability and rationale over Requirements from [D2.2.1] over the High Level Design is shown.

2.4.5 Connections between Design Elements

PANOPTESEC architecture is a complex set of projects and diagrams. Since the Design Object model is the same for all the Design Projects, consistency is ensured. When a change request is proposed toward a Requirement or toward a Design Object then it is possible for WP3 to test if the changes can still be validated over the Trace/Satisfy Links among the Requirements and the Design Object Model elements.

2.4.6 Version control

The SysML Projects repository is contained on a Subversion (SVN) server managed by the University of Lubeck (UoL). Each Design Project has a Version that must be included inside the log comments on every commit to the SVN repository, along with a meaningful summary over the main topics of the update. It is the responsibility of the WP3 Leader to ensure requirements and design models are maintained within the repository.

2.4.7 Conclusions

The change management process adopted within the PANOPTESEC Design will ensure that implementation, testing and design will remain consistent through the entire life of the project. Requirements updates and design updates will be manageable by defining traceability links between each element of the Design Object Model. Even if WP2 and WP3 tasks will be concluded after the first year of the project, their work will be carried on by WP7 for last two years of the Project.

2.5 Quality assurance

2.5.1 Quality criteria

The QA in the PANOPTESEC project relies on the assessment of a work product (i.e. deliverable) according to lists of QA checks (QA checklists) established by a QAM, validated at a Consortium level and centralized in the Project Handbook [PH15].
For the purpose of the QA of the D3.1.2, the deliverable MUST be assessed according the following checklists:

- PEER REVIEW (PR) QA CHECKLIST: the D3.1.1 deliverable is a report; it then requires a proper peer review according to the checks defined in this checklist;
- DESIGN REVIEW (RR) QA CHECKLIST: the D3.1.1 deliverable is also a Design document, it then requires the assessment of the checks including in this checklist.

2.5.2 Validation process

For the final validation of work products (i.e. deliverables) within the PANOPTESEC project, a final QA review process MUST be used before the issuing of a final version. This QA validation process follows the Quality Review Procedure established by the QAM and validated by the Consortium in order to guarantee the high quality level of work products and to validate its adequacy according to the defined quality criteria chosen and defined for each deliverable (see Section 2.5.1). The Quality Review Procedure itself and the selection of the QA Review Committee are described in the PANOPTESEC Project Handbook [PH15]. It is specifically detailed in a PANOPTESEC Quality Review Procedure document available on the Project SVN. The QA validation process is scheduled in the QA Schedule [QAS15] managed by the QAM. And, the detailed results obtained after the process took place are captured and stored in the Project log in a Quality Review Summary Report also available on the Project SVN.

3 SYSTEM OVERVIEW

The PANOPTESEC System is a beyond-state-of-the-art prototype of a cyber-defence decision support system, demonstrating a risk based approach to automated cyber defence that accounts for the dynamic nature of information and communications technologies (ICT) and the constantly evolving capabilities of cyber attackers.

Current commercial solutions do not meet the requests of modern networks and systems: in particular, a lack of connection between organizations 'business services and the ICT cyber security field is stated. Organizations have become more and more dependent on networks and ICT system to support their operations: many of them perform critical services and the impact of a cyber-security threat can affect thousands, if not millions of people.

The PANOPTESEC System prototype addresses these challenges by proactively and reactively evaluating system weaknesses, identifying potential attack paths, providing a list of possible mitigation actions and acting in order to enforce these actions in to the system.
The PANOPTESEC System prototype continuously monitors the system and provides response capability to prevent, detect, manage and react to cyber incidents.

The resulting structure of the PANOPTESEC System prototype should be modular: an integrated set of components and technologies that collectively deliver the required functionalities. Figure 1 provides a Logic View of the PANOPTESEC System.

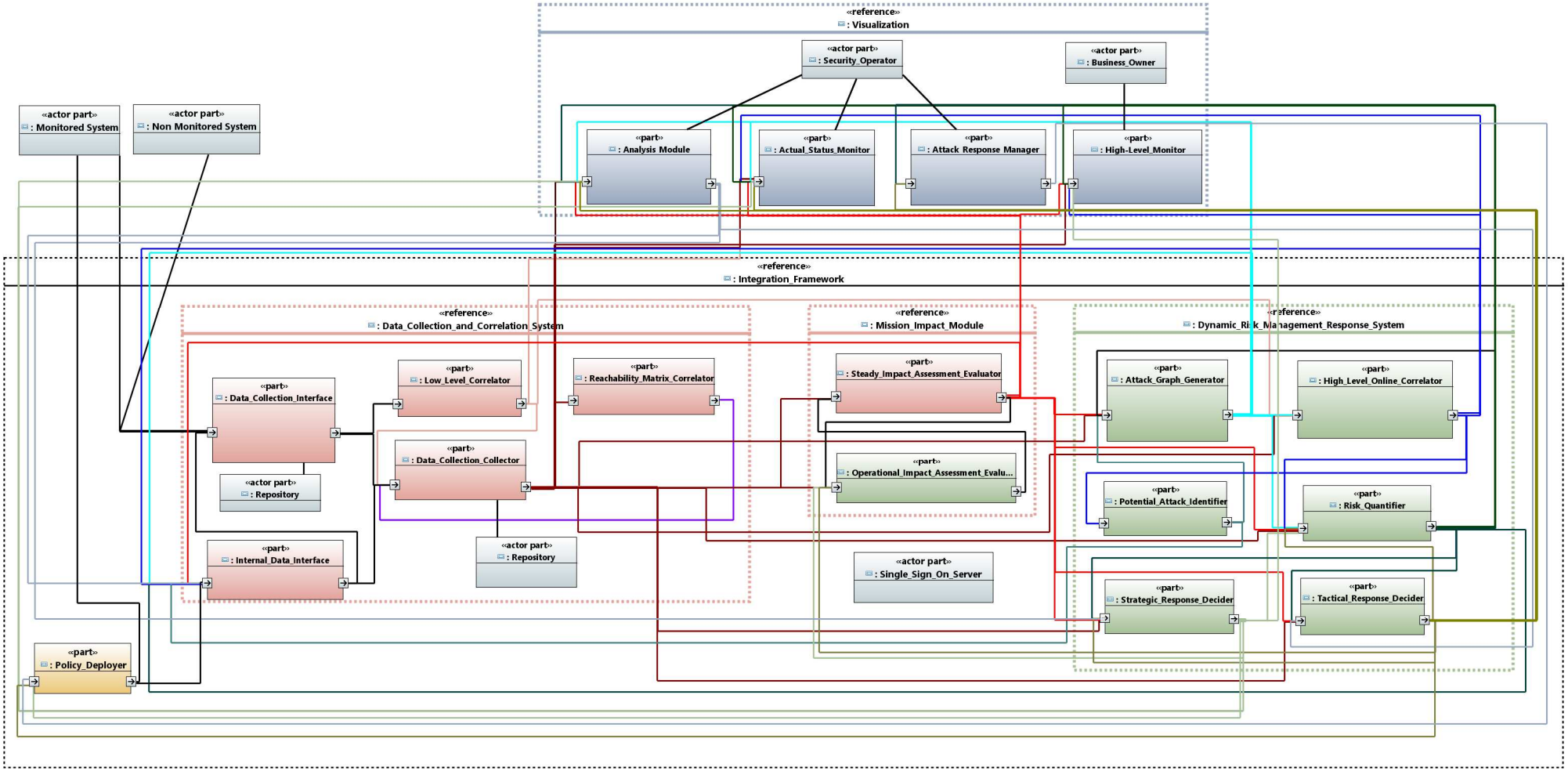


Figure 1: PANOPTESEC System Logic View

The PANOPTESEC System is then composed of five main components:

- Data Collection and Correlation System (developed in Work Package 4)
- The Mission Impact Model (developed within Work Package 4 and 5)
- Dynamic Risk Management Response System (developed in Work Package 5)
- Visualization System (developed in Work Package 6)
- Integration Framework (developed in Work Package 7)

3.1 Data Collection and Correlation System

Data Collection and Correlation System (described also in [D4.1.2], while Specialized Requirements are in [D4.1.1]) is responsible for collecting, normalizing and processing data from the Monitored (for example, an ICT or SCADA/ICS network) and from non-Monitored System(s). Inputs for the Data Collection and Correlation can be any kind of raw source from the System(s): the component is responsible for processing these data and computing Network Inventory, Reachability Matrix and Vulnerability Inventory (among many standardized PANOPTESEC objects produced by the Data Collection and Correlation System and analysed in Section 5.2.3), needed by the rest of the PANOPTESEC System in order to compute the reactive/proactive chain. The PANOPTESEC Data Collection and Correlation System includes various correlation processes, each of them with a specific purpose. In particular, a subset of the information is translated into an abstract ontology language to facilitate the correlation tasks to obtain the Reachability Matrix. In the Reachability Matrix Correlator component, various procedures are carried out to extract knowledge needed to compute the Matrix.

During the implementation phase, the Data Collection and Correlation System will work on a strict relationship with the Simulation Environment (depicted in [D7.1.1]).

3.2 Mission Impact Model

Mission Impact Model (described also in [D4.1.2], with Specialized Requirements in [D4.1.1]) represents one of the most promising innovations within the PANOPTESEC Project. The Mission Impact Model has in charge the evaluation of the critical system of the Monitored System(s), in order to focus the efforts of the Dynamic Risk Management Response System. The Mission Impact Model aims to keep a track of ongoing business processes (or missions) inside a company and connect the business world with the underlying ICT or SCADA/ICS world. Besides threats posed by adversaries to the ICT or ICS environment, it further minimizes potential “collateral damage” posed by the PANOPTESEC engine to the business layer.

Inside a company different Missions or Business Processes must be accomplished. Part of these missions or business process might be several ICT or SCADAICS devices, or rather functions provided by them. Without the information, availability or integrity of these functions, a business process might be compromised. The compromise of a business process might prohibit the accomplishment of a business process/mission. The dependencies of functions provided by devices and their uses inside a BP/Mission are provided in a single

Mission Graph. The purpose of the Mission Impact Module is to represent this general business structure and build a bridge towards the underlying ICT or SCADA/ICS environment needed in other modules. It furthers provides a mapping of potential threats against ICT or SCADA/ICS environment that impact the business structure. This is heavily important as one threat might globally affect several business processes.

The Mission Impact Model is responsible for providing these evaluations to the Dynamic Risk Management Response System in order to be considered in the Attack, Risk and Response Modelling phases.

3.3 Dynamic Risk Management Response System

Dynamic Risk Management Response System (described also in [D5.1.2], Specialized Requirements in [D5.1.1]) holds the core Attack, Risk and Response modelling components. The system can work in two different (but converging) perspectives: proactive and reactive.

The **proactive risk management chain of treatment** consists of managing dynamically and continuously the risk during the life-cycle of a system. On a daily basis, new vulnerabilities are discovered, with a continuous growth in the complexity and the dynamicity of today's ICT systems. The intent of the Dynamic Risk Management Response System is then to automate this proactive risk management process, which enables Security Officers to continually update their risk evaluation process and deploy the needed security measures.

The **reactive risk management chain of treatment** acts complementary to the proactive risk management one. Indeed, some of the risks might be accepted without being eliminated. However, during operations, some of the detected attacks may occasionally see their risk rise to an unacceptable level, e.g. when the likelihood of an attack increase drastically towards its success, for instance the attacker used a zero day resulting in the detection of an unforeseen anomaly in the system. It may also happen that new opportunities appear for some attackers and open to them new means to threaten critical assets of the monitored ICT system (e.g. if the filtering rules of a compromised firewall are modified by an attacker). This kind of situation stresses the need of a reactive part for any organization willing to continually and dynamically manage their risks.

By contrast with the proactive risk management system, the reactive risk management system refers to the risk management after the detection of an on-going attacks. Consequently, an insightful decision to deploy an efficient reactive response has to be determined in order to eliminate, or mitigate, the ongoing attack. Then, possible conflicts must be managed to produce a security policy instantiation that complies with both strategic and tactical response requirements before enforcement.

3.4 Visualization System

The Visualization System (described in [D6.1.2], Specialized Requirements in [D6.1.1]), manages all human interactions with the PANOPTESEC System. The goal of the Visualization System is to design, develop, and validate an innovative visual analytics environment for the PANOPTESEC System with the capability of analysing the system model, the attack models, and the actual and historical network data. Moreover, the visual analytics component will

show the proposed and automatic decisions made by the proactive and reactive systems together with the matching between the actual network state and the closest attack models. The final goal of the visual analytics component is to support the network administrator in analysing the network configuration, monitoring the network actual status, analysing the automatic scenario proposed by the system, and supervising the system reaction. The Visualization System will lead the final decision about the proactive and reactive actions proposed by the Dynamic Risk Management Response System.

3.5 Integration Framework

The Integration Framework is responsible for integrating, deploying and maintaining every component of the PANOPTESEC System. The Integration Framework will handle every communication between components and will cover many of the Non-Functional PANOPTESEC Requirements. The Integration Framework represents the foundation and the glue of the PANOPTESEC System.

4 STAKEHOLDERS AND ARCHITECTURALLY SIGNIFICANT REQUIREMENTS

In this Section PANOPTESEC System Stakeholders and Actors (already defined in [D2.2.1]) that has a role in the definition of the architecture are summarised. PANOPTESEC System Architecturally Significant Requirements (ASRs) (e.g. a requirement that will have a profound impact on the architecture), are identified.

4.1 PANOPTESEC System Stakeholders and Actors

PANOPTESEC Stakeholders represent any person or entity that may be affected by or may affect the scope and intended use of the system or software developed in the PANOPTESEC project.

In [D2.2.1] an extended analysis of PANOPTESEC project Stakeholders has been conducted. In the present deliverable focus is on Stakeholders playing a role in the definition of the architecture of the PANOPTESEC System:

- **Technical Project Manager**: This partner stakeholder manages the technical work between the other Partners involved in the various Work Packages. He propose processes that ensure the smooth running of the technical progress, validates the produced results according to the global objectives of the Project as described in the [DoW2013] to ensure technical high quality and consistency, and enforces the technical processes and the technical schedule of the project according to the [DoW2013]. RHEA is the Technical Project Manager Partner stakeholder for the PANOPTESEC project.
- **Project Coordinator**: This partner stakeholder coordinates all the aspects of the work between the other Partners involved in the Project, propose processes that ensure the smooth running of the Project within the Consortium and outside the Consortium, validate the produced results according to the global objectives of the Project as described in the [DoW2013] to ensure high quality, consistency and pertinence, and enforces the processes and the schedule of the project according to

the [DoW2013]. IMT is the Project Coordinator Partner stakeholder for the PANOPTESEC project.

- **Work Package Leaders**: These partner stakeholders coordinate the work between the other Partners involved within a Work Package, validate the produced results according to the technical objectives of the Work Package as described in the [DoW2013]. Their goal is to ensure technical high quality, and enforce the schedule of their Work Package according to the [DoW2013].
- **User Partner**: This partner stakeholder provides to other partners the operational context and requirements (e.g. use cases, scenarios, and experiment dataset and test bed) and controls the appropriateness of the solution proposed by Solution Providers to this operational context and requirements. It also tests the software architecture designed the IT Architect and developed by the Software Developers based on realistic cases. ACEA is the User Partner for the PANOPTESEC project.
- **Solution Providers**: These partner stakeholders propose scientific or technical solutions for which they are is skilled and recognized in their community, to fulfill one or several of the objectives of the sub-system (e.g. Dynamic Risk Management Response System, Visualization, etc.) researched, designed and developed in the purview of each Work Package. Being able to develop new concepts and tools as well as testing them is very important for PANOPTESEC solution providers.
- **IT Architect**: This partner stakeholder is interested in the quality, stability and performance of the PANOPTESEC overall architecture. Particularly, he is responsible of choosing and organizing the Integration Framework that will bind the various components of the PANOPTESEC System together.
- **Software Developers**: These partner stakeholders are interest in the quality, reliability, adequacy and performance of the various components of the system. Since their main focus is on each component, they need to rely on the Integration Framework for integration with the other components. Being able to use adequate and efficient programming languages and IDE is also very important for them.

In [D2.2.1] an extended analysis of PANOPTESEC Actors has been conducted. In the present deliverable focus is on Actors playing a role in the definition of the preliminary architecture of the PANOPTESEC System:

- **PANOPTESEC Security Operators**: Persons that are the primary users of the PANOPTESEC system. Operators have both proactive and reactive responsibilities. Proactive functions include reviewing vulnerabilities within the Monitored System(s), reviewing potential mission impact, reviewing and selecting of mitigation actions and their execution. Reactive functions include reviewing indications of suspected cyber-attacks, reviewing potential mission impacts, reviewing and selecting of mitigation actions and approval of execution.
- **PANOPTESEC System Manager**: A person involved in setup and configuration of the data sensors interface with the PANOPTESEC system as well as any pre-configuration or configuration changes needed by the PANOPTESEC system (e.g., input the list of pre-approved mitigation actions). His focus is on keeping the PANOPTESEC up and running securely and in being able to make it evolve when necessary.

- **Monitored System**(s): This actor includes SCADA/ICS systems (e.g., database and/or SCADA servers, Front End Gateways, etc.) and ICT systems (e.g., Network devices, database servers, Domain Name Servers, computers used to access the infrastructure to perform operational roles, etc.). It is an information source for the PANOPTESEC System that collects numerous pieces of information about its configuration. It also receives mitigations actions from the PANOPTESEC System for deployment.
- **Non-Monitored System**(s): this actor includes any data source for the PANOPTESEC System that is not controlled in any mean by the PANOPTESEC System itself. For example:
    - Public Database of knows Vulnerability (NVD, CVE). The Data Collection System will download vulnerabilities and update from public vulnerabilities databases with a scheduled frequency.
    - Any external service that will be used in order to integrate useful information for the PANOPTESEC.
- **Cyber Sensors**: This actor corresponds to the sources of cyber relevant data (such as configuration scanners, IDS and log generators) within the Monitored System used by PANOPTESEC for security analysis and mitigation action planning and execution.

4.2 PANOPTESEC System Architecturally Significant Requirements

The goal of the PANOPTESEC System architecture design is to build a system that satisfies Requirements of a cyber-defence decision support system, demonstrating a risk based approach to automated cyber defence. Some of these Requirements have a far more profound impact on the architecture, and are commonly defined as Architecturally Significant Requirements, or Architecture Drivers. ASRs are derived from Functional and Non-Functional Requirements for the PANOPTESEC System, defined in [D2.2.1].

ASRs are derived as a combination of:

- Functional Requirements
- Non-Functional Requirements (also defined as Quality Attributes Requirements, due to their direct connection with Quality Attributes)

In essence, the style of the architecture is determined by the Quality Architectural Requirements while the functional instances of the elements types defined by that style are determined by the Functional Architectural Requirements.

Within the PANOPTESEC project, Functional and Non-functional requirements have been evaluated by the Stakeholders (as depicted in [D2.2.1]) in terms of Importance and Reachability. Importance is rated on a scale of 1 to 3, representing a priority ranking (1 being least important and 3 being most important) to guide the implementation of features. Reachability is rated on a scale of 1 to 3 and represents the degree of research, implementation or integration required to achieve the requirement (1 representing only integration of existing components and 3 representing a full requirement for detailed

research, implementation and integration). Complete details of these rating schemes are provided in [D2.2.1].

ASRs are identified by evaluating the Importance of the identified System's Requirements, assuming that these Requirements already had been prioritized by the Stakeholders. In addition, the Architectural Impact is evaluated on a scale between 1 (Low Impact – little effect on the architecture) and 3 (High Impact – profoundly affect the architecture). Requirements with both High Impact and Importance have to be considered among the ASRs.

Each iteration of the design methodology identified in Section 2 considers the highest remaining priorities among the ASRs. At the end of the iterations, all ASRs are evaluated and considered. At that stage, all remaining Requirements are taken into account in order to finalize the architecture. In the current document all [D2.2.1] Requirements are taken into account, and the Preliminary Architecture depicted in [D3.1.1] is further evolved.

The lists of the ASRs identified among the Non-Functional and Functional Requirements in [D2.2.1] are presented in ANNEX A and ANNEX B. In the Preliminary Design presented in [D3.1.1] document only the most relevant fraction of these ASRs is considered (the result of the first design iterations). The current document depicts the complete design considering all ASRs (High Importance and Medium Architectural Impact and Medium Importance and High Architectural Impact) and the coverage over the non-architecturally significant requirements.

5 SYSTEM ARCHITECTURE

This section provides a high-level overview of how the functionality and responsibilities of the System are partitioned and then assigned to subsystems or components. It also provides a rationale in terms of Architectural Decisions in order to justify the architecture over the ASRs previously described. In Section 6 a complete list of Requirements from [D2.2.1] whose satisfaction is justified in this deliverable is presented.

5.1 Architectural Decisions

Documenting architectural decisions is a critical point in an architecture documentation. Architecture decisions are visualized in the results of the architecture design, which are reflected in the architecture documentation.
Important subjects of architecture decisions are:

- Selection of ASRs;
- Selection of key abstractions (significant abstractions of the functional domain that must be handled by the system to be realized). By combining the key abstractions and the most important identified Functional ASRs it is possible to derive the first functional architecture;

- Selection of specific Architecture Means (Architectural Principles, Patterns and Tactics) that fulfil the previously identified ASRs;
- How selected Architectural Means are applied in order to fulfil the identified ASRs;
- How system's building blocks are structured in order to implement selected architectural means.

This analysis had been conducted in [D3.1.1] (also provided in Annex C), where a set of Architectural Principles and Patterns had been identified and applied in order to fulfil the Non-Functional ASRs over the Functional ASRs. The resulting Design is a Preliminary High Level Design (Annex C) used as the foundation of the System High Level Design presented in the current deliverable.

5.1.1 PANOPTESEC System Preliminary High Level Design

The High Level Design presented in this document has its foundation in the preliminary results from [D3.1.1] (also in Annex C), where a set of ASRs were preliminary fulfilled after the first analysis.
In Figure 2 an overview of the resulting Preliminary Design (Logic View) is presented:

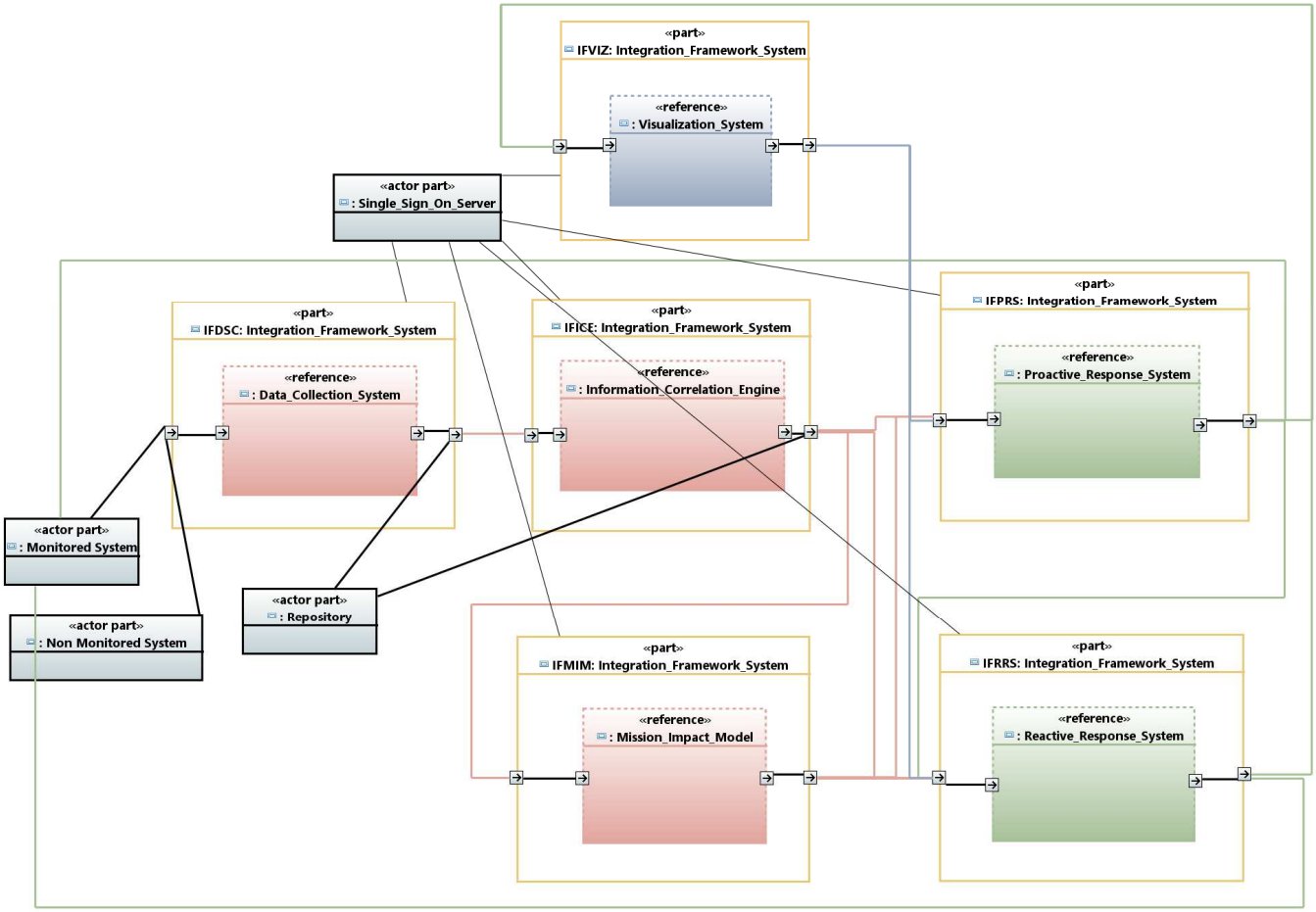


Figure 2: PANOPTESEC System Preliminary High Level Overview from [D3.1.1]

From the Preliminary High Level Design, a set of functional components had been designed, along with a Preliminary High Level Data Model and a set of interactions among them.

In the following paragraphs a brief summary of the functionality of this preliminary decomposition is given:

The purpose of the Data Collection System is to collect raw data from the Monitored and Non-Monitored System(s) (Section 4.1). Topology data, vulnerabilities data and alerts, for instance, are collected from different sources in order to be available for the Information and Correlation Engine computations.

The Information Correlation Engine manages a knowledge base and integrates, normalizes and correlates input coming from the Data Collection System, including historic and actual data. The Information and Correlation Engine computes and provides the necessary input for other components of the PANOPTESEC System.

The Mission Impact Model is responsible for the evaluation of the critical system of the Monitored System(s), in order to focus the efforts of the Proactive and the Reactive Response System.

The proactive risk management chain of treatment (cf. Section 3.3) is managed by the Proactive Response System. The Proactive Response System automates this proactive risk management process, which enables the Security Operators (Section 4.1) to continually update their risk evaluation process and deploy the needed mitigation actions.

The reactive risk management chain of treatment (cf. Section 3.3) is managed by the Reactive Response System.

The Visualization System is a visual analytics environment that analyses current and historical cyber-security situation of a Monitored System(s) protected by a PANOPTESEC System. The Visualization System also shows the mitigation actions proposed by the Proactive and Reactive Response Systems together with the matching between the actual network state and the closest attack model. The Visualization System also supports the Security Operator in analysing the proposed mitigation actions, confirming or refusing such actions.

One of the main consequences related to the evaluation of the top-priority Non-Functional ASRs and the choice of the Architectural Patterns (Annex C) is the need for the PANOPTESEC System to be able to implement a Broker Architecture, to have the capability of building Pipe-and-Filter chains (possibly by using orchestration mechanisms within the technology used in order to implement the Broker) and to use Service Oriented Principles if needed, in order to communicate with external components. This issue is solved by implementing an **Integration Framework**.

Integration Frameworks are used to integrate different technologies, applications and products without needing to develop complex glue code. Connectors, implicit type converters, domain specific languages and Enterprise Integration Patterns are already

implemented in the framework. The PANOPTESEC System Integration Framework manages communications between PANOPTESEC System components, security, and components runtime.

5.1.2 From the Preliminary High Level Design to the High Level Design

After the first decomposition of the PANOPTESEC System, the iterative design method depicted in Section 2.1 had been carried out analysing each of the main components within the PANOPTESEC System Preliminary High Level Design, in order to better decompose the System by pointing out well defined functionalities. Architectural Principles analysed in [D3.1.1] had been applied (see Annex C for a summary from [D3.1.1]), with a particular focus on:

1. **Loose Coupling**: this Principle states that the coupling between building blocks should be kept as low as possible. The less strongly a building block is coupled with other building blocks, the easier it is to understand, design and maintain the building block in order to improve modifiability and compatibility of the system.
2. **Separation of Concerns**: one of the most important uses of this Principle (tightly coupled with the Loose Coupling Principle) is to support modularization. This primarily means identifying parts of a software system responsible for specific concerns, aspects or tasks and encapsulating them as separate building blocks.
3. **Information Hiding**: this Principle is a fundamental concept for structuring and understanding complex systems. This Principle (when applied to software architecture) states that a building block should show to client building blocks ONLY that part of information that is really necessary for the client's tasks. This brings to the definition of well-designed interfaces between components. This Principle is strictly coupled with the Loose Coupling Principle.

The different functionalities of the Data Collection System and the Information Correlation Engine have then been divided in four different components (depicted in details in Section 5.2), inside the main package **Data Collection and Correlation System** (developed by the Work Package 4). While the scope of the Data Collection System remains the same as the Preliminary High Level Design, the functionalities of the Information Correlation Engine had been explored in detail during various design iterations and it seemed necessary to subdivide the component in order to fulfil the selected Architectural Principles.

The Mission Impact Model component and the Visualization System expose well defined functionalities and maintain their scope (except for a change in naming: **Mission Impact Module**).

After the analysis of the Proactive and the Reactive Response System requested functionalities, common behaviours had been observed: while the reactive chain depends on the Attack Paths from the proactive chain, the Risk Quantification functionalities can be carried out by a single component (subdivided in various other components). The Risk Quantification process is if fact a functionality that must be provided by both chains. Due to that, the Proactive Response System and the Reactive Response System functionalities have

been summarized in a main package within the PANOPTESEC System (with a functional internal subdivision), the **Dynamic Risk Management Response System** (developed by the Work Package 5). Even if the Response processes seem similar in both the proactive and the reactive chains, the scope of the produced proactive and reactive Mitigation Actions is different. In the proactive chain, the PANOPTESEC System improves the overall cyber-security assessment of the Monitored System(s), by using a policy driven process that may take some time in order to be computed and deployed. In the reactive chain the PANOPTESEC System is managing one or more on-going attacks and the reaction must be as fast as possible.

In Figure 3 a Block Diagram of the PANOPTESEC System is shown, depicting the functional inheritance between blocks from the Preliminary High Level Design (empty blocks with coloured borders) and blocks from the High Level Design.

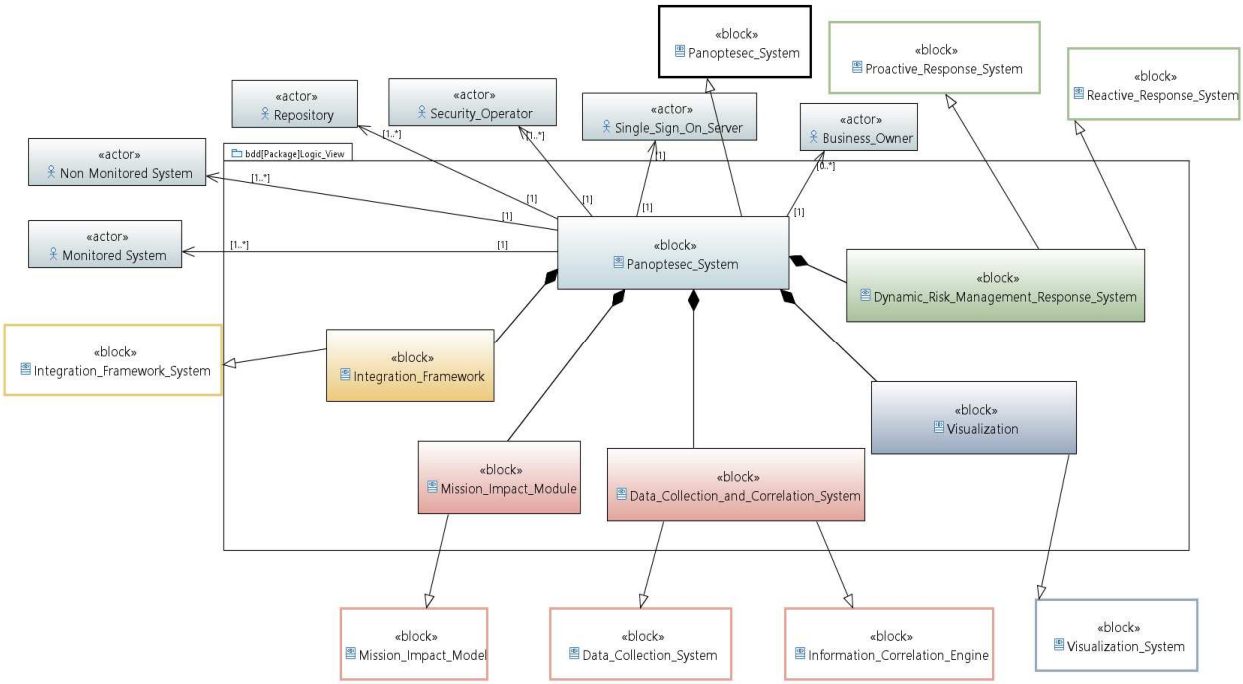


Figure 3: PANOPTESEC System High Level Overview

This evolution from the Preliminary High Level Design encompasses a better understanding of the PANOPTESEC System functionalities. ADD iterations over the main PANOPTESEC identified packages conducted within WP3, WP4, WP5 and WP6 further analysed the needs and functionalities of each sub-component of the main PANOPTESEC packages.

Due to the *generalization* links between blocks of the two projects, a direct connection is created. In addition, the PANOPTESEC High Level Data Model (Section 5.2.4) is directly derived from the PANOPTESEC Preliminary Data Model. In Section 5.2 a detailed decomposition of the PANOPTESEC System is presented.

5.2 Architecture Views

Documenting an architecture is also a matter of documenting relevant Views in which the architecture can be depicted and then adding documentation that applies to more than one View, in order to maintain consistency throughout the overall document. After the development of the Preliminary High Level Design ([D3.1.1]) it is possible to continue the design of the System by following the iterative method depicted in Section 2.1 and collecting the resulting design in Views.

- A View is a representation of a whole system from the perspective of a set of concerns [IEEE 2007].
- A View is a representation of a coherent set of architectural elements, as written by and read by system stakeholders [SAiP].

5.2.1 Architecture Viewpoint

With regards to architecture views, there must be a differentiation between their specification and their concrete use as an instance of their specification. [IEEE 2007] introduced the important concept of the Viewpoint for the specification of Architecture Views:

- [IEEE 2007] differentiates between an architecture view and its specification with regard to the terms and the concepts.
- In [IEEE 2007], a viewpoint corresponds to the specification for a specific architecture view (e.g., Logic View) independently of a specific system.
- In [IEEE 2007], an architecture view corresponds to the instance of its corresponding specification for a specific system.

In order to follow these suggestions, a focus on specific Views is decided, in order to best describe the High Level Design of the PANOPTESEC System. The chosen views for the PANOPTESEC architecture include:

- Requirements View;
- Data View;
- Logic View;
- Process View.

These four Views are consistent and tightly coupled. Each of them has been developed with respect to the others and by using the same PANOPTESEC Modelling Object Model that is at their foundation. A change of a Diagram of a particular View results in changes in the others.

5.2.2 General Notes

The PANOPTESEC System is composed of different components designed and developed by different Work Packages within the Project. Different colours have been used in the Diagrams in order to give a clear distinction between Design Elements managed by different Work Packages. In detail:

- Red elements belong to Work Package 4
- Green elements belong to Work Package 5
- Blue elements belong to Work Package 6
- Orange elements belong to Work Package 7

5.2.3 Requirements View

Below the Viewpoint of the Requirements View for the PANOPTESEC System Design is shown:

- **Purpose**:
  Documentation of the PANOPTESEC System requirements
- **Stakeholders**:
  Technical Project Manager, Project Coordinator, Work Package Leaders, User Partners, IT Architects, Software Developers, Solution Providers
- **Concern(s):**
  What are the essential requirements the system must satisfy?
- **Artefacts:**
  Stakeholders, Requirements

The Requirements View of the PANOPTESEC System High Level Design is depicted in [D2.2.1]. In the present document, High Level Design coverage over the identified Functional and Non-Functional Requirements is provided and summarized in Section 6.
As stated in [D2.2.1] and explained in Section 2.2, all PANOPTESEC Requirements are modelled in SysML project and will be traced to the Design.

5.2.4 Data View

Below the Viewpoint of the Data View for the PANOPTESEC System Design is shown:

- **Purpose**:
  Documentation of aspects with regard to main data structures and data flows in the PANOPTESEC System
- **Stakeholders**:
  Work Package Leaders, IT Architects, Software Developers
- **Concern(s):**
  Which are the data structures and data flows of the system?
- **Artefacts:**
  Data models

5.2.4.1 PANOPTESEC High Level Data Model

The PANOPTESEC High Level Data Model is the skeleton on which all PANOPTESEC High Level Designs in [D3.1.1], [D3.1.2], [D4.1.2], [D5.1.2] and [D6.1.2] are defined. The PANOPTESEC High Level Data Model describes all High Level SysML ValueTypes used in all Logic and Process Views of the PANOPTESEC High Level Design, the Data Collection and Correlation System High Level Design ([D4.1.2]), the Dynamic Risk Management Response System High Level Design ([D5.1.2]) and the Visualization System High Level Design

([D6.1.2]). By developing a common High Level Data Model accepted by all the partners of the consortium, it is possible to work on the different High Level Designs without naming inconsistencies. All interfaces and connectors between the components described in the deliverables are defined on behave of the High Level Data Model.

The High Level Data Model does not define Class Diagrams or low level structures: every ValueType can be considered like a concept, a common and stated understanding of a type of data defined in the system. Deliverables [D4.1.2], [D5.1.2] and [D6.1.2] define some lover level ValueTypes and some Class Diagrams that inherits from these High Level ValueTypes, maintaining the overall consistency throughout the High Level Designs. Even if ValueTypes are a higher level concept than Class Diagrams, they are designed using similar UML rules. Generalization, Composition and Association ([OMGUML]) links are widely used in order to define relationships between ValueTypes.

The PANOPTESEC High Level Data Model is organized with respect to the main component introducing a particular ValueType. For each ValueType we propose a brief explanation and a rationale explaining the Requirement directly traced to it, or the reason behind adding it to the design. Since all PANOPTESEC High Level Designs are based on the PANOPTESEC High Level Data Model, a clear understanding of this chapter is fundamental.
The PANOPTESEC High Level Data Model is described by using Block Definition Diagrams.

In this document a complete version of the PANOPTESEC High Level Data Model is depicted: since the PANOPTESEC High Level Design derives directly from the PANOPTESEC Preliminary High Level Design, the complete PANOPTESEC High Level Data Model derives, and includes, the Preliminary High Level Data Model.
This ensures consistency between the Design Projects: the PANOPTESEC High Level Design can be seen as a much more detailed version of the Preliminary Design.

The main source of the PANOPTESEC High Level Data Model is the Requirements list (Annex A and Annex B with the identified ASRs and [D2.2.1] for the complete list). Some Data Models had been created after the ADD iterations over the identified components. A rationale will be given for each of the Data inside the High Level Data Model.

In Figure 4 common ValueTypes (common concepts for the overall PANOPTESEC High Level Design) are depicted (from the Preliminary High Level Data Model):

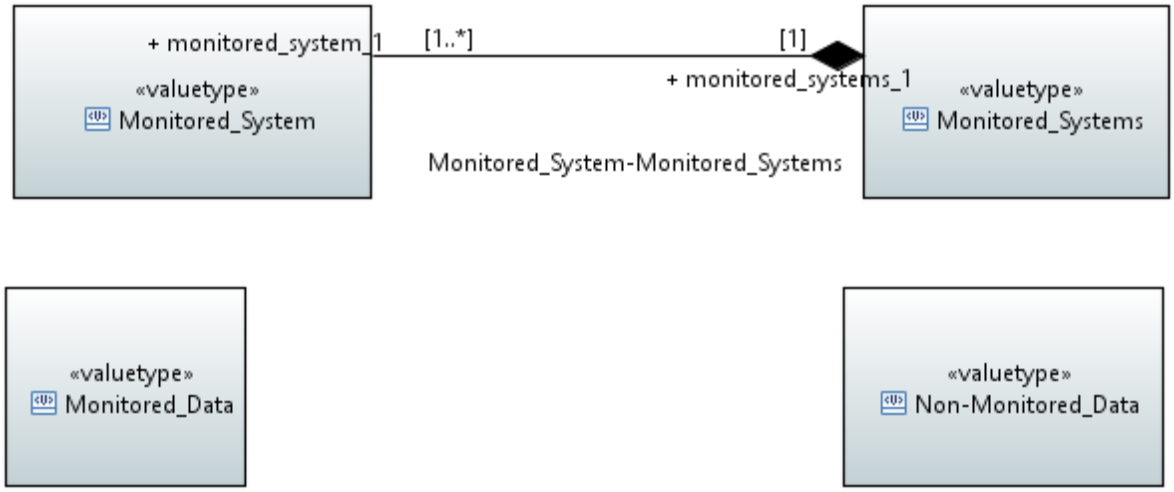


Figure 4: PANOPTESEC System common ValueTypes

Monitored_Data
This ValueType represents any kind of raw data coming from the Monitored System(s). For example: firewall logs, topology scanners data, and vulnerability scanners data. In that sense, the meaning of *raw data* can be stated as *data not standardized in a PANOPTESEC data format.*

Non-Monitored_Data
This ValueType represents any kind of raw data coming from Non-Monitored System(s). For example: vulnerability information from a public database.

Monitored_System
This ValueType represents a Monitored System. Of course in this context the meaning of Monitored System is no more related to the concept of *actor* of the PANOPTESEC but to the concept of a *collection of information identifying a particular Monitored System with which the PANOPTESEC System interacts*. This ValueType is needed in order to model and identify a single Monitored System that the PANOPTESEC System is interacting with.

Monitored_Systems
This ValueType represents a collection of Monitored Systems.

In Figure 5 and Figure 6 ValueTypes related to Data Collection and Correlation System are depicted. ValueTypes from the Preliminary High Level Data Model are depicted in empty blocks with red border.

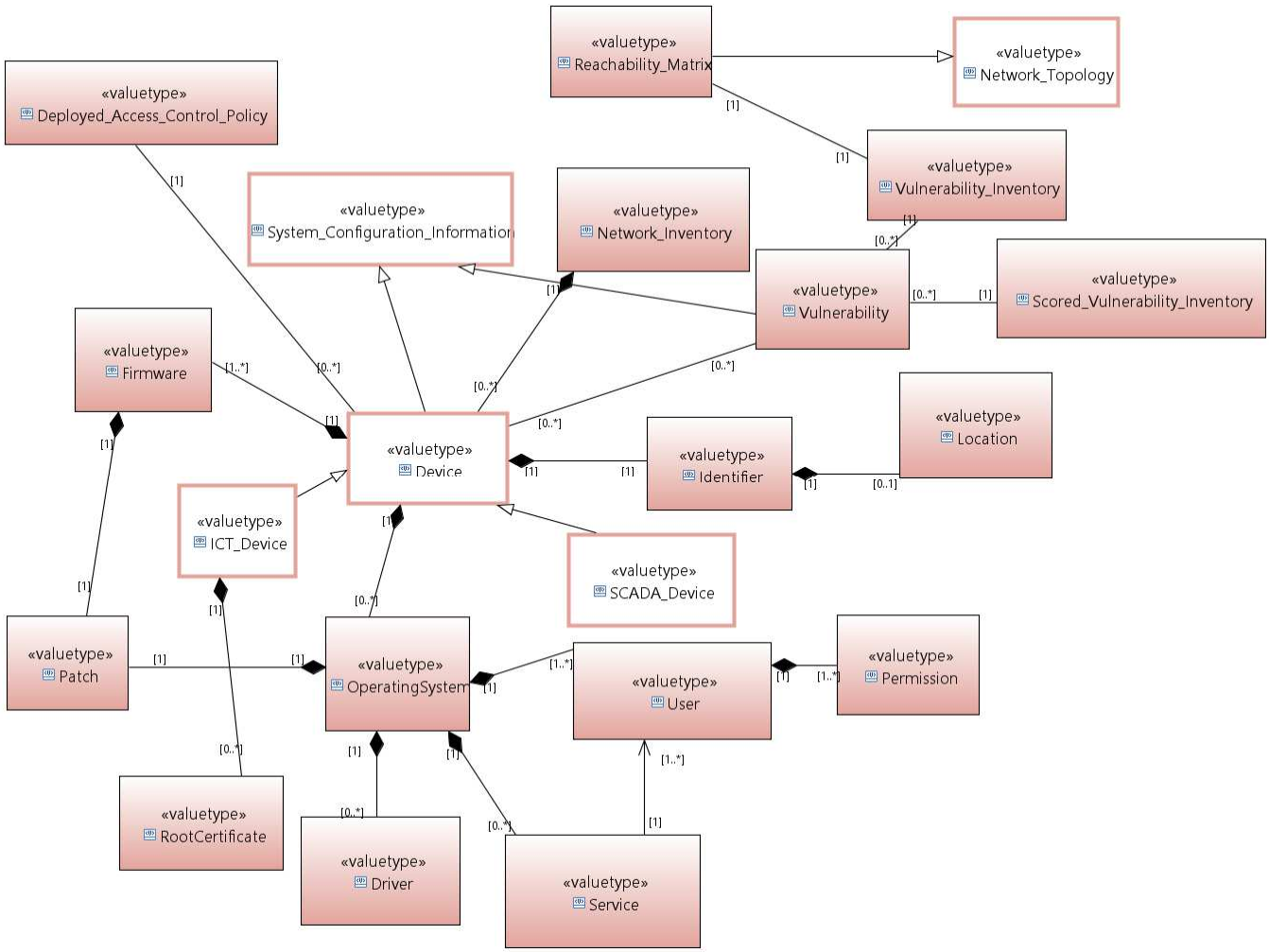


Figure 5: Data Collection and Correlation System ValueTypes

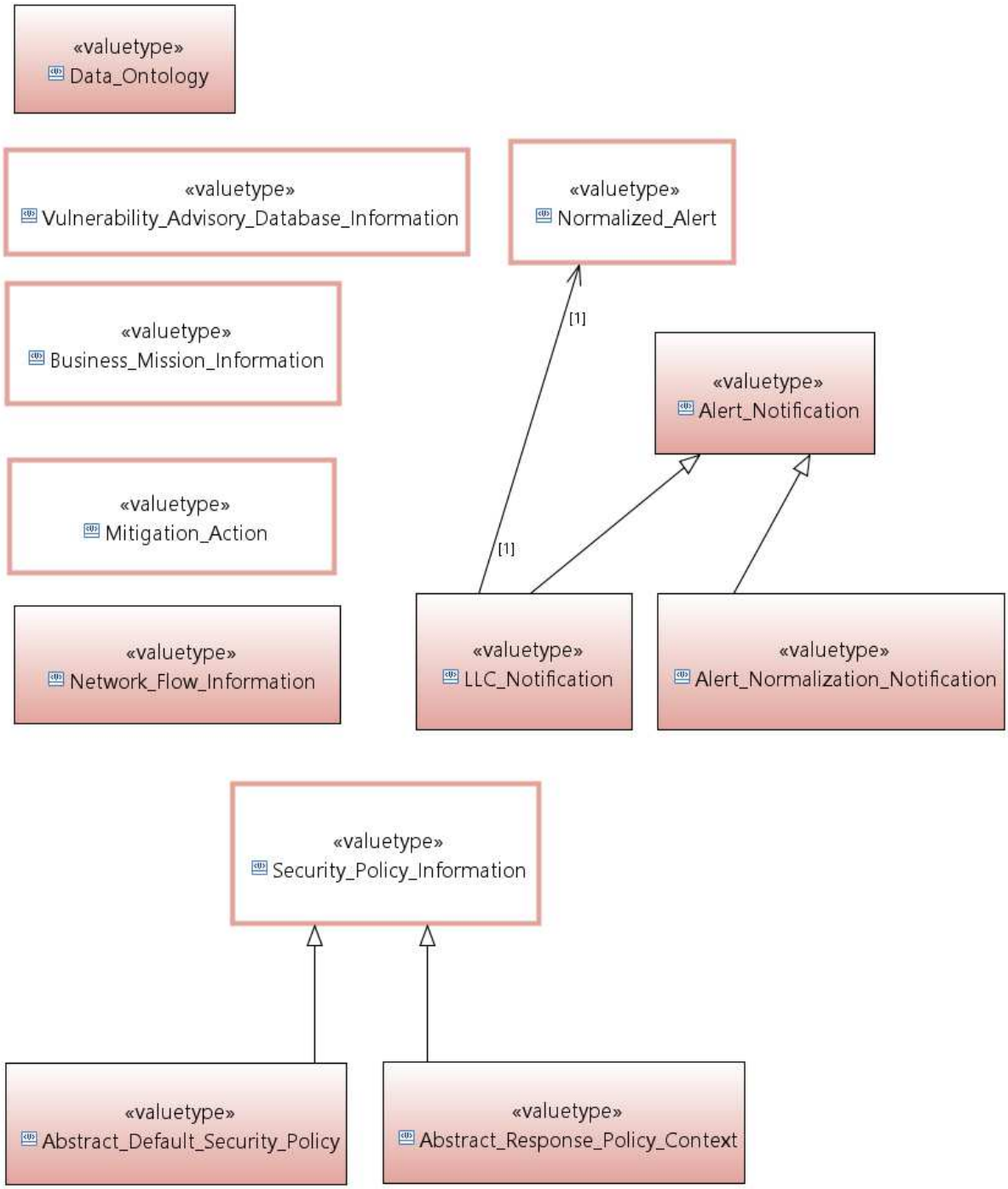


Figure 6: Data Collection and Correlation System ValueTypes

System_Configuration_Information

A High Level representation of a standardized and normalized set of topology or vulnerability information of the Monitored System(s). Identified in **DSC006**.

Device

A *Device* High Level Data represents all collected information about a single node (ICT or SCADA/ICS) in the Monitored System(s): topology (at least Level 3) information, inventory

information, vulnerabilities information, routing tables and interfaces information. The *Device* Data Model is at the bottom of the process of data correlation in the Data Collection and Correlation System that produces complex objects like the *Reachability_Matrix*. Every *Device* has a set of properties, in form of ValueTypes. Most of these properties can be associated both to ICT or SCADA Devices:

- *Firmware* ValueType satisfies **DCS011** and **DCS020**.
- *Patch* ValueType store information regarding the actual version of the patches of a Firmware or an Operating System. Although not requested in [D2.2.1] Requirements, it is a useful information in order to associate specific vulnerabilities to the *Device*.
- *Driver* ValueType satisfies **DSC009**. It may apply also to SCADA Devices.
- *Service* ValueType satisfies **DSC012** and **DSC019**.
- *Identifier* ValueType satisfies **DSC013** and **DSC021**. Specialized identification information will be assigned to ICT Devices and SCADA Devices.
- By considering **DSC031**, information about the physical location (in form of geographical coordinates) of the single Devices are also considered (*Location* ValueType).
- *User* and *Permission* ValueTypes satisfy **DSC015** and **DSC016**.
- *RootCertificate* ValueType satisfies **DSC010**.
- *OperatingSystem* ValueType satisfies **DSC008**.

Vulnerability_Advisory_Database_Information
A PANOPTESEC internal standardized and normalized version of publicly available information sources regarding Vulnerabilities. It allows identification of applicability. This ValueType is linked to the Non-Monitored System Actor (advisory vulnerability databases are of course external to the Monitored System(s)). Identified in **DSC028**.

Business_Mission_Information
Information about gateways between business levels and ICT and SCADA/ICS infrastructures inside a Monitored System(s). Identified in **DSC023**. *Business_Mission_Information* handles information regarding critical systems inside the Monitored System(s).

Mitigation_Action
An atomic action performed by the PANOPTESEC System over the Monitored System(s) after a proactive or reactive risk management treatment chain. Mitigation Action deployed by the PANOPTESEC System must be part of a collection of Authorized Mitigation Actions configured by the Security Operator by using the Visualization System. Examples of Mitigation Actions are the deployment of a patch on a Device or the closure of a Port on a firewall. Identified in **DSC026**: all possible *Mitigation_Action* deployed by the PANOPTESEC System MUST derive from a configurable set of Authorized Mitigation Actions.
Mitigation_Action will consist of:

- Ask a Device to install a software patch;
- Close a TCP Port on a Device.

Security_Policy_information
Organizational security guidelines (e.g., static and dynamic authentication, authorization and reaction guidelines), from a corporate perspective, to settle the structured rules (e.g., in XOrBAC format) to be deployed by the PANOPTESEC System in the Monitored System(s). Identified in **DSC017** and specified in **DSC022**. **PRS017** specifies that this Value_Type is a needed input in order to implement the Proposed Mitigation Actions.

The XOrBAC format is an XML schema based on OrBAC that structures its data storage and communication. More information about XOrBAC are in [D5.1.2].

Abstract_Default_Security_Policy
Default organizational security guidelines (e.g., static authentication, authorization and reaction guidelines), from a corporate perspective, to settle the structured static rules (e.g., in XOrBAC format) to be deployed by the PANOPTESEC System in the protected infrastructure.

This ValueType is a more specialized version of the *Security_Policy_Information* ValueType.

Abstract_Response_Policy_Context
Contextualized organizational security requirements (e.g., dynamic authentication, authorization and reaction guidelines), from a corporate perspective, to settle the structured dynamic rules (e.g., in XOrBAC format) to be deployed by the PANOPTESEC System in the protected infrastructure. It also contains the definition of the dynamic expressions confirming each context expected to activate the rules.

This ValueType is a more specialized version of the *Security_Policy_Information* ValueType.

ICT_Device
A description of an object inside an ICT environment, including comparable information about hardware and software configurations. Allows unique identification. Identified in **DSC007**.

SCADA_Device
A description of an object inside a SCADA environment, including comparable information about hardware, firmware and software configuration. Allows unique identification. Identified in **DSC018**. A SCADA device is generally part of an ICS. From a networking perspective, it is expected that the *SCADA_Device* will also be an Internet-Protocol (IP) accessible device (e.g. Profinet compliant), possibly a TCP/IP device (e.g. Modbus or DNP3 compliant).

Vulnerability
A description of a known flaw inside a SCADA or ICT Devices. Carries identifiers to a *Vulnerability_Advisory_Database_Information*. Identified in **DSC028**. Every *Device* collected

by the PANOPTESEC System will be associated with a set of known *Vulnerability*.

Network_Inventory
An exhaustive collection of ICT and SCADA devices in a given environment and corresponding applicable Vulnerabilities.

Deployed_Access_Control_Policy
Generic representation of all the filtering rules implemented on Devices (usually, firewalls, but also routing tables associated to other Devices) of a Monitored System(s).
In order to create the *Network_Topology* (as requested in **DSC014**) it is mandatory to collect all filtering rules of the Monitored System(s) and evaluate the reachability of each *Device*.

Reachability_Matrix
Structured format that provides the knowledge of which *Device* can communicate with each other *Device* of the Monitored System(s), using which source and destination ports, or specific protocols. This format should integrate not only the logical network topology (i.e. routing possibilities, ISO OSI Level 3) but should also provide the ISO OSI level 4 communication possibilities deriving from (i) *Network_Inventory* and (ii) the *Deployed_Access_Control_Policy* implemented in the system. The format should better be structured in order to ease the algorithm chosen for the Attack Path Generation process. The *Reachability Matrix* is a central ValueType for all the PANOPTESEC and encompasses information given by a standard *Network_Topology* (*Reachability_Matrix* covers the information requested by **DSC014** and **VIZ008** and adds information regarding the list of *Vulnerability* associated to each node of the Matrix).

ICA005 identifies the need of the PANOPTESEC System for a process of normalization and correlation over the raw topology data in order to produce a standardized PANOPTESEC data representing the Level 3 network topology: the *Reachability_Matrix* has been identified as the main result of this correlation process. This ValueType also carries *Location* information and, in case of ICT Devices, *Identifier* information regarding ISO OSI Level 2 (covering partially **DSC032**, **DCS033, VIZ029** and **VIZ030**).

Vulnerability_Inventory
List of the *Vulnerability* currently existing on all the devices of a Monitored System that should be taken in account for the Attack Path Generation process; the knowledge should be structured so that for each vulnerabilities, (i) the requirements for an attacker to successfully exploit the vulnerabilities (e.g. requires a given privilege level on the device, requires the access to the device on a specific port or with a specific protocol), and (ii) the consequences gained by an attacker on the system by exploiting successfully the vulnerability are explicitly described (e.g. privilege escalation up to the highest powered level on the system). Directly referenced by the *Reachability_Matrix*. This ValueType is needed in order to compute the *Attack_Path* for the Monitored System(s), as stated in **PRS006**.

Scored_Vulnerability_Inventory
Structured representation of the intrinsic characteristics of the vulnerabilities that could be exploited in, each step of the potential attack scenarios, or in future steps of ongoing attack scenarios. These characteristics should enable the computation of the likelihood dimension and the impact nature of Elementary Risks. This knowledge is basically composed of the values of metrics as expressed in public vulnerability scoring system, like the FIRST Common Vulnerability Scoring System. Scored Vulnerability Inventory can be seen as a directly enriched Vulnerability Inventory with additional information from Vulnerability Advisory Database Information (CVSS scores, e.g.).

This ValueType has been evaluated as a needed input in order to compute the Static and Dynamic Risk assessment.

Normalized_Alert
An alert generated in the course of a low or high level correlation process. Described in a format close to IDMEF (Intrusion Detection Message Exchange Format). A *Normalized_Alert* is the result of a normalization and correlation process in the Data Collection and Correlation System of the raw alerts data collected by the Data Collection System from the Monitored System(s). **DSC029** identifies the need of the PANOPTESEC System to collect raw alerts from the Monitored System(s). **ICA005** identifies the need of the PANOPTESEC System for a process of normalization and correlation over the raw alert data in order to produce a standardized set of PANOPTESEC System data representing alerts.

Network_Flow_Information
Enrichment of a *Reachability_Matrix* by the respective normalized degree of actual usage of an identified Reachability over a given period of time. *Network_Flow_Information* are collected from the Monitored System(s) (from firewall logs or from directly analyzing the network data traffic of the System) and then normalized in the Data Collection Interface component in order to extract relevant information (for example, the amount of bytes exchanged between two devices in a fixed period of time).

This ValueType has been evaluated as a needed input for the Mission Impact Module component, in order to determine a heuristic of dependency between two *Device(s)* or between two *Service(s)*.
For example, an Intranet-Webserver is identified as a *Mission_Critical_System* (cf. [D3.1.1] on a Monitored System. During the identification, the Security Operator does not recognize that another crucial component of the Webserver is a Database Server behind it. As the Database Server will probably send a huge amount of data to the Webserver, it is possible to measure a dependency between the two *Device(s)*. Any Proposed *Mitigation_Action* affecting the Database Server will then be evaluated with respect of this dependency.

Data_Ontology

A Data Ontology is a representation of knowledge of a specific domain, done in an abstract machine readable language. The language to be used will be determined after a research activity in Task 4.2 of the Work Package 4. The ontology allows creation of the domain knowledge base and performing semantic queries on it.

The Data Ontology used in Data Collection and Correlation System by the Reachability Matrix Correlator represents *Network_Inventory*, *Deployed_Access_Control_Policy*. The ontology is imported from an external file via the Visualization System and stored in a Graph Database in the Reachability Matrix Correlator component. The need for a *Data_Ontology* comes from **ICA002**.

Alert_Notification and the derived values (LLC_Notification and Alert_Normalization_Notification)
A notification which indicates the occurrence of a specific new *Normalized_Alert*. Generated at the same time than the referenced alert and by the same component. Informs any other interested function in order to analyze or to store the corresponding alert. While the *Alert_Normalization_Notification* contains some duplicated attributes of the associated *Normalized_Alert* to speed up the future analysis process and avoid unnecessary access to the whole description of the alert (in order to try to fulfill **PRF006**), *LLC_Notification* carries a complete *Normalized_Alert*.

In Figure 7 ValueTypes related to the Mission Impact Model are depicted. ValueTypes from the Preliminary High Level Data Model are depicted in empty blocks with red border.

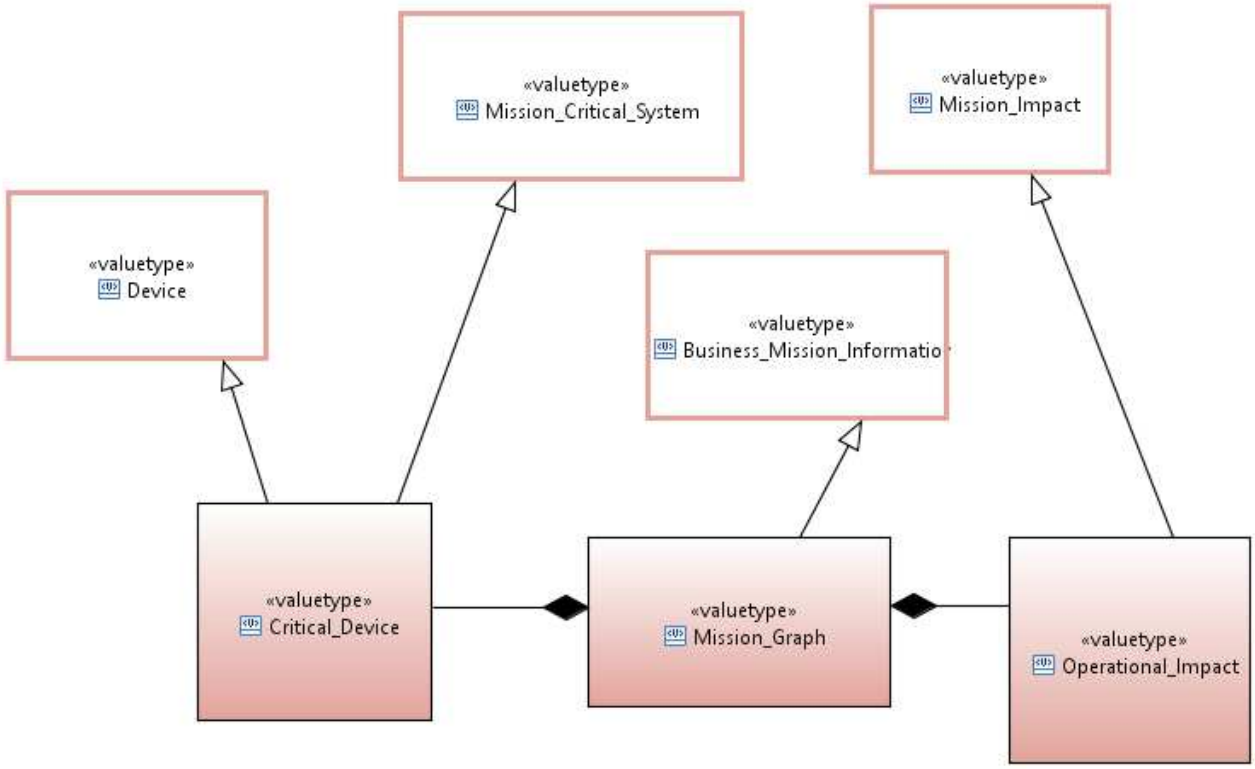


Figure 7: Mission Impact Model ValueTypes

Mission_Graph
A machine readable, compact representation of *Business_Mission_Information*. A Mission Graph is a structured representation of the link existing between business entities identified as critical for the organization (i.e. assets), and technical devices supporting those assets. It also should provide the knowledge on the feared events (i.e. detrimental events) and an assessment of the impact and nature of the feared event on the assets for the business of the organization. Identified in **ICA011, ICA012, ICA013, ICA014, ICA015, ICA016, ICA017, ICA018** and **ICA019**.

Critical_Device
A *Device* identified at a business level as crucial for maintaining the ability to work of an environment. Identified in **ICA015** and specified in **ICA016**, **ICA017** and **ICA018**.

Operational_Impact
A measurement for the reduction of working ability at business level due to unforeseen events or taken actions. Identified in **ICA019** and **ICA020**.

In Figure 8 ValueTypes related to Dynamic Risk Management Response System are depicted. ValueTypes from the Preliminary High Level Data Model are depicted in empty blocks with green/red border.

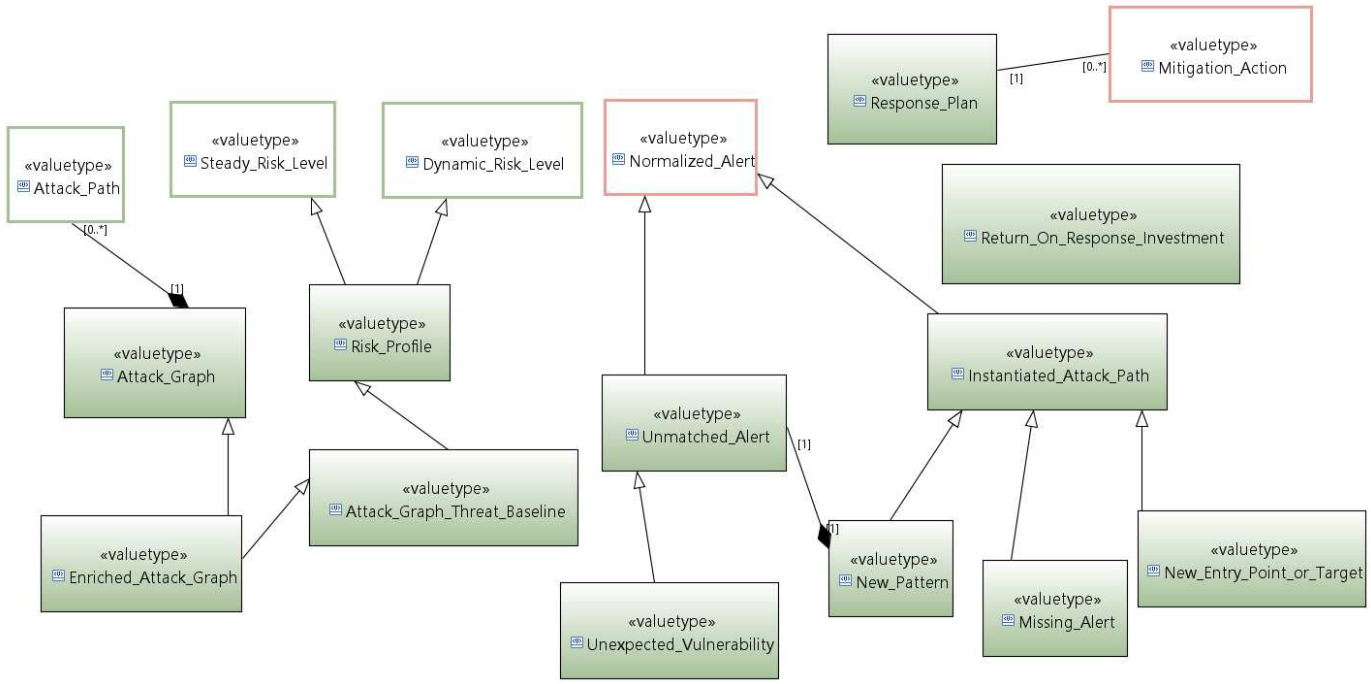


Figure 8: Dynamic Risk Management Response System ValueTypes

Response_Plan

A *Response_Plan* is a structure that represents the possibility to mitigate the various identified threats with a set of *Mitigation_Action*. Depending on the deployed devices that enable to enforce mitigation actions (e.g. firewall, NAC, switch etc.), several possible options could be provided to mitigate the same risk situation. Different Response Plans could also be envisaged on a Proactive or Reactive perspective at the same time. A Response Plan can be seen as a collection of *Mitigation_Action* created in response to an (on-going) threat in an environment.

It is possible to distinguish between Proposed Response Plans (i.e. Response Plan that have to be evaluated by the Mission Impact Model and by the Risk Quantifier component in the Dynamic Risk Management Response System), Enriched Response Plans (i.e. Response Plans with Financial and Operational Impact evaluation that are sent to the Visualization System for human evaluation) and Selected Response Plans (Enriched Response Plans that must be deployed in the Monitored System(s)). Since *Response_Plan* are collections of *Mitigation_Action* proposed in Proactive/Reactive chain, they can be traced to **PRS010**, **PRS011**, **PRS012**, **PRS014**, **PRS015**, **PRS016**, **PRS017**, **PRS022** and to **RRS009**, **RRS010**, **RRS011**, **RRS013**, **RRS014**, **RRS015**, **RRS024**.

Attack_Graph

An *Attack_Graph* represents (on a structured way) the list of all possible attack scenarios (*Attack_Path* defined in the Preliminary High Level Data Model) that can append for each Monitored System. One *Attack_Graph* for each Monitored System should be provided by an Attack Graph Generator component. Identified in **PRS002** and specified in **PRS003**, **PRS004**, **PRS005** and **PRS006**. The *Attack_Graph* can also be enriched by adding a risk evaluation to each *Attack_Path*. The resulting High Level Data is called *Enriched_Attack_Graph*.

Instantiated_Attack_Path
Is a description of the progress state of an attack along an attack path, with the related set of *LLC_Notification*. The *Attack_Path* can be partially or fully exploited. In the first case, it is also characterized by the probability that an attacker is following the attack path given the set of *LLC_Notification*. Identified by **RRS017**, **RRS025** and **RRS026**.

Risk_Profile
A Risk Profile represents the structured list of proactive and reactive Elementary Risks computed by the component, grouped by detrimental events. This should encompass for each Detrimental Event the two dimension of the risk, together with the risk computed values or level and the nature of the impact on the detrimental event. The *Risk_Profile* ValueType encompasses the *Steady_Risk_Level* and the *Dynamic_Risk_Level* identified in the Preliminary High Level Data Model in a unique data structure. Identified in **PRS007** and **RRS002,** specified in **PRS008** and **PRS009**, **RRS003**, **RRS004**, **RRS005**, **RRS006**, **RRS007** and **RRS008**.

An Elementary Risk is defined ([KPD2014]) as:
"*The quantum of risk inflicted by a single Detrimental Event to an Asset through the exercise of a single Attack Scenario on one single Supporting Asset contributing to that Asset*."

- Asset is a strategic entity, which is required by an organization to perform its business or accomplish its mission. While on a general perspective, an Asset is not necessarily a technical entity, in the ICT or SCADA/ICS domain, an Asset often relates to ICT or SCADA/ICS services or information. In the context of the PANOPTESEC System, the Assets will be the Business Processes defined in the *Mission_Graph.*
- Supporting Asset is any technical component of the Monitored System(s) that ensures and supports the proper operation of an Asset. A supporting Asset may be a *Device*, a part of a device or another network service composed of several device and parts of devices. Note that an 'n-to-n' relationship exists between Supporting Assets and Assets: a Supporting Asset may participate to the delivery of the service or accomplishment of the objective/mission of several Assets, and vice versa. In the context of the PANOPTESEC project, Supporting Assets are the *Device*(s) that are associated to a Business Process in the *Mission_Graph*.
- Attack Scenario is the technical sequence of malicious actions undertaken by an adversary to compromise ultimately a Supporting Asset. An attack scenario often consists of several steps, which are intermediate elementary attacks that compromise a technical entity of the monitored systems (e.g. router, server, firewall etc.) by exploiting one or several of its vulnerabilities. Each steps of an Attack Scenario enables the adversary to get closer (topologically) to the exploitation of vulnerability of a Supporting Asset, leading to the violation of one of its security properties (i.e. Confidentiality, Integrity or Availability). Compromising a Supporting Asset ultimately affects its related Assets and may incur Detrimental Events, which have the nature of the security properties violation on the Supporting Asset. In the context of the PANOPTESEC project, an Attack Scenario is formalized with an

Attack_Path, which may be potential (i.e. proactive perspective) or on-going (i.e. reactive perspective).

- Detrimental Event is the fact of harming the accomplishment of an organization's objective or mission. In the context of the PANOPTESEC System, it relates to the fact that a Business Process in the *Mission_Graph*, cannot accomplish one of the Missions to which it relates. A Detrimental Event (DE) may then arise when one of the *Device(s)* (i.e. Supporting Assets) associated with the Business Process (i.e. Asset) that relates to the Detrimental Event through the *Mission_Graph* modelling is targeted by an *Attack_Path*. A Detrimental Event is characterized by two attributes: (i) the magnitude of its impact on the organization (which can be quantitative or qualitative); and (ii) whether it results from a violation of Confidentiality (i.e. ViolC), Integrity (i.e. ViolI) and/or Availability (i.e. ViolA). A Detrimental Event will then be affected by an *Attack_Path* if it leads to a violation of one (or several) security properties of the terminating Supporting Asset which share (at least) a common nature with this DE. Considering in the *Mission_Graph* we may have 'n-to-n' relationships between Supporting Assets, Assets and Detrimental Event, a single Attack Scenario can result in multiple ERs.

Unlike the classical definition of Risk, an ER relates to the contribution to the risk of occurrence of a specific event that would be caused by one (and only one) of the different possibilities of causing this specific event. Nevertheless, conforming to the classical definition of Risk, the ER is computed by assessing its two dimensions: likelihood and impact.

Additional information about Elementary Risks and Detrimental Events are in [D5.1.1] and [D5.1.2].

Unmatched_Alert
A reference to an alert contained in the flow of *LLC_Notification* that does not allow to progress in the recognition of any searched attack scenario. Allows the Potential Attack Identifier component to perform further investigations on the set of alerts whose analysis has previously generated no significant change in the state of the correlation engines. Identified in **RRS027**.

Unexpected_Vulnerability
One of the *Normalized_Alert* generated by the Potential Attack Identifier component. Provides additional analysis of the most frequent vulnerabilities that are indicated in the recently observed *Unmatched_Alert*. Allows identification of the unknown vulnerabilities (the corresponding attribute of the alert has no specified value or an unexpected value) and the known vulnerabilities that are not mentioned in the *Attack_Graph* but seems to be frequently exploited by some attackers. Traced to **RRS023**.

New_Pattern
One of the *Normalized_Alert* generated by the Potential Attack Identifier component. Provides an additional analysis of the most frequent short patterns that combines recently observed *Unmatched_Alert*. Traced to **RRS023**.

Missing_Alert
One of the *Normalized_Alert* generated by the Potential Attack Identifier component. Identifies an attack corresponding to a particular approximation of one of the *Attack_Path* specified in the *Attack_Graph*. Enables to raising an alert when all the steps of a given attack path, except one, called missing LLC alert, have been observed. Traced to **RRS023**.

New_Entry_Point_or_Target
One of the *Normalized_Alert* generated by the Potential Attack Identifier component. Identifies an attack corresponding to a particular approximation of one of the attack path specified in the *Attack_Graph*. Enables to raising an alert when all the steps of a given attack path have been observed but with a small difference, namely either the entry point or the target is not the node identified in the *Attack_Graph*. Traced to **RRS023**.

5.2.5 Logic View and Process View

In order to give a more direct understanding of the relationships between the Logic View representing the structure of the PANOPTESEC System Architecture from a component and interface perspective, and the Process View representing the behaviour and the processes among the components identified in the Logic View, the Views are depicted together in association with the expressed concepts and not in separate Sections.

Below the Viewpoint of the Logic View for the PANOPTESEC System Design is shown:

- **Purpose**:
  Documentation of the architecture design
- **Stakeholders**:
  Technical Project Manager, Work Package Leaders, IT Architects, Software Developers
- **Concern(s):**
  Which are the logical structures of the system?
- **Artefacts:**
  Architecture overview/vision
  System context
  Key abstractions (with behaviour)
  Functional system building blocks
  Technical system building blocks

Below the Viewpoint of the Process View for the PANOPTESEC System Design is shown:

- **Purpose**:
  Documentation of the control and coordination of the building blocks identified in the Logic View
- **Stakeholders**:
  Technical Project Manager, Work Package Leaders, IT Architects, Software Developers
- **Concern(s):**
  Which are the concurrent building blocks of a system?
  How the building blocks behave during system's operations?
- **Artefacts**
  Processes
  Inter-process communication

In Figure 1 a PANOPTESEC System Overview, in form of a Logic View, is shown, resulting from the functional evolution over the Preliminary High Level Design described in Section 5.1.2.

A second Logic View is then produced, after considering Figure 1 and the identified PANOPTESEC High Level Data Model from Section 5.2.4 and the analysis over every single component conducted within WP3, WP4 WP5 and WP6. This process represents one of the iterative design steps presented in Section 2.1.

Since the provided Diagram is too big and complex in order to fit in this document in a suitable way, it is split over its components. Nevertheless, the complete view is maintained in as part of the SysML project. A rationale concerning the coverage of the design over other Functional and Non-Functional ASRs is also given and the processes involving these blocks are depicted by using Process Views in form of Sequence Diagrams. In addition, for every analysed block a description of its functionalities is given.

PANOPTESEC System communication model
Components in PANOPTESEC System should be able to communicate by using Push or Pull Communication Strategies. A registration process may occur at component start-up: the consumer component registers for a list of Monitored Systems managed by the producer component for a particular data update, in order to be able to receive notifications of a particular data update (or directly the data update in *push mode*) or to request (*pull mode*) data updates from the provider.

Due to the adoption of an Integration Framework managing all the interactions between the components, a wide range of Enterprise Integration Patterns can be used in order to manage the communications between integrated components, in order to fulfil **CMP002** (c.f. Section 5.2.5.5 for additional information).

5.2.5.1 Data Collection and Correlation System

In Figure 9 a High Level Logic View of the PANOPTESEC System with focus on the Data Collection and Correlation System is presented.

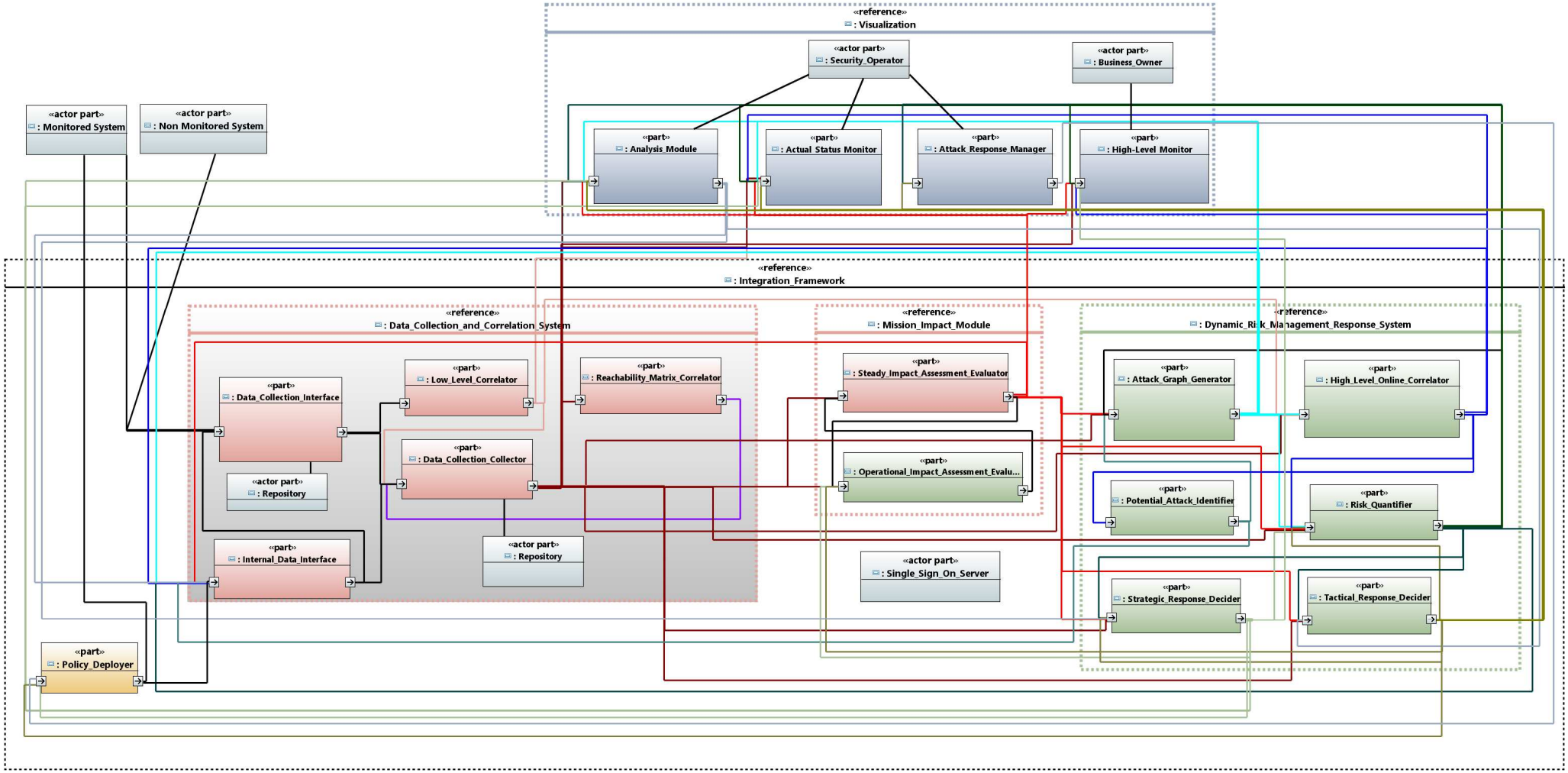


Figure 9: PANOPTESEC System Logic View with focus on Data Collection and Correlation System

The Data Collection and Correlation System is the PANOPTESEC System package described as collecting information from Monitored and Non Monitored Systems. In Figure 9 it is evident that the Data Collection and Correlation System architecture satisfies **CMP001**, **MNT001**, and **PRT004**. This package is not designed as a monolithic multipurpose block, but as a set of functional blocks glued by the Integration Framework, with respect to **CMP002**, **CMP003**, **CMP004** and **PRT005**.

As already presented in Figure 3, the Data Collection and Correlation System encompasses the functionalities requested in the Functional ASRs of the Data Collection System and of the Information Correlation Engine (Section 5.1.1).

The Data Collection and Correlation System is composed of five main components:

1. Data Collection Interface
2. Data Collection Collector
3. Low Level Correlator
4. Reachability Matrix Correlator
5. Internal Data Interface (this component has the purpose of being an **Input** interface between the rest of the PANOPTESEC System and Data Collection and Correlation System components, without adding any complex processing. For this reason, its description is left to the Data Collection and Correlation System Design described in [D4.1.2]).

Some possible external data sources (from Monitored and Non-Monitored System(s)) for the PANOPTESEC System are identified and briefly explained:

- **Topology Scanners**:
  Software that analyses networks at ISO/OSI Layer 2/3 (for example, Nmap). The result of the scan is a MAC/IP mapping of on-line devices, usually in form of log files. There are two family of topology scanners: active scanners (they use ISO/OSI Layer 3 protocols, such Ping for example, in order to verify hosts availability and map devices) and passive scanners (they analyse flows of traffic in order to extract information about hosts).
- **Inventory Scanners**:
  Software that analyses device assets. In order to perform this detailed scan, usually an agent needs to be installed in host machine OS. Some inventory scanners are able to remotely analyse the hosts, using user permissions (typically Administrator or root).
- **Vulnerability Advisory Databases**:
  A Vulnerability Database is a platform aimed at collecting, maintaining, and disseminating information about discovered vulnerabilities targeting computer systems.

- **BPMN description files**:
  Business Process Model and Notation (BPMN) is a standard for business process modeling. The BPMN 2.0 standard is defined by an XML schema.
- **Security Scanners**, **Vulnerability Scanners**:
  Software that analyses vulnerabilities on hosts (widely used vulnerability scanners are Nessus or OpenVas). These software usually perform TCP/UDP port scan in order to analyse services (and version) open/running. After the scanning, they search in a Vulnerability Database (usually not remote) if correspondences exist.
- **Log Server Firewall**:
  Firewall Devices produce log files that are usually persisted on server in the Monitored System(s). These servers can be accessed in order to retrieve these files for analysis.
- **Syslog Servers**:
  Servers persisting syslog messages from Devices (using the standard Syslog Service Protocol). Syslog is an extremely prevalent protocol for logging, both for networking equipment and servers (*ICT_Device*) and for more specialized objects such as UPS or intelligent meters (*SCADA_Device*).
- **SNMP Servers**:
  Simple Network Management Protocol is a standard protocol that manages network devices. A SNMP Server collects Devices information via SNMP Protocol.
- **SIEM Servers**:
  Security Information and Event Management Servers are a software solution (sometimes with hardware appliance) that collects security information from a heterogeneous environment. The goal of SIEMs is to correlate security alert and network alert events in order to describe or analyze a security incident.
- **IDS logs**:
  Intrusion Detection Systems listen the network traffic and analyze possible matching with known vulnerabilities. They can usually work in passive mode (detecting attacks but not blocking them) or active (detecting and blocking). IDS usually produce log files of their analyses. These logs files can be accessed by the PANOPTESEC System in order to start the reactive chain (the aim of the PANOPTSEC, of course, is to be able to understand far more complex attacks then a simple IDS).

In order to give a more precise understanding of the Data Collection and Correlation System, a sub-set of the Diagram in Figure 9 is given.

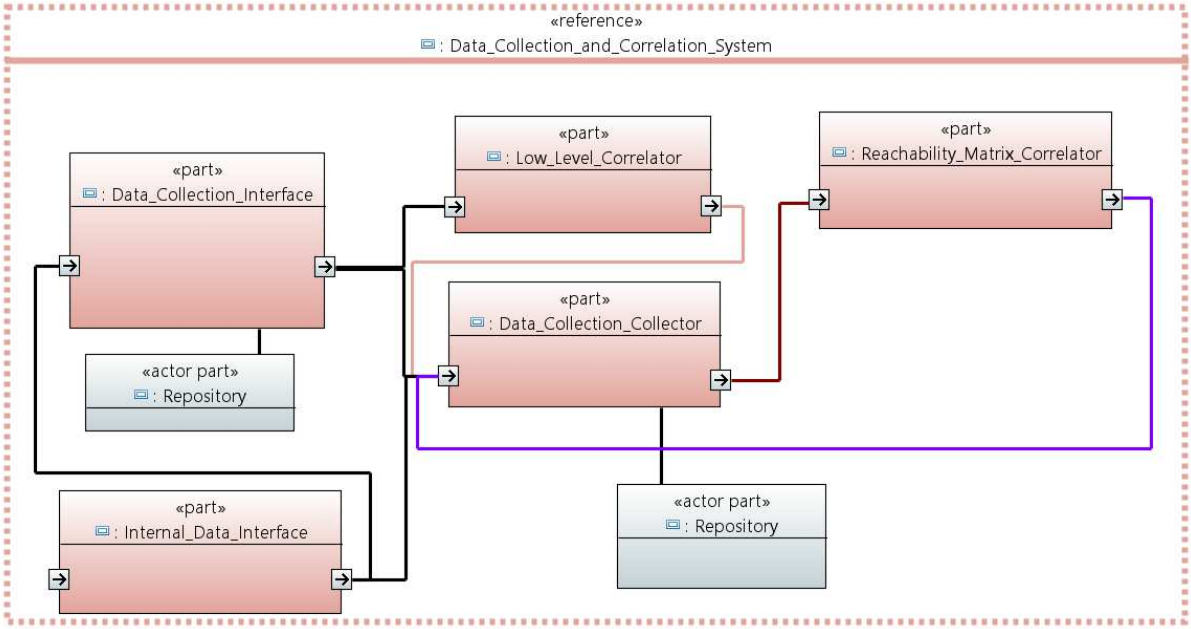


Figure 10: Data Collection and Correlation System High Level Logic View

Data Collection Interface

In Figure 11 a Logic View of the Data Collection Interface (DCI) component is presented.

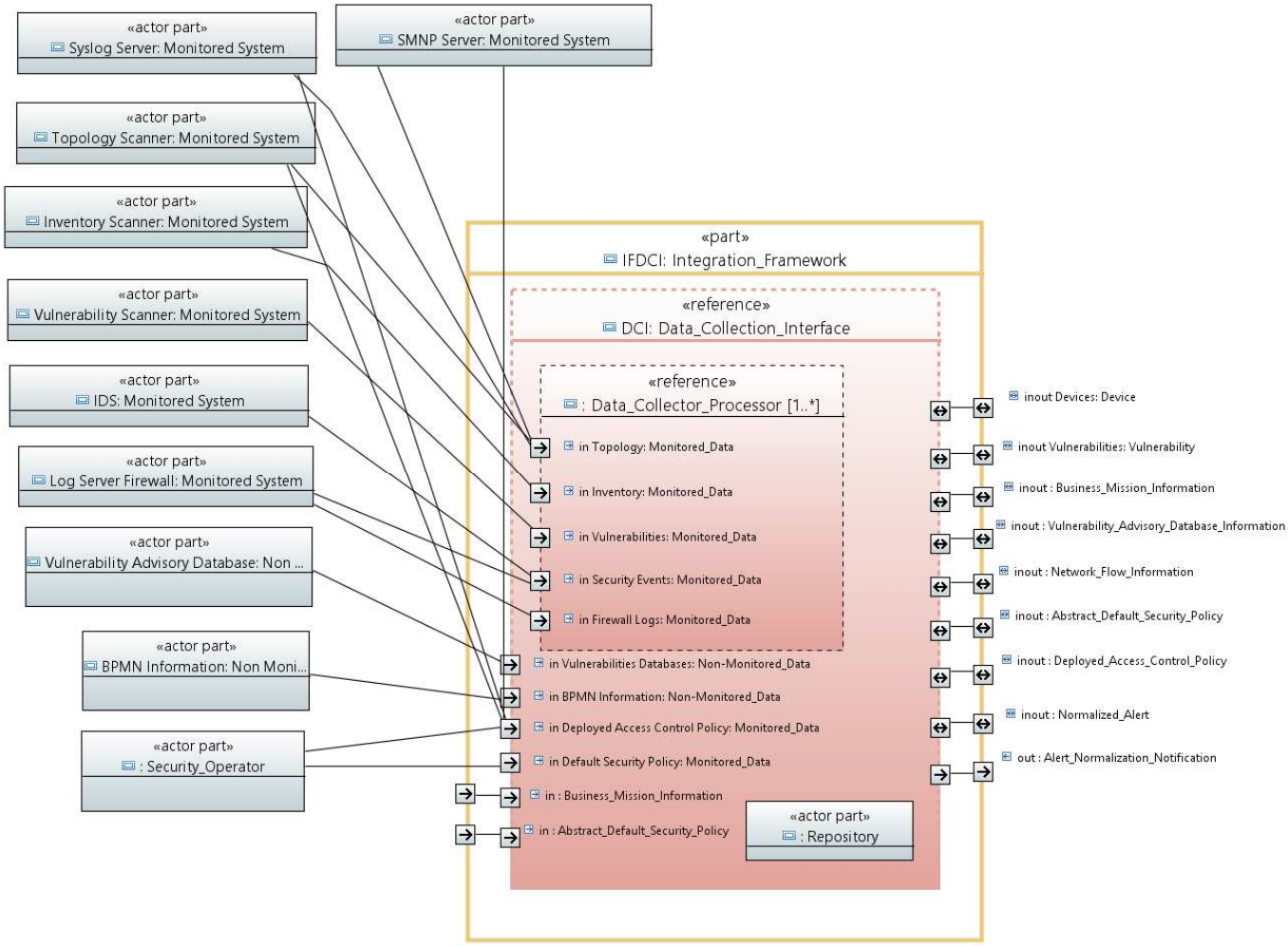


Figure 11: Data Collection Interface Logic View

The Data Collection Interface component, part of the Data Collection and Correlation System, is composed of several subcomponents and is responsible to collect, normalize and pre-process data from the Monitored and the Non-Monitored System(s). Figure 11 shows DCI as a component inside an Integration Framework wrapper that manages the external connections and the runtime of the component.

DCI has a direct connection with the Data Collection Collector component that manages the persistency of the PANOPTESEC System: all PANOPTESEC standardized data normalized in DCI are then collected and persisted in the Data Collection Collector component. A sub-component of DCI (the **Data Collection Processor**) is responsible for directly connecting with the Monitored System(s) and collecting raw data (topology, vulnerabilities, alerts data sources,) and can be deployed in multiple instances throughout the Monitored System(s).

DCI has been designed such that it does not depend on a specific Monitored System (for example, the ACEA environment): the Data Collection Processors Design covers the possible sources of data of a complex network with ICT and SCADA Devices. That solves one of the risks identified in Section 5.3 of [D3.1.1].

All raw data collected are then stored in a repository (following **DSC005)** that can be accessed by normalization processors (managed by DCI). The output is a set of PANOPTESEC standardized data.

The chosen repository technology will need to be able to be backed up on regular bases (**RLB006**), and, possibly, redundant (**RLB005**). The repository is important in order to allow IT Architects use a Data-Shared Architectural Pattern (Annex C) and decouple the Data Collection process (managed by the Data Collection Processors) with the normalization processes. In order to increase performance and meet **PRF006** (related to the reactive chain) it is possible to envision a direct processing and normalization of a stream of alerts as soon as they are collected.

The Data Collection Interface component encompasses the Data Collection System block identified in [D3.1.1] (with the Data Collection Processors) and the normalization functionalities behind the Information Correlation Engine ([D3.1.1]).

Related Functional ASRs are **DSC002**, **DSC003** and **DSC004**. **CMP005** and **CMP006** also states that the Data Collection Processors must be able to collect alerts in IDMEF format and should be able to translate any IDS log in IDMEF format, if not native. Since the *Normalized_Alert* High Level Data, as it is stated in Section 5.2.4, is a format close to IDMEF, a translation to IDMEF will have to be provided anyhow: we then consider these two ASRs as satisfied by the Data Collection Processors design.

Among the *Monitored_Data* collected from the Monitored System(s), the Data Collection Processors is in charge to collect information regarding *Devices* (**DSC006**), both *ICT* (**DSC007**) and *SCADA/ICS* (**DSC018**), with all the identified properties (**DSC008**, **DSC009**, **DSC010**, **DSC011**, **DSC012**, **DSC013**, **DSC015**, **DSC016**, **DSC019**, **DSC020** and **DSC021**). Information collected will have to be sufficient in order to let the correlation components reconstruct the *Network_Topology/Reachability_Matrix* (**DSC014**) and to identify their physical location (**DSC031**). The Data Collection Processors must also collect topology information regarding IDSes and sensors (**DSC030**), in order to describe them as *Device*(s) in the *Network_Topology/Reachability_Matrix*.

Cyber security alerts should be collected by the Data Collection Processors from IDS logs and firewall logs (**DSC029**).

The Data Collection Interface is also responsible to collect and normalize information from the external vulnerabilities databases, as stated in **DSC028**. These data are downloaded on a regular basis (e.g., every day), translated into a common PANOPTESEC System format and persisted, in order to be accessible by other components (in particular, DCC components assessing the *Vulnerability_Inventory*, the *Scored_Vulnerability_Inventory* and the *Network_Inventory*). This process may need Internet connectivity in order to be performed (with the risk of violating **SEC002**, which has however less priority among ASRs).

The Data Collection Interface is responsible, following **DSC023** and **DSC025**, for collecting *Business_Mission_Information* from the Visualization System or directly from a Security Operator (it may be an XML file, for example). **DSC024** states that DCI may collect *Business_Mission_Information* directly from a BPMN infrastructure (a BPMN server, for example). In that case a normalization process over the BPMN collected information may be needed.

The Data Collection Interface is responsible, following **DSC026** and **DSC027**, for collecting a list of Authorized *Mitigation_Action* from Security Operator (it may be a loaded XML file, or a set of messages from the Visualization System).

The Data Collection Interface is in charge to collect *Abstract_Default_Security_Policy* (**DSC017**). This kind of data usually is loaded from a configuration file directly taken from a shared folder or from the Visualization System, loaded by the Security Operator. A normalization process over these files may need to take place in order to ensure consistency.

Deployed_Access_Control_Policy is needed information in order to construct the *Reachability_Matrix* presented in the Data View in Section 5.2.4. *Device* routing tables can be collected by a topology scanner, while firewalling rules may be loaded in the PANOPTESEC System by human activity.

Due to the adoption of the separation of concerns Architectural Principle inside the DCI, **MNT002** can be addressed. If a new data source must be considered, it may request a new Data Collector Processor able to actually collect these data, if not already present. In addition, it may be necessary to update the DCI single sub-components managing the normalization phase, and the PANOPTESEC data model (or, in case of adoption of semantic technologies for the data normalization phase, of the Data Ontology). The global architecture of the DCI, however, would be maintained. The impact of a new data source is then reduced by the modular interface of the DCI.

After the collection phase, the Data Collection Interface provides the normalization phase (a part of the scope of the Information Correlation Engine defined in [D3.1.1]). Raw data are retrieved from the database by set of normalization processors that parse the collected logs and create a consistent set of PANOPTESEC System standardized data (following **ICA003** and **ICA004**). Normalization components may need to access to the central PANOPTESEC System persistency (managed in the Data Collection Collector component, DCC) in order to process the collected data. It is possible to identify two main normalization processes:

- Device Normalization Process, with the aim of translating collected topology and inventory data from different sources, in order to build a standardized set of *Device* for the PANOPTESEC System;
- Alerts Normalization Process, with the aim of translating each raw alert (IDMEF format for raw alerts is preferred, as stated in **CMP005,** but a translator to IDMEF will be provided in case the Monitored System would be able to only provide IDS

with proprietary format, as requested by **CMP006**) and each field of a raw alert in a common standardized format, in order to treat them independently from the specific format of the source of the alerts and to compare fields of different alerts.

In case of the Alert Normalization Process, the resulting *Normalized_Alert* is persisted in the DCC and an *Alert_Normalization_Notification* message is sent to the Low Level Correlator (LLC) component, signaling that a new *Normalized_Alert* is ready to be retrieved from the persistency for further computations. Since the Data Collection Interface set of components will be managed by the Integration Framework (as it is possible to see in Figure 11), it is easy to envision a Publish/Subscribe Enterprise Integration Pattern between DCI and LLC.

In order to fulfil **PRT005**, a set of fixed interfaces for the Data Collection Interface component is then defined.

IN Interface ValueTypes with the Monitored System(s)
Topology Data
Inventory Data
Security Events Data
Vulnerability Data
Firewall Logs Data
Abstract Default Security Policy Data
Deployed Access Control Policy Data

IN Interface ValueTypes with the Non-Monitored System(s)
Vulnerability Advisory Database Data
BPMN Files

IN Interface ValueTypes with the Internal Data Interface component
Business_Mission_Information (collected by human activity through the Visualization System)
Abstract_Default_Security_Policy (collected by human activity through the Visualization System)

IN Interface ValueTypes with the Data Collection Collector component
Device(s)
Vulnerability(s)
Business_Mission_Information
Vulnerability_Advisory_Database_Information
Network_Flow_Information
Abstract_Default_Security_Policy
Deployed_Access_Control_Policy

OUT Interface ValueTypes with the Data Collection Collector component
Device(s)
Vulnerability(s)

Business_Mission_Information (after a possible normalization between
Business_Mission_Information collected from BPMN 2.0 files and
Business_Mission_Information collected from the Visualization System)
Normalized_Alert(s)
Vulnerability_Advisory_Database_Information
Network_Flow_Information
Abstract_Default_Security_Policy (after a possible normalization between
Abstract_Default_Security_Policy directly collected from the Monitored System and
Abstract_Default_Security_Policy collected from the Visualization System)
Deployed_Access_Control_Policy

OUT Interface ValueTypes with the Low Level Correlator component
Alert_Normalization_Notification

PLEASE NOTE that some interfaces between DCI and Data Collection Collector component are INOUT interfaces: this is due to the fact that DCI, during the normalization processes, may need to access and update already persisted data in the Data Collection Collector.

Interaction between DCI and other components of the System are shown in sequence Diagrams in Figure 15, Figure 16, Figure 18, Figure 21: Visualization System-DCI-DCC-MIM (Business Mission Information Initialization) Process View and Figure 27

More details about DCI can be found in [D4.1.2].

Data Collection Collector
In Figure 12 a Logic View of the Data Collection Collector is presented.

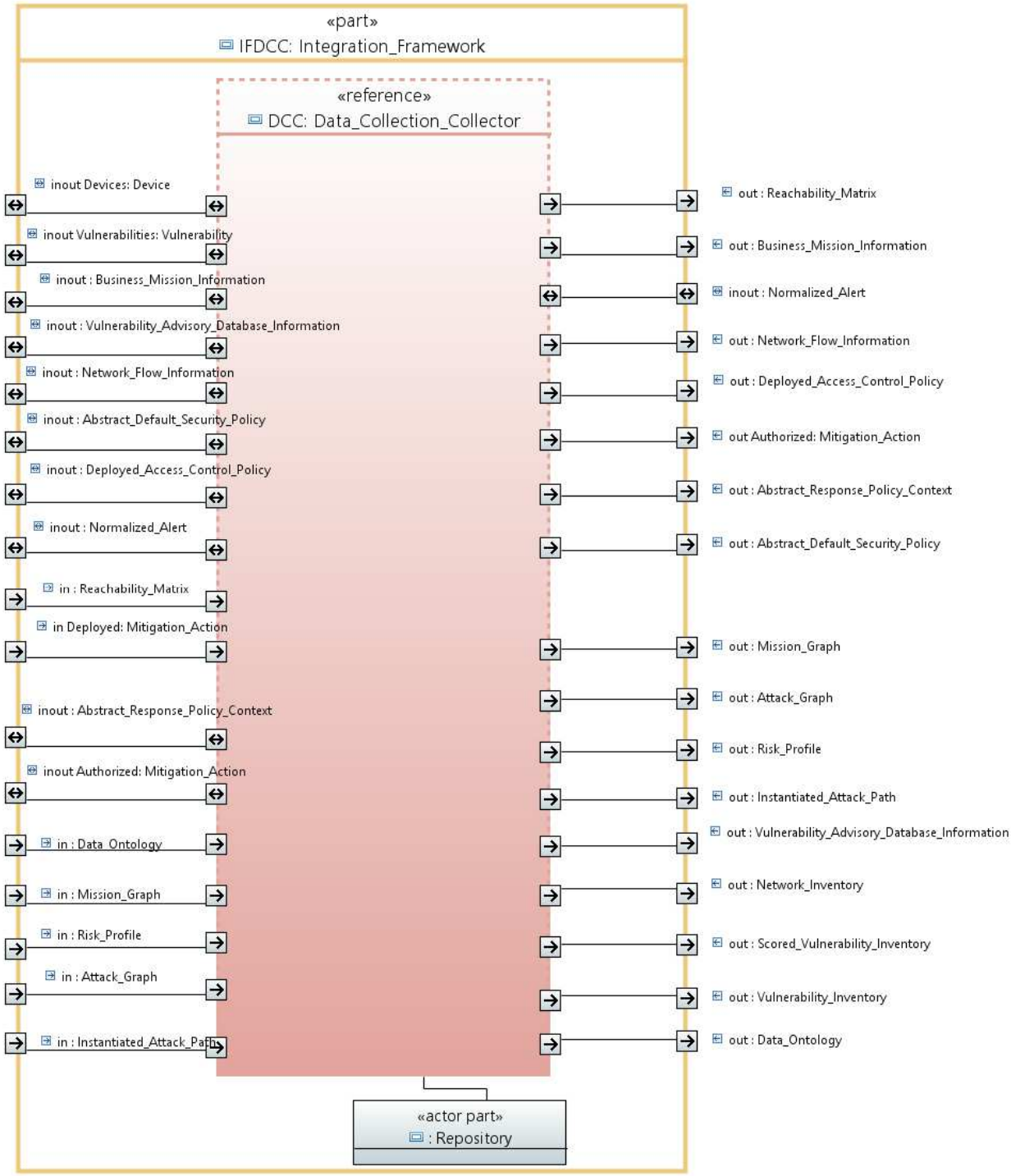


Figure 12: Data Collection Collector Logic View

Data Collection Collector (DCC) is a component of the Data Collection and Correlation System and is responsible for managing data persistency for most of the standardized PANOPTESEC data. Figure 12 shows DCC as a component inside an Integration Framework wrapper that manages the external connections and the runtime of the component.

DCC then plays a central role in all the PANOPTESEC System, since it can be a data source and destination for the Dynamic Management Response System, the Mission Impact Model and the Visualization System. The Persistency Manager component inside DCC uses a repository for persistent storage of all PANOPTESEC standardized data.

The need for a persistency manager for PANOPTESEC System standardized data produced by the normalization and correlation components inside the Data Collection and Correlation System arises from **ICA006**. Persistency Manager will be able to store these data in encrypted form, following **SEC001** (a suitable repository technology needs to be chosen by WP4).

In the operational context it will be possible that the performances overhead behind encryption would be unsustainable with respect to **PRF001**, **PRF004** and **PRF006**. This will need to be evaluated as part of later phases of the PANOPTESEC project.

Since **VIZ023** states that it should be possible for a Security Operator to interact with the Visualization System to analyse historical PANOPTESEC data, the need of a common repository for all PANOPTESEC System computed data structure become relevant. The DCC and its internal Persistency Manager are then in charge to maintain the global persistency of the PANOPTESEC System. Since the PANOPTESEC System proactive chain, as stated in [D3.1.1], works by considering collected snapshots of the Monitored System(s), these snapshots need to be persisted with a unique timestamp allowing to retrieve past proactive situation (snapshot concept is related to the proactive chain in partial fulfilment of **RBL010**). Data related to the reactive chain must also be persisted: these data have a direct relationship with the proactive data set and carries reference IDs to that data set (for example, any *Instantiated_Attack_Path* is computed over a given *Attack_Path* from the proactive chain and carries a reference ID to it). It will then be possible to retrieve, for any persisted proactive snapshot, all reactive evaluations based on it.

RBL004 concerns fault-tolerance capabilities of the PANOPTESEC System. Due to the fact that it maintains the global data persistency, DCC may play an important role in improving fault-tolerance capabilities of the PANOPTESEC System (but, as the Data-Shared Pattern states, the repository would become a possible critical single point of failure of the PANOPTESEC System). The chosen repository technology must be able to be backed up on regular bases (**RLB006**), and, possibly, redundant (**RLB005**).

Sub-components on DCC are also responsible for producing relevant data structures from the standardized data produced by the DCI: the *Network_Inventory*, the *Vulnerability_Inventory* and the *Scored_Vulnerability_Inventory*. These data structure are needed in order to compute the *Reachability_Matrix*. In order to perform these computations, DCC may need to access to the *Vulnerability_Advisory_Database_Information* persisted in the repository.

Due to **RBL011** it appears that the Data Collection and Correlation System needs to be able to provide to the subsequent components in the proactive and reactive chains a consistent

set of data (for example, referred to the same data snapshot from the DCI). Data Collection and Correlation System correlation results (in particular the *Reachability_Matrix*, the *Network_Inventory*, the *Vulnerability_Inventory* and the *Scored_Vulnerability_Inventory*) are produced by different functional components. This raises the requirement for designing a synchronization mechanism ensuring that this set of data is published in synchrony. Applying a Data-Shared Pattern is useful in this situation: a new Data Collection and Correlation proactive computation result will be available only when all computed consistent data will be produced.

In order to fulfil **PRT005**, a set of fixed interfaces for the Data Collection Collector component is then defined.

IN Interface ValueTypes with the Data Collection Interface component
Device(s)
Vulnerability(s)
Business_Mission_Information
Normalized_Alert(s)
Vulnerability_Advisory_Database_Information
Network_Flow_Information
Abstract_Default_Security_Policy
Deployed_Access_Control_Policy

IN Interface ValueTypes with the Internal Data Interface component
(Authorized) Mitigation_Action(s) (from Visualization System)
Data_Ontology (from Visualization System)
Abstract_Response_Policy_Context (from Visualization System)
(Deployed) Mitigation_Action(s) (from the Policy Deployer component)
Historical information regarding *Attack_Graph(s), Risk_Profile(s), Instantiated_Attack_Path(s) and Mission_Graph(s)* (from the Dynamic Risk Management Response System)

IN Interface ValueTypes with the Reachability Matrix Correlator component
Reachability_Matrix

IN Interface ValueTypes with the High Level Online Correlator component
Normalized_Alert(s)

IN Interface ValueTypes with the Low Level Correlator component
Normalized_Alert(s)

OUT Interface ValueTypes with the Data Collection Interface component
Device(s)
Vulnerability(s)
Business_Mission_Information
Normalized_Alert(s)

Vulnerability_Advisory_Database_Information
Network_Flow_Information
Abstract_Default_Security_Policy
Deployed_Access_Control_Policy

OUT Interface ValueTypes with the Low Level Correlator component
Normalized_Alert(s)
Vulnerability_Advisory_Database_Information
Network_Inventory
OUT Interface ValueTypes with the Reachability Matrix Correlator component
Network_Inventory
Deployed_Access_Control_Policy
Data Ontology

OUT Interface ValueTypes with the Internal Data Interface component
(Authorized) Mitigation_Action(s)
Abstract_Response_Policy_Context

OUT Interface ValueTypes with the Attack Graph Generator component
Reachability_Matrix
Vulnerability_Inventory

OUT Interface ValueTypes with the High Level Online Correlator component
Normalized_Alert(s)

OUT Interface ValueTypes with the Risk Quantifier component
Scored_Vulnerability_Inventory

OUT Interface ValueTypes with the Strategic Response Decider component
Reachability_Matrix
Abstract_Default_Security_Policy
Abstract_Response_Policy_Context
(Authorized) Mitigation_Action(s)

OUT Interface ValueTypes with the Tactical Response Decider component
Deployed_Access_Control_Policy
(Authorized) Mitigation_Action(s)

OUT Interface ValueTypes with the Visualization System
Reachability_Matrix
Network_Inventory
Vulnerability_Inventory
Scored_Vulnerability_Inventory
(Deployed) Mitigation_Action(s)
Vulnerability_Advisory_Database_Information

(Authorized) Mitigation_Action(s)
Network_Flow_Information
Attack_Graph(s)
Risk_Profile(s)
Instantiated_Attack_Path(s)
Mission_Graph(s)

OUT Interface ValueTypes with the Mission Impact Module
Reachability_Matrix
Network_Inventory
Business_Mission_Information
Network_Flow_Information

PLEASE NOTE that some interfaces between DCC and other component are INOUT interfaces: this is due to the fact that data persisted in DCC may need to be accessed and updated by various PANOPTESEC components (for example, the Low Level Correlator component may need to retrieve and enriched already persisted *Normalized_Alert(s)*).

Due to its central role in the PANOPTESEC System, WP3 advice on behalf the Data Collection Connector component is to adopt the state-of-the-art of persistency management in real time systems (for example, hybrid No-SQL/SQL repositories).

Interactions between DCC and other components of the System are shown in Sequence Diagrams in Figure 14, Figure 15, Figure 16, Figure 18, Figure 21: Visualization System-DCI-DCC-MIM (Business Mission Information Initialization) Process View, Figure 27, Figure 28, Figure 29, Figure 33 and Figure 39.

More details about DCC can be found in [D4.1.2].

Reachability Matrix Correlator
In Figure 13 a Logic View of the Reachability Matrix Correlator (RMC) component is presented.

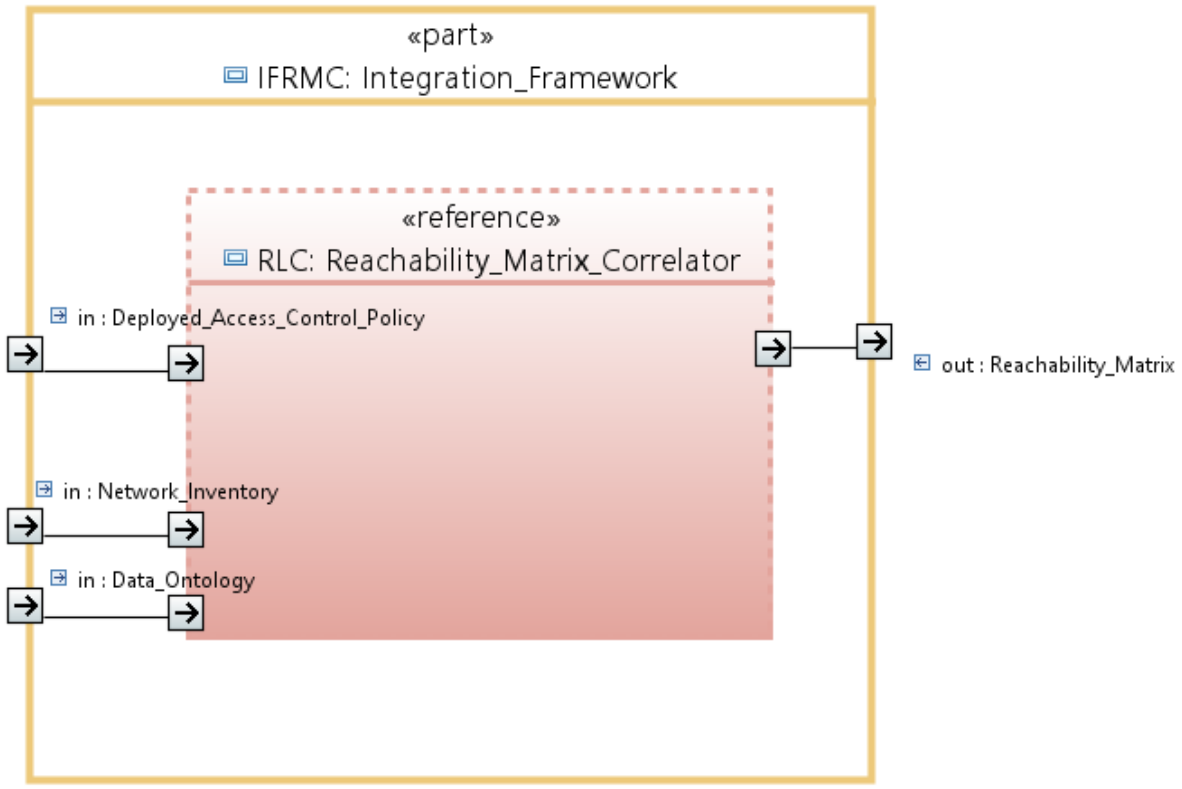


Figure 13: Reachability Matrix Correlator Logic View

Figure 13 shows RMC as a component inside an Integration Framework wrapper that manages the external connections and the runtime of the component.

The Reachability Matrix Correlator (RMC) computes the *Reachability_Matrix*, needed by the Dynamic Risk Management Response System, the Mission Impact Model and the Visualization System. This Correlator component performs the reachability computation across data collected from the Monitored System(s) to deduct if two nodes are reachable from each other in the network, for all pairs of nodes representing *Device(s)* of the Monitored System(s).

RMC satisfies **ICA005** and **ICA002** (due to the semantic technologies used in RMC): the produced *Reachability_Matrix* represents the complete connectivity and reachability of every *Device* of the Monitored System(s) at the ISO/OSI Level 3. RMC covers part of the correlation functionalities of the Information Correlation Engine depicted in the PANOPTESEC System Preliminary High Level Design [D3.1.1].

In order to compute the *Reachability_Matrix* a complete *Network_Inventory* of the Monitored System(s) is needed. In particular, information about *Device* Interfaces and routing tables are a fundamental input, along with Firewalling rules (also called *Deployed_Access_Control_Policy*).

In order to fulfil **PRT005**, a set of fixed interfaces for the Reachability Matrix Correlator component is defined.

IN Interface ValueTypes with the Data Collection Collector component
Network_Inventory
Deployed_Access_Control_Policy
Data_Ontology

OUT Interface ValueTypes with the Data Collection Collector component
Reachability_Matrix

Interactions between RMC and other components of the System are shown in Sequence Diagrams in Figure 14, Figure 15 and Figure 16.

More details about RMC can be found in [D4.1.2].

Interactions between Visualization System-DCC-RMC (Data Ontology Initialization)
In Figure 14 a Process View (a Sequence Diagram) of the interactions between Visualization System, DCC and RMC is presented.

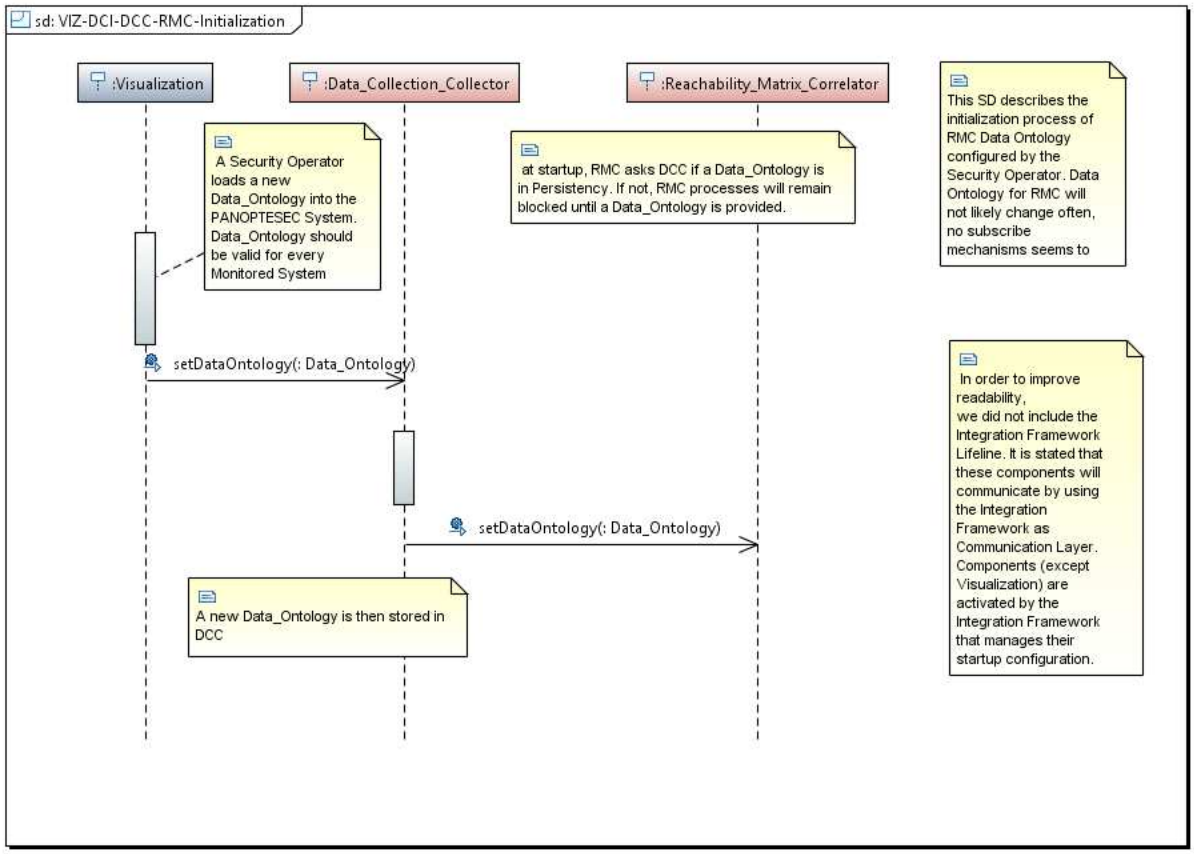


Figure 14: Visualization System-DCC-RMC (Data Ontology Initialization) Process View

In order to reduce the size of the diagrams for documentation reasons, Integration Framework lifelines are not shown in this Diagram. As can be seen in the previous Logic

Diagrams, the Integration Framework will be directly managing all communication between the components. More details are available in Section 5.2.5.5.

In order to compute the *Reachability_Matrix*, RMC needs to be loaded with a *Data_Ontology* representing the knowledge of the domain. This will happen through interaction between the Security Operator, the Visualization System, DCC and RMC. At start-up RMC interrogates DCC and the persistency in order to retrieve a *Data_Ontology*. If a *Data_Ontology* is not stored, RMC cannot compute.

If a Security Operator needs to load a new *Data_Ontology* he needs to use the Visualization System in order to send it to the Data Collection and Correlation System in push strategy. A new *Data_Ontology* is then stored in DCC. DCC will then push the *Data_Ontology* to RMC. If RMC is in a processing phase it will likely stop the process in order to load the new *Data_Ontology*. Due to the static nature of the knowledge base, it is unlikely that this process will occur often (that is why it is manageable to eventually stop RMC on-going processes and load a new ontology without risking racing conditions).

Please note that it would be possible to make DCI loading the *Data_Ontology* from a shared folder, or and FTP server. The decision around using the Visualization System relies on the fact that a limited Ontology editor may be provided in order to directly modify and update the knowledge base from a View of the Visualization System.

Interactions between DCI-DCC-RMC (Deployed Access Control Policies Initialization)
In Figure 15 a Process View (a Sequence Diagram) of the interactions between DCI, DCC and RMC is presented.

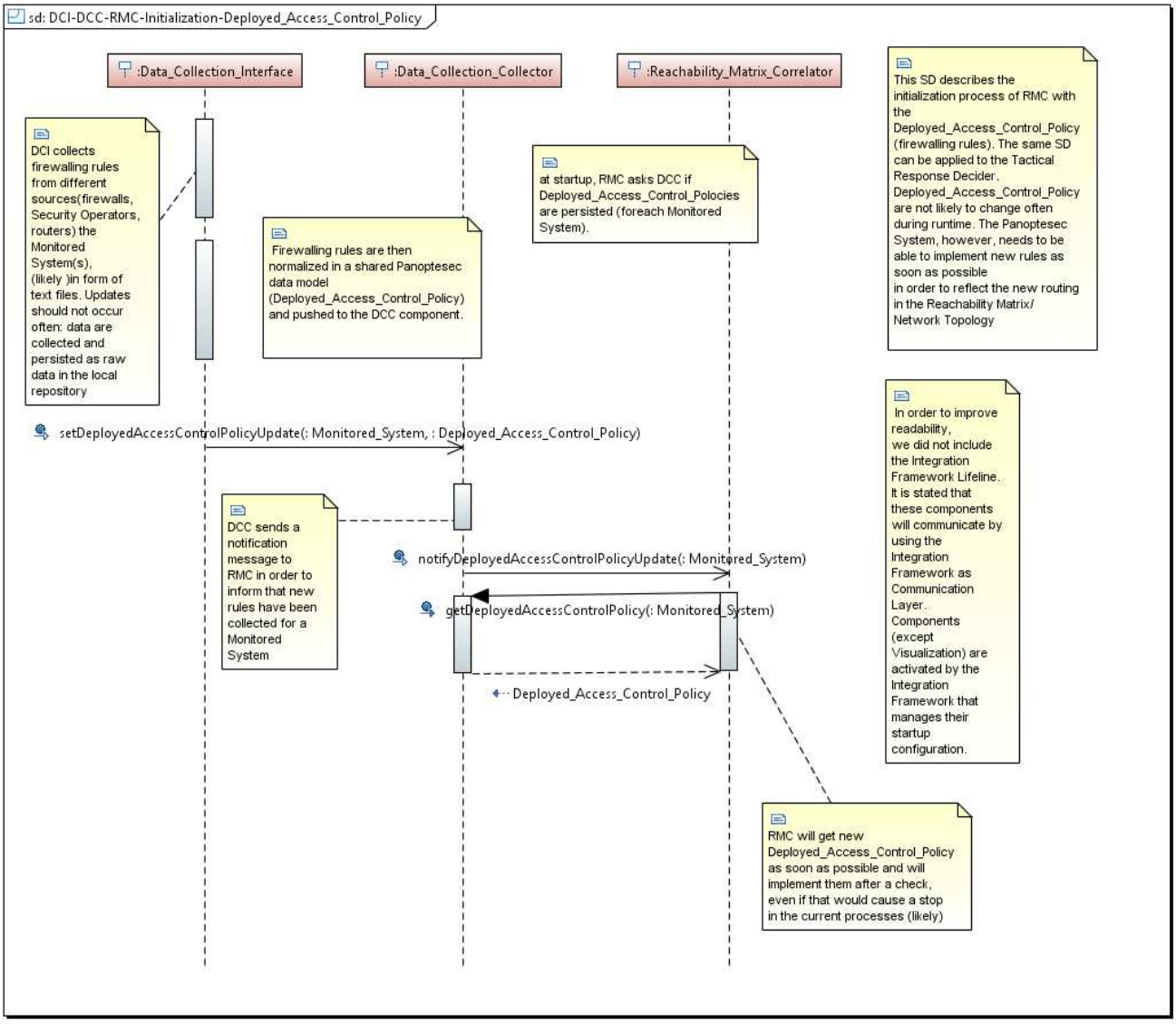


Figure 15: DCI-DCC-RMC (Deployed Access Control Policies Initialization) Process View

In order to reduce the size of the diagrams for documentation reasons, Integration Framework lifelines are not shown in this Diagram. As can be seen in the previous Logic Diagrams, the Integration Framework will be directly managing all communication between the components. More details in Section 5.2.5.5.

In order to compute the *Reachability_Matrix*, RMC benefits of the knowledge of the firewalling rules of the Monitored System(s). At component start-up, RMC interrogates RMC in order to retrieve a last version of these rules (for each Monitored_System). RMC can compute without the rules, but results will be likely non consistent with the real situation of the Monitored system(s).

DCI monitors the Monitored System(s), collecting firewalling rules raw data from possible different sources (Topology Scanners, Syslog Servers, SNMP servers, Security Operators) in form of text files. Raw data are collected and persisted in the local repository. The

normalization process for the *Deployed_Access_Control_Policy* analyses the raw firewalling rules data and standardize them in *Deployed_Access_Control_Policy.*

Deployed_Access_Control_Policy are saved in the persistency in DCC that sends a notification message to RMC. If RMC can implement new *Deployed_Access_Control_Policy* it accesses the persistency in order to retrieve and compute. In case a *Reachability_Matrix* processing is on-going, it will be likely stopped in order to implement the new *Deployed_Access_Control_Policy.* Due to the quite static nature of the firewalling rules (changes usually are not so frequent on a Monitored System), it is unlikely that this process will occur often. For this reason, it is manageable to stop RMC on-going processes and load a new set of rules without risking racing conditions.

Interactions between DCI-DCC-RMC (topology data set processing)
In Figure 16 a Process View (Sequence Diagram) of the interactions between DCI, DCC and RMC is presented.

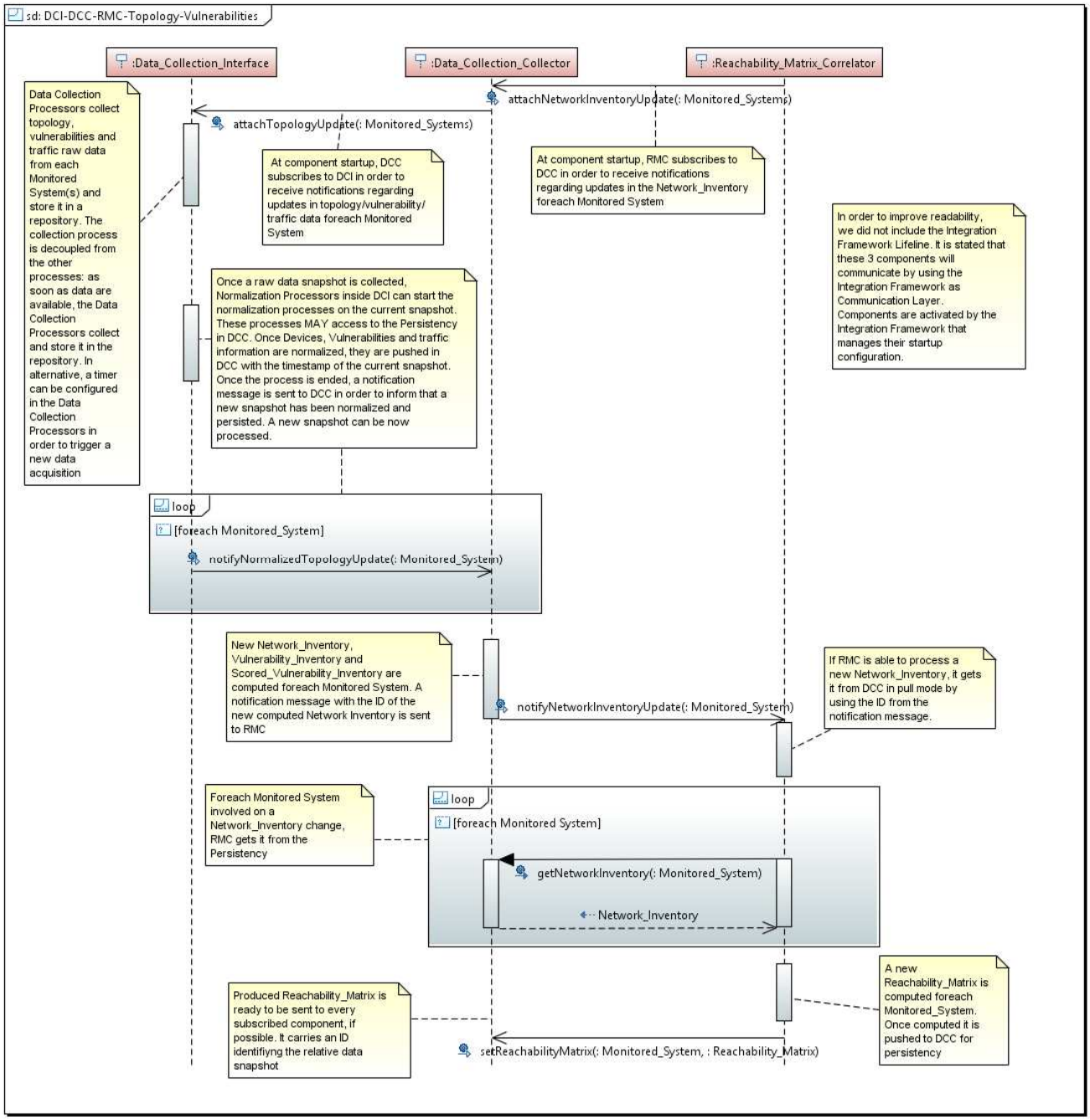


Figure 16: DCI-DCC-RMC (topology data set processing) Process View

In order to reduce the size of the diagrams for documentation reasons, Integration Framework lifelines are not shown in this Diagram. As can be seen in the previous Logic Diagrams, the Integration Framework will be directly managing all communication between the components. More details in Section 5.2.5.5.

At components start-up (managed by the Integration Framework) RMC subscribes for *Network_Inventory* updates with DCC, while DCC subscribes to Monitored System topology updates with DCI.

The Data Collection Processors monitor the Monitored System, collecting raw data as soon as they are available (or the data collection may be triggered by a timer). Data are stored in a repository (likely, a NoSQL repository). A timestamp should be added to every new collected data set.

Once a new data set has been collected, Normalization Processes can start: raw data are parsed and analysed in order to normalize and standardize them into the PANOPTESEC System data model (*Device*(s), *Vulnerability* (is), *Network_Flow_Information*). Persistency in DCC may be accessed in the process in order to retrieve already saved information. Once data are standardized, they are saved in the persistency with a timestamp identifying the snapshot. At the end of the processes a notification message will be sent to DCC in order to state that a new Monitored System snapshot has been persisted, with a specified timestamp ID.

If DCC is free to process a new data setit will compute a new *Network_Inventory*, a new *Vulnerability_Inventory* and a new *Scored_Vulnerability_Inventory* (identified by the timestamp ID for the analysed snapshot). If DCC is already processing an older data set it will not stop the computation in order to get the updated information, in order not to risk racing conditions.

A notification message will be send to RMC, informing the component that a new *Network_Inventory* is ready to process, with a specified timestamp ID. If RMC is free to process a new data set (if RMC is already processing an older data set it will not stop the computation in order to get the updated information, in order not to risk racing conditions), it will access the persistency and get the *Network_Inventory*. A new *Reachability_Matrix* is then computed and pushed in the persistency in DCC (with the timestamp ID). The new set of *Reachability_Matrix*, *Network_Inventory*, *Vulnerability_Inventory*, *Scored_Vulnerability_Inventory* related to the same snapshot are then ready to be notified to the subsequent components in the proactive chain (or to the Visualization System).

It is possible to see the benefits of the use of the Data-Shared Pattern (and of course the benefits related to the internal subdivision in different processes applied during design ADD iterations over the Preliminary High Level Design component Information Correlation Engine), applied to that section of the proactive chain. In [D3.1.1], Section 5.2.5.4, a Control Loop (**Control Loop 01**) had been identified: the data normalization was locked to the proactive chain. Due to the evolution of the internal architecture, data normalization and the computation of the base proactive topology/vulnerability dataset are no more coupled with the proactive chain. As can be seen, the Data Collection and Correlation System can proceed on the topology/vulnerability correlation without waiting for a notification message from the end of the proactive chain. When the proactive chain will be able to process a new snapshot, a notification message will be sent and DCC will be able to push/notify to every subscriber in the proactive chain the snapshot ID of the more recent dataset (according to **RBL011** and **RBL010**).

Low Level Correlator

In Figure 17 a Logic View of the Low Level Correlator (LLC) component is presented.

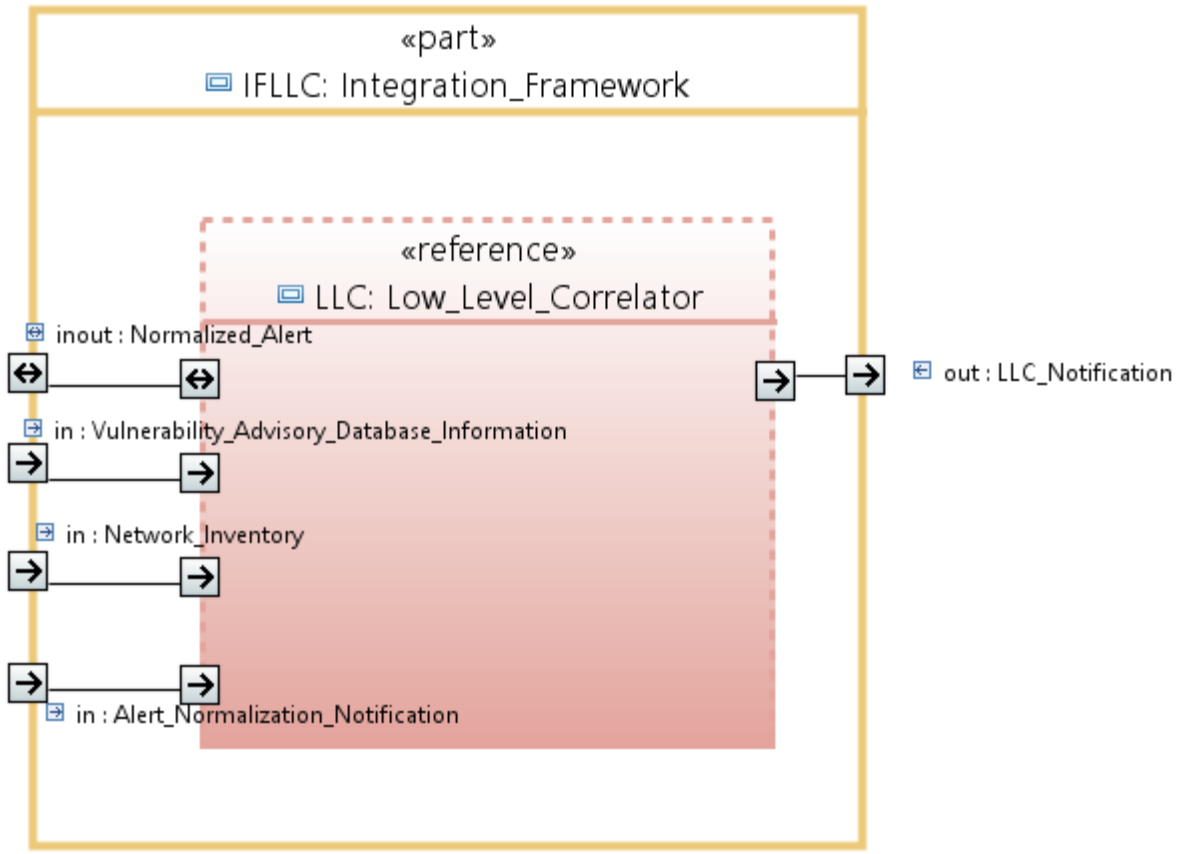


Figure 17: Low Level Correlator Logic View

Figure 17 shows LLC as a component inside an Integration Framework wrapper that manages the external connections and the runtime of the component.

The Low Level Correlator (LLC) performs *Normalized_Alert* processing. The Low Level Correlation Process can be decomposed in five sub-processes:

- **Alert enrichment** consists of adding external knowledge to the *Normalized_Alert*. For example the IP address of the attacked machine if the alert is generated by a host IDS that does not provide this information;
- **Alert verification**: consists of determining if an attack has been successful, i.e., if the *Normalized_Alert* corresponds to a real successful attack (intrusion identification, false positive elimination);
- **Alert fusion**: this consists of merging information carried by *Normalized_Alert*(s) produced by several probes of the same type for a given attack;

- **Elementary attack identification**: this consists of merging *Normalized_Alert* that characterize a given type of attack;
- **Intrusion scenarios identification**: this consists of merging *Normalized_Alert* that characterize each single action in an attack scenario;

LLC covers part of the correlation functionalities of the Information Correlation Engine depicted in the PANOPTESEC System Preliminary High Level Design [D3.1.1].

LLC is a part of the reactive chain of treatment for the PANOPTESEC System. Fulfillment of **PRF003** is a direct responsibility of LLC: the Low Level Correlated *Normalized_Alert(s)* from LLC should be a subset of the *Normalized_Alert(s)* computed by the Alert Normalization Processes in the DCI component.

The output of LLC is a stream of Low Level Correlated Alerts (*LLC_Notification*) that is directly sent to the High Level Online Correlator (a component of the Dynamic Risk Management Response System package) for further analysis. Each *LLC_Notification* message carries a *Normalized_Alert* object (*Normalized_Alert* computed by LLC are also saved in to the persistency in order to be accessed if necessary). LLC_Notification are also sent to the Visualization System. A Publish/Subscribe mechanism carried out by the Integration Framework (along with a fast messaging protocol) is a possible solution in order to guarantee performances in the reactive chain (as stated in **PRF006**).

In order to fulfil **PRT005**, a set of fixed interfaces for the Low Level Correlator component is then defined.

IN Interface ValueTypes with the Data Collection Collector component
Normalized_Alert(s)
Vulnerability_Advisory_Database_Information
Network_Inventory

IN Interface ValueTypes with the Data Collection Interface component
Alert_Normalization_Notification

OUT Interface ValueTypes with the Data Collection Collector component
Normalized_Alert(s)

OUT Interface ValueTypes with the High Level Online Correlator component
LLC_Notification

OUT Interface ValueTypes with the Visualization System
LLC_Notification

Interactions between LLC and other components of the System are shown in Sequence Diagrams in Figure 18, Figure 33 and Figure 40.

More details about LLC can be found in [D4.1.2].

Interactions between DCI-DCC-LLC (alerts processing)
In Figure 18 a Process View (Sequence Diagram) of the interactions between DCI, DCC and LLC is presented.

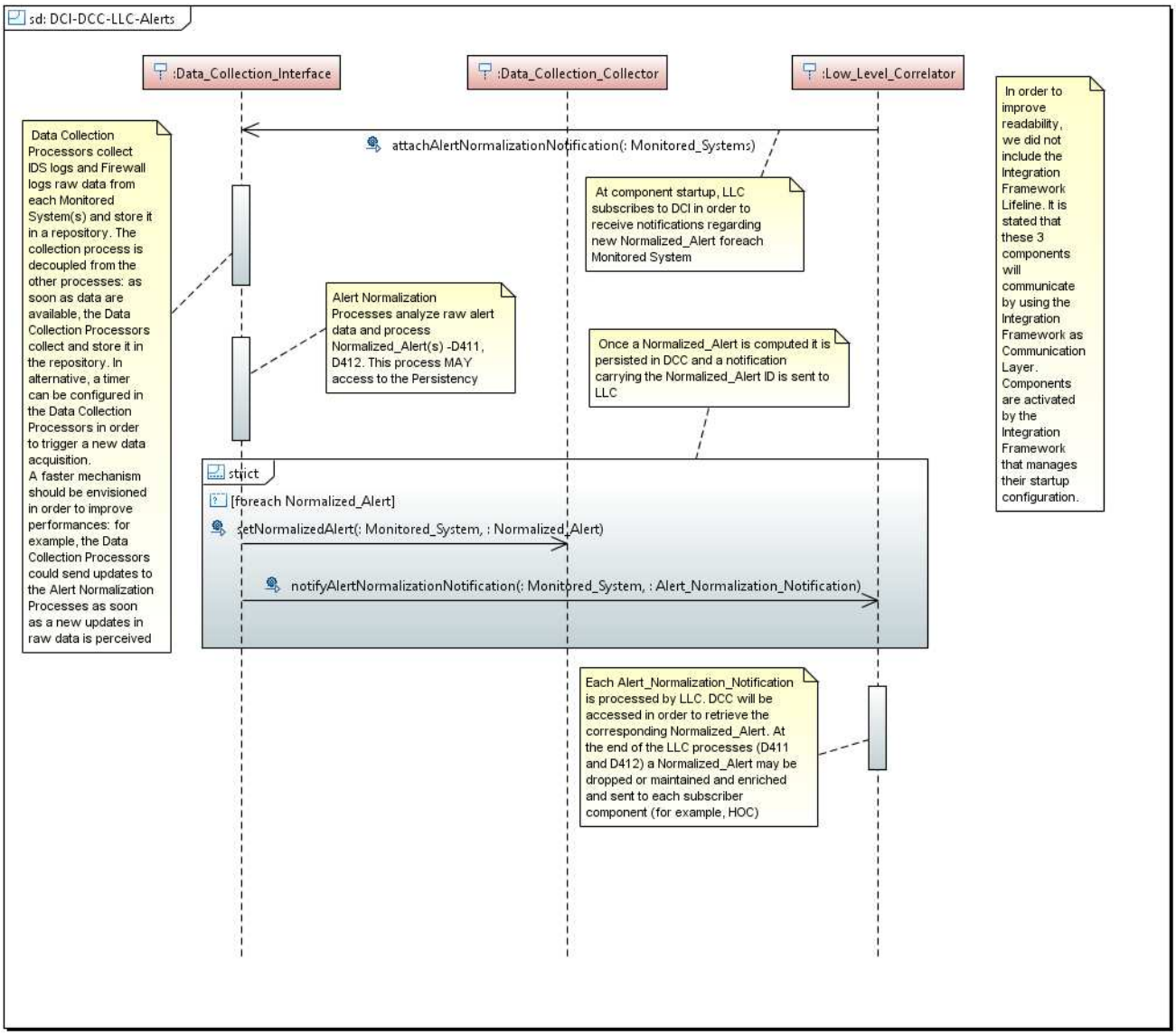


Figure 18: DCI-DCC-LLC (alerts processing) Process View

In order to reduce the size of the diagrams for documentation reasons, Integration Framework lifelines are not shown in this Diagram. As can be seen in the previous Logic Diagrams, the Integration Framework will be directly managing all communication between the components. More details in Section 5.2.5.5.

At components start-up (managed by the Integration Framework) RMC subscribes to *Alert_Normalization_Notification* updates with DCI. The Data Collection Processors monitor the Monitored System, collecting raw data from IDS and firewalls as soon as they are

available. Data are stored in a repository (likely, a NoSQL repository). A timestamp should be added to every new collected data set.

Once a new data set has been collected, Normalization Processes should start immediately: raw data are parsed and analysed in order to normalize and standardize them into the PANOPTESEC System data model *Normalized_Alert*). Persistency in DCC may be accessed in the process in order to retrieve already saved information (for example, information related to the topology, or vulnerabilities). Once a *Normalized_Alert* is produced, it is stored in the persistency in DCC with its unique ID. A notification message is then sent to LLC (carrying an *Alert_Normalization_Notification* with the ID of the persisted *Normalized_Alert)*, that will be able to access it in DCC and compute the next phases of the Low Level Correlation over the *Normalized_Alert.*

Due to the much more rigid performance constraints of **PRF006**, this process should be as fast as possible (so as soon as an update on the log has been perceived it should be collected and normalized and the notification sent to LLC). Due to **ICA006**, a possible loss in performance can be envisioned (if every *Normalized_Alert* must be persisted and accessed for computation). Since **ICA006** priority is higher than **PRF006** one, the current design consider DCC and the persistency of *Normalized_Alert(s)* as a stable solution, but the impact of this choice should be further evaluated.

During the Low Level Correlation processes, persistency in DCC may be accessed in the process in order to retrieve already saved information (for example, information related to the topology, or vulnerabilities)

5.2.5.2 Mission Impact Module

In Figure 19 a High Level Logic View of the PANOPTESEC System with focus on the Mission Impact Module is presented

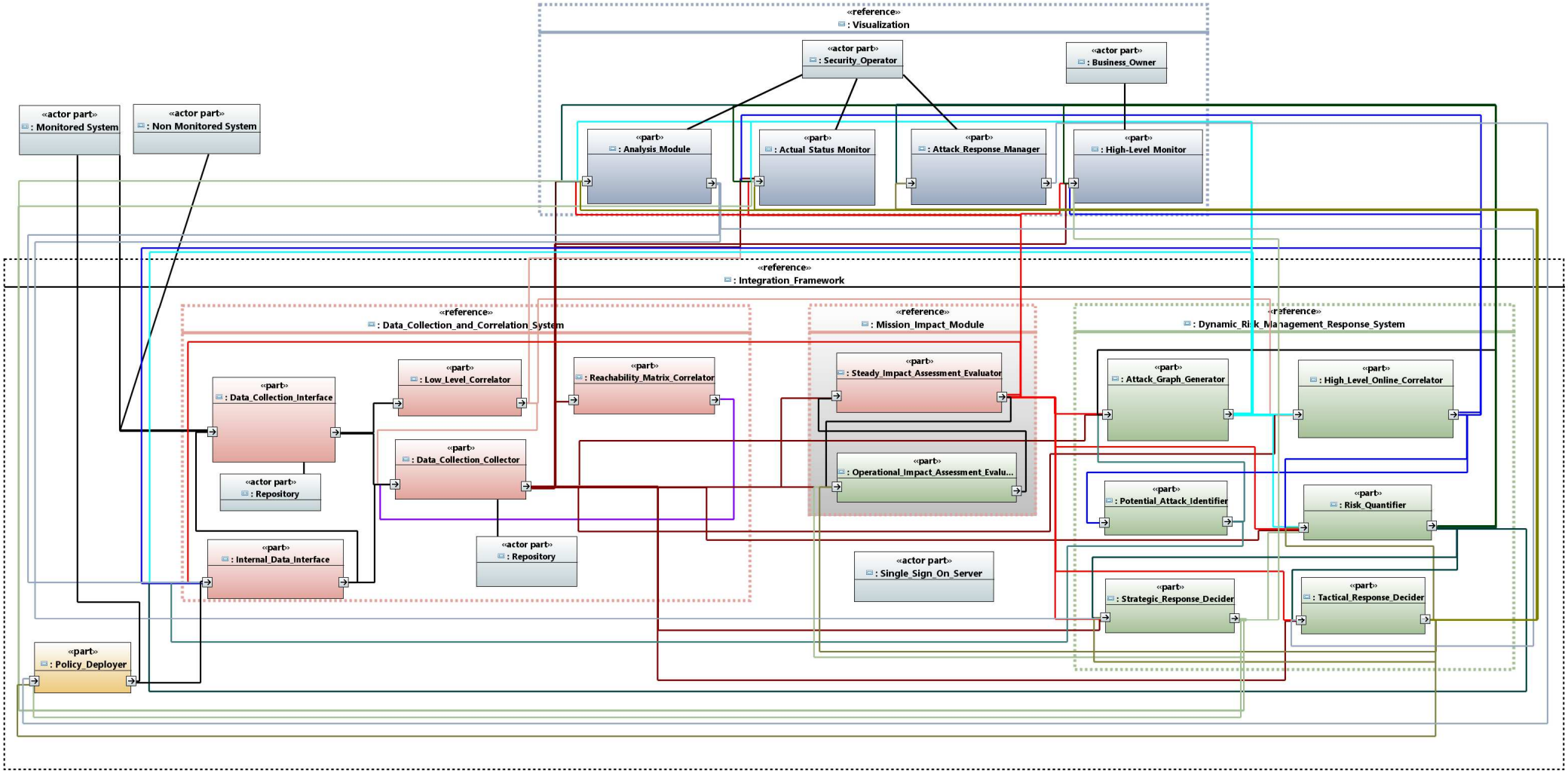


Figure 19 PANOPTESEC System High Level Logic View with focus on Mission Impact Module

In Figure 20 a more detailed Logic View of the Mission Impact Module (MIM) component is presented.

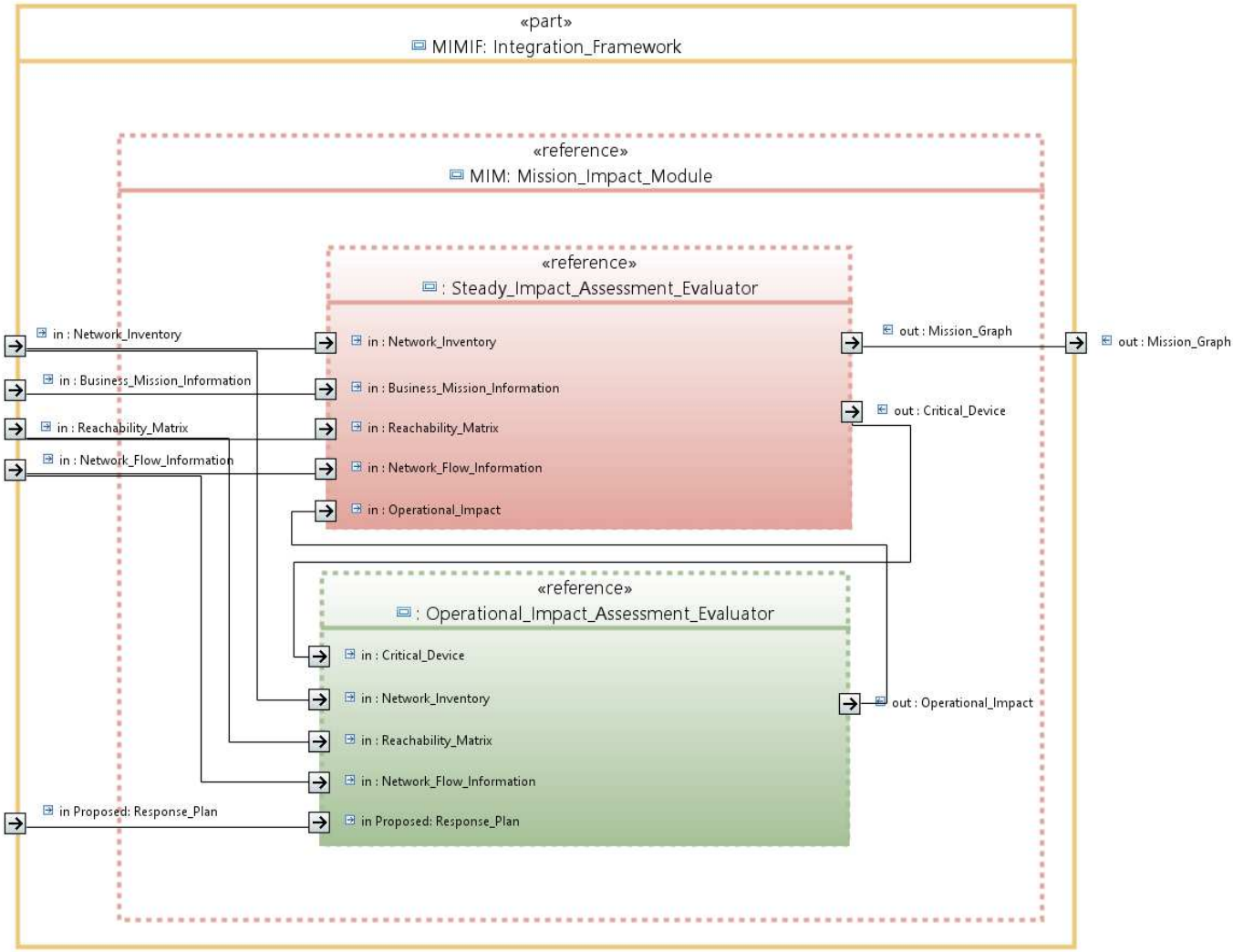


Figure 20: Mission Impact Module Logic View

Figure 20 shows MIM as a component inside an Integration Framework wrapper that manages the external connections and the runtime of the component. MIM will be implemented as a unique component with different, but related, functionalities. The Mission Impact Module directly derives from the Mission Impact Model, identified component in the PANOPTESEC System Preliminary High Level Design depicted in [D3.1.1].

The Mission Impact Module has responsibility for the evaluation of the business critical supporting systems of the Monitored System(s), in order to focus the efforts of the proactive and the reactive chains of risk management.

MIM keeps a track of ongoing business processes (or missions) inside a company and connect the business world with the underlying ICT or SCADA/ICS world (**ICA011, ICA012, ICA013** and **ICA014**). MIM must then correlate *Business_Mission_Information* collected by

the Data Collection and Correlation System and persisted in the DCC component, with the *Reachability_Matrix* from the RMC component, in order to compute the *Mission_Graph* (**ICA015, ICA016, ICA017** and **ICA018**). The *Mission_Graph* also provides the knowledge of the feared events (i.e. detrimental events) and an assessment of the impact and nature of the feared event on the assets for the business of the organization (**ICA019)**. This evaluation has a particular impact in the proactive chain of risk management: the *Mission_Graph* is one of the main inputs for the Attack Graph Generator component (in the Dynamic Risk Management Response System).

In addition, MIM will evaluate the *Mission_Impact* related to *Proposed Mitigation_Action* arising from both in the proactive and reactive chains proposed by the Strategic Response Decider and the Tactical Response Decider (**ICA020**), in order to minimize potential "collateral damage" posed by the PANOPTESEC System to the business layer.

MIM is composed of two main sub-components:

- Steady Impact Assessment Evaluator (developed under Work Package 4)
- Operational Impact Assessment Evaluator (developed under Work Package 5)

The purpose of the Steady Impact Assessment Evaluator (SIAE) component is to represent general business structures and build a bridge towards the underlying ICT or SCADA/ICS environment and topology. It furthers provides a mapping of potential threats at against ICT or SCADA/ICS environments towards that impact the business structure. This is heavily important as one threat in the ICT or SCADA/ICS environment might globally affect several business processes.

The Operational Impact Assessment Evaluator (OIAE) component is responsible for the implementation of the Response Operational Impact Assessment (ROIA) process, which assesses the *Operational_Impact* (**ICA020**) of a *Response_Plan* (after a request by the Strategic/Tactical Response Decider from the Dynamic Risk Management Response System). The key feature of the OIAE component is that it is able to address negative side effects of Response Plans proposed by other modules (e.g. Strategic Response Decider or Tactical Response Decider components inside the Dynamic Risk Management Response System, as requested by **PRS011** and **RRS010**). A specific example would be that, in order to mitigate a potential attack scenario, the best Response Plan would be to shut down a specific connection. The OIAE then analyses this Response Plan in order to assess the Impact of this action onto all ongoing business processes. In fact, this shut down might heavily prohibit another device from working properly, which then affects one or several business processes.

In order to fulfil **PRT005**, a set of fixed interfaces for the Mission Impact Module component is then defined.

IN Interface ValueTypes with the Data Collection Collector component
Reachability_Matrix
Network_Inventory
Business_Mission_Information

Network_Flow_Information

In Interface ValueTypes with the Strategic Response Decider component
(Proposed) *Response_Plan*

In Interface ValueTypes with the Tactical Response Decider component
(Proposed) *Response_Plan*

OUT Interface ValueTypes with the Attack Graph Generator component
Mission_Graph

OUT Interface ValueTypes with the Risk Quantifier component
Mission_Graph

OUT Interface ValueTypes with the Visualization System
Mission_Graph

OUT Interface ValueTypes with the Strategic Response Decider component
(*Operational_Impact*) *Mission_Graph*

OUT Interface ValueTypes with the Tactical Response Decider component
(*Operational_Impact*) *Mission_Graph*

Interactions between MIM and other components of the System are shown in Sequence Diagrams in Figure 21: Visualization System-DCI-DCC-MIM (Business Mission Information Initialization) Process View, Figure 29, Figure 33 and Figure 39.

More details about MIM can be found in [D4.1.2] and [D5.1.2].

Interactions between Visualization System-DCI-DCC-MIM (Business Mission Information initialization)
In Figure 21: Visualization System-DCI-DCC-MIM (Business Mission Information Initialization) Process View a Process View (Sequence Diagram) of the interactions between the Visualization System, DCI, DCC and MIM is presented.

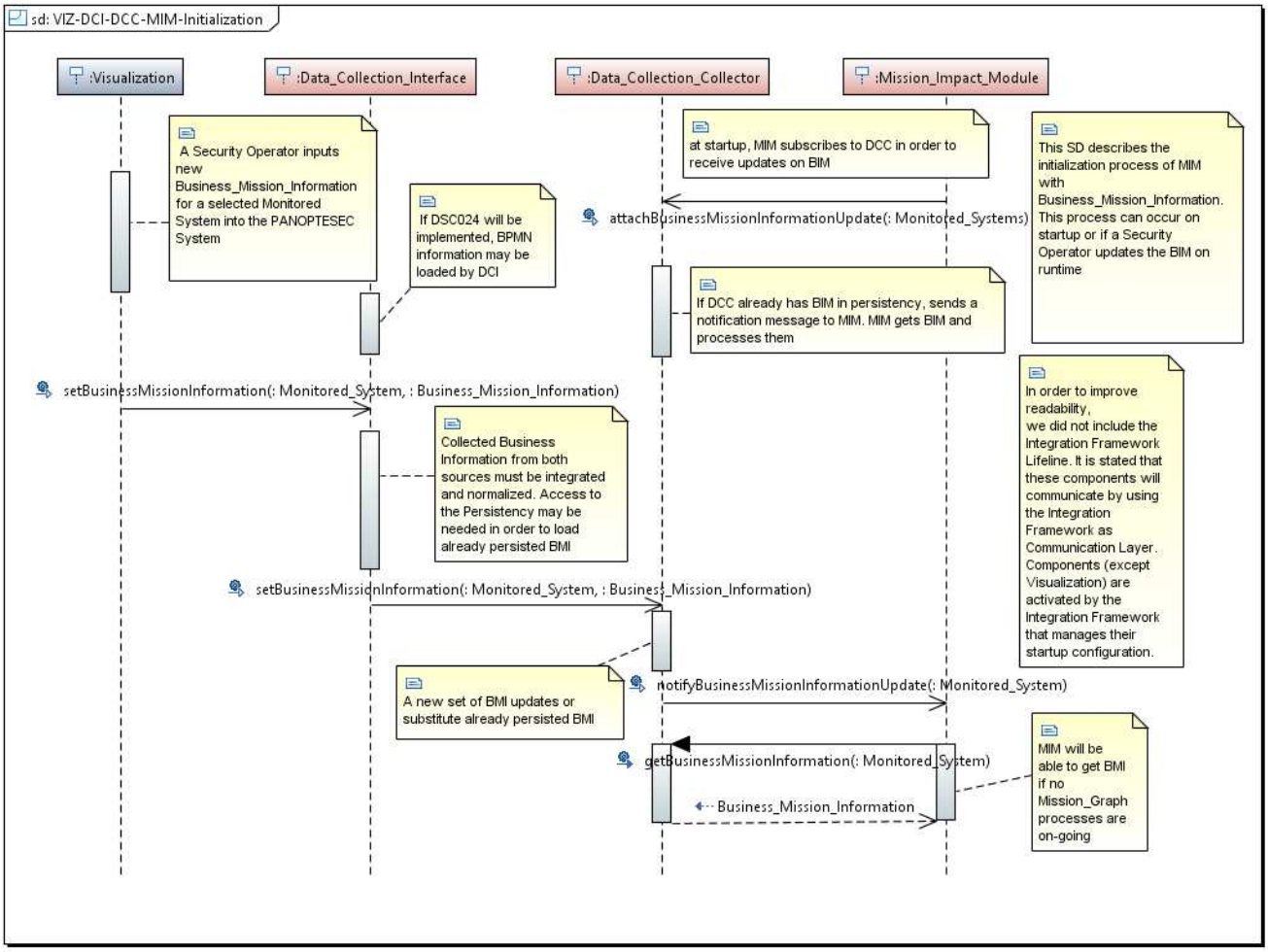


Figure 21: Visualization System-DCI-DCC-MIM (Business Mission Information Initialization) Process View

In order to reduce the size of the diagrams for documentation reasons, Integration Framework lifelines are not shown in this Diagram. As can be seen in the previous Logic Diagrams, the Integration Framework will be directly managing all communication between the components. More details in Section 5.2.5.5.

As stated in **VIZ018**, **VIZ019** and **DSC025** the Visualization System of the PANOPTESEC System needs to be able to configure and update information related to the critical infrastructures of the Monitored System(s). It will be then possible to insert, manipulate and update *Business_Mission_Information* by using the Visualization System.

Since **DSC025** envisions the possibility to automatically collect *Business_Mission_Information,* the design covers this Requirement by adding a Business Mission data input in DCI (for example, BPMN 2.0 files). Since it is then possible to have two data input for *Business_Mission_Information*, a consistency/normalization mechanism must be designed.

At start-up, the MIM subscribes to DCC in order to get updates on *Business_Mission_Information.* If *Business_Mission_Information* are already stored in the persistency, DCC sends a notification message to MIM and MIM is able to load the data from the persistency. However, when a Security Operator updates or add new *Business_Mission_Information* by using the Visualization System, it then pushes *Business_Mission_Information* to DCI, while DCI collects Business Mission raw files from the Monitored System(s).

After a normalization/consistency phase, the results are stored by the Persistency Manager. DCC sends a notification message to MIM in order to inform it that a new set of *Business_Mission_Information* is available. Depending on its configuration, MIM can stop any on-going process and compute the new data set, or wait for the end of the computation in order to get the new *Business_Mission_Information.* Since *Business_Mission_Information* should not change often, the risk of racing conditions is avoided even is MIM would stop its computation if new *Business_Mission_Information* are transmitted.

5.2.5.3 Dynamic Risk Management Response System

In Figure 22 a High Level Logic View of the PANOPTESEC System with focus on the Dynamic Risk Management Response System is presented.

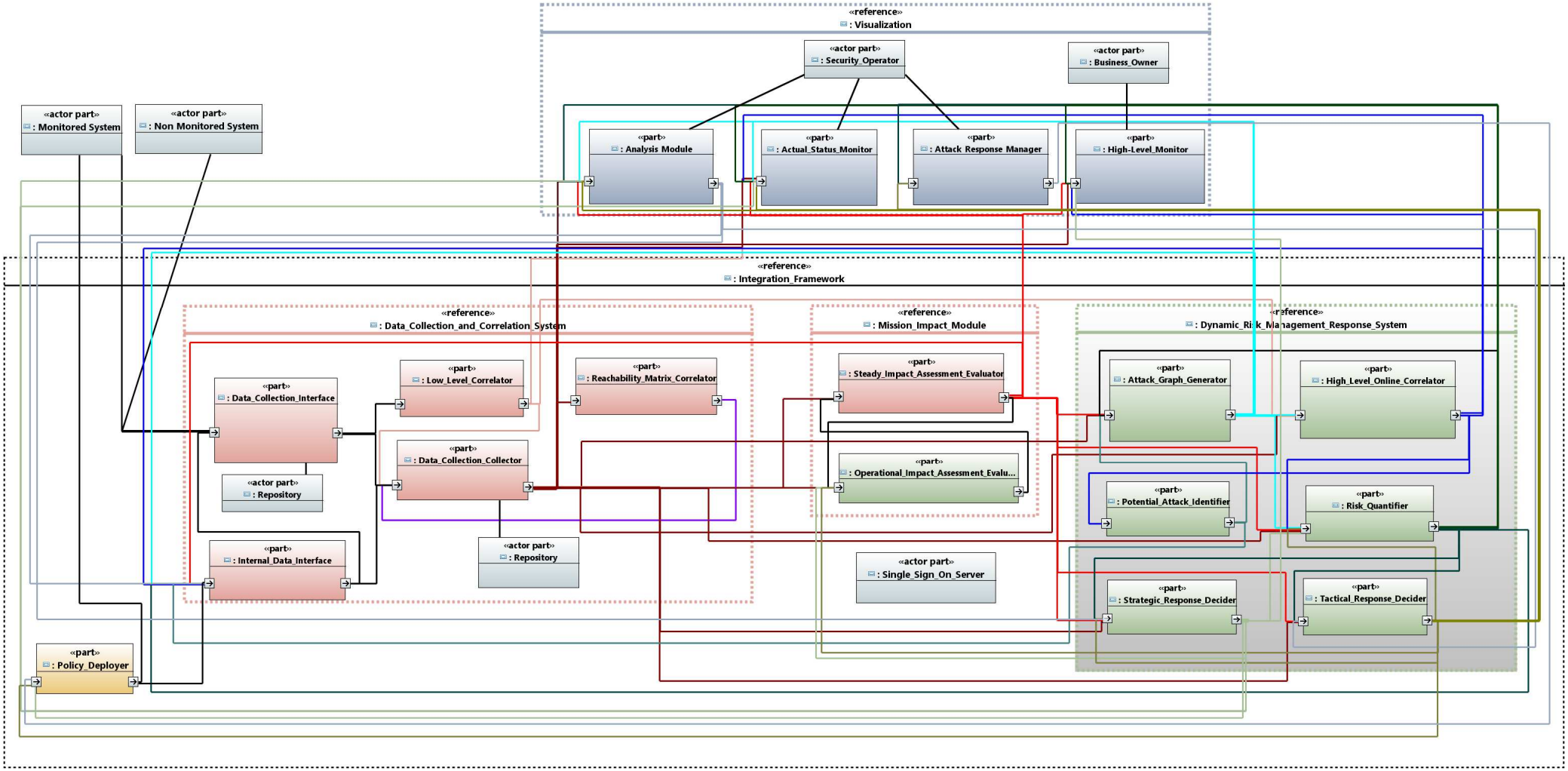


Figure 22: PANOPTESEC System Logic View with focus on Dynamic Risk Management Response System

The Dynamic Risk Management Response System is the PANOPTESEC System package depicted on managing the reactive and proactive chains. In Figure 22 it is evident that the Dynamic Risk Management Response System architecture satisfies **CMP001**, **MNT001**, and **PRT004**. This package is not designed as a monolithic multipurpose block, but as a set of functional blocks glued by the Integration Framework, with respect of **CMP002**, **CMP003**, **CMP004** and **PRT005**.

As already presented in Figure 3, the Dynamic Risk Management Response System encompasses the functionalities requested in the Functional ASRs of the Proactive Response System and of the Reactive Response System (Section 5.1.1).

Dynamic Risk Management Response System is composed of six main components:

- Attack Graph Generator
- High Level Online Correlator
- Potential Attack Identifier
- Risk Quantifier
- Strategic Response Decider
- Tactical Response Decider

The components of the Dynamic Risk Management Response System that specifically participate to the proactive management chain are: the Attack Graph Generator (AGG), the Response Financial Impact Assessor (a sub-component inside the Risk Quantifier component) and the Strategic Response Decider (SRD). Additionally, the Risk Quantifier and the Operational Impact Assessment Evaluator (Section 5.2.5.2), provide the Response Operational Impact Assessment (ROIA) evaluations, participate to the proactive perspective on a non-exclusive way.

The components of the Dynamic Risk Management Response System that specifically participate to the reactive management chain are: the High-Level Online Correlator component (HOC), the Potential Attack Identifier and the Tactical Response Decider (TRD). Additionally, the Risk Quantifier and the Operational Impact Assessment Evaluator (Section 5.2.5.2), provide the Response Operational Impact Assessment (ROIA) evaluations, participate to the reactive perspective on a non-exclusive way.

The following decomposition of the Dynamic Risk Management Response System begins with the analysis of the components commonly used by the proactive and the reactive chain (Risk Quantifier and its sub-components), continues with components involved in the proactive chain (Attack Graph Generator and Strategic Response Decider) and with components involved in the reactive chain (High Level Online Correlator, Potential Attack Identifier and Tactical Response Decider).

In order to give a more precise understanding of the Data Collection and Correlation System, a sub-set of the Diagram in Figure 22 is given.

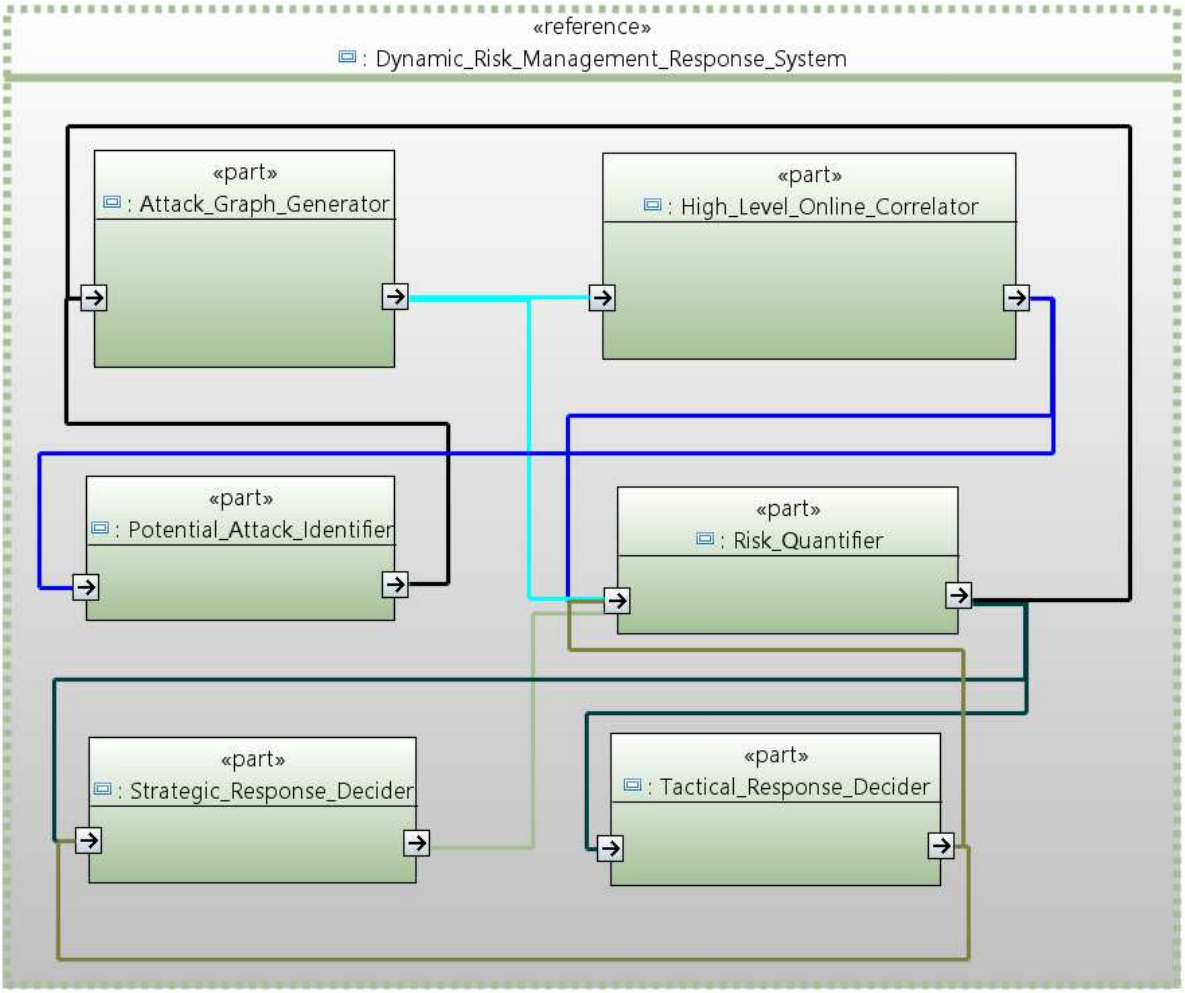


Figure 23: Dynamic Risk Management Response System High Level Logic View

Risk Quantifier

In Figure 24 a Logic View of the Risk Quantifier (RQ) component is presented.

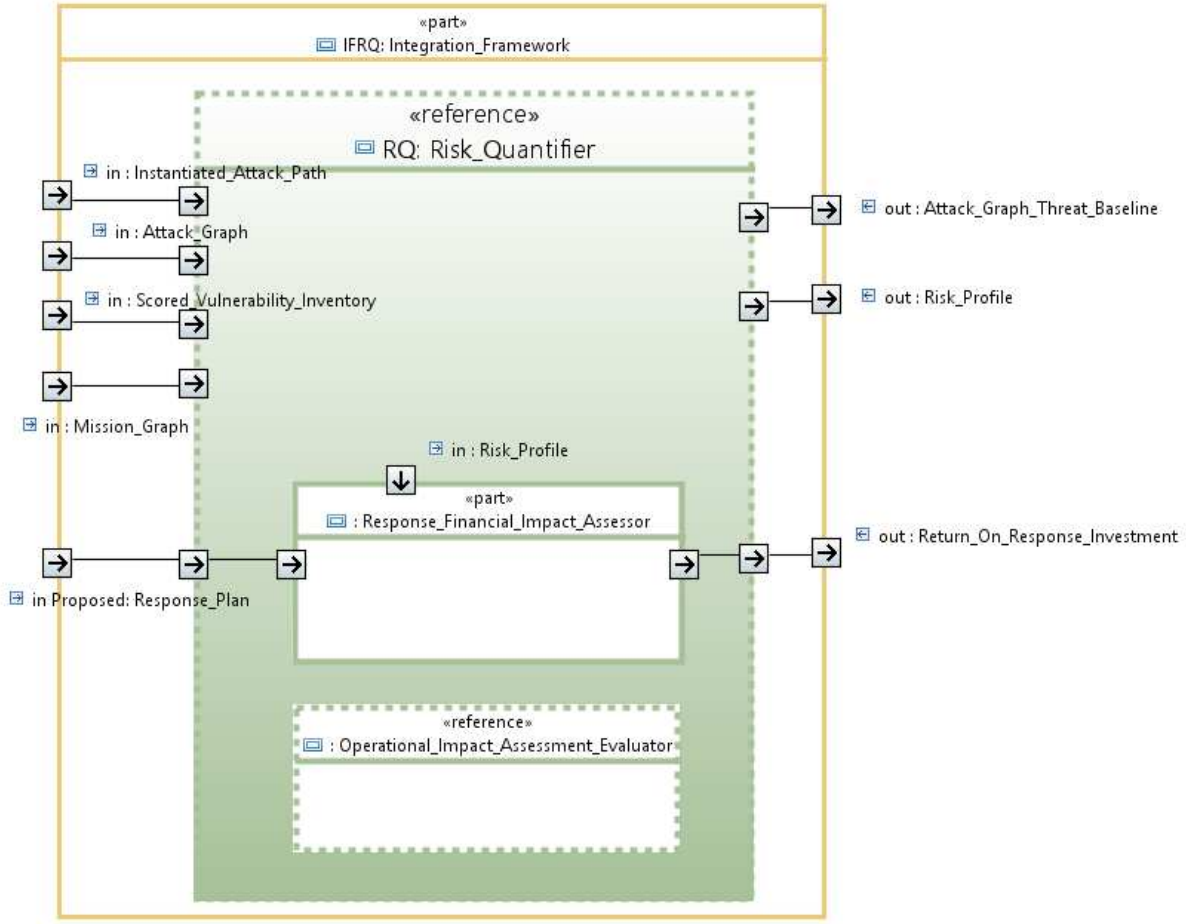


Figure 24: Risk Quantifier Logic View

Figure 24 shows RQ as a component inside an Integration Framework wrapper that manages the external connections and the runtime of the component.

The Risk Quantifier component has three main purposes (and it is consequently divided in three sub-components):

- Quantifying risk values or levels (*Risk_Profile*) for the threats identified at the Attack Awareness stage (AGG, HOC): the component will compute and maintain the most exhaustive list of the contribution to the risk (c.f. Section 5.2.4.1, *Risk_Profile*, for further information about the applied definition of Risk in the PANOPTESEC Project) of each identified threat both on proactive and reactive perspective.

  In the proactive perspective, this evaluation takes into account *Mission_Graph* information from the Mission Impact Module component (**PRS008**) and the *Attack_Graph* produced by the Attack Graph Generator component (**PRS007**). The computed *Risk_Profile* is then available for current state analysis by, for example, Security Operators using the Visualization System. In proactive perspective, RQ component does not consider *Normalized_Alert(s)* or *Instantiated_Attack_Path* in order to calculate the *Risk_Profile*, as stated in **PRS009**.

In the reactive perspective, the process of evaluation of the *Risk_Profile* is triggered by the input of *Instantiated_Attack_Path(s)* from the High Level Online Correlation component (**RRS003**). This process relies also on vulnerabilitiesinformation, *Scored_Vulnerability_Inventory* from the Data Collection and Correlation System (**RRS006)**, devices information from the *Attack_Graph* from AGG (**RRS004**) and *Mission_Graph* from the Mission Impact Module (**RRS002** and **RRS008**). In this perspective, RQ evaluates the *Risk_Profile* associated to the current attack(s) (as stated in **RRS007**).

- Assessing the *Operational_Impact* of proposed *Response_Plans* computed by the Strategic/Tactical Response Decider components (as stated in **PRS010**, **PRS011**, **RRS009**, **RRS010**), on the Assets identified in a *Mission_Graph*. As can be seen in Figure 24, the RQ component manages an instance of the Operational Impact Assessment Evaluator in order to process the *Operational_Impact* and the ROIA process (cf. Section 5.2.5.2, Operational Impact Assessment Evaluator);
- Assessing the Financial Impact (i.e. *Return_On_Response_Investment*) of *Mitigation_Action*(s) composing (Proposed) *Response_Plan*, as stated in **PRS011**. This function (carried out by the internal sub-component Response Financial Impact Assessor) provides an assessment about the potential impact that some given *Response_Plan*, in terms of mitigation of the assessed risks, may cause to an organization (from a financial perspective).

Contrary to the proactive chain, the reactive chain has more demanding time constraints in order to respond to on-going attack scenarios (as stated in **PRF006**). Particularly, the High-Level Online Correlator component must cope with such constraints while being able to track the progress of multiple and simultaneous attack scenarios. Considering that an *Attack_Graph* can contain thousands of *Attack_Path*(s), the High-Level Online Correlator component will use prioritization mechanisms in order to handle the most imminent and riskiest attack scenarios. AGG will then tag each *Attack_Path* by their risk level as calculated by Risk Quantifier component and returned to AGG using the *Attack_Graph_Threat_Baseline* ValueType before sending them to the High-Level Online Correlator component. This process is also stated in **RRS017**.

In order to fulfil **PRT005**, a set of fixed interfaces for the RQ component is then defined.

IN Interface ValueTypes with the Data Collection Collector component
Scored_Vulnerability_Inventory

IN Interface ValueTypes with the High Level Online Correlator component
Instantiated_Attack_Path

IN Interface ValueTypes with the Attack Graph Generator component
Attack_Graph

IN Interface ValueTypes with the Mission Impact Module
Mission_Graph

IN Interface ValueTypes with the Strategic Response Decider
(Proposed) *Response_Plan*

OUT Interface ValueTypes with the Strategic Response Decider component
Return_On_Response_Investment

OUT Interface ValueTypes with the Internal Data Interface component
Risk_Profile

OUT Interface ValueTypes with the Visualization System
Risk_Profile

OUT Interface ValueTypes with the Attack Graph Generator component
Attack_Graph_Threat_Baseline

OUT Interface ValueTypes with the Strategic Response Decider component
Risk_Profile

OUT Interface ValueTypes with the Tactical Response Decider component
Risk_Profile

Interactions between RQ and other components of the System are shown in Sequence Diagrams in Figure 29, Figure 33, Figure 39 and Figure 40.

More details about RQ can be found in [D5.1.2].

Attack Graph Generator
In Figure 25 a Logic View of the Attack Graph Generator (AGG) component is presented.

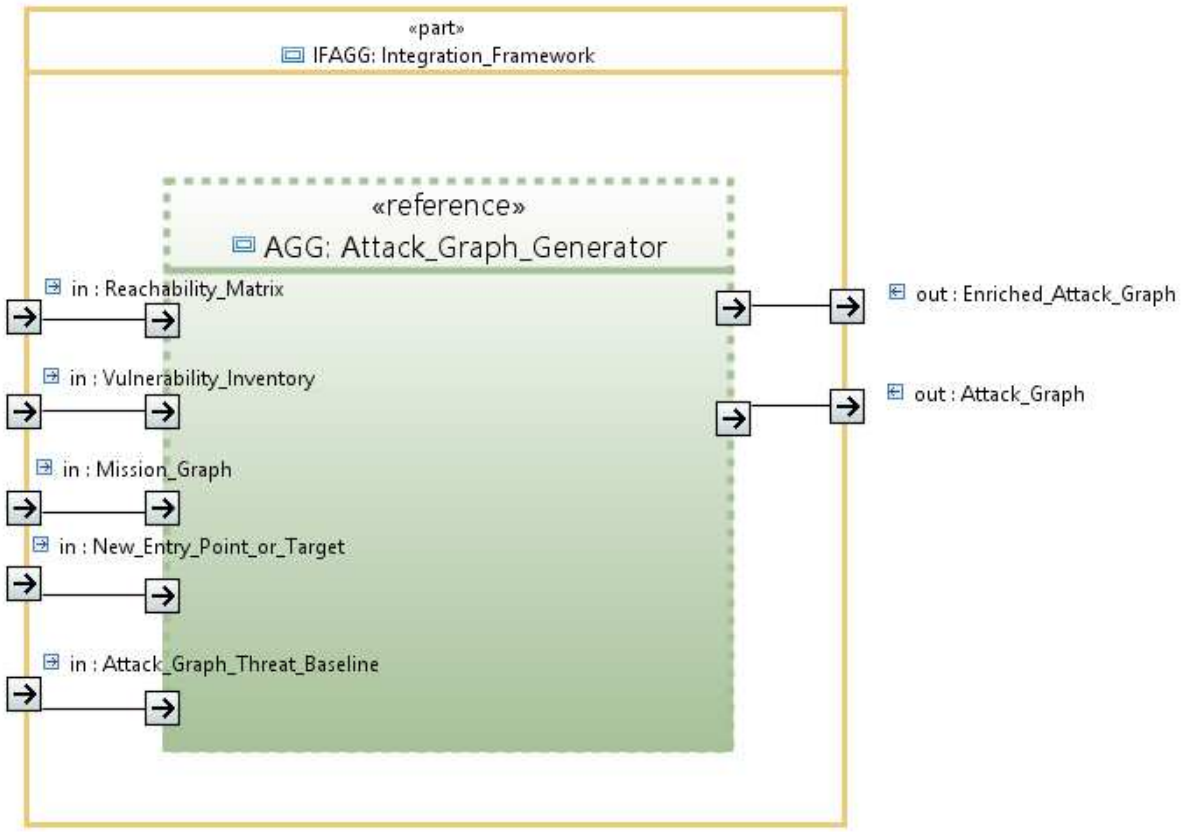


Figure 25: Attack Graph Generator Logic View

Figure 25 shows AGG as a component inside an Integration Framework wrapper that manages the external connections and the runtime of the component.

The objective of the Attack Graph Generator (AGG) component is to capture the exposure of the Monitored System(s) to potential and ongoing attack scenarios, whether they are internal or external attacks based on the present of known vulnerabilities and logical connectivity. This step is crucial for both proactive and reactive risk management chains. Thus, the Attack Graph Generation component offers three major features for the proactive and reactive risk management chains:

- Attack Graph Generation (stated in **PRS002**);
- Construction of internal vulnerabilities structure (within a node), stated in **PRS006.** Vulnerabilities information for the Monitored System(s) are embedded in the *Reachability_Matrix* ValueType and detailed in the correspondent *Vulnerability_Inventory*;
- Attack Graph Enrichment (stated in **RRS017.** This process will increase the global performances of the reactive chain in order to fulfil **PRF006**).

Regarding the proactive chain, AGG will generate the *Attack_Graph* of each Monitored System by considering its topology/connectivity and existing vulnerabilities (stated in **PRS004** and **PRS005**. This information is contained in the *Reachability_Matrix* received by

the Data Collection Collector component). An *Attack_Graph* consists of at least one *Attack_Path*, which starts from an identified entry points (identified in the *Mission_Graph* provided by the Mission Impact Module, as stated in **PRS003**) and leads to the identified critical component/machine in the topology (i.e. supporting assets). An *Attack_Graph* depicts the exposure of Monitored System(s) by enabling the Risk Quantifier component to evaluate the likelihood, impact and the risk level of each potential attack scenario. Hence, The AGG component will generate the *Attack_Graph* and send them to the Risk Quantifier component.

For the reactive risk management chain, *Attack_Graph* can also depict the progress of an on-going attack in one of the Monitored System(s). By depicting potential attack scenarios, the generated *Attack_Graph* are also used by the High-level-correlation engines in order to track the progress of on-going attacks. Hence, AGG will provide the generated *Attack_Graph* for the reactive risk management chain through the High-Level Correlation Engines. In the reactive chain, an accurate depiction of attack scenarios is needed in order to enable correlation components to match between *Attack_Path*(s) and received *Normalized_Alert*(s).

Contrary to the proactive chain, the reactive chain has more demanding time constrains in order to respond to ongoing attack scenarios (as stated in **PRF006**). Particularly, the High-Level Online Correlator component must cope with such constraints while being able to track the progress of multiple and simultaneous attack scenarios. Considering that an *Attack_Graph* can contain thousands of *Attack_Path*(s), the High-Level Online Correlator component will use prioritization mechanisms in order to handle the most imminent and riskiest attack scenarios. AGG will then tag each *Attack_Path* by their Risk level as calculated by Risk Quantifier component, using the *Enriched_Attack_Graph* ValueType, before sending them to the High-Level Online Correlator component.

Another input for the Attack Graph Generator comes from the Potential Attack Identifier component, in order to fulfil **RRS023**. PAI processes over the reactive chain may discover new possible entry points for the *Attack_Path* processing in AGG: if new entry points are discovered, PAI sends the information to AGG (*New_Entry_Point_or_Target).*

In order to fulfil **PRT005**, a set of fixed interfaces for the AGG component is then defined.

IN Interface ValueTypes with the Data Collection Collector component
Reachability_Matrix
Vulnerability_Inventory

IN Interface ValueTypes with the Mission Impact Module component
Mission_Graph

IN Interface ValueTypes with the Potential Attack Identifier component
New_Entry_Point_or_Target

IN Interface ValueTypes with the Risk Quantifier component
Attack_Graph_Threat_Baseline

OUT Interface ValueTypes with the Internal Data Interface component
Attack_Graph

OUT Interface ValueTypes with the High Level Online Correlator component
Enriched_Attack_Graph

OUT Interface ValueTypes with the Risk Quantifier component
Attack_Graph

OUT Interface ValueTypes with the Visualization System
Attack_Graph

Interactions between AGG and other components of the System are shown in Sequence Diagrams in Figure 29 and Figure 39.

More details about AGG can be found in [D5.1.2].

Strategic Response Decider
In Figure 26 a Logic View of the Strategic Response Decider (SRD) component is presented.

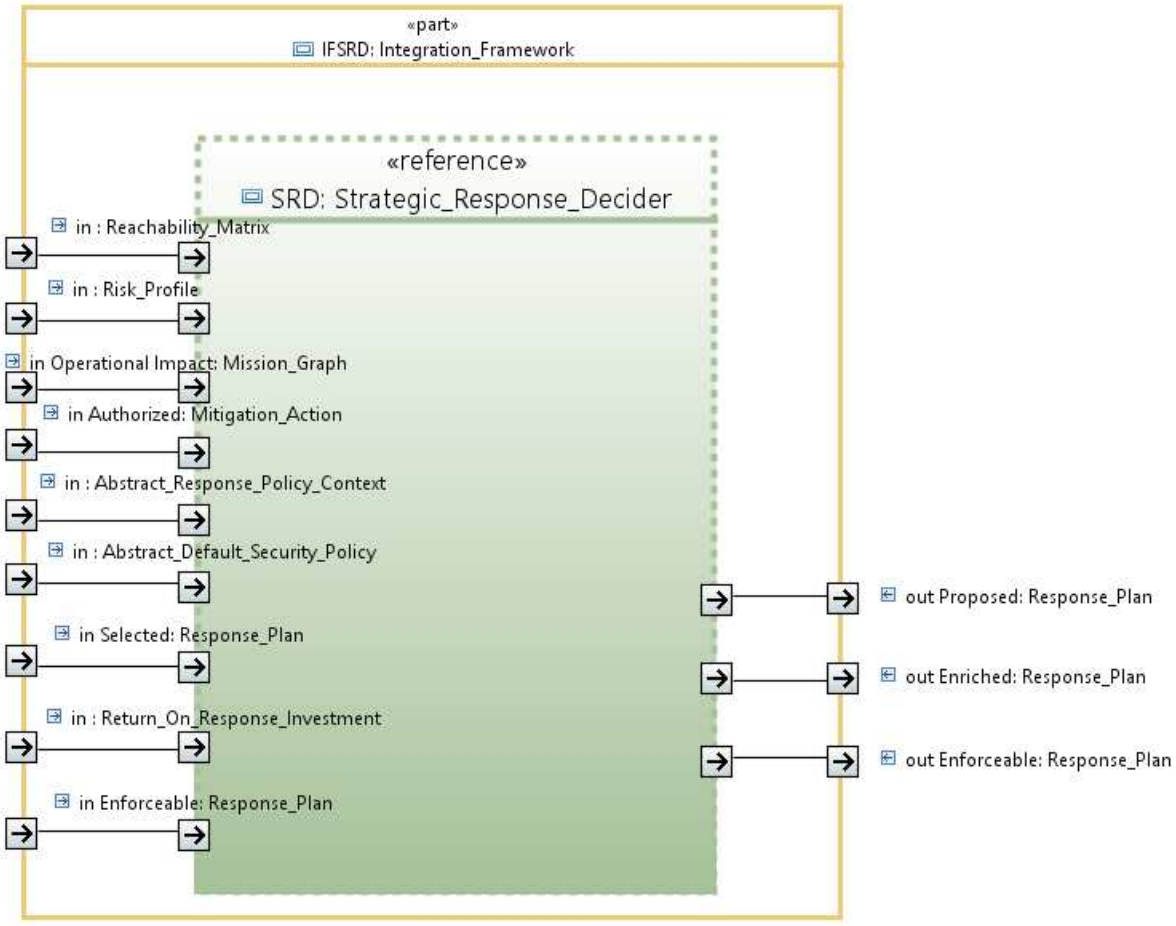


Figure 26: Strategic Response Decider Logic View

Figure 26 shows SRD as a component inside an Integration Framework wrapper that manages the external connections of the component.

The Strategic Response Decider (SRD) component, part of the proactive chain, relies on policy-driven mitigation. It handles identified risks (in form of *Risk_Profile*(s) from the Risk Quantifier component) and provides *Mitigation_Action*(s) on a long-term proactive perspective. Proposed *Mitigation_Action*(s) are evaluated using a *Return_On_Response_Investment* (RORI) index in order to minimise the current proactive level of Risk (stated in **PRS010**) and the *Operational_Impact* on identified *Critical_Device*(s) (stated in **PRS011**).

It is mandatory that the Proposed *Mitigation_Action*(s) are selected from a list of Authorized *Mitigation_Action*(s), loaded from the DCC component in the Data Collection and Correlation System (**PRS012**).

The Strategic Response Decider (SRD) component translates abstract policies (*Abstract_Response_Policy_Context* and *Abstract_Default_Security_Policy*) and contextual data into concrete policy instances (**PRS017**). Then, it provides them as concrete

Response_Plan(s) on a long-term proactive perspective. *Response_Plan*(s) must be validated by Security Operators in the Visualization System prior enforcement (**PRS014** and **PRS015**). The instantiation of security policies will be performed via the Organization-based Access Control (OrBAC) model ([ABO03], [MIEI05]). OrBAC allows modelling abstract security policies to organizational entities (e.g., organizations, roles, activities, views, contexts), as well as concrete security policies to instantiated entities (e.g., subjects, actions, objects). More information about the OrBAC model and its expected application with the PANOPTESEC project are also available in [D2.1.1].

The aim of the resulting application is the automatic guidance of pre-defined security directives and attributes, provided by the PANOPTESEC System, and their eventual combination with ongoing *Risk_Profile*(s) from the Risk Quantifier component. The attributes are mapped with the high level (abstract) policies, in order to automatically infer *Mitigation_Action*(s).

Mitigation_Action(s) are grouped into series of concrete *Response_Plan*(s) expected to be enforced (**PRS016**) by already deployed policy enforcement points, such as firewalls (access control lists) and routers (routing-based mitigation).

The enforcement of the concrete *Response_Plan*(s) per policy enforcement point is conducted by a Policy Deployer component directly managed by the Integration Framework. The Strategic Response Decider will also take into account in its evaluation every already enforced *Response_Plan* from the Tactical Response Decider.

In order to fulfil **PRT005**, a set of fixed interfaces for the SRD component is then defined.

IN Interface ValueTypes with the Data Collection Collector component
Reachability_Matrix
Abstract_Default_Security_Policy
Abstract_Response_Policy_Context
(Authorized) Mitigation_Action(s)

IN Interface ValueTypes with the Risk Quantifier component
Risk_Profile (only a notification of the production of a new (proactive) *Risk_Profile* is send to SRD)

IN Interface ValueTypes with the Visualization System
(Selected) Response Plan

OUT Interface ValueTypes with the Risk Quantifier component
(Proposed) Response Plan

OUT Interface ValueTypes with the Operational Impact Assessment Evaluator component (inside MIM)
(Proposed) Response Plan

OUT Interface ValueTypes with the Visualization System
(Enriched) Response Plan

OUT Interface ValueTypes with the Policy Deployer component
(Enforceable) Response Plan

Interactions between SRD and other components of the System are shown in Sequence Diagrams in Figure 27, Figure 28 and Figure 29.

More details about SRD and OrBAC can be found in [D5.1.2].

Interactions between Visualization System-DCI-DCC-SRD (Security Policy Initialization)
In Figure 27 a Process View (Sequence Diagram) of the interactions between Visualization System, DCI, DCC and SRD is presented.

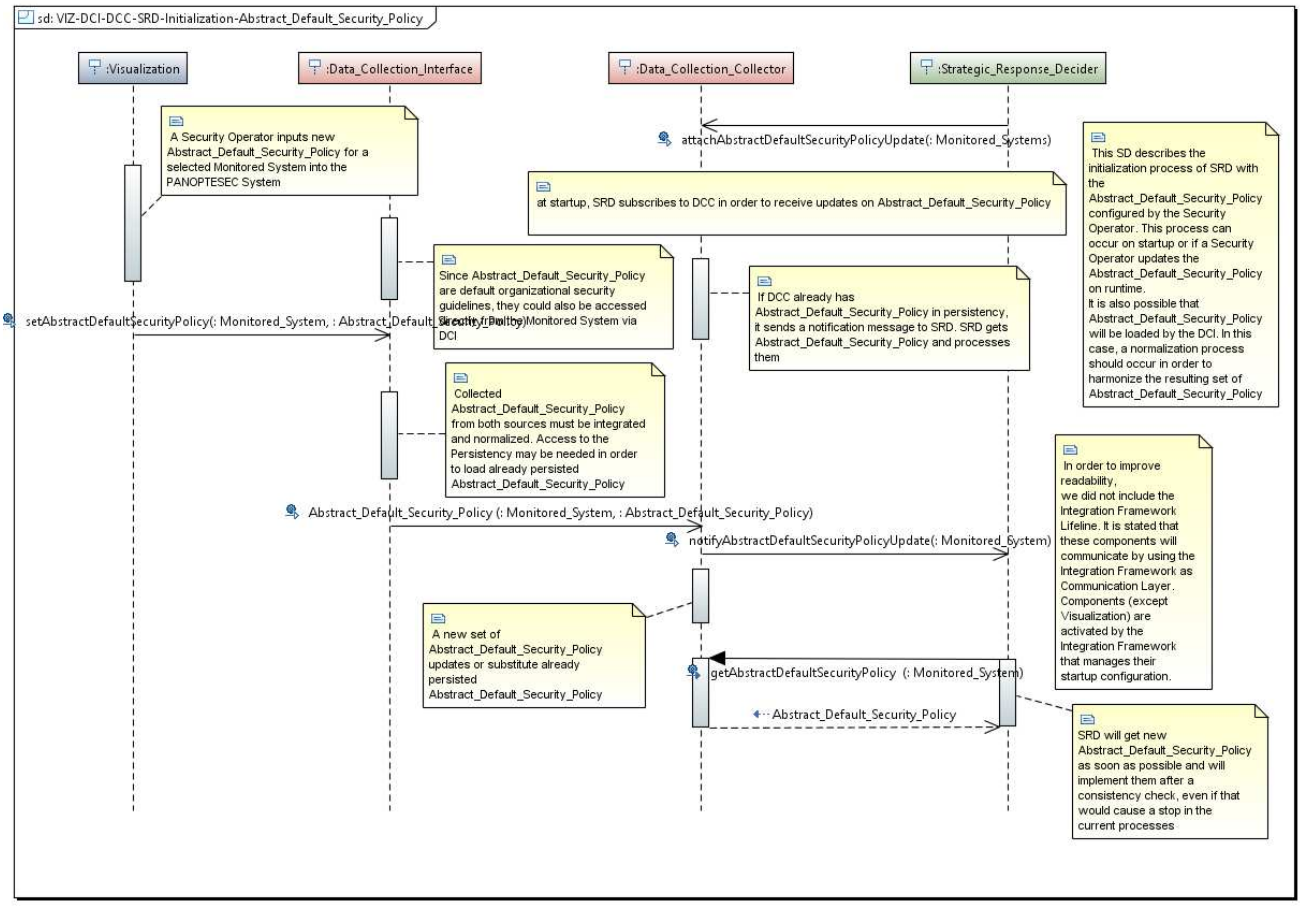


Figure 27: Visualization System-DCI-DCC-SRD (Security Policy Initialization) Process View

In order to reduce the size of the diagrams for documentation reasons, Integration Framework lifelines are not shown in this Diagram. As can be seen in the previous Logic Diagrams, the Integration Framework will be directly managing all communication between the components. More details in Section 5.2.5.5.

To evaluate the Proposed (*Response_Plan*), SRD relies on *Abstract_Default_Security_Policy* and *Abstract_Response_Policy_Context* (cf. 5.2.4.1 and [D5.1.2] for further information), as it is defined in **PRS017**. Both sets of policies will be loaded by the Visualization System, while *Abstract_Default_Security_Policy* may also be loaded directly from the Monitored System(s). This is done usability reasons as it is evaluated as easier for the Security Operator to have a unique environment by which he can load initialization data for the PANOPTESEC System. *Abstract_Default_Security_Policy* may be loaded from static firewalling rules, routers configurations and tunnelling configurations.

Since it is then possible to have two data input for *Abstract_Default_Security_Policy*, a consistency/normalization mechanism must be designed. At start-up, SRD subscribes to DCC in order to get updates on *Abstract_Default_Security_Policy.* If DCC already has *Abstract_Default_Security_Policy* in the persistency, it sends a notification message to SRD and SRD is able to load the data from the persistency.

When a Security Operator updates or add new *Abstract_Default_Security_Policy* by using the Visualization System, it then pushes *Abstract_Default_Security_Policy* to DCI, while DCI collects Security Policy raw files from the Monitored System(s).

After a normalization/consistency phase, the results are stored in the persistency. DCC then sends a notification message to SRD in order to inform it that a new set of *Abstract_Default_Security_Policy* is available. Depending on its configuration, SRD can stop any ongoing process and compute the new data set, or wait for the end of the computation in order to get the new *Abstract_Default_Security_Policy.*

The initialization process of *Abstract_Response_Policy_Context* in SRD is similar to the process shown in Figure 27, with the difference that *Abstract_Response_Policy_Context* will be loaded just from the Visualization System.

Interactions between Visualization System-DCC-SRD-TRD (Mitigation Action Initialization)
In Figure 28 a Process View (Sequence Diagram) of the interactions between Visualization System, DCI, DCC, SRD and TRD is presented.

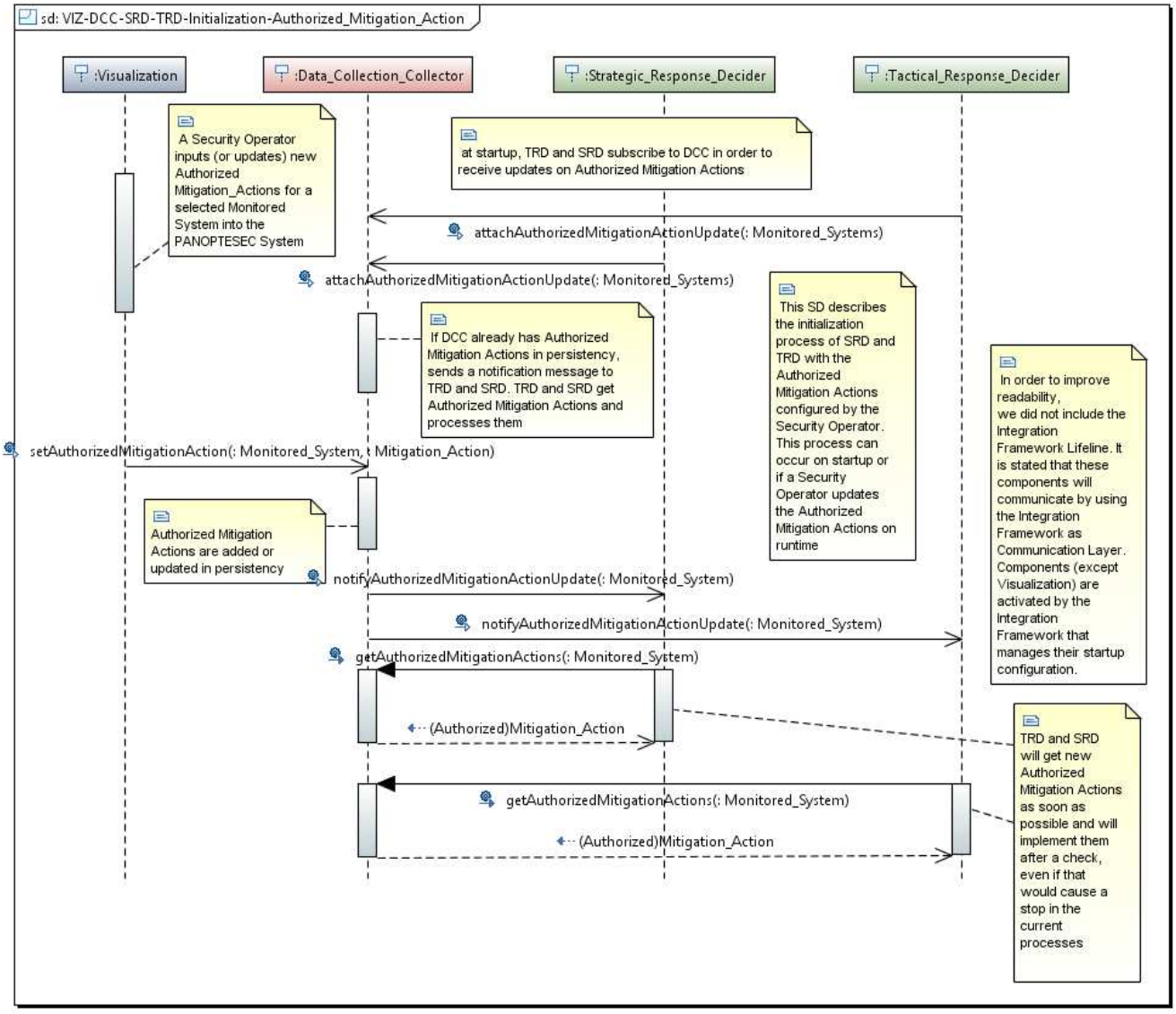


Figure 28: Visualization System-DCC-SRD-TRD (Mitigation Action Initialization)

In order to reduce the size of the diagrams for documentation reasons, Integration Framework lifelines are not shown in this Diagram. As can be seen in the previous Logic Diagrams, the Integration Framework will be directly managing all communication between the components. More details in Section 5.2.5.5.

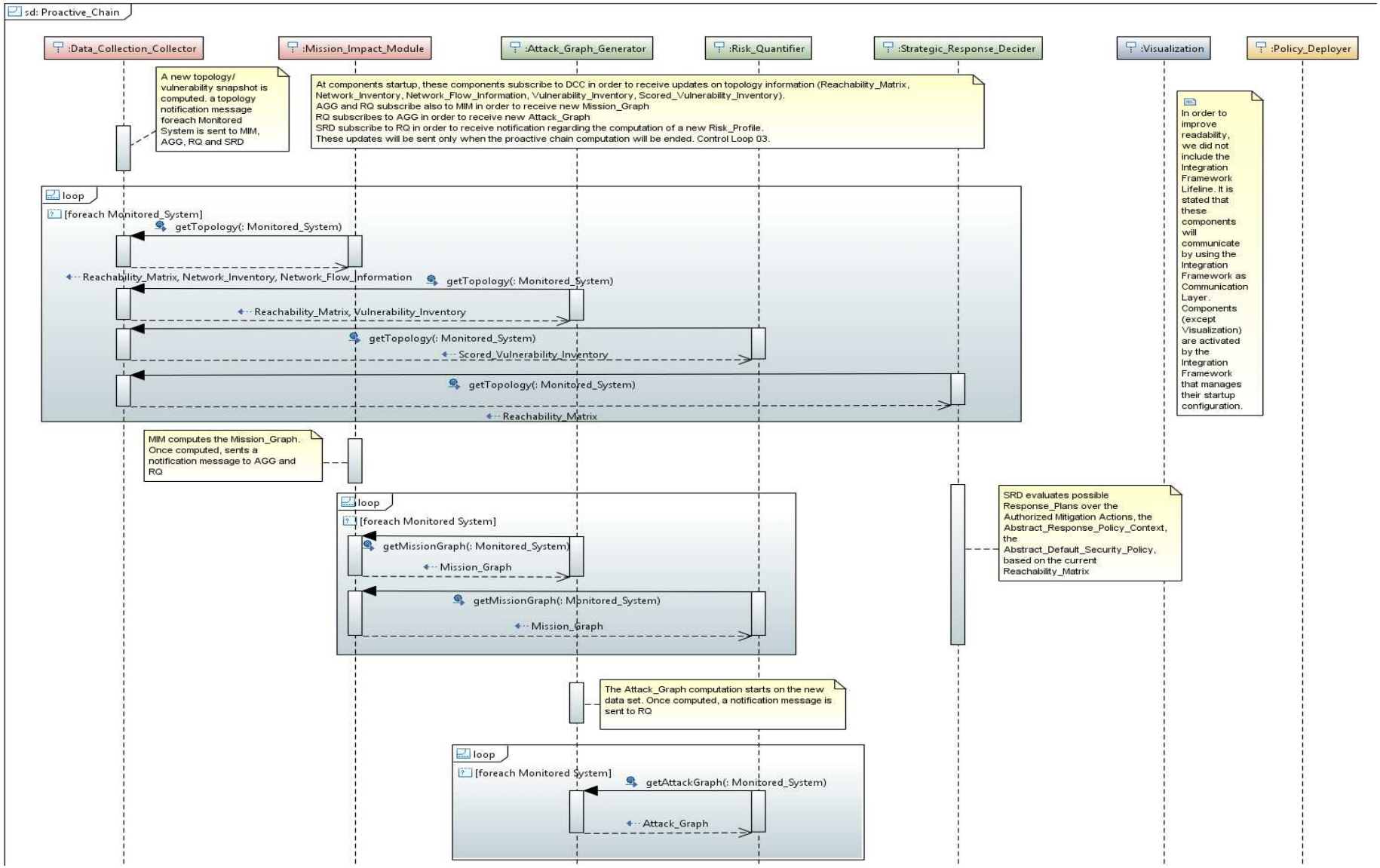
Proposed (*Response_Plan*) from SRD and TRD MUST be evaluated over a given set of (Authorized) *Mitigation_Action*(s) (**PRS012** and **RRS011**). These *Mitigation_Action*(s) will be defined, validated and loaded by Security Operators by using the Visualization system (as it suggests **DSC027**).

At start-up, SRD and TRD subscribe to DCC in order to get updates on (Authorized) *Mitigation_Action*(s). If DCC already has (Authorized) *Mitigation_Action*(s) in the persistency, it sends a notification message to SRD and TRD. The two components are then able to load the data from the persistency.

When a Security Operator updates or add new (Authorized) *Mitigation_Action*(s) by using the Visualization System, it then pushes them in the persistency. DCC then sends a notification message to SRD and TRD in order to inform them that a new set of (Authorized) *Mitigation_Action*(s) is available. Depending on their configuration, SRD and TRD can stop any ongoing process and compute the new data set, or wait for the end of the computation in order to get the new (Authorized) *Mitigation_Action*(s).

Proactive Chain Sequence Diagram

In Figure 29 a Process View (Sequence Diagram) of the proactive chain is presented. Due to its complexity, it is divided in two pages. It can be directly compared to Figure 19 in [D3.1.1].


sd: Proactive_Chain
:Data_Collection_Collector
:Mission_Impact_Module
:Attack_Graph_Generator
:Risk_Quantifier
:Strategic_Response_Decider
:Visualization
:Policy_Deployer
A new topology/ vulnerability snapshot is computed. a topology notification message foreach Monitored System is sent to MIM, AGG, RQ and SRD
At components startup, these components subscribe to DCC in order to receive updates on topology information (Reachability_Matrix, Network_Inventory, Network_Flow_Information, Vulnerability_Inventory, Scored_Vulnerability_Inventory).
AGG and RQ subscribe also to MIM in order to receive new Mission_Graph
RQ subscribes to AGG in order to receive new Attack_Graph
SRD subscribe to RQ in order to receive notification regarding the computation of a new Risk_Profile.
These updates will be sent only when the proactive chain computation will be ended. Control Loop 03.
In order to improve readability, we did not include the Integration Framework Lifeline. It is stated that these components will communicate by using the Integration Framework as Communication Layer. Components (except Visualization) are activated by the Integration Framework that manages their startup configuration.
loop
[foreach Monitored_System]
getTopology(: Monitored_System)
Reachability_Matrix, Network_Inventory, Network_Flow_Information
getTopology(: Monitored_System)
Reachability_Matrix, Vulnerability_Inventory
getTopology(: Monitored_System)
Scored_Vulnerability_Inventory
getTopology(: Monitored_System)
Reachability_Matrix
MIM computes the Mission_Graph. Once computed, sents a notification message to AGG and RQ
loop
[foreach Monitored System]
getMissionGraph(: Monitored_System)
Mission_Graph
getMissionGraph(: Monitored_System)
Mission_Graph
SRD evaluates possible Response_Plans over the Authorized Mitigation Actions, the Abstract_Response_Policy_Context, the Abstract_Default_Security_Policy, based on the current Reachability_Matrix
The Attack_Graph computation starts on the new data set. Once computed, a notification message is sent to RQ
loop
[foreach Monitored System]
getAttackGraph(: Monitored_System)
Attack_Graph

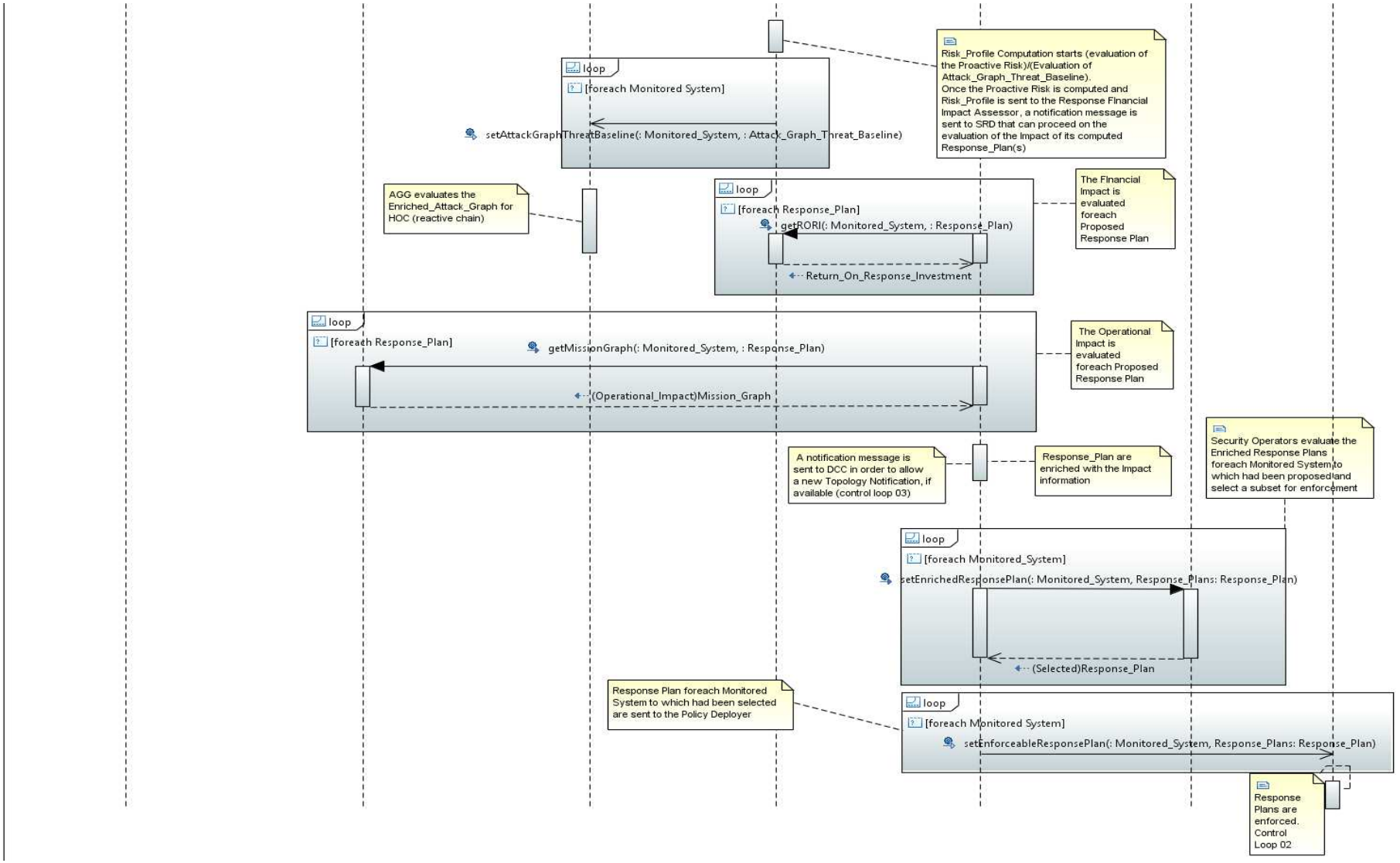


Figure 29: Proactive Response Chain Process View

In order to reduce the size of the diagrams for documentation reasons, Integration Framework lifelines are not shown in this Diagram. As can be seen in the previous Logic Diagrams, the Integration Framework will be directly managing all communication between the components. More details in Section 5.2.5.5.

Interactions depicted in Figure 29 cover most of the possible interactions between PANOPTESEC System components involved in the proactive chain. In [D5.1.2] it is possible to have a more detailed view of the interactions between each component of the Dynamic Risk Management Response System and the others. In the Sequence Diagram in Figure 29 the complete proactive interaction is shown in order to give a complete idea of how the PANOPTESEC System proactive chain will work.

At components start-up, MIM, AGG, RQ and SRD subscribe to DCC (for a set of Monitored System) in order to receive updates on topology information (*Reachability_Matrix, Network_Inventory, Network_Flow_Information, Vulnerability_Inventory, Scored_Vulnerability_Inventory*). AGG and RQ subscribe also to MIM in order to receive new *Mission_Graph.* RQ subscribes to AGG in order to receive new *Attack_Graph.* SRD subscribes to RQ in order to receive notification regarding the computation of a new (Proactive) *Risk_Profile*.

DCC computes and correlates a new snapshot of a Monitored System to which proactive components subscribed to and sends a notification message (with the ID of the snapshot). MIM, AGG, RQ and SRD can now get the needed data, linked to the identified snapshot from DCC.

MIM starts the computation of the *Mission_Graph*, while SRD starts the evaluation of the Monitored System(s) in terms of possible deployable *Response_Plan*(s). As soon as MIM computes the *Mission_Graph* sends a notification to AGG and RQ and they can pull the computed *Mission_Graph*.

AGG starts the evaluation of a new *Attack_Graph*. Once computed, AGG notifies to RQ that a new *Attack_Graph* is ready. RQ can pull it and starts two parallel computations: the first process evaluates the proactive *Risk_Profile* of the collected snapshot. The second process evaluates the risk related to the *Attack_Path*(s) part of the *Attack_Graph* and pushes the results to AGG in form of *Attack_Graph_Threat_Baseline*. AGG prioritizes the *Attach_Path*(s) by their evaluated risk level and, once completed, sends the produced *Enriched_Attack_Graph* to HOC (reactive chain in Figure 33).

Once RQ calculates the new proactive *Risk_Profile*, it pushes it in its internal component, RFIA, and notifies the information to SRD. SRD is then able to ask RQ (and then, RFIA) a RORI impact evaluation on the (Proposed) *Response_Plan*(s) it computed. After the first impact evaluation SRD asks MIM the *Operational_Impact* evaluation on the (Proposed)*Response_Plan*.

Once both evaluations are concluded, SRD enriches the *Response_Plan(*s) with both impact evaluations and sends the (Enriched)*Response_Plan*(s) to the Visualization System for the evaluation by the Security Operator. SRD is now free to send a notification message to DCC in order to inform it that the proactive chain may receive a new topology snapshot, if available.

The Security Operator evaluates the (Enriched)*Response_Plan*(s) and selects some of them, that the Visualization System pushes back to SRD. The SRD needs to know which *Response_Plan* will be enforced in order to better compute the next proactive evaluations. SRD is now able to send the selected *Response_Plan* to the Policy Deployer component asking for the actual deploy in the Monitored System(s).

In conclusion, the proactive chain satisfies **RBL010** and **RBL011**. None PANOPTESEC proactive component is working on a set of data based on a different snapshot of the Monitored System.

The impact of this behaviour on performances can be critical: the proactive chain is able to compute a new Monitored System(s) snapshot only after the computation of MIM, AGG, RQ and SRD. Since Performance ASRs have a less Importance then Reliability ASRs (**RBL011** and **RBL010** are both respected), this risk on performances is considered, at the moment, acceptable.

It is possible to see the benefits of the use of the Data-Shared Pattern (and of course the benefits related to the internal subdivision in different processes applied to the Preliminary High Level Design component Information Correlation Engine), applied to the section of the proactive chain related to the production of the topology information. In [D3.1.1], Section 5.2.5.4, a Control Loop (**Control Loop 01**) had been identified: the data normalization was locked to the proactive chain. Due to the evolution of the internal architecture, data normalization and the computation of the base proactive topology/vulnerability dataset are no more coupled with the proactive chain. As can be seen, the Data Collection and Correlation System can proceed on the topology/vulnerability correlation without waiting for a notification message from SRD. A Control Loop (**Control Loop 03**) is still identified, between SRD and DCC. It is similar to **Control Loop 01**, but with limited scope. A further analysis of the controls loops is provided in Section 5.4.

The PANOPTESEC Consortium will evaluate a possible upgrade of the proactive chain or order to improve performances through iterative development and prototype refinement.

A second Control Loop (**Control Loop 02**), already evaluated in the Preliminary High Level Design, is identified again between the proactive chain and the Monitored System(s). When the proactive chain commands the enforcement of a *Response_Plan*, this process will affect the Monitored System(s) and, following the PANOPTESEC System architecture, it will be reflected in future PANOPTESEC System topology/vulnerabilities computations. This update of the Monitored System will finally be perceived by the Scanners and computed by the Data Collection and Correlation System as part of the new topology and reachability

snapshot of the Monitored System(s). Since the topology/vulnerabilities data collection process is not real time, at least in the first prototype of the System, but it happens on scheduled timings dependent on the implemented scanners and the performance capabilities of the DCI, the risk related of **Control Loop 2** is that there can be some delay from the moment a *Response_Plan* is deployed on the Monitored System and the moment in which it is perceived by the PANOPTESEC System.

High-Level Online Correlator

In Figure 30 a Logic View of the High-Level Online Correlator (HOC) component is presented.

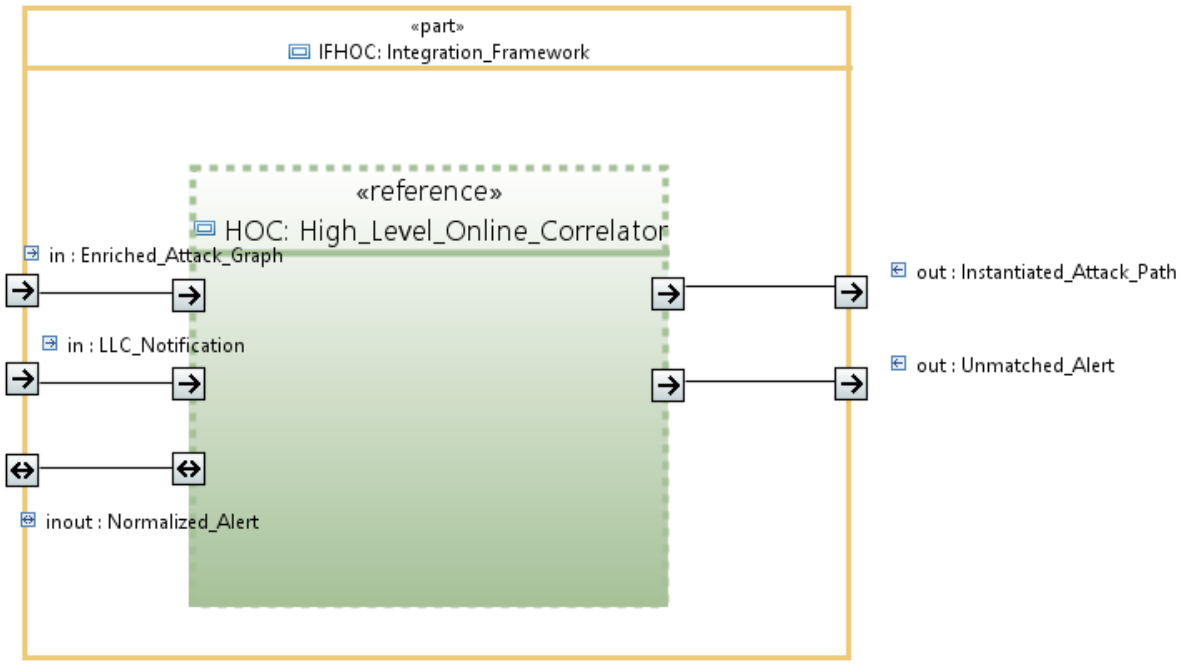


Figure 30: High Level Online Correlator Logic View

Figure 30 shows HOC as a component inside an Integration Framework wrapper that manages the external connections and the runtime of the component.

The aim of High-level Online Correlation component (HOC) is to correlate information about events generated by sensors and IDSs in the Monitored System(s) with vulnerabilities of the system itself to provide an accurate estimation of the possible on-going attacks.

HOC evaluates two major inputs: the *Enriched_Attack_Graph* from the AGG component and a stream of Low Level *Normalized_Alert*(s) from the LLC component (carried out by *LLC_Notification* messages). The *Enriched_Attack_Graph* provide static information about complex attack scenarios and identify all possible known *Attack_Path*(s), along with an evaluation of their Risk level carried out by the Risk Quantifier component. The Low Level *Normalized_Alert*(s) provide real-time information about the current status of the system.

Each Low-Level *Normalized_Alert* corresponds to an observation, while each node of the *Enriched_Attack_Graph* represents an elementary attack action. Therefore a first sub-

component of the HOC is responsible for transforming the *Enriched_Attack_Graph* representation into a set of correlation rules. The resulting correlation rules identify, on one side, all the possible targets for each elementary action of an attacker and, on the other side, the sources and the format of all the matching observations. At runtime, these rules are used by the second sub-component of the HOC component that correlates *Normalized_Alert(s)* in order to discover hidden or implicit temporal and spatial relationships defining a possible attack pattern and tries to match such pattern on a known *Attack_Path* (as stated in **RRS025**). This possible matching then defines the main output of the module, named *Instantiated_Attack_Path* (stated in **RRS026**).

The High Level Online Correlator component will also able to tolerate IDS false positive (a legitimate event is considered as an attack) and false negative (don't detect on-going attacks) when correlating alerts, as stated in **RRS027**.

In order to fulfil **PRT005**, a set of fixed interfaces for the HOC component is then defined.

IN Interface ValueTypes with the Attack Graph Generator component
Enriched_Attack_Graph

IN Interface ValueTypes with the Data Collection Collector component
Normalized_Alert(s)

IN Interface ValueTypes with the Low Level Correlator component
LLC_Notification

OUT Interface ValueTypes with the Data Collection Collector component
Normalized_Alert(s)

OUT Interface ValueTypes with the Potential Attack Identifier component
Instantiated_Attack_Path
Unmatched_Alert

OUT Interface ValueTypes with the Risk Quantifier component
Instantiated_Attack_Path

OUT Interface ValueTypes with the Internal Data Interface component
Instantiated_Attack_Path

OUT Interface ValueTypes with the Visualization System
Instantiated_Attack_Path

Interactions between HOC and other components of the System are shown in Sequence Diagrams in Figure 33 and Figure 40.

More details about HOC can be found in [D5.1.2].

Potential Attack Identifier

In Figure 31 a Logic View of the Potential Attack Identifier (PAI) component is presented.

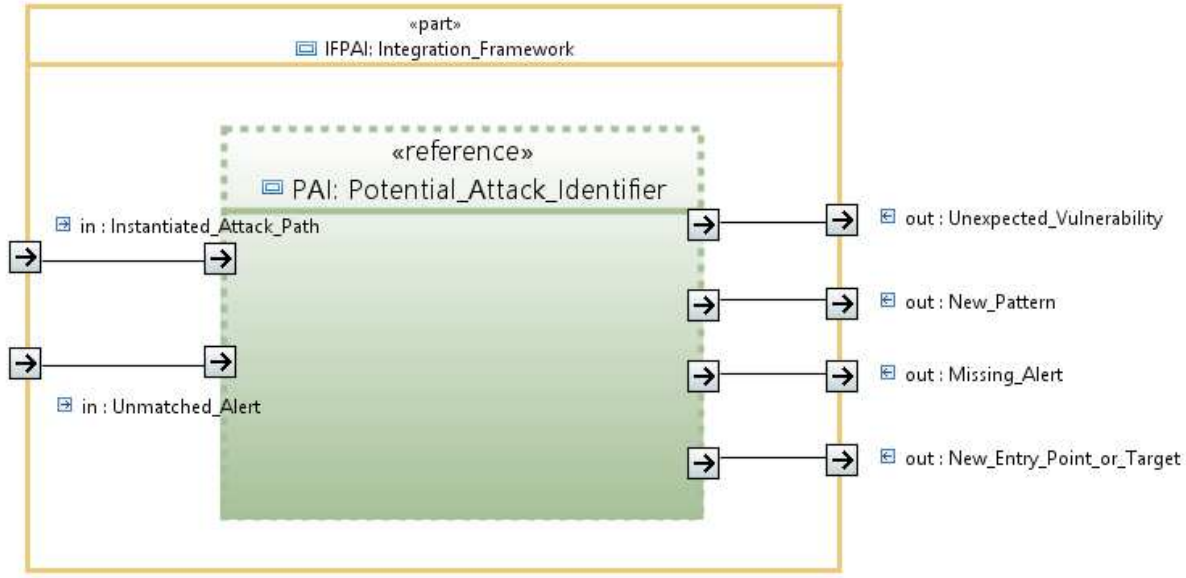


Figure 31: Potential Attack Identifier Logic View

Figure 31 shows PAI as a component inside an Integration Framework wrapper that manages the external connections and the runtime of the component.

The Potential Attacks Identifier component (PAI) extracts useful information (from the HOC processes) that may help to identify new attack scenarios or to update the set of already known *Attack_Path(s)* (after a complementary analysis performed by the AGG component). The aim of the component is to fulfil **RRS023**.

The PAI component receives information only from the HOC component. More precisely, during the HOC correlation process, the incoming flow of Low-Level *Normalized_Alert*(s) is automatically subdivided into two subsets. Some alerts do not help the recognition of an on-going attack scenario (e.g. alerts not belonging to an *Attack_Path* or alerts included in an *Attack_Path* but not included in a recent sequence) and they are named *Unmatched_Alert(s)*. Otherwise, an alert corresponds to an expected elementary step in at least one of the identified *Attack_Path*(s): it is a "Matched Alert" (that leads to the processing of the *Instantiated_Attack_Path*(s)). Based on this dichotomy the PAI component performs additional investigations.

The PAI component is then informed of all *Unmatched_Alert(s)*. Among the *Unmatched_Alert(s)*, the PAI can also detect the existence of frequent patterns (e.g. sequences of a few *Unmatched_Alert(s)*). The PAI component is also responsible for

identifying the *Unmatched_Alert(s)* that are potentially preceded by a *Missing_Alert*. This detection can help to discover problems that affect the deployed monitoring devices.

Instantiated_Attack_Path(s)) are another input for the PAI component. The HOC component must then report to the PAI function any *Instantiated_Attack_Path*(s)) that allows progress within the detection of an on-going attack but does not correspond to a well-identified *Vulnerability*. This information is checked and used to update a knowledge base (new *Vulnerability* or new characteristic of a monitoring device). On another hand, each time a complete *Attack_Path* is detected, the involved targets should also be transmitted to the PAI function to enrich the knowledge base with identified *New_Entry_Point_or_Target*. The information of the knowledge base may be provided back to the Attack Graph Generator component to update the *Attack_Graphs* accordingly.

All other information provided by PAI, at the moment, are directly persisted in the Data Collection Collector component (these inputs for DCC are not shown in the diagrams, due to the fact that the component is not mandatory in the reactive chain of the PANOPTESEC System and these ValueTypes, at the moment, are of no use). Further researches will state if the PANOPTESEC System will be able to directly benefit from all PAI outputs.

In order to fulfil **PRT005**, a set of fixed interfaces for the PAI component is then defined.

IN Interface ValueTypes with the High Level Online Correlator component
Instantiated_Attack_Path
Unmatched_Alert

OUT Interface ValueTypes with the Attack Graph Generator component
New_Entry_Point_or_Target

OUT Interface ValueTypes with the Internal Data Interface component
Unexpected_Vulnerability
New_Pattern
New_Entry_Point_or_Target
Missing_Alert
(These interfaces are not depicted in Sequence Diagrams, due to the relative importance – with respect to the ASRs- of the considered data, which will simply be saved in the persistency for possible future uses).

Interactions between PAI and other components of the System are shown in Sequence Diagram in Figure 33.

More details about PAI can be found in [D5.1.2].

Tactical Response Decider
In Figure 32 a Logic View of the Tactical Response Decider (TRD) component is presented.

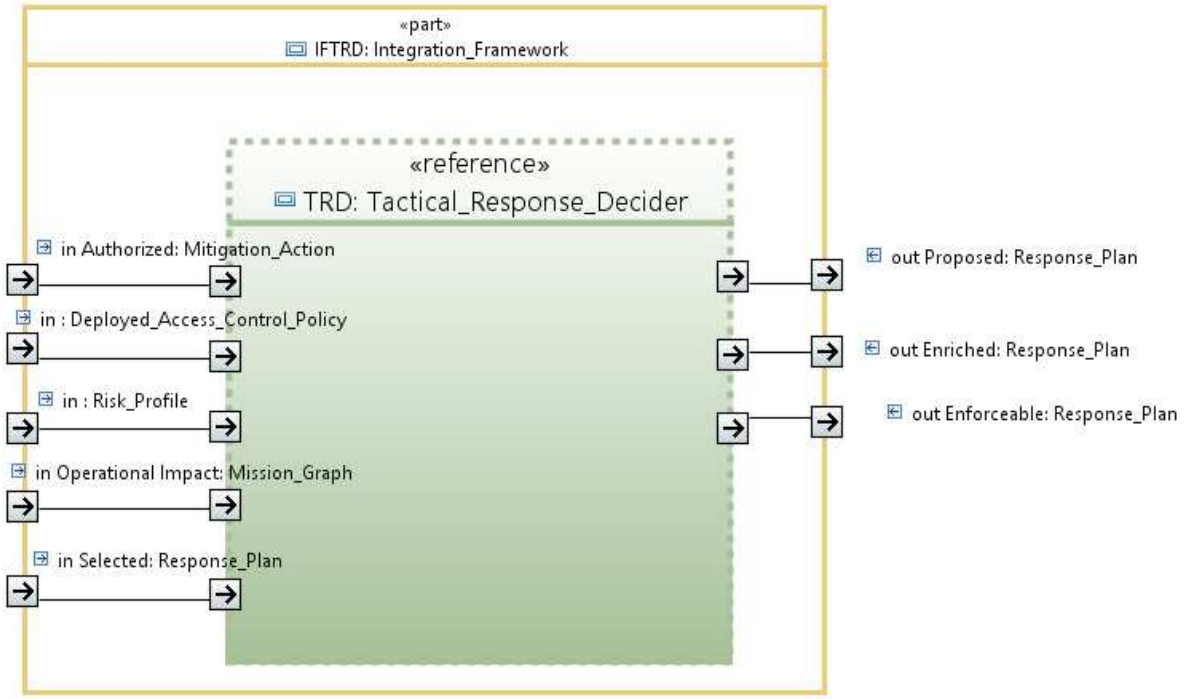


Figure 32: Tactical Response Decider Logic View

Figure 32 shows TRD as a component inside an Integration Framework wrapper that manages the external connections and the runtime of the component.

While the Strategic Response Decider component envisages identified risks mitigation on a long-term proactive perspective, detection of on-going attacks' progressions towards critical Assets of the organization may increase the value of some identified risks beyond an acceptable level. Moreover, detection of new attackers may raise new risks, if the detected entry point of this new attacker is not a node of an already computed and monitored *Attack_Graph*. In these cases, a reprioritization of the risks that should be addressed needs to be performed and *Mitigation_Action*(s) have to be taken with this new reactive perspective (stated in **RRS009**). This is the main function of the Tactical Response Decider component, which obviously has to coordinate its decided Response Plans with the proactive Response Plans established by the Strategic Response Decider component. The Proposed *Response_Plan*(s) must be evaluated and enriched in terms of *Operational_Impact* by the Operational Impact Assessment Evaluator (as stated in **RRS010**).

It is mandatory that the Proposed *Mitigation_Action*(s) are selected from a list of Authorized *Mitigation_Action*(s), loaded from the DCC component in the Data Collection and Correlation System (**RRS011**). TRD is then responsible for deciding which of the risks quantified with a reactive perspective (i.e. reactive risks within the received *Risk_Profile* from the Risk Quantifier component) have to be addressed by the Response System. It also has to issue the decision of the *Mitigation_Action(s)* chosen to mitigate the reactive risks in the most efficient manner.

The enforcement of the Response Plans selected by the Tactical Response Decider (after the enrichment by the Operational Impact Assessment Evaluator component) may depend on human approval (with an interaction similar to the Strategic one) or can be automatic. Due to **RRS013** and **RRS014**, in the actual implementation of the PANOPTESEC System these reactive *Response_Plan*(s) must be selected and activate by a Security Operator via the Visualization System. After the selection, TRD will implement the (Selected) *Response Plan*(s) by sending them to the Policy Deployer component (**RRS015**).

In order to fulfil **PRT005**, a set of fixed interfaces for the TRD component is then defined.

IN Interface ValueTypes with the Data Collection Collector component
Deployed_Access_Control_Policy
(Authorized) *Mitigation_Action*(s)

IN Interface ValueTypes with the Risk Quantifier component
Risk_Profile

In Interface ValueTypes with the Operational Impact Assessment Evaluator component
(Operational_Impact)Mission_Graph

OUT Interface ValueTypes with the Operational Impact Assessment Evaluator component
(Proposed) Response_Plan

OUT Interface ValueTypes with the Visualization System
(Enriched) Response_Plan

OUT Interface ValueTypes with the Policy Deployer component
(Enforceable) Response_Plan

OUT Interface ValueTypes with the Strategic Response Decider component
(Enforceable) Response_Plan

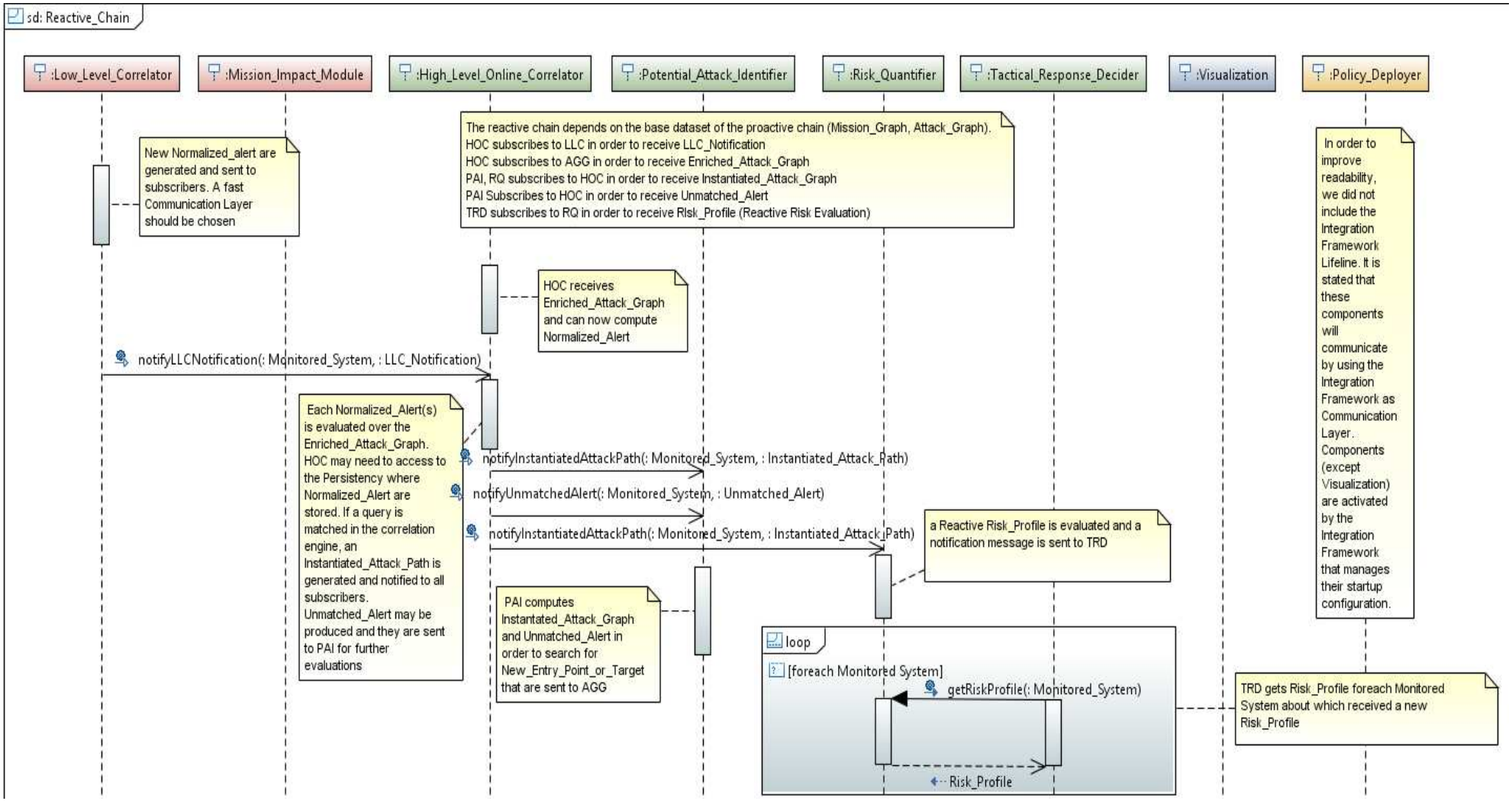
(This interface is not depicted in any Sequence Diagram: when (Enforceable) Response_Plan(s) are pushed to the Policy Deployer, TRD also pushes them also to SRD. This is due to the fact that SRD can benefit of the knowledge of the *Response_Plan* enforced by TRD, for a better understanding of the Monitored System(s). SRD will not stop any internal process in order to compute new *Response_Plan*(s) from TRD, in order to avoid racing conditions).

Interactions between TRD and other components of the System are shown in Sequence Diagram in Figure 28 and Figure 33.

More details about TRD can be found in [D5.1.2].

Reactive Chain Sequence Diagram

In Figure 33 a Process View (Sequence Diagram) of the reactive chain is presented. Due to its complexity, it is divided in two pages. It can be directly compared to Figure 20 in [D3.1.1].


sd: Reactive_Chain
:Low_Level_Correlator
:Mission_Impact_Module
:High_Level_Online_Correlator
:Potential_Attack_Identifier
:Risk_Quantifier
:Tactical_Response_Decider
:Visualization
:Policy_Deployer
New Normalized_alert are generated and sent to subscribers. A fast Communication Layer should be chosen
The reactive chain depends on the base dataset of the proactive chain (Mission_Graph, Attack_Graph).
HOC subscribes to LLC in order to receive LLC_Notification
HOC subscribes to AGG in order to receive Enriched_Attack_Graph
PAI, RQ subscribes to HOC in order to receive Instantiated_Attack_Graph
PAI Subscribes to HOC in order to receive Unmatched_Alert
TRD subscribes to RQ in order to receive Risk_Profile (Reactive Risk Evaluation)
In order to improve readability, we did not include the Integration Framework Lifeline. It is stated that these components will communicate by using the Integration Framework as Communication Layer. Components (except Visualization) are activated by the Integration Framework that manages their startup configuration.
HOC receives Enriched_Attack_Graph and can now compute Normalized_Alert
notifyLLCNotification(: Monitored_System, : LLC_Notification)
Each Normalized_Alert(s) is evaluated over the Enriched_Attack_Graph. HOC may need to access to the Persistency where Normalized_Alert are stored. If a query is matched in the correlation engine, an Instantiated_Attack_Path is generated and notified to all subscribers. Unmatched_Alert may be produced and they are sent to PAI for further evaluations
notifyInstantiatedAttackPath(: Monitored_System, : Instantiated_Attack_Path)
notifyUnmatchedAlert(: Monitored_System, : Unmatched_Alert)
notifyInstantiatedAttackPath(: Monitored_System, : Instantiated_Attack_Path)
a Reactive Risk_Profile is evaluated and a notification message is sent to TRD
PAI computes Instantated_Attack_Graph and Unmatched_Alert in order to search for New_Entry_Point_or_Target that are sent to AGG
loop
[foreach Monitored System]
getRiskProfile(: Monitored_System)
Risk_Profile
TRD gets Risk_Profile foreach Monitored System about which received a new Risk_Profile

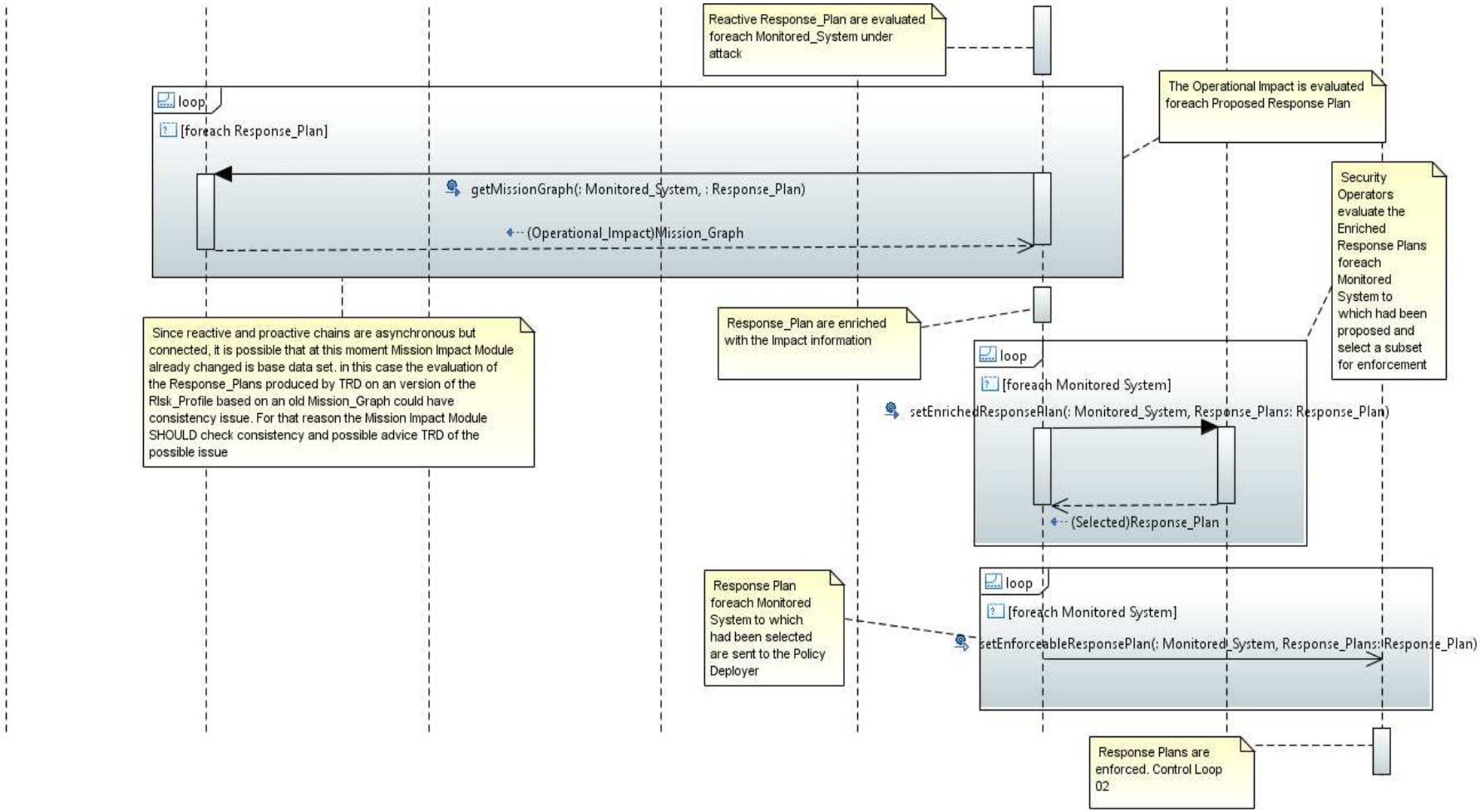


Figure 33: Reactive Response Chain Process View

In order to reduce the size of the diagrams for documentation reasons, Integration Framework lifelines are not shown in this Diagram. As can be seen in the previous Logic Diagrams, the Integration Framework will be directly managing all communication between the components. More details in Section 5.2.5.5.

Processes depicted in Figure 33 cover most of the possible interactions between PANOPTESEC System components involved in the reactive chain. In [D5.1.2] a more detailed view of the interactions between each component of the Dynamic Risk Management Response System and the others is given. In the Sequence Diagram in Figure 33 the reactive interaction is shown in order to give a complete idea of how the PANOPTESEC System will work on a reactive perspective.

The reactive chain depends on the base dataset of the proactive one (*Mission_Graph*, *Attack_Graph*). This means that the reactive chain cannot compute if at least one iteration of the proactive one has been computed.
At System start-up, components related to the reactive chain subscribe to their data producers:

- HOC subscribes to LLC in order to receive *LLC_Notification*.
- HOC subscribes to AGG in order to receive *Enriched_Attack_Graph.*
- PAI, RQ subscribe to HOC in order to receive *Instantiated_Attack_Graph.*
- PAI Subscribes to HOC in order to receive *Unmatched_Alert.*
- TRD subscribes to RQ in order to receive *RIsk_Profile* (Reactive Risk Evaluation).

As described in Figure 18, LLC correlates *Normalized_Alert*(s) after the raw data normalization phase. LLC sends *LLC_Notification*(s) to subscribers (HOC, for example). While receiving *LLC_Notification(s),* HOC tries to match them with the *Attack_Path*(s). This possible matching then defines the main output of the module, named *Instantiated_Attack_Path* (stated in **RRS026**). HOC pushes *Instantiated_Attack_Path* to subscribers, as well as *Unmatched_Alert*(s) to PAI.

PAI computes these inputs in order to find *New_Entry_Point_or_Target(s)* and send them to AGG (in push mode or in pull after a notification). AGG will compute *New_Entry_Point_or_Target* if no other processes are on-going (by acting in this way, there is no risk of racing conditions between the components): **RRS023** (which is linked to PAI processes) is not a mandatory Requirement.

In RQ, the Reactive *Risk_Profile* is evaluated for a given set of *Instantiated_Attack_Path*(s). Once computed, a notification message is sent to TRQ that pulls the *Risk_Profile* from RQ.

Due to the components and the processes involved in the reactive chain, it can be stated that **RRS016** is fulfilled.

TRD evaluates a set of possible *Response_Plan*(s) in order to lower the risk level of the current reactive situation. Once this evaluation ends, TRD asks MIM the *Operational_Impact* evaluation on the (Proposed)*Response_Plan*.

Once this evaluation are concluded, TRD enriches the *Response_Plan(*s) with the impact evaluations and sends the (Enriched)*Response_Plan*(s) to the Visualization System for the evaluation by the Security Operator. The Security Operator evaluates the (Enriched)*Response_Plan*(s) and selects some of them that the Visualization System pushes back to TRD (TRD needs to know which *Response_Plan* will be enforced in order to better compute the next reactive evaluations). TRD is now able to send the selected *Response_Plan* to the Policy Deployer component asking for the actual deploy in the Monitored System(s).

This approach to the reactive chain takes into account **PRF004** more than **RBL010**. The reactive chain risks to work with alert data referring to a Monitored System situation that is not depicted in the stateful set of data managed by the proactive chain (*Reachability_Matrix*, *Mission_Graph*, *Vulnerability_Inventory*, *Scored_Vulnerability_Inventory*). This is a deliberate choice in order to increase performances for the sake of risking some inconsistent results.

For example, inconsistent results may arise mainly due to changes in the Monitored System that are not perceived in the proactive chain. Since the reactive chain is designed in order to be as fast as possible, it can happen that some *Normalized_Alert*(s) processed by HOC would refer to a Monitored System situation that can't be entirely reflected in the current *Enriched_Attack_Graph* (because it is newer). It is possible to envision three different cases:

1. *Normalized_Alert*(s) received by the HOC still matches an *Attack_Path,* but the *Attack_Path* that HOC is considering is no more the same (i.e. in the new Monitored System situation it is shorter or longer);
2. *Normalized_Alert*(s) received by the HOC does not find any match in the current set of *Attack_Path(s)* (but a match would have been found with an updated *Enriched_Attack_Graph)*;
3. *Normalized_Alert*(s) received by the HOC matches one of the existing *Attack_Path* that would not exists anymore if the *Enriched_Attack_Graph* would have been updated in real-time.

Case 1 does not create issues: an *Instantiated_Attack_Path* will be eventually triggered.

In order to mitigate issues caused by Case 2, the PAI module, responsible for collecting all the *Unmatched_Alert(s)* and manage them.

In order to mitigate issues caused by Case 3, HOC will define internal procedures in order to manage new *Enriched_Attack_Graph* following a view-based approach (without risks of racing conditions, since *Enriched_Attack_Graph* come from the proactive chain).

A second Control Loop (**Control Loop 02**), evaluated in the Preliminary High Level Design, is identified again between the reactive chain and the Monitored System(s). When the reactive chain commands the enforcement of a *Response_Plan*, this process will affect the Monitored System(s) and following the PANOPTESEC System architecture, it will be reflected in future PANOPTESEC System computations.

Even if **Control Loop 02** is similar from proactive to reactive situation, it will be important to evaluate the different impacts due to the fact that reactive *Response_Plan*(s) will likely be deployed far faster than proactive ones. This is due to the fact that the reactive chain will try to optimize the deployment time by using simple sets of *Mitigation_Action*, faster than the set proposed by the proactive chain (e.g., not including a patch upgrade *Mitigation_Action*, but only updates on Ports one particular Devices). In this respect, the feedback on the Monitored System will be faster and DCI will be able to collect network information with these *Mitigation_Action* already deployed faster than for the proactive chain.

Policy Deployer

In Figure 34 a Logic View of the Policy Deployer (PDP) component is presented.

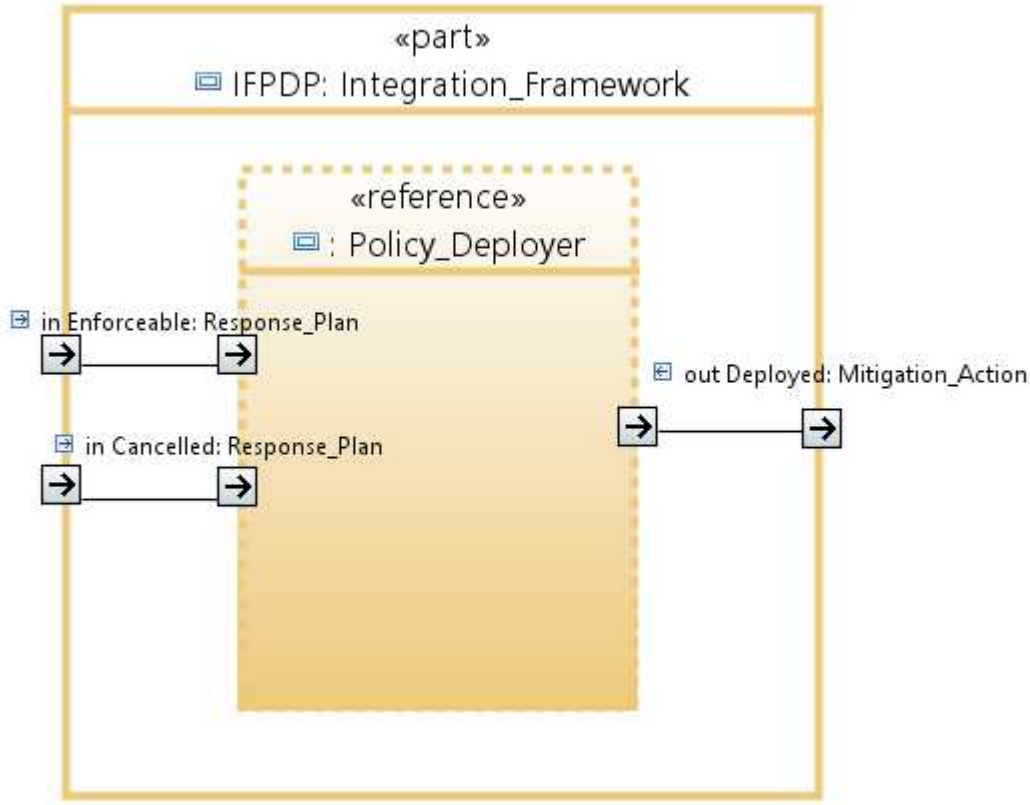


Figure 34: Policy Deployer Logic View

Figure 34 shows PDP as a component inside an Integration Framework wrapper that manages the external connections and the runtime of the component.
PDP is not part of the Dynamic Risk Management Response System but it is the final component of the proactive and the reactive chains.
PDP is responsible to fulfil **PRS022** and **RRS024**.

The Policy Deployer component (PDP) is responsible for the enforcement of Enforceable *Response_Plan*(s) (from the Strategic Response Decider or from the Tactical Response Decider).

The Policy Deployer will receive *Response_Plan*(s) in XOrBAC format (an XML schema based on OrBAC that structures its data storage and communication. More information about XOrBAC are in [D5.1.2]).

During the enforcement of concrete security rules, the Policy Deployer component should look up for relevant elements in the XOrBAC file passed as an input. Then, it should process the information contained on those elements to be able to proceed to the translation of these rules into Policy Enforcement Point configurations commands for the network components on the Monitored System(s) on which the rules shall be enforced.

The Policy Deployer component, if requested by the Visualization System, will attempt to stop any *Response_Plan*(s) enforcement process, if possible, as requested by **VIZ026**. The enforcement process, however, can be impossible to be stopped (for example, the reconfiguration of a firewall policy). The Policy Deployer component will then need a direct network connection with the Monitored System(s) and may rely on remote agents installed on *Devices*. After the policy enforcement process the Policy Deployer component will store the Deployed *Mitigation_Actions* in the Data Collection and Correlation persistency module.

In Figure 29 and in Figure 33 the role of the Policy Deployer inside the proactive/reactive chain is shown.

In order to fulfil **PRT005**, a set of fixed interfaces for the PDP component is then defined.

IN Interface ValueTypes with the Strategic Response Decider component
Enforceable Response_Plan(s)

IN Interface ValueTypes with the Tactical Response Decider component
Enforceable Response_Plan(s)

IN Interface ValueTypes with the Visualization System
(Cancelled) Response_Plan
(This interface is not shown in any Sequence Diagram. Visualization System –due to **VIZ026**- may ask the Policy Deployer to stop the enforcement of specific *Response_Plan*. This request is directly pushed to the Policy Deployer).

OUT Interface ValueTypes with the Internal Data Interface component
(Deployed) Mitigation_Action(s)
(This interface is not shown in any Sequence Diagram. The Policy Deployer may *push* information regarding all deployed *Mitigation_Action*(s) to the persistency in DCC, for further uses).

Interactions between PDP and other components of the System are shown in Sequence Diagram in Figure 29and Figure 33.

5.2.5.4 Visualization

In Figure 35 a High Level Logic View of the PANOPTESEC System with focus on the Visualization System is presented.

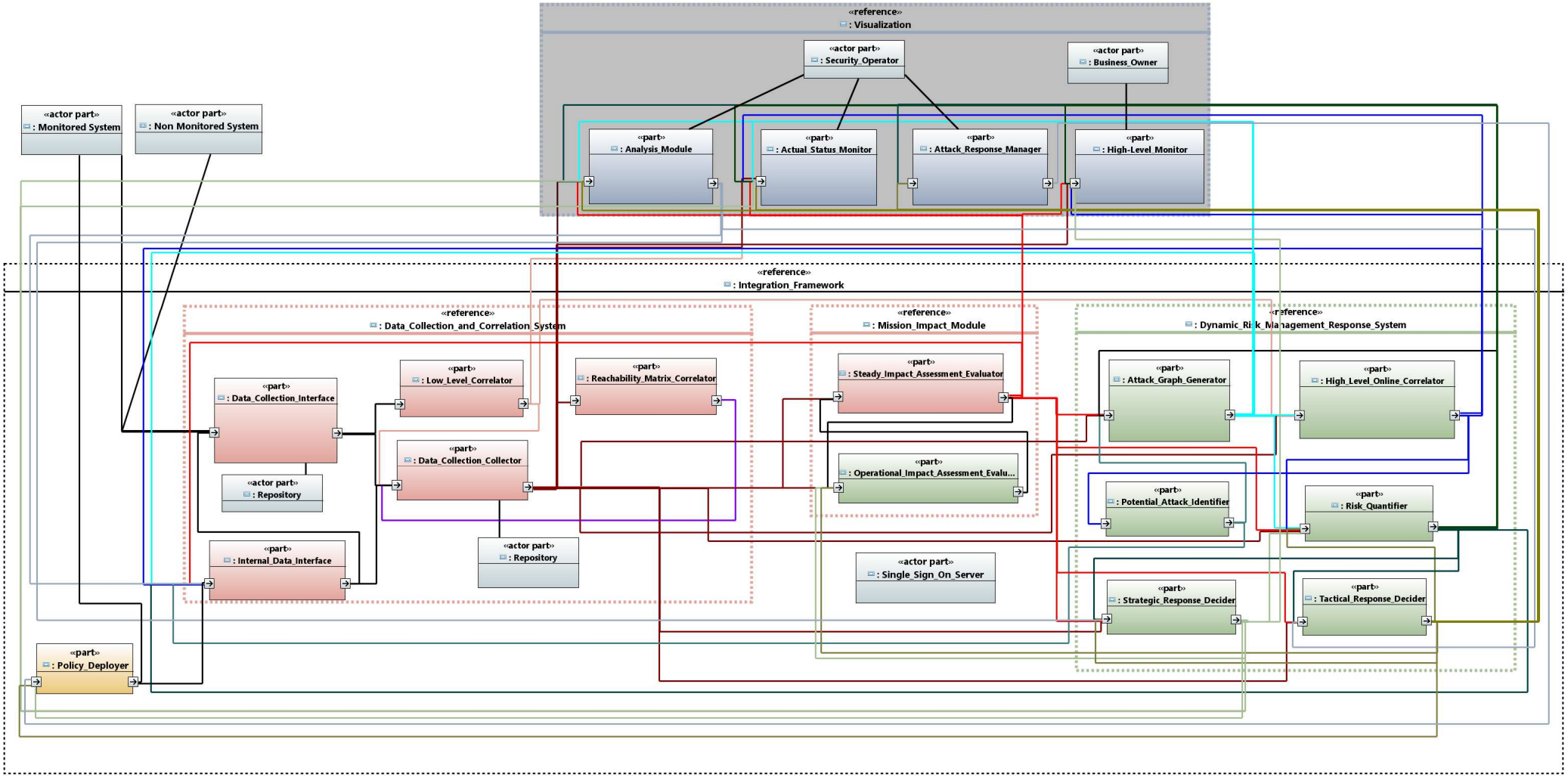


Figure 35: Visualization System Logic View

The Visualization System is a visual analytics environment that analyses current and historical cyber-security situation of a Monitored System protected by a PANOPTESEC System. The Visualization System also shows the mitigation actions proposed by the proactive and reactive chains together with the matching between the actual network state and the closest attack model. The Visualization System also supports the Security Operator in analysing the proposed mitigation actions, confirming or refusing such actions.

USG001 states that various version of the user interface must be provided in order to address different needs for different Stakeholders (for example, Security Operators and Business Owners). Due to that Requirement and due to the provided expertise of WP6 owners (UROME) in Web-based visual analytics, the Visualization System will be based on Web technologies that should also fulfil **USG003**. The Visualization System website will be hosted on an Application Server. Due to that, the Integration Framework will not directly integrate the internal components and views of the Visualization System, as can be seen in Figure 35.

Since every other component of the PANOPTESEC System will be integrated and managed by the Integration Framework, communications between PANOPTESEC System components and the Visualization System will be directly managed by the Integration Framework (it is possible to envision a *Visualization System Proxy* managed by the Integration Framework with the task of concentrate and re-route messages to/from the Visualization System, using REST APIs and JSON format, as stated in **CMP009**). Since the Visualization System will rely on industrial Application Server (for example, Apache Web Server), it will be possible to easily guarantee **RBL004** and **RBL008** for the Visualization System.

Visualization System is composed by four different views:

- Analysis Module
- Actual Status Monitor
- Attack Response Manager
- High-Level Monitor (this view is intended for executive activities –or Business Owner Stakeholders- and provides high-level information for long term/strategic decisions and it is based on a subset of the data set of the Analysis Module –for this reason, it is not analysed in detail in this deliverable. More information can be found in [D6.1.2])

More information about the Visualization System can be found in [D6.1.2].

Analysis Module

In Figure 36 a Logic View of the Analysis Module (ANM) view is presented.

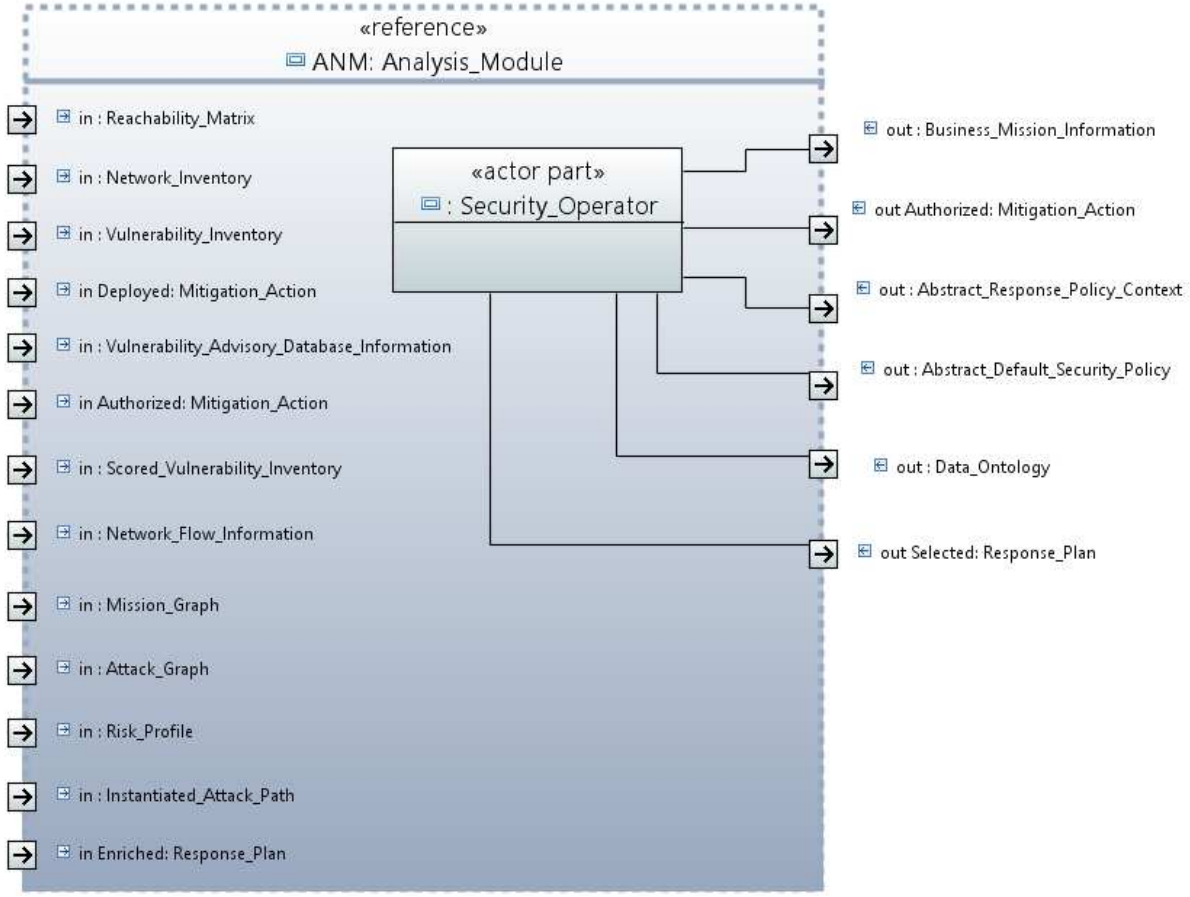


Figure 36: Analysis Module Logic View

The Analysis Module (ANM) depicted in the Logic View in Figure 36 is responsible for analysing the actual cyber security characteristics of the Monitored System(s) along with their network characteristics: network topology (at least, at ISO/OSI Layer 3) and reachability (**VIZ007** and **VIZ008.** The *Reachability_Matrix* and the *Network_Inventory* carries also ISO/OSI Layer 2 information, partially fulfilling **VIZO029**), vulnerabilities on *Device*(s) (**VIZ007, VIZ009, VIZ021** and **VIZ022**), critical assets depicted with *Mission_Graph* (**VIZ006**), current processed *Attack_Graph* (**VIZ010**). ANM has to provide users with effective visualization and interaction in order to get useful insights about all provided inputs.

ANM is also responsible for enabling historical data analysis for all the provided inputs, as stated in **VIZ023, VIZ024** and **VIZ025** (due to the current architecture of the Data Collection and Correlation System, a persistency manager for all PANOPTESEC System computed data is set. Data related to the proactive chain will be persisted with a timestamp identifying the related snapshot of the Monitored System(s). Data computed by the reactive chain, for example a set of *Instantiated_Attack_Path*(s), will be related to the corresponding proactive situation).

ANM is responsible for showing the Security Operator the list of already (Deployed)*Mitigation_Action*(s). This information is collected in *pull* mode from the DCC.

ANM component is also responsible for assisting the Security Operator on the evaluation of (Enriched) *Response_Plan*(s) after the proactive/reactive chain processed by the Dynamic Risk Management Response System, as stated in **VIZ012** and **VIZ014**. When a new set of *Response_Plan*(s) are proposed to the Security Operator, ANM must be able to show all the desired information: actual situation awareness, actual *Response_Plan*(s) *Operational_Impact*, actual *Risk_Profile*. ANM will allow the Security Operator select a set of (Enriched) *Response_Plan*(s) (as stated in **VIZ013** and **VIZ015**). The Selected *Response_Plan*(s) will be then activated in order to return to the Strategic/Tactical Response Decider for enforcement (**VIZ016**).

Due to **DSC025** and **DSC027,** Data related to *Business_Mission_Information* should be collected by the Visualization System (this is also stated in **VIZ018** and **VIZ019**) along with the list of *Authorized Mitigation_Action* (these data can be input in the System by loading an external file containing the set of *Authorized Mitigation_Action*).

In order to fulfil **PRT005**, a set of fixed interfaces for ANM is then defined. All communications with other components of the PANOPTESEC System will probably be concentrated and re-routed by a Visualization System Proxy managed by the Integration Framework.

IN Interface ValueTypes with the Data Collection Collector component
Reachability_Matrix
Network_Inventory
Vulnerability_Inventory
Scored_Vulnerability_Inventory
(Deployed) Mitigation_Action(s)
Vulnerability_Advisory_Database_Information
(Authorized) *Mitigation_Action*(s)
Network_Flow_Information (historical records)
Historical information regarding *Mission_Graph*(s), *Attack_Graph*(s), *Risk_Profile*(s) and *Instantiated_Attack_Path*(s)

IN Interface ValueTypes with the Mission Impact Module
Mission_Graph (last version)

IN Interface ValueTypes with the Attack Graph Generator component
Attack_Graph (last version)

IN Interface ValueTypes with the Risk Quantifier component
Risk_Profile (last version)

IN Interface ValueTypes with the Strategic Response Decider component

(Enriched) Response_Plan

IN Interface ValueTypes with the Tactical Response Decider component
(Enriched) Response_Plan

OUT Interface ValueTypes with the Internal Data Interface component
Data_Ontology
Abstract_Response_Policy_Context
(Authorized) Mitigation_Action(s)
Abstract_Default_Security_Policy
Business_Mission_Information

OUT Interface ValueTypes with the Strategic Response Decider component
(Selected) *Response_Plan*

OUT Interface ValueTypes with the Tactical Response Decider component
(Selected) *Response_Plan*

More details about AMD can be found in [D6.1.2]

Actual Status Monitor
In Figure 36 a Logic View of the Actual Status Monitor (MON) view is presented.

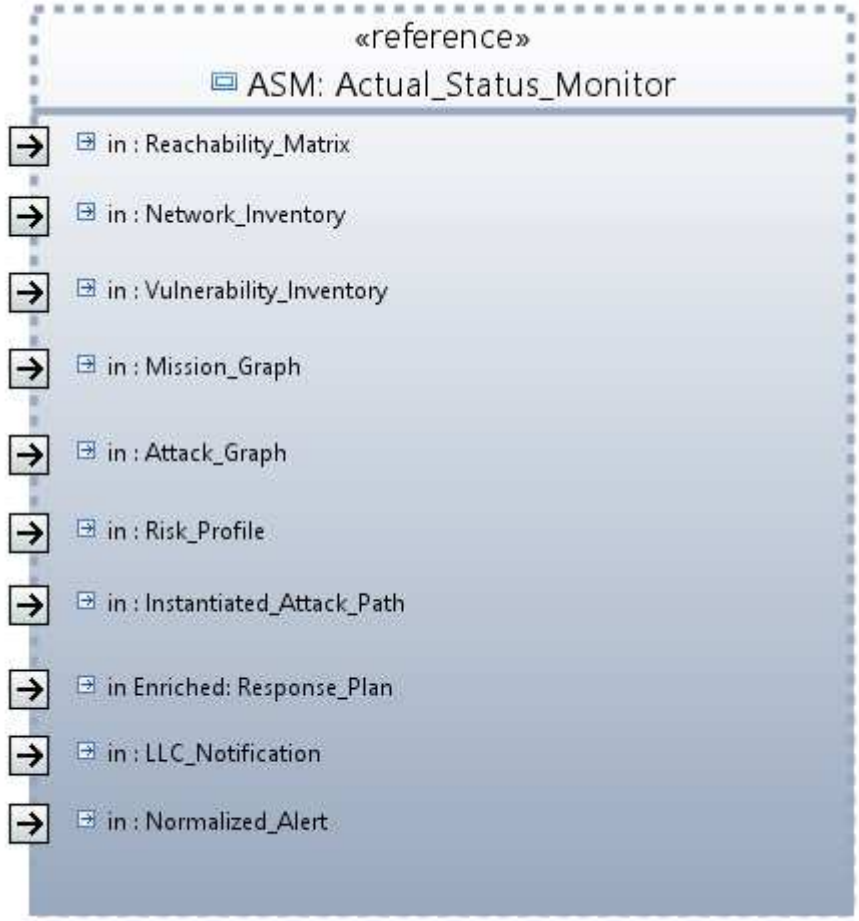


Figure 37: Actual Status Monitor Logic View

The Actual Status Monitor is responsible for visualizing the actual network and cyber security situation of the Monitored System(s). Therefore it has to provide users with effective visualizations and interaction in order to get useful insight about:

- Actual proactive Risk Modelling assessments (current computed *Risk_Profile* -**VIZ002**- and related *Attack_Graph* and *Mission_Graph* -**VIZ003**).
- Actual reactive Risk Modelling assessments in case of on-going attack (current computed *Risk_Profile* -**VIZ004**- and related *Instantiated_Attack_Path(s)* and *Mission_Graph* -**VIZ005**). MON will also show (Enriched) *Response_Plan*(s) from the reactive perspective (while the selection of these *Response_Plan*(s) will be carried out within the ANM).
- Incoming *Normalized_Alert* applied to reactive Attack Modelling situation assessments, as stated in **VIZ011**. This Requirement is difficult to be completely fulfilled. At this moment of the analysis, although the raw alert data collection process by Data Collection and Correlation System collects data every relatively short period (seconds), the Reactive Chain is not strictly *real time*.

In order to fulfil **PRT005**, a set of fixed interfaces for ANM is then defined. All communications with other components of the PANOPTESEC System will probably be concentrated and re-routed by a Visualization System Proxy managed by the Integration Framework.

IN Interface ValueTypes with the Data Collection Collector component
Reachability_Matrix
Network_Inventory
Vulnerability_Inventory
Normalized_Alert(s)

IN Interface ValueTypes with the Mission Impact Module
Mission_Graph (last version)

IN Interface ValueTypes with the Attack Graph Generator component
Attack_Graph (last version)

IN Interface ValueTypes with the Risk Quantifier component
Risk_Profile (last version)

IN Interface ValueTypes with the High Level Online Correlator component
Instantiated_Attack_Path(s) (on-going attacks)

IN Interface ValueTypes with the Low Level Correlator component
LLC_Notification(s)

IN Interface ValueTypes with the Tactical Response Decider component
(Enriched) Response_Plan

More details about MON can be found in [D6.1.2]

Attack Response Manager
In Figure 38 a Logic View of the Attack Response Manager (ARM) view is presented.

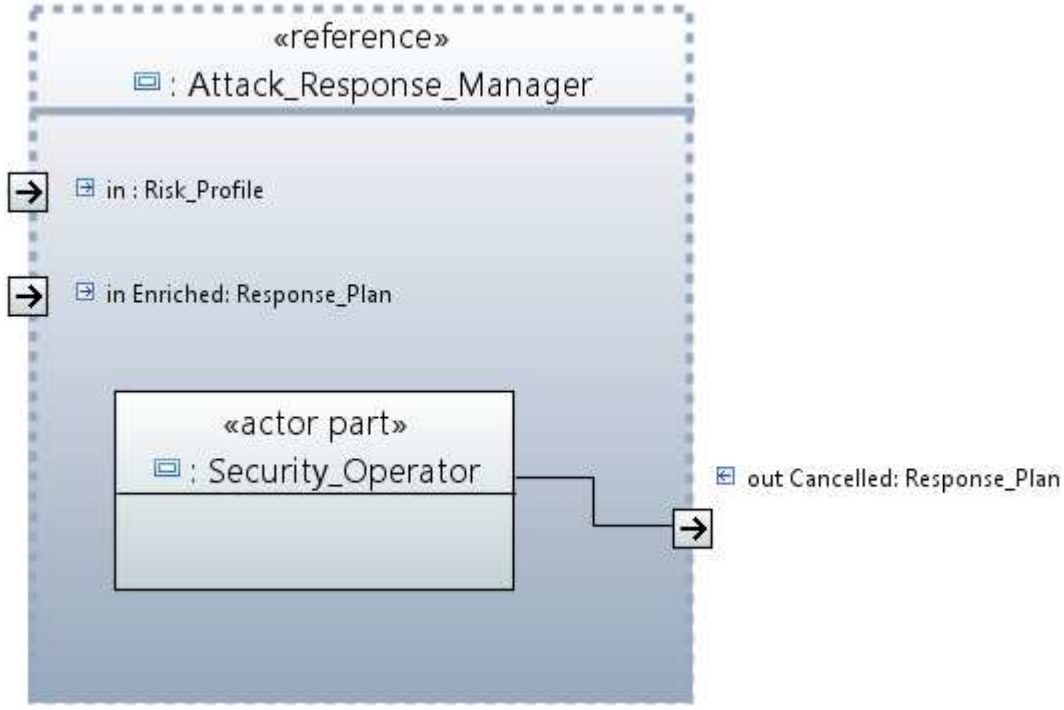


Figure 38: Attack Response Manager Logic View

The Attack Response Manager (ARM) component is responsible for monitoring consequences related to an on-going attack (as stated in **VIZ027**). The Security Operator will have the possibility to ask the Policy Deployer component for trying to stop a selected Response Plan from the enforcement process (this cancelling request may not be able to process, since the policy enforcement process, however, can be impossible to be stopped - for example, the reconfiguration of a firewall policy), as stated in **VIZ026**. ARM will rely on the same set of data of the MON view.

In order to fulfil **PRT005**, a set of fixed interfaces for ANM is then defined. All communications with other components of the PANOPTESEC System will probably be concentrated and re-routed by a Visualization System Proxy managed by the Integration Framework.

IN Interface ValueTypes with the Risk Quantifier component
Risk_Profile

IN Interface ValueTypes with the Tactical Response Decider component
(Enriched) Response_Plan

OUT Interface ValueTypes with the Policy Deployer component
(Cancelled) Response Plan
More details about ARM can be found in [D6.1.2]

In order to receive the needed information to display to the Security Operators and to the Business Owners, the Visualization System needs to interact with most of the components of the PANOPTESEC System.

The Visualization System has to show information about the actual proactive (Figure 39) and reactive (Figure 40) situation:

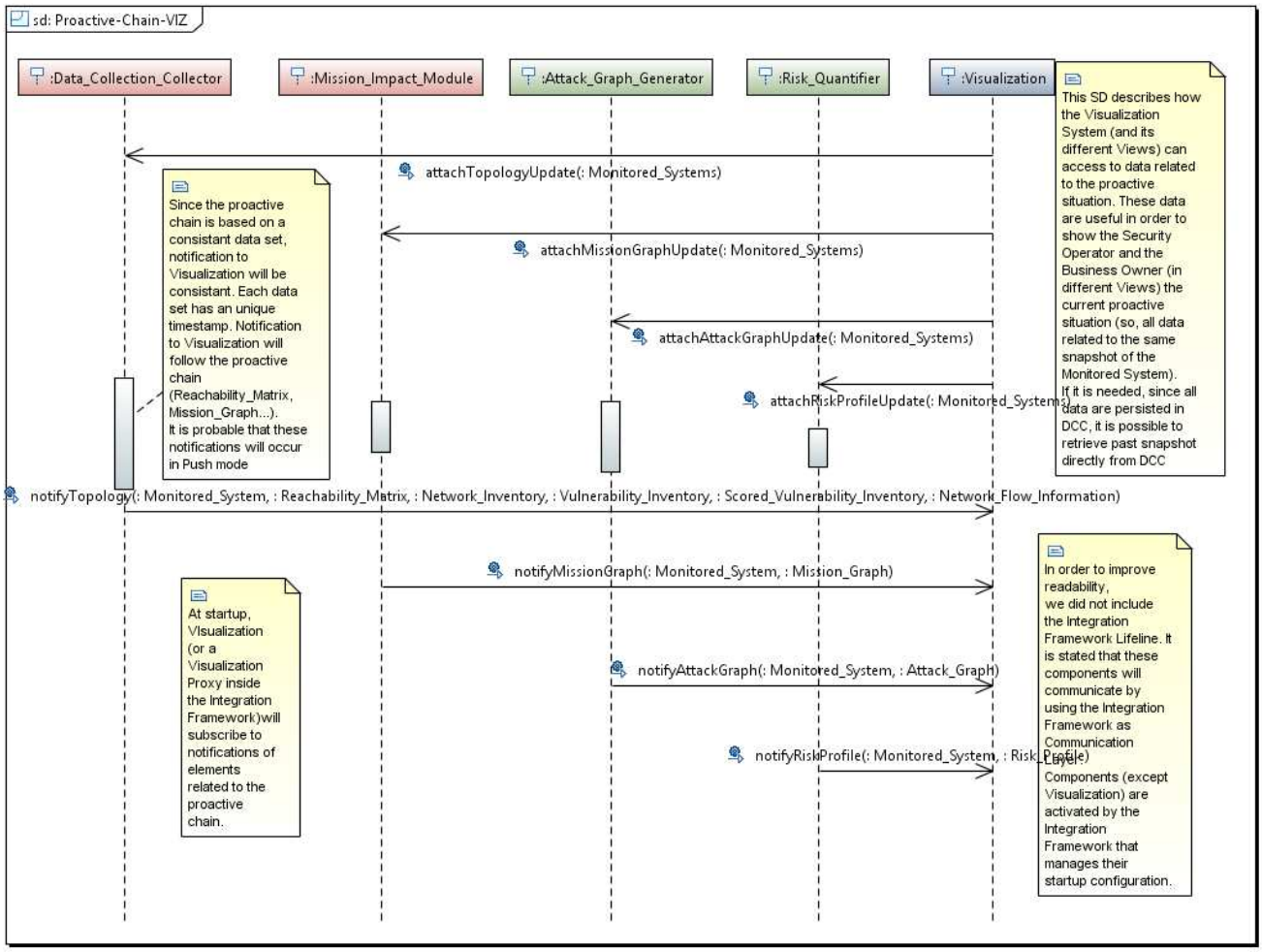


Figure 39: Visualization System proactive data collection

In order to reduce the size of the diagrams for documentation reasons, Integration Framework lifelines are not shown in this Diagram. As can be seen in the previous Logic Diagrams, the Integration Framework will be directly managing all communication between the components. More details in Section 5.2.5.5.

Visualization (via the Integration Framework) subscribes to notifications from DCC, MIM, AGG and RQ for each Monitored System. As soon as these components produce new set of data, they are pushed to the Visualization System. Since a timestamp ID identifies a common snapshot, the data set will remain consistent and the Visualization System will be able to show the current proactive situation of the Monitored System(s).

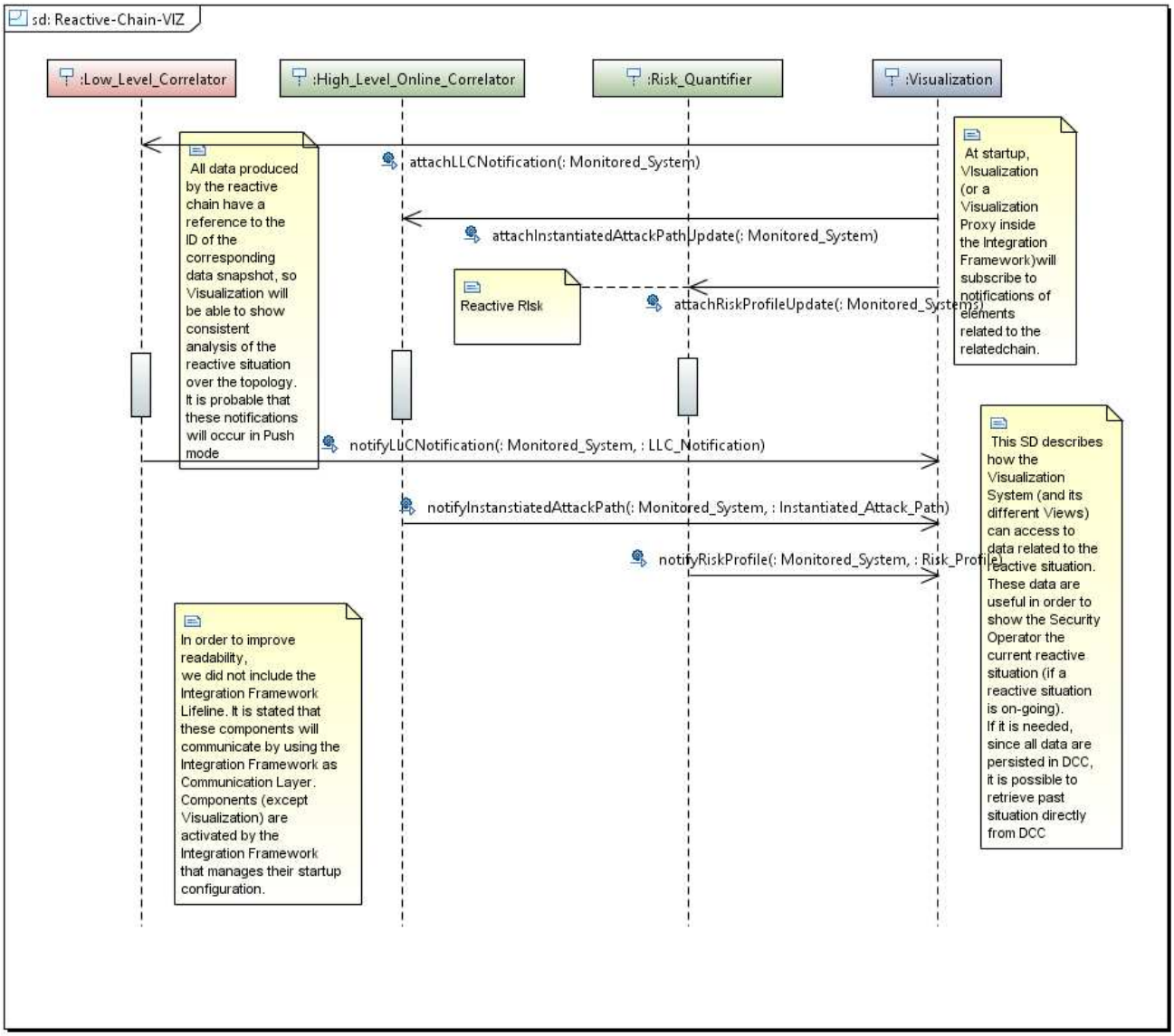


Figure 40: Visualization System reactive data collection

In order to reduce the size of the diagrams for documentation reasons, Integration Framework lifelines are not shown in this Diagram. As can be seen in the previous Logic Diagrams, the Integration Framework will be directly managing all communication between the components. More details in Section 5.2.5.5.

Visualization (via the Integration Framework) subscribes to notifications from LLC, HOC and RQ for each Monitored System. As soon as these components produce new set of data during an on-going perceived cyber-attack, they are pushed to the Visualization System. It will be then possible for the Visualization System to correlate the reactive data to the proactive situation (for example, showing a network map with identified *Attack_Path*(s) and enlightened *Instantiated_Attack_Path*(s) and their reactive *Risk_Profile*(s) in order to depict the possible consequences of an attack).

VIZ023, **VIZ024** and **VIZ025** states that the Visualization System should be able to show historical situations of the Monitored System(s). These ASRs are handled by using the persistency in DCC. DCC subscribes to the proactive and reactive data producers in the same way as Figure 39 and Figure 40 show, collecting all data from the components. The Visualization System will then access to DCC in *pull* mode in order to retrieve past proactive and reactive situation.

The Visualization system is involved in various initialization and configuration activities, depicted in Figure 14, Figure 21: Visualization System-DCI-DCC-MIM (Business Mission Information Initialization) Process View, Figure 27 and Figure 28.

The Visualization System is also involved in the proactive and in the reactive chain: the Security Operator has to choose the *Response_Plan*(s) he wants to deploy in the Monitored System(s) after the evaluation from SRD and TRD (Figure 29 and Figure 32).

5.2.5.5 Integration Framework

The need for an Integration Framework arises from the analysis over the ASRs conducted in the PANOPTESEC Preliminary High Level Design, as stated in [D3.1.1] and ANNEX C. A reasoned choice on the Integration Framework technologies allows the architect to fulfil many of the Non-Functional ASRs. The choice of the technological stack depends on the selected Non-Functional ASRs and the initial analysis in [D2.1.1] and in [D3.1.1]. The Integration Framework will be depicted by using Architecture Viewpoints (cf. Section 5.2.1).

Integration Framework Component View

In Figure 41 a Component View of the Integration Framework within the PANOPTESEC System is shown.

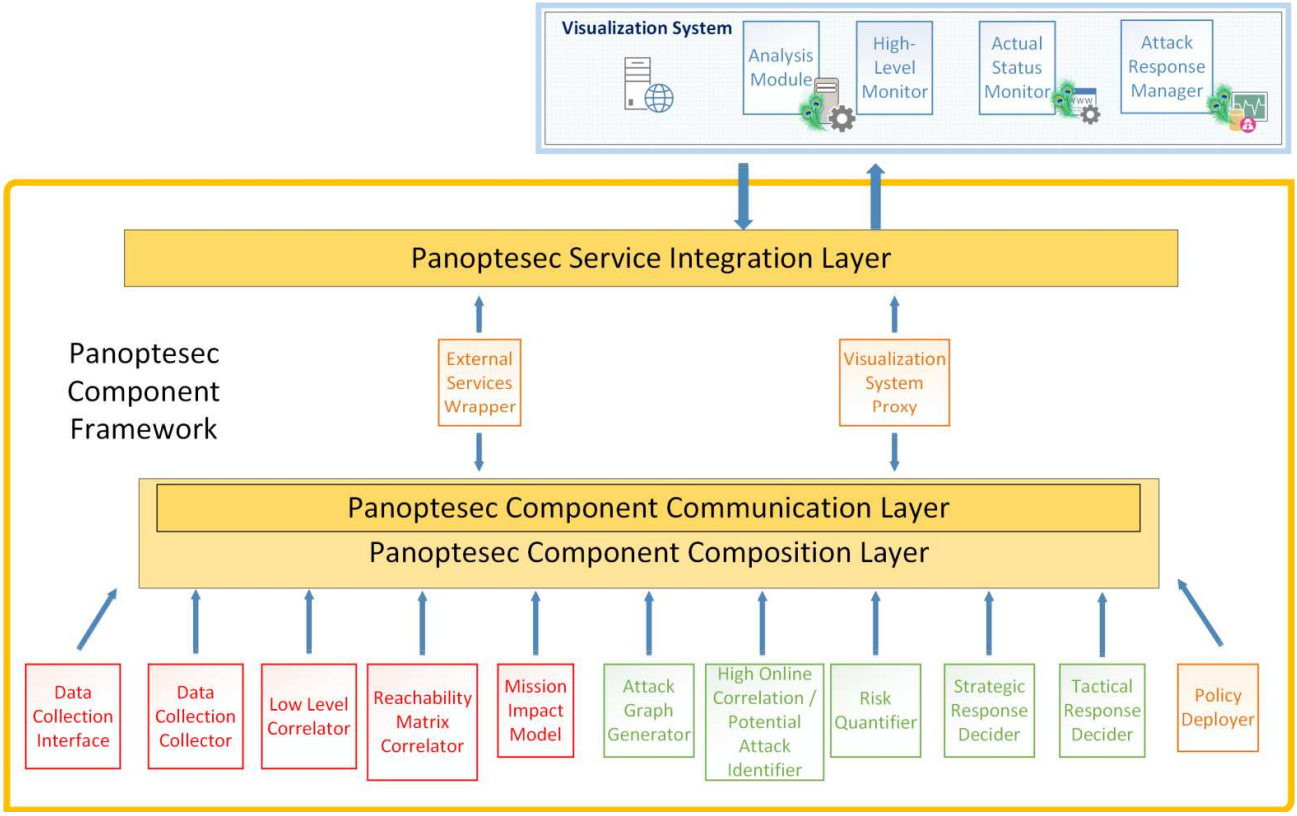


Figure 41: Integration Framework Component View

The PANOPTESEC System implementation with the Integration Framework has a modular structure (as stated in **CMP001**, **MNT001** and **PRT004**), composed of software units built or wrapped under the same standard. Modules are assembled and deployed in the PANOPTESEC Component Framework runtime platform. That modular vision allows the deployment over a distributed system (as stated in **PRT006** and **CMP007**) and low coupling design, following the chosen Architectural Principles (Annex C).

PANOPTESEC Component Composition Layer supports the assembling of the components providing the APIs necessary for the communication and interaction between modules. It includes also the binding interfaces to specific data communication mechanisms. This layer has a modular structure and it is the higher level view of the interfaces linked to components' internal interfaces.

The internal *PANOPTESEC Component Communication Layer* uses message oriented middleware technologies to implement a message bus for distributed high performance components (as stated in **CMP002**, following the Broker Architecture Pattern adopted in [D3.1.1]). Communication is realized using different approaches or EIP (Enterprise Integration Pattern) (e.g. producer-consumer, publishing-registration, request-reply) and even different technologies depending on the nature of the communication (as stated in **CMP004**).

PANOPTESEC Service Integration Layer supports the visibility of PANOPTESEC System services to external systems (if needed) and to the Visualization System. This layer allows

communication with a higher level of abstraction providing integration of services to be used by external application. It exposes PANOPTESEC services to external system and application, through the service integration bus.

The *Visualization System Proxy* component handles connections and communications from/to the PANOPTESEC System with the Visualization System, which runs on a separate Application Server (possibly, on a different machine). The proxy will connect and interact with the necessary visualization functions with the message broker for high performance connections, or using a more standard integration approach using mechanisms provided by service integration layer (**SEC002** states that PANOPTESEC System should limit service exposure. Due to that, the connection with the Visualization System Application Server will probably not rely on an Enterprise Service Bus but directly on the features of the message broker, by using the envisioned Visualization System Proxy).

The External Agent Wrapper component is an envisioned Service Integration Component that can manage connection and communication with the Network and Security Agents Platforms deployed in the Monitored System, as theorized in [D7.1.1].

The PANOPTESEC Component Composition Layer, the PANOPTESEC Component Communication Layer and the PANOPTESEC Service Integration Layer are depicted in the SysML diagrams describing the High Level Logic View of the PANOPTESEC (Figure 25, for example. The wrapper that includes the internal functional component represents the Integration Framework APIs and interfaces needed to integrate and manage the component).

Integration Framework Deployment View

The PANOPTESEC Integration Framework will support modularization of components and flexible deployment (as stated in **PRT006** and **CMP007**), on both local and distributed environments, allowing load balancing distribution. Modules are encapsulated independently and can be deployed on different machines. The diagram in Figure 42 shows an example of local deployment of the PANOPTESEC System in which components run in the same hardware platform, to ensure that the system can be used with minimal hardware requirements and simple configuration approach.

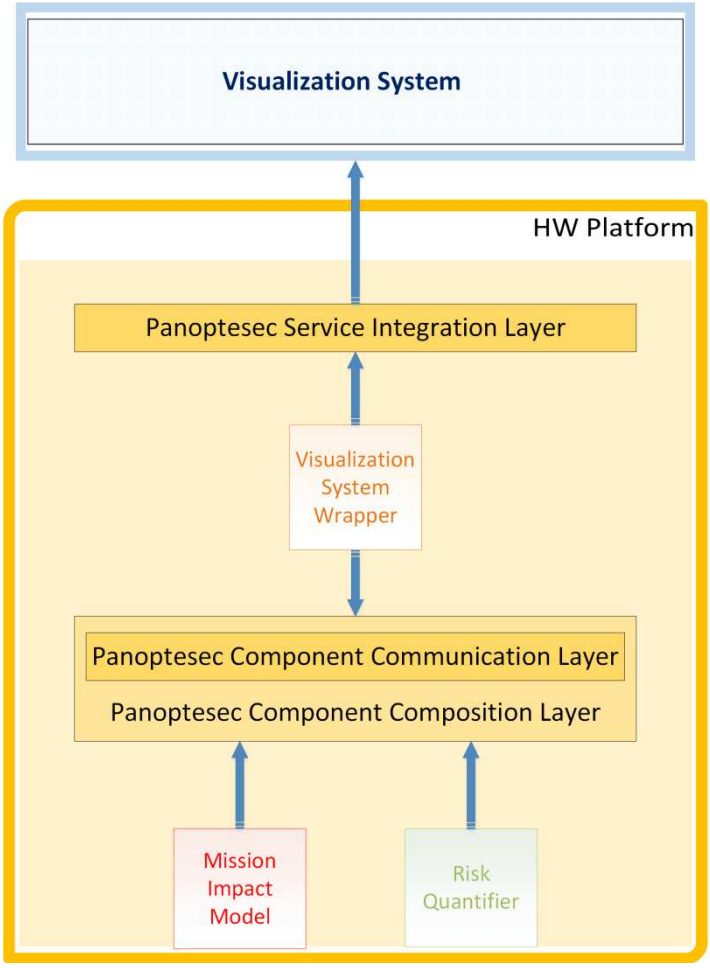


Figure 42: Integration Framework Deployment View (single HW deployment)

The diagram in Figure 43 shows an example of distributed deployment of the PANOPTESEC System, in which components run in different hardware platforms, each one with an independent component runtime container, connected by the Component Communications and Composition Layer and network communication protocols implementation. The Component Composition and Communication Layer binds the interface to components in different platforms to network based mechanisms provided by the message broker.

The Component Communication and Composition Layer has the same implementation for both kinds of deployment: it acts as a link between different modules, with a data communication protocol implementing it.

This architecture supports the management of large amount of data and improves system performance as the load can be distributed on different HW Platform. It also simplifies evaluation and system testing in both local and distributed environments (in order to fulfil **RBL001, RBL002** and **RBL003**). Components can be dynamically managed, stopped, started or replaced without the need of a system reboot (as stated in **CMP008**).

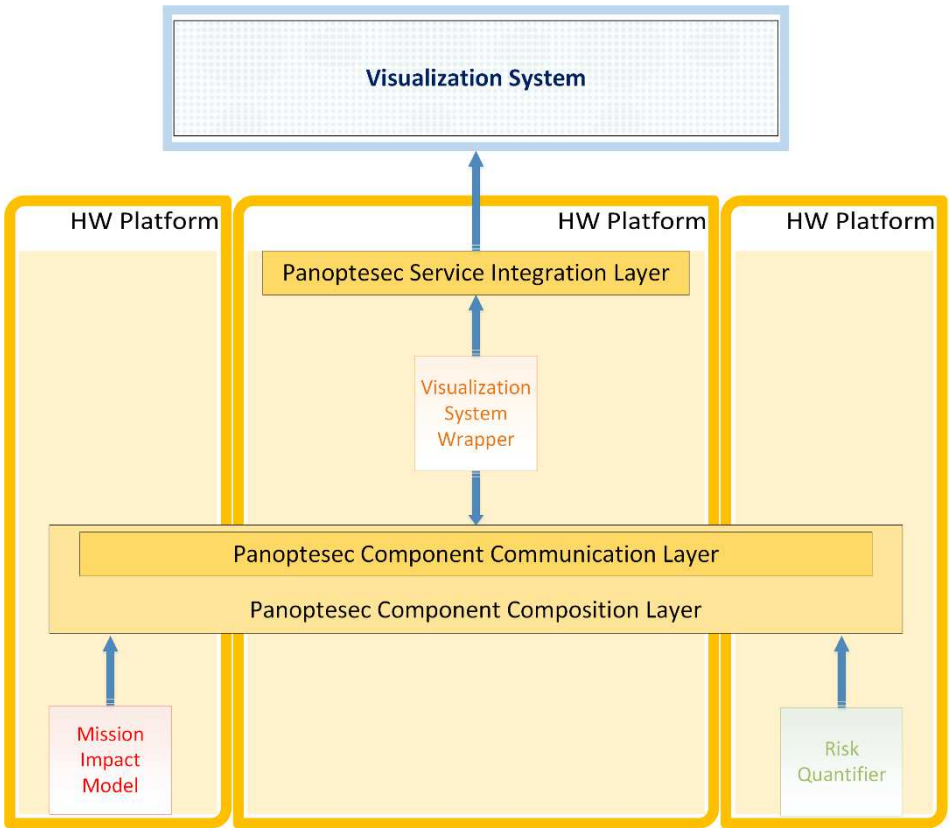


Figure 43: Integration Framework Deployment View (multiple HW deployment)

Integration Framework Integration View

The diagrams in Figure 44 are a high level representation example of the integration of PANOPTESEC System components into the system architecture. The components are plugged into the PANOPTESEC Composition Layer to provide their interfaces internally and to the service layer if the interfaces are exposed as services.

Components are all wrapped and deployed into the runtime platform for execution. The components deployment environment is a container that manages each component independently, along with the needed component configuration (the Integration Framework will then manage the configuration of each integrated component, fulfilling **MNT003)**. It allows taking action over the component (start, stop, etc.) without affecting the lifecycle of the others.

The deployment orchestration will contain the information about which interfaces are provided by which components, and which is the communication technology or mechanism involved. The deployment orchestration is important to ensure that components can locate the others, and it needs to be generated to optimize the communications.

The components expose their interfaces through the PANOPTESEC Composition Layer, which allows binding them to the communication mechanism, supporting also composition of components. This allows any component in the system to communicate with others,

using the different communication patterns supported (e.g. request-response, and publish-subscribe). The component can access any interface provided by other components by using the message middleware technologies.

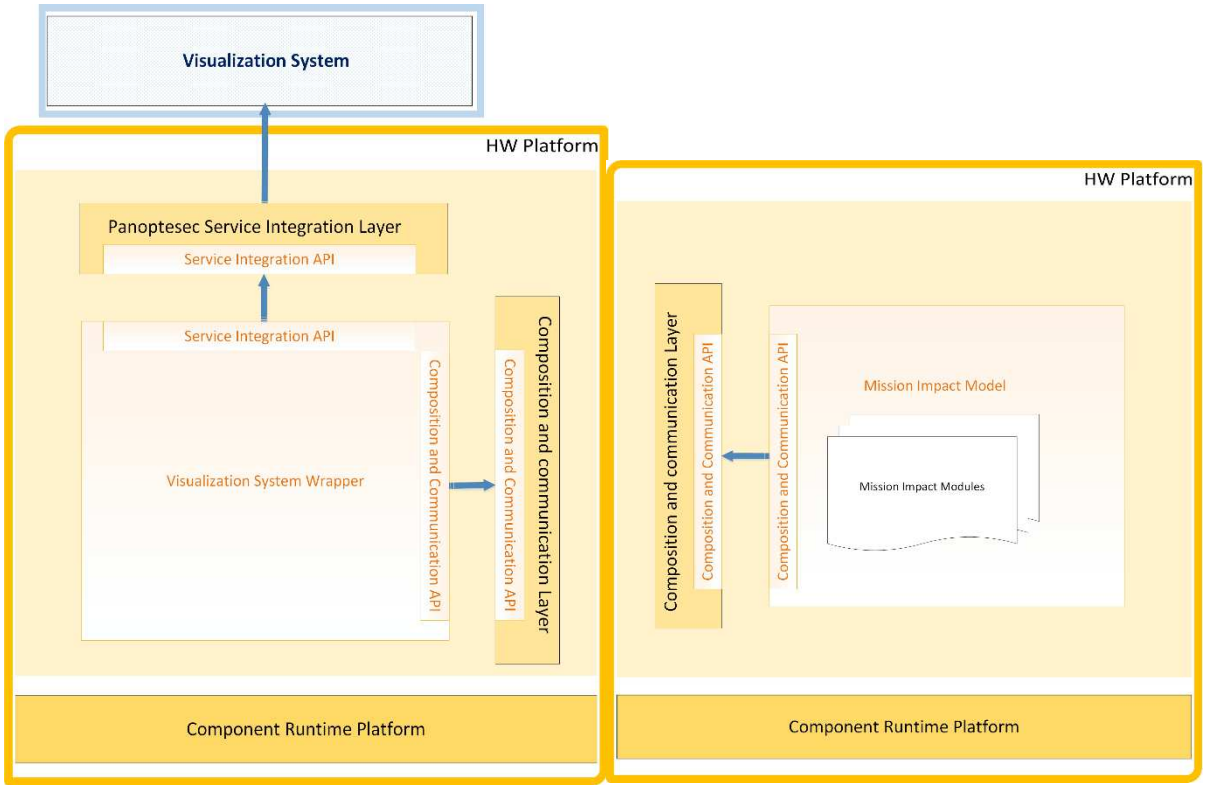


Figure 44: Integration Framework Integration View (example)

The PANOPTESEC System Integration will be developed using the following mechanism:

- *PANOPTESEC Component Modules*: they are plain Java classes organized in packages, which have no dependency on the component framework or deployment technologies. They are mainly concerned about processing logic (as implied in **CMP002** and **CMP003**).
  As **PRT001** states, it is not mandatory that every PANOPTESEC System component is developed in Java language, but it would easier the integration. As requested in **PRT002**, all non-Java components will have Java compliant interfaces –SOAP, REST interfaces, for example. The Integration Framework will be able to wrap these components in order to fulfil **PRT002** with respect of the other PANOPTESEC System components.
- *PANOPTESEC Components*: they are deployable components which internally use the Component Modules for the logic, and provide a set of defined interfaces, and consume interfaces of other components. They are dependent on the PANOPTESEC Component Runtime Platform. The communication between components is realized through the PANOPTESEC Communication Layer.

- *PANOPTESEC Composition and Communication APIs*: they are the result of assembling PANOPTESEC Components interfaces which are known to the rest of PANOPTESEC Components (as stated in **PRT005**). They also bind the components interfaces to the communications technologies and expose composite interfaces where necessary. In addition they can expose their interfaces in the service bus, which implies that the component deployment includes the publishing of its interfaces in order to be found by any other component related to that service bus in the PANOPTESEC Communication Layer.
- *PANOPTESEC Service Integration APIs*: They expose the composite interfaces in the form of technology independent services through a service bus.
- *Component Runtime Platform*: it is the container that will manage components lifecycle inside the system. Each hardware platform will contain one instance of a container and components will be deployed based on an orchestration plan. The environment allows dynamic changes on components and hot deployment without a negative impacting on whole system activity. The main technology supporting that management is OSGi (cf. [OSGi], and a lightweight OSGi container (Apache Karaf) will be used for PANOPTESEC Runtime Platform (the use of OSGi fulfils **CMP008**).
  The Component Runtime Platform add basic fault tolerance capabilities to the integrated system (as stated in **RBL004**). In case of fault, the Runtime Platform can be configured in order to restart the faulted component. Of course, other measures need to be taken in order to ensure complete fault tolerance capabilities to the PANOPTESEC System (the common data persistency described on behalf of the Data Collection Collector component in Section 5.2.5.1 will be able to feed compliant restarted PANOPTESEC System integrated components with the needed data sets).

Integration Framework Security View

PANOPTESEC security approach relies on multiple mechanisms, implemented at different levels of the functional flow. Security Non-Functional ASRs should to be satisfied for any information exchanged between components, or required by external systems. Since the Importance of these Requirements has been set at 2, it will not be mandatory to fulfil all Security Requirements (in particular, **SEC004** can add a significant overhead to the global performances and needs to be balanced with Performances Non-Functional ASRs, for example **PRF006**).

The diagram in Figure 45 is a high level representation example of components and security modules.

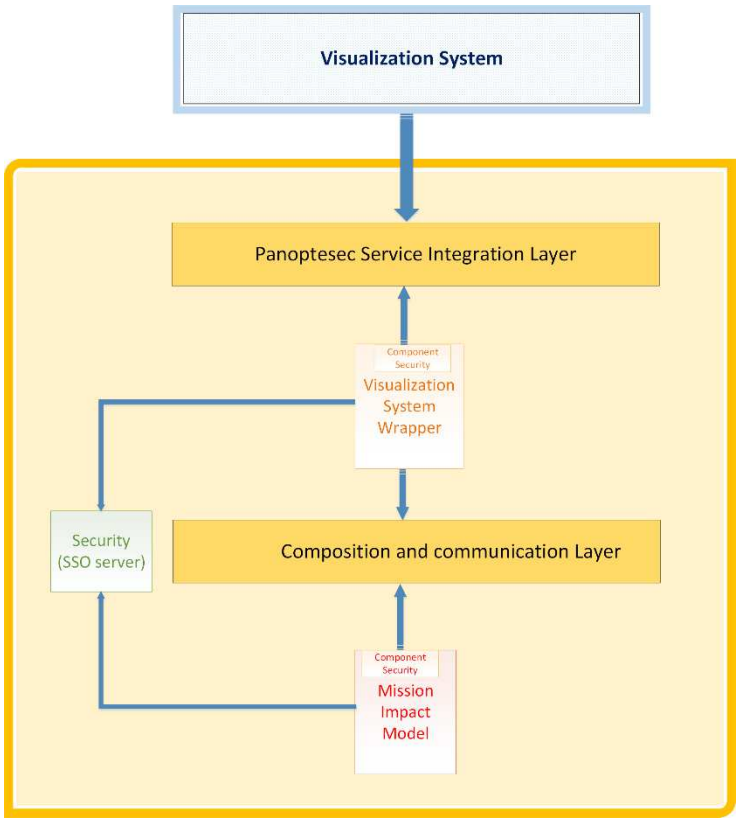


Figure 45: Integration Framework Security View (example)

The PANOPTESEC security approach is implemented in three levels:

- *Service level security*: The ESB (if implemented) provides security mechanisms to authorize the access of components to the services exposed. This is a first level of security which ensures that only authorized components can access the services. This level is necessary because external components may not be trustable enough to connect to the bus to be verified in the next levels internally in the PANOPTESEC System.
- *Component level security*: It provides the mechanism to authenticate components and to provide authentic and encrypted communications between them (as stated in **SEC003** and **SEC004**). Components must be authenticated at start up time, the provided login information will be checked with a *SSO* (single sign-on) mechanism. As already stated in [D3.1.1], a Single Sign-On Server may be added to the PANOPTESEC System and interact with the Integration Framework. The authenticated component is able to access public key infrastructure artefacts which are used to sign and encrypt outgoing messages. Therefore, a component receiving a communication from another will check the authenticity of the message received using the public key infrastructure based cryptography functions.

  The diagrams in Figure 46 shows examples of how component level security will be realized. It displays a message exchange situation.

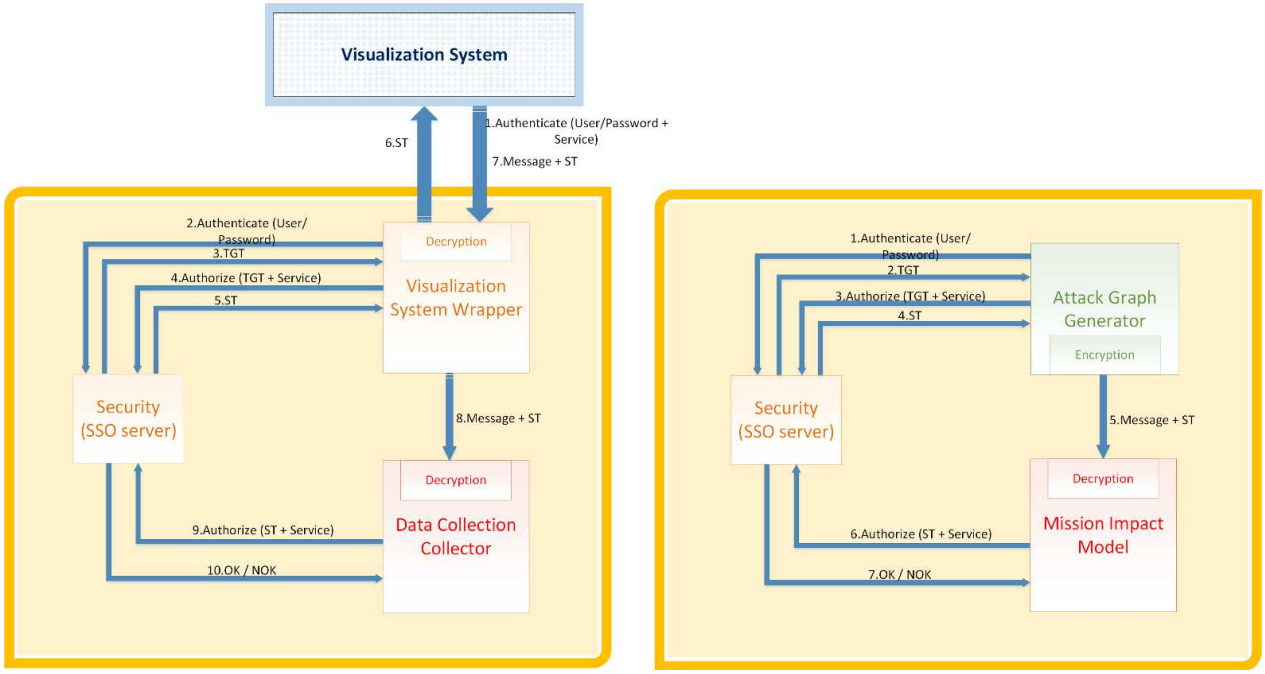


Figure 46: Integration Framework Security Approach (example)

The left diagram in Figure 46 shows the interactions between components and the Security Server. It is presented a complete sequence involving the exchange of tickets between the different elements. The design is based on a SSO (Single Sign-On) mechanism.

Main functionalities are component Authentication and messages exchange Authorization, supporting a single authentication which gives the rights to perform the following actions. This flow of actions includes all the information exchanged, and can be identified as the start-up situation. The Authentication step can be initially performed when two components connect each other. The following message exchanges can be simplified by caching tickets and performing Authorization based on previews Authentication.

Components Authenticate at start-up, gaining a TGT (Ticket Granting Ticket) that prove the positive outcome. TGT can be used in combination with a service (message typology) to obtain the ST (Service Ticket) that represents the authorization for performing that Service. The difference between the two diagrams is that the wrapper components (example the Visualization System Wrapper) will forward messages to the interested component using the External Component ST (Service Ticket).

It is possible to add SSL encryption to each exchanged message.

Integration Framework Technological Stack

Chosen technological stack for the Integration Framework has already been shown in [D3.1.1]. It is based on widely used Open-Source technologies, analysed in [D2.1.1].

The chosen technological stack is compliant with the Integration Framework description in the previous paragraphs of this deliverable. The PANOPTESEC System Integration Framework implementation will be mainly based on Apache Technologies, which provide all the features needed in a single distribution. Below the list of the frameworks and the technologies that will be applied in the realization of the PANOPTESEC Integration Framework is provided.

- **Apache ServiceMix** is the service integration ESB (previously identified as PANOPTESEC Service Integration Layer). Service Mix contains a collection of middleware services that allow to route and transform messages between components.
- **Apache Karaf** for the OSGi container. OSGi brings functionalities such as runtime management, hot deploy, dynamic configuration, logging system. Karaf is the Service Mix Kernel and the distribution used is based on **Apache Felix** OSGi core runtime. Karaf realises the PANOPTESEC Component Composition Layer.
- **Apache Camel** as a flexible way to build routes to connect components together. Camel is the routing and mediation engine which implements the Enterprise Integration Patterns. Camel allows architects to implement orchestration between components. Camel (and the underlying messaging frameworks) implement the PANOPTESEC Component Communication Layer.
- **Apache ActiveMQ** as a MOM (Message Oriented Middleware) to provide a JMS implementation and transport. ActiveMQ will be used in order to implement the Broker Architecture Pattern extablished in ANNEX C. XML or JSON can be easily used for message formatting.
- **ZeroMQ** is a performance oriented point-to-point messaging library. ZeroMQ is not a MOM, and will be used between components where performances could play a major role (for example, in the reactive chain). XML or JSON can be easily used for message formatting.
- **Apache CXF** the web service framework, that allows building and developing services based on different protocols (SOAP, RESTful, etc.). XML or JSON can be easily used for message formatting.
- **Maven** as the tool that supports building and dependencies management of OSGi components.
- **Jenkins** is the framework providing continuous integration services (also regression and integration tests) for software development.

Apache Service Mix is the implementation of an Enterprise Service Bus based on OSGi technology. Service Mix provides packaging of the above technologies, including a Core container that supports deployment and runtime schema for organizing groups of OSGi bundles into ensembles. Relying on Karaf container, Apache ServiceMix allows hot deployment, dynamic configuration and runtime of OSGi bundles. Below are listed some of the features included in Karaf runtime container:

- Karaf Console provides graphical overview of the runtime.

- Logging component allows centralized logging for all OSGi bundles and log filters.
- Karaf provides a Deployer that is able to deploy plain blueprint configuration files.
- Karaf includes a Provisioning component that allows loading of components from repositories.
- Admin console supports creation and management of Karaf instances on the same machine.
- Karaf has its Security component that handles users' authorization and access for bundles' deployment and management.

Service Mix deploys technology components as Apache Camel runtime, Apache CXF runtime and Apache ActiveMQ broker. It is not mandatory to use all technologies provided by Apache ServiceMix: WP7 IT Architects and Developers will choose the most suitable subset (for example, the ESB features of ServiceMix will not be used).

5.3 Identified Risks on the PANOPTESEC High Level Design

The possible risks related to the chosen Architecture, with respect to the ASRs, are summarized in this section. It is possible to identify a set of Risks related to the chosen Architectural Patterns applied to the functional/non-functional needs of the PANOPTESEC System:

- Risks related to the debugging/testing of the Pipe-and-Filter chains (component testing is among the top priority ASRs with **RLB001**). Since many components rely on other components produced data, it can be difficult to effectively debug or test the complete chains. From one hand, each component is designed in order to be able to work independently from the others, while the Integration Framework manages the overall functionality of the System, but in order to obtain useful outputs, a careful orchestration between components needs to be implemented. Testing or debugging these orchestrations can be difficult. The integration strategies depicted in [D7.2.1] need to take into account this possible risk.
- Performances Requirements are not top priority ASRs for the PANOPTESEC System prototype. The use of a Broker Architecture Pattern (and the subsequent adoption of a broker middleware in the Integration Framework, cf. Section 5.2.5.5) can add some overhead in performances that must be taken into account in case of industrial development of the PANOPTESEC System.
- Since the Shared-Data Pattern had been applied to the Data Collection and Correlation System (cf. Section 5.2.5.1), some possible issues related to **RBL010** and **RBL011** had been solved from the Preliminary High Level Design. The shared data store may be a single point of failure for the PANOPTESEC System. While in a prototype that consideration may be not relevant, it can be an issue in the industrial environment. The satisfaction of **RBL005** and **RBL006** can mitigate this possible issue.

The Publish-Subscribe Pattern has been evaluated to be not feasible as a common Architectural Pattern for the PANOPTESEC System (cf. Annex C). Due to the adoption of the Integration Framework (cf. Section 5.2.5.5) in order to manage the communications

between components, it had been evaluated by WP3 that it will possible to use publish-subscribe mechanisms (via Enterprise Integration Patterns) in order to handle the interactions among some components, as Tactics in order to solve development complexity issues. This decision can bring several benefits to the implementation phase (with respect to **CMP002**, **CMP003** and **CMP004**), since several components need to transmit the same set of data (or notification messages) to various other components. The risk is related to synchronization and orchestration of the transmitted data. **RBL011** and **RBL010** make a clear statement about the importance of working on stateful set of data (shared among various components). The presented Design manages this request, but the integration phase managed by WP7 will have to be planned over a consistent Low Level Design depicting in detail every communication between the different components in order to avoid racing conditions and deadlocks.

Another set of possible risks is related to the direct functionalities and needs of the PANOPTESEC System, analyzed in Section 5.2.5.

- Although **PRT004** states that the PANOPTESEC components should be independent (and the Architecture clearly fulfil the Requirement), a possible risk is related to a strict coupling of some components to the data produced from other components. For example, it should be possible to imagine a PANOPTESEC System working without the Mission Impact Model. In that sense, AGG and RQ should be able to compute results independent of the presence of the *Mission_Graph*. This risk needs to be kept in mind during the internal design of each component of the System.
- Due to **RBL011** and the consequent **Control Loop 03**, various components on the proactive chains become strictly coupled to the data set they use in order to perform their computations, partially violating the **Loose Coupling** Principle. This need can cause issues related to the runtime management of the PANOPTESEC System (related to **RBL004**): if a proactive chain component faults and is restarted by the Integration Framework, it needs to access to a consistent data set with the other components. At the same time, the rest of the System should be aware of the missing component (this can be handled by the Integration Framework if the faulted component is integrated).
  During the design of the integrated PANOPTESEC System ([D7.2.1] and [D7.3.1]) these issues must be taken into account. Detailed Component/Deployment Views of the System must been designed in order to guide developers.
- **ICA006** states that all correlation results must be persisted. This is also applied in the processes between LLC and DCC (cf. Sequence Diagram in Figure 18). *Normalized_Alert*(s) are persisted in DCC during the processes. This may lead to a loss in performances, in case of a huge streaming of alerts. It must be evaluated if this procedure is necessary in any case.
- **DSC028** states that public vulnerabilities databases information must be collected by the PANOPTESEC System. Since this operation should occur at least daily based, via Internet, it may be a cause of Security problems (related to **SEC002**).
- In order to fulfil **RBL010** and **RBL011** a control feedback has been added to the proactive chain. While working, this approach can be weak in case of components

fault (since the Integration Framework will manage these messages, it is possible to state that the chosen technological stack –cf. Section 5.2.5.5- will offer enough reliability assurance in order to couple with temporary network issues, packet losses, etc.). Further studies must be conducted in order to improve this mechanism.

- In order to improve performances related to attacks management, the reactive chain is not locked to the proactive one. This can cause inconsistencies between the raw IDS data and the Topology/Vulnerability Monitored System(s) current snapshot (the reactive chain is likely faster than the proactive one). The PANOPTESEC Consortium will have to conduct a careful analysis over risks and benefits of this approach as part of detailed design activities.

5.4 Identified Control Loops on the PANOPTESEC System High Level Design

There are three control loops identified in the System High Level Design. These control loops have been analyzed as follows:

- Since data consistency and overall Reliability are a major concern in the Non-Functional Requirements, in the High Level Design the proactive chain is entangled with a control loop (**Control Loop 03**) between SRD and DCC. This will lock PANOPTESEC System proactive performances to, at least, the sum of the computation time of each single component in the chain. In addition, some components will remain inactive until the chain is not ended (this is acceptable because Reliability ASRs have a higher priority than Performance ASRs). By applying the Data-Shared Pattern over the **Control Loop 01** identified in [D3.1.1] it is possible to decouple the Data Collection and Correlation processes from the proactive chain (cf. Section 5.2.5.1), but more research must be conducted in order to reduce the performance loss caused by this Control Loop (cf. Section 5.3).
- Another risk can be related to **Control Loop 02** (between SRD/TRD-PDP-the Monitored System(s) and DCI. In theory, every Mitigation Action enforced by the proactive and reactive chains should immediately affect the Monitored System and should be directly perceived by the PANOPTESEC System (giving the Security Operator an immediate feeling about the adopted solution to a proactive or reactive solution). This hypothetical situation could cause possible racing conditions and affect the overall stability of the System, if the System would be designed in order to stop every topology/vulnerability/proactive running computations any time a change is perceived, in order to start a new computation. This is not the case of the actual design. Even in the case of immediate perception of updates in the Monitored System(s), no racing conditions could occur in the PANOPTESEC System.

  As a further consideration, Mitigation Actions deployment can be time consuming and this overhead can be hardly predictable because the deployment processes may be carried out by external services while the PANOPTESEC System' perception of the Monitored System strictly depends on the reactivity of the external tools used for raw data collection. This chain of possible delays can cause the PANOPTESEC System Monitored System(s) status perception to be out of date compared with the real situation of the Monitored System. Due to these considerations, it would be possible

that the PANOPTESEC System would propose the same Mitigation Actions over subsequent data collection iterations. As explained in SRD and TRD paragraphs in Section 5.2.5.3, these two components will maintain an internal persistency of the already (Enforced)*Response_Plan*(s) in order to avoid this possible issue.

6 FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS TRACEABILITY

In this Deliverable a consistent analysis over the identified ASRs and over non-ASRs Functional and Non-Functional Requirements has been conducted in Section 4 and 5. It is necessary to state the level of satisfaction of the current High Level Design over the complete set of Requirements, Functional and Non-Functional. Requirements traceability matrixes are presented in Annex D. The evaluation of the architecture (Section 2) is based on the current level of Requirements satisfaction.

Functional Requirements coverage is easily obtained: in Section 5 the Functional Design elements have been depicted with emphasis of the Requirements driving the functionalities that they will have to show. The table will then show which Design Elements will have to cover the requested functionalities.

Most of the Functional Requirements had been satisfied by the current Design, except:
DSC033
PRS013
RRS012
RRS018
VIZ017
VIZ020
VIZ028
VIZ030

Among them, the only ASR (and the only with Importance 2) is **VIZ028** (with a very low priority in the ASRs list). The other non-satisfied Requirements have Importance 1. These Requirements will be eventually addressed in the more detailed Designs from WP4, WP5 and WP6. Due to their low Importance, the lack of their satisfaction is not considered a risk for the current architecture.

Regarding Non-Functional Requirements, due to their own nature that often encompasses a single Design Element, a rationale of the architecture coverage is given, along with, when possible, a description of the application of Architecture Tactics adopted in order to meet the Requirement. Some Non-Functional Requirements, although, will be fulfilled by some Design Elements (usually, the Integration Framework). A first analysis regarding [D2.2.1] Requirements coverage had been conducted within [D3.1.1] and is further developed in the present deliverable (since some Design Elements are changed and some processes have been better explored). It is unlikely that all Non-Functional ASRs could be satisfied: Performances ASRs and Reliability ASRs, for example, usually are in conflict and the IT Architect must decide which Quality Attribute needs to be addressed with priority.

The evaluation of the satisfaction of Non-Functional Requirements has three possible values:

- SATISFIED (the current Design covers the Requirement)
- PARTIALLY SATISFIED (the current Design only partially covers the Requirement. In alternative, a Test Case that can verify the Requirement is provided)

- NOT SATISFIED (the current Design cannot cover the Requirement and a Test Case that can verify it has not been yet envisioned).

Among the Non-Functional Requirements, this Design does not satisfy:
MNT004
PRF005

MNT004 is related to the validation of the overall behaviour of the PANOPTESEC System after the integration of the components: it is simply impossible to make a statement, at this stage of the Project, and it will have to be evaluated again (with a complete set of Test Cases).

PRF005 is related to the deployment of the PANOPTESEC System components. Even if it is already possible to make some assumptions (cf. Table 11), WP3 still does not have sufficient elements in order to be able to make a final statement. Since the Requirement is not rigid, WP3 is not particularly worried in fulfilling it in the future).

Among the top-priority Non-Functional ASRs (ASRs with Importance 3 and Architectural Impact 3), two of them are evaluated as PARTIALLY SATISFIED:
RLB001
PRT004

In Section 5.3 are summarized some possible risks related to these Requirements.

Among the ASRs with Importance 3 and Architectural Impact 2, three of them are evaluated as PARTIALLY SATISFIED:
PRF008
RLB002
RLB003

PRF008 will have to be fully satisfied during the next phases of the Project, by depicting meaningful Test Cases. **RBL002** and **RBL003** are also related to the testing of the PANOPTESEC System. The Integration Framework can access to a set of technologies that will help developers implement the selected tests (component tests, acceptance tests). Except the possible risks related to **RBL001** in Section 5.3, there is no particular concern with the fulfilment of these Requirements during the Low Level Design Phase.

A rationale of the satisfaction/partial satisfaction, and related issues, of all the other Non-Functional Requirements from [D2.2.1] is given in Annex D.

In summary, it has been demonstrated that the proposed architecture satisfies Functional ASRs and the most important Non-Functional ASRs, trying to find a balance between all of them. All Non-Functional ASRs depicted as PARTIALLY SATISFIED or NOT SATISFIED need to be addressed again in [D7.2.1] and [D7.3.1].

7 CONCLUSIONS

7.1 Significant results achieved

This deliverable describes the High Level Design of the PANOPTESEC System, by analysing different Views of the System, functionalities and architectural decisions encompassing all the PANOPTESEC System. Although the Design Projects in SysML remains a fundamental source of information for PANOPTESEC developers and architects, this deliverable contains a sufficient degree of rationale and detailed explanations on the final High Level Design and the followed architectural methodology. A common High Level Data Model is then presented in order to harmonize the High Level Designs made by WP4, WP5 and WP6.

A detailed analysis over Requirements in [D2.2.1] is conducted in order to demonstrate that the proposed architecture covers most of the ASRs and non-ASRs.

A risk analysis over the established architecture is then presented, with details over identified control loops within the PANOPTESEC System.

7.2 Recommendations

This deliverable is the foundation of the PANOPTESEC System architecture. A deep comprehension of the High Level Design is fundamental for any architect and developer of the PANOPTESEC Project. The proposed methodologies will have to be followed with the guidance of the WP3 Leader.

7.3 Deliverable validation

This deliverable has been validated in accordance with the quality assurance plan of the PANOPTESEC project as outlined in the [PH15] and following supporting quality review checklists and procedures. The outcomes of the quality review are located on the project SVN.

8 REFERENCES

[DoW2013] PANOPTESEC Consortium – "*Annex I - Description of Work*" – European Union Seventh Framework Programme, DoW of the PANOPTESEC project, Grant Agreement no: 610416, 19 Sep 2013.

[D2.1.1] PANOPTESEC Consortium - *"Deficiency Evaluation"* - Project deliverable D2.1.1, version 1.0, April 30th, 2014.

[D2.2.1] PANOPTESEC Consortium - *"Operational Requirements Analysis"* - Project deliverable D2.2.1, version 2.0, March 27th, 2015.

[D3.1.1] PANOPTESEC Consortium - *"System High-Level Preliminary Design"* - Project deliverable D3.1.1, version 2.0, March 27th, 2015.

[D3.1.2] PANOPTESEC Consortium – *"System High-Level Design"* - Project deliverable D3.1.2, version 1.0, version 2.0, March 27st, 2015.

[D4.1.1] PANOPTESEC Consortium – *"Data Collection and Correlation Functional Requirements"* – Project deliverable [D4.1.1], version 2.0, March 27st, 2015.

[D4.1.2] PANOPTESEC Consortium – *"Data Collection and Correlation High Level Design"* – Project deliverable [D4.1.2], version 1.0, October 31st, 2014.

[D5.1.1] PANOPTESEC Consortium - *"Response System for Dynamic Risk Management Requirements"* - Project deliverable [D5.1.1], version 2.0, March 27st, 2015.

[D5.1.2] PANOPTESEC Consortium – *"Response System for Dynamic Risk Management Models and High-Level Design"* – Project deliverable D5.1.2, version 1.0, October 31st, 2014**.**

[D6.1.1] PANOPTESEC Consortium - *"Visualization Component Requirements"* - Project deliverable [D6.1.1], version 2.0, March 27st, 2015.

[D7.1.1] PANOPTESEC Consortium – *"Simulation Environment"* – Project deliverable D7.1.1, version 1.0, October 31st, 2014**.**

[D7.2.1] PANOPTESEC Consortium – *"Integration Framework Prototype I"* – Project deliverable D7.2.1, version 1.0, July 31st, 2015**.**

[D7.3.1] PANOPTESEC Consortium – *"Integration Framework Prototype II"* – Project deliverable D7.2.1, version 1.0, April 31st, 2016**.**

[PH15] PANOPTESEC Consortium – "PANOPTESEC Project Handbook" – Project internal document, version 0.1, March 27th 2015.

[QAS15] PANOPTESEC Consortium – "PANOPTESEC QA Schedule" - Project internal document, available online at https://gotika.ifis.uni-luebeck.de/panoptesec/WP01/Project%20Handbook/Quality%20Assurance/QA%20Schedule

[SAiP] L.Bass, P. Clements, R. Kazman, *Software Architecture in Practice 3rd ed.*, Addison Wesley, 2013.

[DSA] P.Clemens, L.Bass, F.Bachmann, *Documenting Software Architectures, Views and Beyond 2rd ed.*, Addison Wesley, 2011.

[SWA] O.Vogel, I.Arnold, A.Chughtai,T.Kehrer, *Software Architecture, a comprehensive Framework and Guide for Practitioners*, Springer

[Coplien] J.Coplien, *A Pattern Definition*, http://hillside.net/patterns/222-design-pattern-definition, 2004

[Harrison] N.Harrison, P.Avgeriou, *Leveraging Architecture Patterns to Satisfy Quality Attributes*, http://www.cs.rug.nl/paris/papers/ECSA07.pdf

[IEEE 2007] IEEE, *Recommended Practice for Architectural Description of Software-intensive Systems*, http://www.iso-architecture.org/ieee-1471, 2007
[KPD2014] Kanoun Waël, Papillon Serge, Dubus Samuel – "*Calculating & Composing Elementary Risks: Novel Decision Support Systems for Cyber Defence*" – NATO SAS-106 Symposium on Analysis Support to Decision Making in Cyber Defence, 9-11 June 2014, Tallinn, Estonia.
[ABO03] Abou-El-Kalam, A., Baida, R., Balbiani, P., Benferhat, S., Cuppens, F., Deswarte, Y., Miège, A., Saurel, C., Trouessin, G.: "*Organization Based Access Control. In: IEEE 4th International Workshop on Policies for Distributed Systems and Networks*". (2003) 120–131.
[MIE05] Miège, Alexandre -"*Definition of a formal framework for specifying security policies. The OrBAC model and extensions*" - PhD thesis, Ecole Nationale Superieure des Télécommunications, Paris, 2005
[OSGi] http://www.osgi.org/Main/HomePage, last accessed on 22/03/2015
[OMGSYSML] http://www.omgsysml.org/, last accessed on 10/03/2015.
[PAPYRUS] http://www.eclipse.org/papyrus/, last accessed on 14/03/2015.
[NVD] http://nvd.nist.gov, last accessed on 13/03/2015.
[OMGUML] http://www.uml.org/, last accessed on 08/03/2015.

ANNEX A: ASR FROM NON-FUNCTIONAL REQUIREMENTS

All Non-Functional Requirements in [D2.2.1] have been evaluated by Importance for stakeholders, Reachability in terms of research and engineering efforts and Architectural Impact. Focus is on Importance and Architectural Impact.

Table 2: ASR FROM NON-FUNCTIONAL REQUIREMENTS

Id	Importance	Reachability	Architectural Impact
CMP001	3	2	3
CMP002	3	2	3
MNT001	3	2	3
PRT002	3	2	3
PRT004	3	2	3
RLB001	3	2	3
RLB011	3	3	3
PRT005	3	2	3
USG001	3	3	2
RLB012	3	3	2
CMP005	3	1	2
PRF008	3	2	2
RLB002	3	2	2
RLB003	3	2	2
RLB009	3	3	2
USG002	3	3	2
RLB013	3	3	2
RLB014	3	3	2
CMP003	2	2	3
CMP009	2	2	3
CMP007	2	1	3
CMP008	2	1	3
MNT004	2	2	3
PRF001	2	3	3
PRF004	2	3	3
PRF006	2	3	3
PRF007	2	3	3
PRT001	2	2	3
PRT003	2	2	3
RLB004	2	3	3

RLB010	2	3	3
SEC001	2	1	3
SEC002	2	1	3
SEC003	2	2	3
SEC004	2	2	3
SEC005	2	2	3
USG003	2	3	3
CMP004	2	2	2
CMP006	2	2	2
MNT002	2	2	2
MNT003	2	2	2
PRF002	2	3	2
PRF003	2	3	2
PRF005	2	3	2
PRT006	2	2	2
RLB005	2	1	2
RLB008	2	1	2
SEC006	2	2	2

ANNEX B: ASR FROM FUNCTIONAL REQUIREMENTS

All Functional Requirements in [D2.2.1] have been evaluated by Importance for stakeholders, Reachability in terms of research and engineering efforts and Architectural Impact. Focus is on Importance and Architectural Impact. Not all Functional Requirements are ASRs: Functional Requirements with Importance 1 and/or Architectural Impact 1 are not considered ASRs.

Table 3: ASR FROM FUNCTIONAL REQUIREMENTS

Id	Importance	Reachability	Architectural Impact
DSC001	3	1	3
DSC006	3	1	3
DSC029	3	1	3
ICA001	3	1	3
ICA006	3	2	3
ICA010	3	1	3
PRS001	3	1	3
PRS022	3	3	3
RRS001	3	1	3
RSS024	3	3	3
VIZ001	3	1	3
DSC002	3	2	2
DSC003	3	2	2
DSC004	3	3	2
DSC007	3	1	2
DSC014	3	2	2
DSC017	3	2	2
DSC018	3	2	2
DSC023	3	1	2
DSC025	3	3	2
DSC026	3	3	2
DSC027	3	3	2
DSC028	3	1	2
DSC030	3	1	2
DSC031	3	1	2
ICA003	3	3	2
ICA004	3	3	2
ICA005	3	3	2
ICA011	3	3	2

ICA012	3	3	2
ICA013	3	3	2
ICA014	3	3	2
ICA015	3	3	2
ICA020	3	3	2
PRS002	3	1	2
PRS003	3	3	2
PRS004	3	2	2
PRS005	3	3	2
PRS006	3	2	2
PRS007	3	1	2
PRS008	3	3	2
PRS009	3	1	2
PRS010	3	1	2
PRS011	3	3	2
PRS012	3	3	2
PRS014	3	3	2
PRS015	3	3	2
PRS016	3	2	2
PRS017	3	1	2
RRS002	3	3	2
RRS003	3	3	2
RRS004	3	2	2
RRS005	3	3	2
RRS006	3	2	2
RRS007	3	3	2
RRS008	3	3	2
RRS009	3	1	2
RRS010	3	3	2
RRS011	3	3	2
RRS013	3	3	2
RRS014	3	3	2
RRS015	3	3	2
RRS016	3	2	2
RRS017	3	3	2
RRS025	3	3	2
RRS026	3	3	2
VIZ002	3	3	2

VIZ003	3	3	2
VIZ004	3	3	2
VIZ005	3	3	2
VIZ006	3	3	2
VIZ007	3	2	2
VIZ008	3	2	2
VIZ009	3	1	2
VIZ010	3	2	2
VIZ011	3	2	2
VIZ012	3	3	2
VIZ013	3	3	2
VIZ014	3	3	2
VIZ015	3	3	2
VIZ016	3	2	2
VIZ018	3	3	2
VIZ021	3	2	2
VIZ026	3	3	2
DSC005	2	2	3
VIZ023	2	3	3
VIZ024	2	3	2
DSC032	2	2	2
ICA002	2	3	2
RRS023	2	3	2
RRS027	2	3	2
VIZ027	2	3	2
VIZ028	2	1	2
VIZ029	2	2	2

ANNEX C: ARCHITECTURE PRINCIPLES AND PATTERNS FROM [D.3.1.1]

In [D3.1.1] a complete analysis over top-priority ASRs (Requirements with Importance 3 and Architectural Impact 3) had been conducted. Here a summary of this analysis is presented (with the aim of keeping the current deliverable as independent as possible, due to its Public Dissemination Level).

First functional architecture

The most important key abstractions related to the PANOPTESEC System are shown in in Figure 47. They are mainly deducted from the Description of Work Objectives [D2.2.1] where DOW Objectives derive from the Framework Project 7 ICT-2013.1.5 Item (c) objectives and extensive domain knowledge of cyber defence concepts and needs.

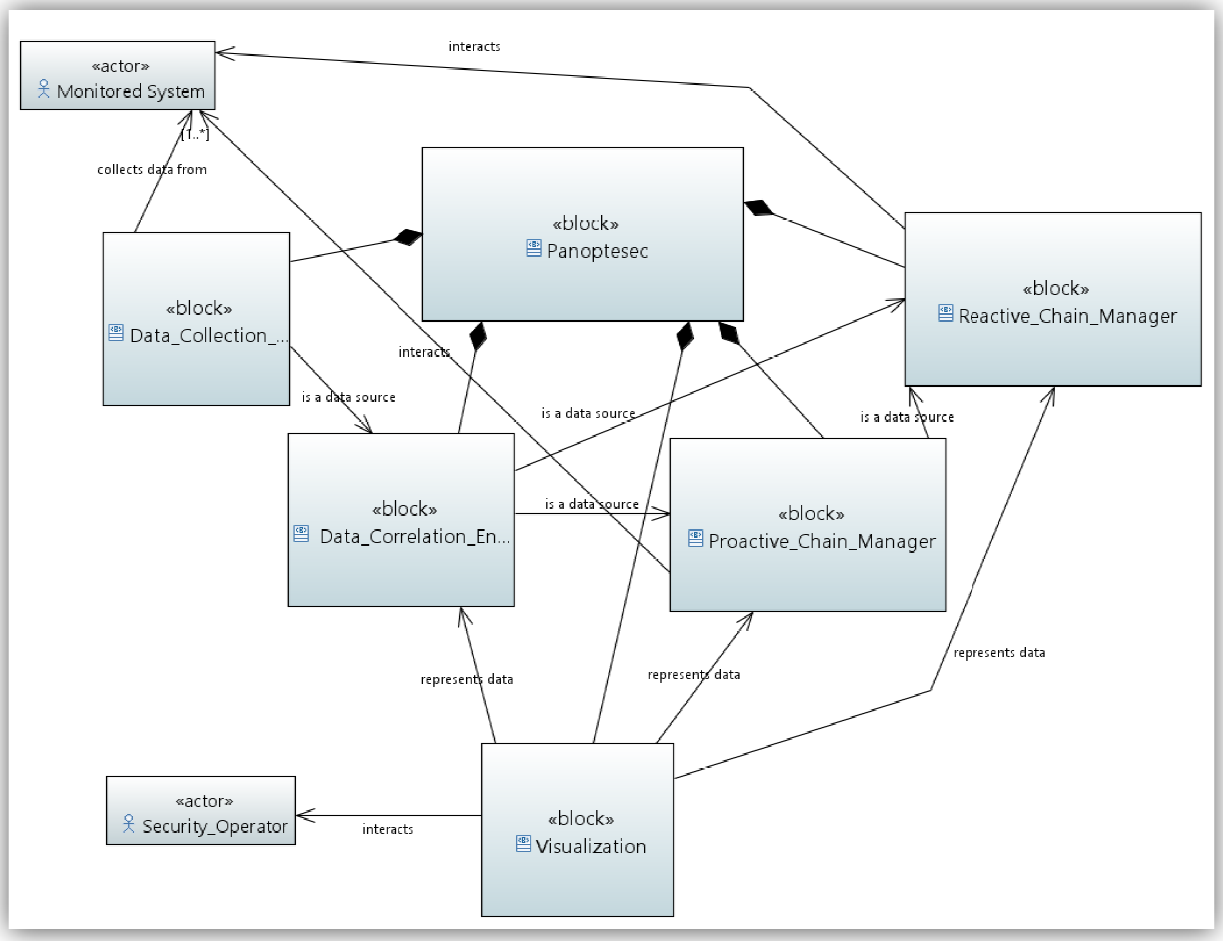


Figure 47: PANOPTESEC Key Abstractions

Considering the top Functional ASRs (Requirements with Importance 3 and Architectural Impact 3) for the PANOPTESEC System in 3, it is possible to derive the first High Level functional architecture (Figure 48).

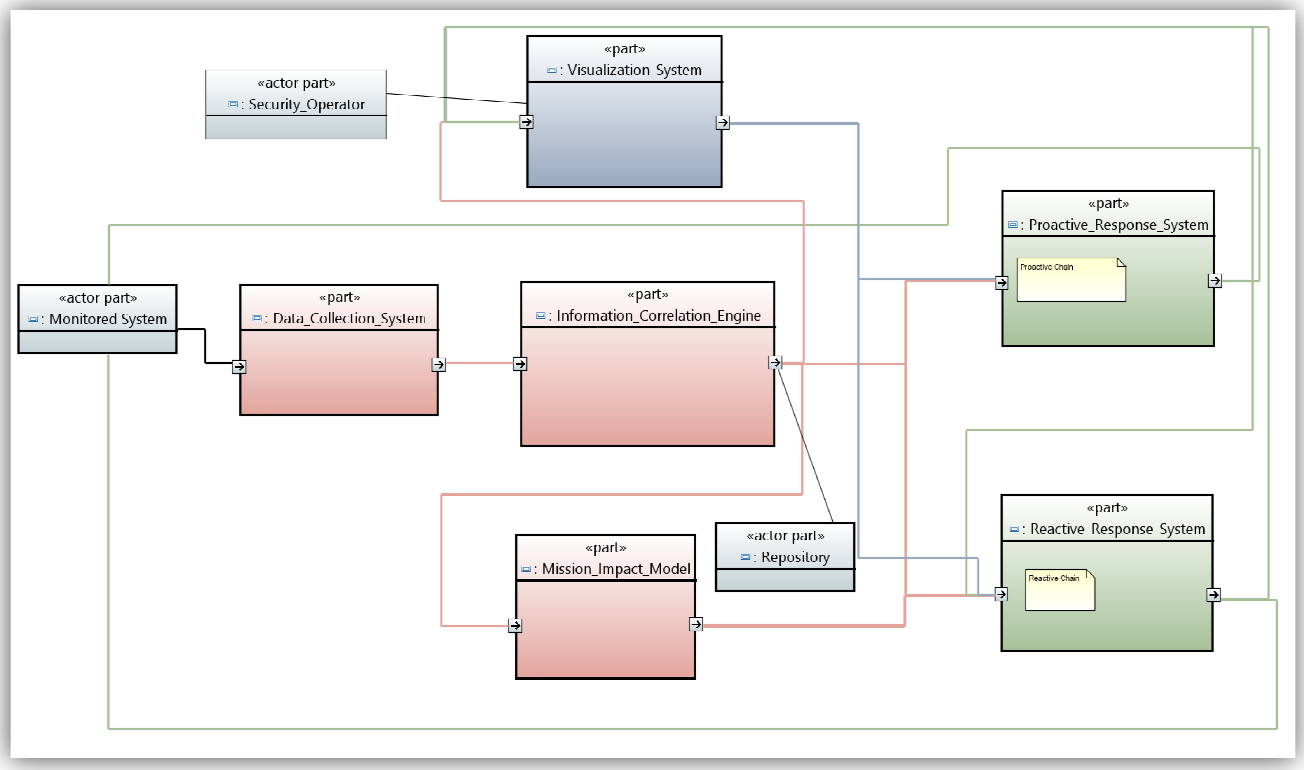


Figure 48: PANOPTESEC High Level Functional Architecture

This functional architecture covers all these top Functional ASRs and encompasses the PANOPTESEC System key abstractions.

The Data Collection System block satisfies **DSC001**, **DSC006** and **DSC029**. This block is responsible for collecting raw data (with a specific focus on topology data, vulnerability data and cyber-security alerts from IDSs and firewalls) from the Monitored System(s).

The Information Correlation Engine block satisfies **ICA001** and is responsible for normalization and correlation of raw data coming from the Data Collection System in order to create a unified view of the monitored system. A “Repository” Actor is added in order to satisfy **ICA006**.

The Mission Impact Model block satisfies **ICA010**. The Mission Impact Model is responsible for the evaluation of the critical system of the Monitored System(s), in order to focus the efforts of the Proactive and the Reactive Response System. The Mission Impact Model aims to keep a track of on-going business processes (or missions) inside a company and connect the business world with the underlying ICT or SCADA world. The Mission Impact Model

obtains relevant data concerning business functions and missions via the Information Correlation Engine.
The Proactive Response System block satisfies **PRS001** and manages the proactive risk management chain of treatment. **PRS022** states that the selected Mitigation Actions computed from the Proactive Response System must be deployed in the Monitored System. In order to satisfy this Requirement, an output link between the Proactive Response System and the Monitored System has been added. This task consists of determining possible actions over the Monitored System, like deploying a patch on a device or opening or closing a Port on a device (these tasks can be carried out by remote services installed on the devices, where possible). The Proactive Response System obtains relevant data from the Information Correlation Engine and the Mission Impact Model.

The Reactive Response System block satisfies **RRS001** and manages the proactive risk management chain of treatment. **RRS024** states that the selected Mitigation Actions computed from the Reactive Response System must be deployed in the Monitored System(s). In order to satisfy this Requirement, an output link between the Reactive Response System and the Monitored System(s) has been added. This task consists of determining possible actions over the Monitored System, like opening or closing a port on a node (these tasks can be carried out by remote services installed on the nodes). The Reactive Response System obtains relevant data from the Information Correlation Engine, the Mission Impact Model and the Proactive Response System.

The Visualization System block satisfies **VIZ001**. It is responsible for providing a visual interface for the PANOPTESEC Security Operator monitoring the actual network and cyber-security status of the Monitored System and analysing the current situation in terms of risk and on-going attacks. The Visualization System is also involved in the Response Management process. The PANOPTESEC Security Operator will decide which Mitigation Action proposed by the proactive and reactive chains will be deployed in the Monitored System.

Selection and application of Architectural Means

The selection of Architectural Means is a key procedure in every software project. Common Principles, Patterns, Tactics give the architect powerful drivers for building the architecture with respect of the Top priority Non-Functional ASRs (Importance 3 and Architectural Impact 3)

Architectural Principles

Architecture Principles are general principles that should be considered when designing architecture. According to the top priority Non-Functional ASRs, some Principles have been taken into particular account during the design of the PANOPTESEC System.

Loose Coupling: this Principle states that the coupling between building blocks should be kept as low as possible. The less strongly a building block is coupled with other building blocks, the easier it is to understand, design and maintain the building block in order to

improve modifiability and compatibility of the system. This Principle is applied to the PANOPTESEC architecture: every block identified in Section 5.1.1 has its own functionalities and purpose and represents a single step (or a set of steps) in the System's complete functionalities.

Loosely coupled architecture can increase complexity or resource consumption. By analysing the top priority ASRs, it can be easily stated that these concerns are not a main issue, so the Loose Coupling Principle is widely used in the PANOPTESEC System design.

Design for Change: this Principle is tightly coupled with the Loose Coupling Principle. Design for Change makes the architect design the system in order to be able to manage probable changes. The Principle of Loose Coupling allows the system to be ready for change: updated in functionalities can be concentrated only over the identified building blocks.

Separation of Concerns: one of the most important uses of this Principle (tightly coupled with the Loose Coupling Principle) is to support modularization. This primarily means identifying parts of a software system responsible for specific concerns, aspects or tasks and encapsulating them as separate building blocks. As can be seen in Section 5.1.1, this Principle has already been applied in the first functional decomposition of the PANOPTESEC System and it will be kept in mind during the overall design of the architecture from the System level to the sub component level.

Information Hiding: this Principle is a fundamental concept for structuring and understanding complex systems. This Principle (when applied to software architecture) states that a building block should show to client building blocks ONLY that part of information that is really necessary for the client's tasks. This brings to the definition of well-designed interfaces between components. This Principle is strictly coupled with the Loose Coupling Principle.

Consistency: this Principle states that architecture should follow a standard set of rules from beginning to end: naming convention, communication of the system building blocks, structure of the interfaces, etc.
Some Architecture sub-Principles are directly related to the previously mentioned Principles and need to be taken into account during the development of the PANOPTESEC System design:

Explicit Interfaces: this sub-principle states that a system building block should clearly show which other building blocks it communicates with.

Separation of interfaces and Implementation: Interfaces should be described separately from the implementation so that the interface client can rely on the interface without knowing the implementation details.

Language support for abstractions: this sub-principle states that both the design and the programming languages should support the definition of the architectural abstractions (interfaces and components, for example).

Architectural Patterns and Tactics

While the Architectural Principles describe wide guidelines that help the architect developing system's design, Tactics and Patterns offer more practical solutions to architectural issues raised by the Non-Functional ASRs that drive the architecture. Chosen Patterns and Tactics have to respect the Architectural Principles previously identified.

Architectural Patterns: An *Architectural Pattern* is a package of design decisions that is found repeatedly in practice, has known properties that permit reuse and describes a class of architectures. Architectural Patterns are commonly used in order to fulfil a set of Non-Functional ASRs related to different but related Quality Attributes.

As defined by [Coplien], "*each pattern is a three-part rule that expresses the relationship between a certain context, a certain system of forces which occurs repeatedly in that context, and a certain software configuration which allows these forces to resolve themselves*".

For a complex system, it is not uncommon to rely on multiple architectural patterns: in order to maintain cohesion and consistency, the architect will always keep in mind selected Architectural Principles. Each pattern is meant to solve a set of architectural problems and to enhance a specific set of quality attributes. Each pattern brings solution to the identified needs along with a possible set of weaknesses that must be evaluated by the architect on behalf of the Non-Functional ASRs and the relative quality attributes. In case of uncertainty regarding the choice of the patterns, the architect must also evaluate the first developed functional architecture in Figure 48.

Architectural Tactics: An *Architectural Tactic* has a lesser scope than a Pattern. A Tactic is meant to solve a particular architectural issue related to a single Quality Attribute (so, related to one or more Non-Functional Requirements). It is common to choose one or more Patterns in order to design the global system and then mitigate possible weaknesses in some Quality Attributes by using Tactics.

From the Non-Functional ASRs analysis a particular emphasis on some specific Quality Attributes will emerge including:

- *Compatibility;*
- *Maintainability;*
- *Portability; and*
- *Reliability*

In order to fulfil these Quality Attributes and to match with the identified Architectural Principles it is necessary to apply a set of compliant Patterns [SAiP], [SWA], [Harrison].

For each of the identified applicable Patterns an overview and a rationale are presented in [D3.1.1] Annexes.

Architectural Patterns evaluation in PANOPTESEC Architecture

The of the top priority Non-Functional Requirements enlightens the need for the PANOPTESEC System to be composed of modular components, following the set of Architectural Principles previously identified.

Communications between components need to be lightweight and transparent for the developers and architects of functional components, while a consistent set of interfaces need to be designed and implemented between components. These constraints, of course, are also justified by the multiple and loosely couple organizations within the Consortium. In addition, a modular architecture will allow easier testing for single functionalities (but also it will add a level of complexity with the integration tests and so the need to use a continuous integration framework arises) and, possibly, different recombination of the prototype (for example, it is possible to envision a proactive only version of the PANOPTESEC System).

With respect to the first High Level Functional Architecture, **RLB011** gains particular importance (along with **RBL010** that is less important but directly related). Since the system is loosely coupled but both proactive and reactive chains need to use the same, consistent, data set from the Monitored System among different components that are both data consumers and producers, the need for a control loop mechanism that gives feedback to the Data Collection System arises in order to drive the collection of a new set of data, or to a set of data synchronization features among the components.

Since Performance is not at top priority among the Non-Functional ASRs, a feedback mechanism at the end of the proactive and reactive chain seems to be a reliable solution for managing this possible issue.

Given all these premises, no single Architectural Pattern is able to fulfil all these Non-Functional ASRs, except the Service Oriented Architecture (SOA) Pattern. This pattern is also suggested in the PANOPTESEC Project Description of Work. However, the use of a complete Service Oriented Architecture is considered too complex in order to give the PANOPTESEC System Prototype the desired functionalities. The orchestration mechanisms needed in order to create the proactive and reactive chains, including feedback, can be quite difficult to be described and to be updated. In addition, one of the main benefits of a SOA is the possibility to add a service or component in a later stage of development, or after the release of the product, with a minimal effort (*Service Discovery*). Analysing the First Functional Architecture, a stable set of needed functionalities are already identified and stated and are not anticipated to be augmented during the life of the project. However, the flexibility to add new services in the future supports the evolution of the PANOPTESEC system as part of the future exploitation of the product. Furthermore, one of the main

benefits of a SOA is *Interoperability* (the ability of a service to be invoked by any potential client of the service), is not a priority among the PANOPTESEC System Quality Attributes.

As a result of this summarized analysis, a pure Service Oriented Architecture Pattern will not be relied upon in order to develop the PANOPTESEC System prototype. Instead, a combination of a **Pipe-and-Filter Pattern** and a **Broker Architecture Pattern** will be used. The Pipe-and-Filter Pattern fits with the Functional Architecture and with **RLB011** wherein some of the components of the System rely of the results of the computation of previous modules on the proactive or reactive chain of risk management. The Broker Architecture Pattern fits with most of the Non-Functional top priority ASRs. The communications between Filters will be managed by the Broker. The technology choice in order to apply the Broker will require the possibility to easily implement orchestration mechanisms.

These Patterns are applied taking in consideration the possibility to add a **Service Oriented Architecture Pattern** among some components in order to ease the integration of possible services not developed using Java language.

The **Shared-Data Pattern** is also used in order to fulfil the Non-Functional implications over the Functional ASR **ICA006**. Due to possible data synchronization issues in the proactive and reactive chains, a wider application of the **Shared-Data Pattern** can be a useful solution.

The **Publish-Subscribe Pattern** does not offer the Reliability Quality Attributes that the PANOPTESEC System must fulfil, but the Pattern will be evaluated for limited applications on the communication between specific components.

All these selected Patterns fulfil the identified Architectural Principles.

The Integration Framework

One of the main consequences related to the evaluation of the top-priority Non-Functional ASRs and the choice of the Architectural Patterns is the need for the PANOPTESEC System to be able to implement the chosen Broker Architecture, to have the capability of building the Pipe-and-Filter chains (possibly by using orchestration mechanisms within the technology used in order to implement the Broker) and to use Service Oriented Principles if needed.

This issue can be solved by building an Integration Framework. Integration Frameworks can be used to integrate different technologies, applications and products without needing to write a lot of glue code. Connectors, implicit type converters, domain specific languages and Enterprise Integration Patterns are already implemented in the framework.

Some other benefits of an Integration Framework include:

- More lightweight than an Enterprise Service Bus (that are usually based on Service Oriented Architectures);
- Can be attached as simple libraries (usually, Java based and so perfectly interoperable among different platforms and different Operating Systems) into a project (for example, in Eclipse);

- Are open source and usually perfectly integrated with Java applications;
- Feature great flexibility.

Integration Frameworks are also beneficial because they reduce complexity in intricate integrations. They support architects and developers by realizing routing, connectivity, error handling and testability.

ANNEX D: DESIGN COVERAGE OVER REQUIREMENTS

In this annex a complete list of Functional and Non-Functional Requirements is presented along with the elements of the High Level Design that covers a particular Requirement. This is quite easily obtained for the Functional Requirements. In Section 5.2 the Functional Design elements have been depicted with emphasis on the Requirements driving the functionalities that they will have to implement. The table will then show which Design Elements will have to cover the requested functionality.

Due to the nature of Non-Functional Requirements, which often encompass a single Design Element, a rationale of the architecture coverage is given. This is provided along with, when possible, a description of the application of Architecture Tactics adopted in order to meet the Requirement. However, some Non-Functional Requirements will be fulfilled by some Design Elements: usually, the Integration Framework.

These tables are directly imported from the SysML Design and Requirements Projects.
ASRs of all level of importance are identified in the matrixes, for the Functional Requirements. Non-Functional Requirements are all ASRs.

Table 4: Design coverage over Data Collection Functional Requirements

Id	Design Elements	Architecturally Significant Requirement
DSC001	[Data_Collection_Interface]	X
DSC002	[Data_Collection_Interface]	X
DSC003	[Data_Collection_Interface]	X
DSC004	[Data_Collection_Interface]	X
DSC005	[Data_Collection_Interface, Repository]	X
DSC006	[Device, Monitored_Data, Data_Collection_Interface]	X
DSC007	[ICT_Device, Device, Data_Collection_Interface]	X
DSC008	[Data_Collection_Interface, Device, ICT_Device, OperatingSystem]	
DSC009	[Data_Collection_Interface, Device, ICT_Device, Driver]	
DSC010	[Data_Collection_Interface, Device, ICT_Device, RootCertificate]	
DSC011	[Data_Collection_Interface, Device, ICT_Device, Firmware]	
DSC012	[Data_Collection_Interface, Device, ICT_Device, Service]	
DSC013	[Data_Collection_Interface, Device, ICT_Device, Identifier]	
DSC014	[Reachability_Matrix, Network_Inventory, Data_Collection_Collector, Data_Collection_Interface, Reachability_Matrix_Correlator]	X

DSC015	[Data_Collection_Interface, Device, ICT_Device, User, Permission]	
DSC016	[Data_Collection_Interface, Device, ICT_Device, User, Permission]	
DSC017	[Data_Collection_Interface, Abstract_Default_Security_Policy]	X
DSC018	[SCADA_Device, Device, Data_Collection_Interface]	X
DSC019	[SCADA_Device, Device, Data_Collection_Interface, Service]	
DSC020	[SCADA_Device, Device, Data_Collection_Interface, Firmware]	
DSC021	[SCADA_Device, Device, Data_Collection_Interface, Identifier]	
DSC022	[Data_Collection_Interface, Abstract_Default_Security_Policy]	
DSC023	[Business_Mission_Information, Data_Collection_Interface]	X
DSC024	[Business_Mission_Information, Data_Collection_Interface]	
DSC025	[Business_Mission_Information, Data_Collection_Interface, Visualization]	X
DSC026	[Mitigation_Action, Data_Collection_Collector]	X
DSC027	[Mitigation_Action, Data_Collection_Collector, Visualization]	X
DSC028	[Vulnerability_Advisory_Database_Information, Data_Collection_Interface]	X
DSC029	[Normalized_Alert, Monitored_Data, Data_Collection_Interface]	X
DSC030	[Data_Collection_Interface, Device]	X
DSC031	[Device, Data_Collection_System, Identifier, Location]	X
DSC032	[Reachability_Matrix, Network_Inventory, Data_Collection_Collector, Data_Collection_Interface, Reachability_Matrix_Correlator]	X
DSC033	[]	

Table 5: Design coverage over Information Correlation and Abstraction Functional Requirements

Id	Design Elements	Architecturally Significant Requirement
ICA001	[Data_Collection_Collector, Reachability_Matrix_Correlator, Low_Level_Correlator]	X

ICA002	[Data_Ontology, Reachability_Matrix_Correlator]	X
ICA003	[Data_Collection_Interface]	X
ICA004	[Data_Collection_Interface]	X
ICA005	[Reachability_Matrix, Reachability_Matrix_Correlator]	X
ICA006	[Data_Collection_Collector, Repository]	X
ICA010	[Mission_Impact_Module]	X
ICA011	[Mission_Impact_Module, Mission_Graph]	X
ICA012	[Mission_Impact_Module, Mission_Graph]	X
ICA013	[Mission_Impact_Module, Mission_Graph]	X
ICA014	[Mission_Impact_Module, Mission_Graph]	X
ICA015	[Mission_Impact_Module, Mission_Graph]	X
ICA016	[Mission_Impact_Module, Mission_Graph]	
ICA017	[Mission_Impact_Module, Mission_Graph]	
ICA018	[Mission_Impact_Module, Mission_Graph]	
ICA019	[Mission_Impact_Module, Mission_Graph]	
ICA020	[Mission_Impact_Module, Mission_Graph, Operational_Impact]	X

Table 6: Design coverage over Proactive Response System Functional Requirements

Id	Design Elements	Architecturally Significant Requirement
PRS001	[Attack_Graph_Generator, Risk_Quantifier, Strategic_Response_Decider]	X
PRS002	[Attack_Graph_Generator, Attack_Graph]	X
PRS003	[Attack_Graph_Generator, Attack_Graph]	X
PRS004	[Attack_Graph_Generator, Attack_Graph, Reachability_Matrix]	X
PRS005	[Attack_Graph_Generator, Attack_Graph, Reachability_Matrix]	X
PRS006	[Reachability_Matrix, Vulnerability_Inventory, Attack_Graph_Generator, Attack_Graph]	X
PRS007	[Risk_Quantifier, Risk_Profile, Attack_Graph]	X
PRS008	[Risk_Quantifier, Risk_Profile, Mission_Graph]	X
PRS009	[Risk_Quantifier, Risk_Profile]	X
PRS010	[Risk_Quantifier, Risk_Profile, Response_Plan, Strategic_Response_Decider]	X
PRS011	[Risk_Quantifier, Risk_Profile, Response_Plan, Operational_Impact, Strategic_Response_Decider]	X
PRS012	[Mitigation_Action, Strategic_Response_Decider]	X

PRS013	[]	
PRS014	[Response_Plan, Strategic_Response_Decider, Visualization]	X
PRS015	[Response_Plan, Strategic_Response_Decider, Visualization]	X
PRS016	[Response_Plan, Strategic_Response_Decider]	X
PRS017	[Response_Plan, Strategic_Response_Decider, Abstract_Default_Security_Policy, Abstract_Response_Policy_Context]	X
PRS022	[Response_Plan, Policy_Deployer]	X

Table 7: Design coverage over Reactive Response System Functional Requirements

Id	Design Elements	Architecturally Significant Requirement
RRS001	[High_Level_Online_Correlator, Risk_Quantifier, Potential_Attack_Identifier, Tactical_Response_Decider]	X
RRS002	[Risk_Quantifier, Mission_Graph, Risk_Profile]	X
RRS003	[Risk_Quantifier, Risk_Profile, Instantiated_Attack_Path]	X
RRS004	[Risk_Quantifier, Risk_Profile, Scored_Vulnerability_Inventory, Attack_Graph]	X
RRS005	[Risk_Profile, Response_Financial_Impact_Assessor]	X
RRS006	[Risk_Quantifier, Risk_Profile, Scored_Vulnerability_Inventory]	X
RRS007	[Instantiated_Attack_Graph, Risk_Quantifier, Risk_Profile]	X
RRS008	[Mission_Graph, Risk_Quantifier, Risk_Profile]	X
RRS009	[Risk_Quantifier, Risk_Profile, Response_Plan, Tactical_Response_Decider]	X
RRS010	[Risk_Quantifier, Risk_Profile, Response_Plan, Tactical_Response_Decider, Operational_Impact]	X
RRS011	[Mitigation_Action, Tactical_Response_Decider]	X
RRS012	[]	
RRS013	[Response_Plan, Tactical_Response_Decider, Visualization]	X
RRS014	[Response_Plan, Tactical_Response_Decider, Visualization]	X
RRS015	[Response_Plan, Tactical_Response_Decider]	X
RRS016	[High_Level_Online_Correlator, Risk_Quantifier, Potential_Attack_Identifier]	X
RRS017	[Instantiated_Attack_Path, Risk_Quantifier, Enriched_Attack_Graph, Attack_Graph_Threat_Baseline]	X
RRS018	[]	
RRS023	[Potential_Attack_Identifier, New_Entry_Point_Or_Target]	X
RSS024	[Response_Plan, Policy_Deployer]	X
RRS025	[High_Level_Online_Correlator]	X

RRS026	[High_Level_Online_Correlator, Instantiated_Attack_Path]	X
RRS027	[High_Level_Online_Correlator]	X

Table 8: Design coverage over Visualization System Functional Requirements

Id	Design Elements	Architecturally Significant Requirement
VIZ001	[Visualization]	X
VIZ002	[Risk_Profile, Visualization]	X
VIZ003	[Risk_Profile, Visualization, Attack_Graph, Mission_Graph]	X
VIZ004	[Risk_Profile, Visualization]	X
VIZ005	[Risk_Profile, Visualization, Mission_Graph, Instantiated_Attack_Graph]	X
VIZ006	[Mission_Graph, Visualization]	X
VIZ007	[Network_Inventory, Visualization]	X
VIZ008	[Network_Inventory, Visualization, Reachability_Matrix]	X
VIZ009	[Vulnerability_Advisory_Database_Information, Visualization, Vulnerability_Inventory, Scored_Vulnerability_Inventory, Network_Inventory]	X
VIZ010	[Attack_Graph, Visualization]	X
VIZ011	[Normalized_Alert, Visualization]	X
VIZ012	[Response_Plan, Visualization, Strategic_Response_Decider]	X
VIZ013	[Response_Plan, Visualization]	X
VIZ014	[Response_Plan, Visualization, Tactical_Response_Decider]	X
VIZ015	[Response_Plan, Visualization]	X
VIZ016	[Response_Plan, Visualization]	X
VIZ017	[]	
VIZ018	[Business_Mission_Information, Visualization]	X
VIZ019	[Business_Mission_Information, Visualization]	
VIZ020	[]	
VIZ021	[Reachability_Matrix, Network_Inventory, Vulnerability_Inventory, Scored_Vulnerability_Inventory, Visualization]	X
VIZ022	[Reachability_Matrix, Network_Inventory, Vulnerability_Inventory, Scored_Vulnerability_Inventory, Visualization]	
VIZ023	[Visualization, Data_Collection_Collector, Repository]	X
VIZ024	[Visualization, Data_Collection_Collector]	X
VIZ025	[Visualization]	
VIZ026	[Visualization, Policy_Deployer]	X
VIZ027	[Visualization]	X

VIZ028	[]	X
VIZ029	[Reachability_Matrix, Visualization]	X
VIZ030	[]	

Table 9: Coverage over Compatibility Non-Functional Requirements

Id	Rationale	Design Elements
CMP001	PANOPTESEC High Level Design in D3.1.2 clearly shows that the PANOPTESEC System is implemented in self-contained functional Blocks. Adopted Architectural Patterns (Pipe-and-Filters, Broker Architecture, Service Oriented Architecture), along with the use of an Integration Framework, satisfy this Requirement. **SATISFIED**.	[Integration_Framework]
CMP002	The Integration Framework selected technological stack, with the Broker Architecture Pattern chosen for managing most of the communications between PANOPTESEC components, satisfies this Requirement. **SATISFIED**.	[Integration_Framework]
CMP003	The Integration Framework Component Composition Layer and the Integration Framework Communication Layer will manage all communications between different functional components of the PANOPTESEC System. Data access to collected data (for example, in persistency) will then benefit of these features. **SATISFIED**.	[Integration_Framework]
CMP004	The Integration Framework technological stack chosen is able to manage different Enterprise Integration Patterns. **SATISFIED**.	[Integration_Framework]
CMP005	**SATISFIED**.	[Data_Collection_Interface]
CMP006	**SATISFIED**.	[Data_Collection_Interface]
CMP007	The Integration Framework selected technological stack, with the chosen Broker Architecture Pattern for managing most of the communications between PANOPTESEC components, satisfies this Requirement. Components will be able to be deployed in a common, distributed OSGi environment. **SATISFIED**.	[Integration_Framework]
CMP008	The Integration Framework selected technological stack have the capability of managing the runtime (start, stop, fault, recovery) of the integrated components. Every component directly integrated in the Integration Framework will be available for runtime management by a PANOPTESEC System Administrator. **SATISFIED**.	[Integration_Framework]
CMP009	Apache Camel, Apache ActiveMQ, ZeroMQ and Apache CXF (messaging protocols and brokers that will be widely used for communications among integrated components, are XML/JSON compliant. **SATISFIED**.	[Integration_Framework]

Table 10: Coverage over Maintainability Non-Functional Requirements

Id	Rationale	Design Elements

MNT001	PANOPTESEC High Level Design in D3.1.2 clearly shows that the PANOPTESEC System is implemented in self-contained functional Blocks. Adopted Architectural Patterns (Pipe-and-Filters, Broker Architecture, Service Oriented Architecture), along with the use of an Integration Framework, satisfy this Requirement. **SATISFIED**.	[Integration_Framework]
MNT002	The High Level Data Model depicted in [D3.1.2] is sufficiently generic to be adaptable to new data sources. The Low Level Data Model will have to be modeled in order to depict generic ICT/SCADA concepts and semantics and allow the integration of new data sources. **PARTIALLY SATISFIED**.	[]
MNT003	The Integration Framework selected technological stack have the capability of managing the configuration of the integrated components. Every component directly integrated in the Integration Framework will be available for configuration management by a PANOPTESEC System Administrator. **SATISFIED**.	[Integration_Framework]
MNT004	During the integration phase, acceptance tests will have to be developed and implemented in order to ensure the complete functionalities of the PANOPTESEC System with respect to the Requirements. **NOT SATISFIED**.	[]

Table 11: Coverage over Performance Non-Functional Requirements

Id	Rationale	Design Elements
PRF001	In order to verify the Requirements, a Test Case is designed: The following test involving real users is planned. Starting from a stable configuration of the whole system some changes (e.g., a new vulnerability, a new device, a new entry point in the network etc…) will be injected. Once such changes are in place, the operator will be asked to work as he/she is currently doing (in absence of the PANOPTESEC System) and the time taken to respond to changes will be measures. Then, the Simulation Environment will be reset to the same initial stable configuration, and the same changes will be injected and the reaction time of the operators with the PANOPTESEC System running will be measured. Finally, the coverage of the requirement by comparing the results obtained with and without the PANOPTESEC System will be assessed. **PARTIALLY SATISFIED**.	[]
PRF002	In order to verify the Requirements, a Test Case is designed: The test will proceed by running a test in parallel on two copies of the Simulation Environment: the first running the PANOPTESEC System and the second without PANOPTESEC System. In both the environments the same set of alerts containing real attack traces and a known percentage of false positive and false negative will be injected. Then, the number of incidents/alerts reported from both the simulation environment will be computed and results will be compared with the expected number of real attack contained in the traces. **PARTIALLY SATISFIED**.	[]

PRF003	In order to verify the Requirements, a Test Case is designed: The test will proceed by running a test in parallel on two copies of the Simulation Environment: the first running the PANOPTESEC System and the second without PANOPTESEC System. In both the environments the same set of alerts containing real attack traces and a known percentage of false positive and false negative will be injected. Then, the number of incidents/alerts reported from both the simulation environment will be computed and results will be compared with the expected number of real attack contained in the traces. **PARTIALLY SATISFIED**.	[]
PRF004	In order to verify the Requirements, a Test Case is designed: A set of parameters (size of the Monitored System, the topology, the event rate and the number of vulnerabilities) will be defined. This set of parameters, combined together, will provide a wide range of test cases from which performance metrics will be collected. Once the values at these points are obtained, a regression function will be computed that should provide an assessment of the scalability capability of the PANOPTESEC System. **PARTIALLY SATISFIED**.	[]
PRF005	At the High Level Design stage, it is impossible to evaluate CPU/Memory consumption of the PANOPTESEC System components. Due to the fact that many components (Attack Graph Generator, for example) need to compute a complete mapping and reachability of each Device of the Monitored System(s), it can be envisioned that memory consumption will increase with the number of Devices of the Monitored System(s) and the complexity of their routing tables/firewalling rules. It will be possible to make more precise evaluations during the first development phase. **NOT SATISFIED**.	[]
PRF006	In order to verify the Requirements, a Test Case is designed: The following test involving real users is planned. Starting from a stable configuration of the whole Monitored System an attack will be injected. Once the attack is in place, the operator will be asked to work as he/she is currently doing (in absence of the PANOPTESEC System) system and the time taken to respond to changes will be measured. Then, with the PANOPTESEC Simulation Environment the same initial stable configuration will be started and the same attack will be injected and the reaction time with the PANOPTESEC System running will be measured. Finally, the coverage of the requirement will be assessed by comparing the results obtained with and without the PANOPTESEC System. **PARTIALLY SATISFIED**.	[]
PRF007	In order to verify the Requirements, a Test Case is designed: The following test involving real users is planned. Starting from a stable configuration of the whole Monitored System an attack will be injected. Once the attack is in place, the operator will be asked to work as he/she is currently doing (in absence of the PANOPTESEC System) system and the time taken to respond to changes will be measured. Then, the Simulation Environment will	[]

	be started ing the same initial stable configuration and the same attack will be injected and the reaction time with the PANOPTESEC System running will be measured. Finally, the coverage of the Requirement will be assessed by comparing the results obtained with and without the PANOPTESEC System. **PARTIALLY SATISFIED**.	
PRF008	Possible deployable Mitigation_Action(s) are pre-configured by the Security Operator. The Security Operator acts as decision-maker in the proactive and reactive response. Specific components (Mission Impact Module, Strategic and Tactical Response Deciders) have the explicit task of evaluate the impact on the Monitored System(s) of every proposed Response_Plan. A certain degree of uncertainty remains: a set of Test Cases is envisioned in order to prove that the PANOPTESEC System will fulfil this Requirement. This Requirement is considered **PARTIALLY SATISFIED**	[Mitigation_Action]

Table 12: Coverage over Portability Non-Functional Requirements

Id	Rationale	Design Elements
PRT001	This Requirement is a constraint for design and development of each building block of the PANOPTESEC System. Most of the PANOPTESEC components will be developed in Java, other in Python and the Integration Framework technological stack selected is developed in Java. **SATISFIED**.	[]
PRT002	This Requirement is a constraint for design and development of each building block of the PANOPTESEC System. The Integration Framework is able to adopt Service Oriented Architecture typical standards (e.g. Web Services) in order to improve interoperability with Non-Java Components. **SATISFIED**.	[Integration_Framework]
PRT003	The Integration Framework chosen technological stack can be installed both in Windows and Linux environments. The Java language, that is be adopted as the preferred developing language for the PANOPTESEC System, runs in many different OS. Still there is some degree of uncertainty related to the needs of specific components; this Requirement is considered **PARTIALLY SATISFIED**.	[]
PRT004	PANOPTESEC High Level Design in D3.1.2 clearly shows that the PANOPTESEC System is implemented in self-contained functional Blocks. Adopted Architectural Patterns (Pipe-and-Filters, Broker Architecture, Service Oriented Architecture), along with the use of an Integration Framework, possibly satisfy this Requirement. There is a possible risk related on the dependences on data inputs by some components. Each component should be designed and implemented in order to work in its computation without the need of relying on specific data coming from other components, or, at least, to reduce these dependences. For example, the Proactive Response System should be able to work without the Mission Impact Module. **PARTIALLY SATISFIED**.	[Integration_Framework]
PRT005	D3.1.1, D3.1.2, D4.1.2, D5.1.2 and D6.1.2 offer a clear description	[Integration_Fr

	of a preliminary version of the interfaces of each building block. Chosen Design methodology descripted in D3.1.1 put consistency at a top priority and the definition of a High Level Data Model (strictly related to the Interfaces) shared across all the Consortium ensures that consistency is maintained of each of the PANOPTESEC Design projects. **SATISFIED**.	amework]
PRT006	The Integration Framework selected technological stack satisfies this Requirement. Components will be able to be deployed in a common, distributed OSGi environment. **SATISFIED**.	[Integration_Framework]

Table 13: Coverage over Reliability Non-Functional Requirements

Id	Rationale	Design Elements
RLB001	The technological stack chosen for the Integration Framework is able to implement Integration and Component Tests. A Continuous Integration Framework will be used in order to integrate the functional components and test the integration. **PARTIALLY SATISFIED.**	[Integration_Framework]
RLB002	The technological stack chosen for the Integration Framework is able to implement Integration and Component Tests. A Continuous Integration Framework will be used in order to integrate the functional components and test the integration. **PARTIALLY SATISFIED**.	[Integration_Framework]
RLB003	The technological stack chosen for the Integration Framework is able to implement Integration and Component Tests. A Continuous Integration Framework will be used in order to integrate the functional components and test the integration. **PARTIALLY SATISFIED**.	[Integration_Framework]
RLB004	The Data-Shared Pattern is applied to any data set produced by the PANOPTESEC: data are persisted in the Repository by the Data Collection Collector component. The runtime operations of every component integrated in the Integration Framework will be managed by the Integration Framework itself, which will be able to start other instances of a faulted component (adoption of the Architectural Tactic of "Recover from Faults"). Since it is possible that not every component will be integrated, this Requirement is **PARTIALLY SATISFIED**.	[Repository, Integration_Framework]
RLB005	Chosen Repository technologies will need to be redundant. This Requirement is **PARTIALLY SATISFIED** (adding redundancy is a feasible goal, but it would cause the deployment of an additional – may be virtual- machine. Due to possible budget constraints, it will have to be evaluated).	[Repository]
RLB006	Repositories will be easily backed-up on a daily bases. **SATISFIED**.	[Repository]
RLB007	The PANOPTESEC System will have to be deployed on a dedicated VLAN/IP Domain, in order to not interfere with the Monitored System(s) in case of fault. **SATISFIED**.	[]
RLB008	This Requirement is a constraint for the deployment and	[Integration_Fr

	integration. Runtime management of the components will be managed by the Integration Framework technologies, but recovery time will depend on the functionalities of the integrated components. Recovery after faults will be tested in order to ensure recovery time within the constraints of the Requirement. **PARTIALLY SATISFIED**.	amework]
RLB009	The PANOPTESEC System will have to be deployed on a dedicated VLAN/IP Domain, in order to not interfere with the Monitored System(s). IDS/Firewalls in the Monitored System(s) will be configured in order to gain knowledge of the presence of active Data Collection Processors components/active Policy Deployer components. **SATISFIED**.	[]
RLB010	This Requirement is **PARTIALLY SATISFIED** (it is satisfied only by the proactive chain. The reactive chain works on the most recent alerts, asynchronous with the proactive chain. Due to the architecture of the reactive chain, no racing condition can be envisioned).	[Proactive_Chain]
RLB011	This Requirement is fulfilled by the proactive chain: **Control Loop 03** allows the proactive chain to process the most recently correlated topology/vulnerability snapshot, when it ends a complete computation. The reactive chain is asynchronous with the proactive one and works on the most recent set of alerts collected by the Data Collection Interface. **SATISFIED**.	[Proactive_Chain, Reactive_Chain]
RLB012	Since data consistency and overall Reliability are a major concern in the Non-Functional Requirements, in the High Level Design the proactive chain is entangled with a control loop (**Control Loop 03**) between SRD and DCC, in order to allow every proactive component to work on the same data set, avoiding inconsistencies and racing conditions. **SATISFIED**	[]
RLB013	A Control Loop (**Control Loop 02**) is identified between the Proactive and the Reactive chains and the Monitored System(s). Every Enforced Mitigation Action that the PANOPTESEC System would deploy in the Monitored System(s) will change the topology/vulnerability status of the Monitored System(s). This change of status will be perceived by the followings topology/vulnerabilities data collection. **SATISFIED**.	[]
RLB014	Since data consistency and overall Reliability are a major concern in the Non-Functional Requirements, in the High Level Design the proactive chain is entangled with a control loop (**Control Loop 03**) between SRD and DCC. This will lock PANOPTESEC System proactive performances to, at least, to the sum of the computation time of each single component in the chain. In addition, some components will remain inactive until the chain is not ended (this is acceptable because Reliability ASRs have a higher priority than Performance ASRs). By applying the Data-Shared Pattern over the **Control Loop 01** identified in [D3.1.1] it is possible to decouple the Data Collection and Correlation processes from the proactive chain, but more research must be conducted in order to reduce	

	the performance loss caused by this Control Loop. Another risk can be related to **Control Loop 02** (between SRD/TRD-PDP-the Monitored System(s) and DCI. In theory, every Mitigation Action enforced by the proactive and reactive chains should immediately affect the Monitored System and should be directly perceived by the PANOPTESEC System (giving the Security Operator an immediate feeling about the adopted solution to a proactive or reactive solution). This hypothetical situation could cause possible racing conditions and affect the overall stability of the System, if the System will be designed in order to stop every topology/vulnerability/proactive running computations any time a change is perceived, in order to start a new computation. This is not the case of the actual design. Even in the case of immediate perception of updates in the Monitored System(s), no racing conditions caused by a change in the Monitored System(s) could occur in the PANOPTESEC System. **SATISFIED**.	

Table 14: Coverage over Security Non-Functional Requirements

Id	Rationale	Design Elements
SEC001	The choice of the Repository will encompass the need of encryption (but in the operational context it will be possible that the performances overhead behind encryption would be unsustainable). **SATISFIED**.	[Data_Collection_Collector, Repository]
SEC002	This Requirement is partially contrasted with the chosen Architectural Patterns for the PANOPTESEC System. Since a communication broker is used, service interfaces will be exposed. Since the PANOPTESEC System will be possibly deployed over multiple machines, ensuring this Requirement may be complex. The PANOPTESEC System will have then to be deployed on a dedicated VLAN/IP Domain. **PARTIALLY SATISFIED**.	[]
SEC003	The Integration Framework may rely on use of a Single Sign On Server in order to manage authentication on communications between components. A Single Sign-On Server has been added to the Design. **SATISFIED**.	[Integration_Framework, Single_Sign_On_Server]
SEC004	The Integration Framework Messaging Broker showed in the technological stack supports SSL encryption over exchanged messages. Where the Integration Framework will have the need to use Web Services, it will rely on SOAP and REST Encryption. **SATISFIED**.	[Integration_Framework]
SEC005	The PANOPTESEC System may rely on use of a Single Sign On Server in order to manage User authentication. **SATISFIED**.	[Single_Sign_On_Server]
SEC006	The production version of the PANOPTESEC system will be deployed in an operational environment involving sensitive cyber security information. Consequently, the production version should be evaluated against the Common Criteria for Information Technology	[]

	Security Evaluation (Common Criteria - www.commoncriteriaportal.org) with an Evaluated Assurance Level 2 (EAL2). The Common Criteria provides and standard (ISO/IEC15408-1) for security evaluation of products. Common Criteria provides a structured means for independent verification of the security assurance of a delivered product. EAL2 is applicable in those circumstances where users may require a moderate level of independently assured security. Given the nature of the development environment for PANOPTESEC, involving partners having varied skills and capability for structured development, it is impractical to target a higher assurance level as part of the FP7 project. Also, Common Criteria is not intended to alleviate all other security concerns, therefore, it is anticipated that additional security may be required in some customer environments that require a higher assurance level. In accordance with the Common Criteria for EAL2, several items resulting from a structured requirements process, design and testing environment are required. The structured design methodology provided by WP3 including use of SysML as a modelling language, use of Papyrus as a design modeling tool, use of a structured process for requirements management and traceability, and use of a defined configuration management process support the needs of an EAL2 evaluation. Please refer to Page 35: https://www.commoncriteriaportal.org/files/ccfiles/CCPART3V3.1R4.pdf. **SATISFIED**.	

Table 15: Coverage over Usability Non-Functional Requirements

Id	Rationale	Design Elements
USG001	Visualization will be based on Web technologies (HTML/Javascript) that will allow easier development of different Views for different Actors. The current design for Visualization already satisfies the Requirements. **SATISFIED**.	[Visualization]
USG002	Visualization development team provided regular feedback with the Consortium and the User Agency on behalf of the current state of the user interface of the PANOPTESEC, in order to validate the frontend during the development. This regular feedback will continue during the next phases of the Project. **SATISFIED**.	[]
USG003	Due to the technology stack used for the user interface (HTML/Javascript), due to the expertise of UROME in the domain, and due to the first mock-ups and prototypes tested by WP3, it is expected that this Requirements will be **SATISFIED** in the final version.	[Visualization]