PROJECT DELIVERABLE REPORT

Introducing advanced ICT
and Mass Evacuation Vessel design
to ship evacuation and rescue systems

D7.7 WP7 Public release

A holistic passenger ship evacuation and rescue ecosystem
MG-2-2-2018
Marine Accident Response

“This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 814962”

Document Information

Grant Agreement Number	814962	Acronym	PALAEMON	
Full Title	A holistic passenger ship evacuation and rescue ecosystem			
Topic	MG-2-2-2018: Marine Accident Response			
Funding scheme	RIA - Research and Innovation action			
Start Date	1stJUNE 2019	Duration	36 months	
Project URL	www.palaemonproject.eu			
EU Project Officer	Georgios CHARALAMPOUS			
Project Coordinator	AIRBUS DEFENCE AND SPACE SAS			
Deliverable	D7.7 WP7 Public Release			
Work Package	WP7 – Public Release			
Date of Delivery	Contractual	M36	Actual	M44
Nature	R- Report	**Dissemination Level**	PU- Public	
Lead Beneficiary	Atos Spain S.A.			
Responsible Author	Julia Ruiz	**Email**	julia.ruiz@atos.net	
		Phone	N/A	
Reviewer(s):	Manuel Ramiro (ADV), Alfredo Gardel (UAH)			
Keywords	Communications Platform, Security, Interoperability, Platform Releases, Testing and Results			

Authors List

Name	Organization
Julia Ruiz, David Gómez	ATOS

Revision History

Version	Date	Responsible	Description/Remarks/Reason for changes
0.1	2022/12/02	ATOS	Report write-up
0.2	2022/12/07	ATOS	Section 2
0.3	2022/12/14	ATOS	Section 3
0.5	2022/12/21	ATOS	Section 4
0.6	2023/01/11	ATOS	Section 5
0.7	2023/01/12	ATOS	Executive Summary, Intro + Conclusions
0.8	2023/01/12	ATOS	Version ready for internal review
0.9	2023/01/13	ATOS	Update with internal reviewers' feedback
1.0	2023/01/13	ATOS	Final version

Contents

List of Figures

Abbreviations

ACL	Access Control List
AR	Augmented Reality
DPA	Designated Person Ashore
DSS	Decision Support System
ETL	Extract, Transform and Load
ETSI	European Telecommunications Standardization Institute
HTTP	HyperText Transfer Protocol
ICT	Information and Communication Technologies
MEV	Massive Evacuation Vessel
PaMEAS	PALAEMON Mustering and Evacuation Process Automation System
PIMM	PALAEMON Incident Management Module
RPC	Remote Procedure Call
SRAP	Smart Risk Assessment Platform
VCS	Version Control Systems
VDES	VHF Data Exchange System
VHF	Very High Frequency
VM	Virtual Machine

Executive Summary

This deliverable compiles the main highlights of the activities carried out under WP7's umbrella. We use this report to outline the so-called PALAEMON Communications Platform, which represents how the different components interact to each other in a real deployment. Concerning the interplay mentioned before, we also introduce the information model defined to guarantee a holistic interoperability among data sources and smart evacuation services. Before this development phase was done, the consortium made an agreement that nailed down the principal software development and integration methodologies to follow. Finally, the document presents the outputs after validating the whole ecosystem in a cloud-based environment. After all this, the PALAEMON system aims to be handed to WP8 activities, where the platform will be duly deployed and tested over a real ship infrastructure (i.e., ANEK's Elyros Ro-Pax Ferry).

1 Introduction

> *DISCLAIMER: The reader must know beforehand that, due to an error in WP7's deliverables numbering, none of these documents match their actual sequence. Namely, D7.1 is D6.6, D7.3 is D7.2, D7.5 is D7.4 and D7.7 is D7.6. Moreover, it is worth highlighting that we reuse part of the content already present in those deliverables, keeping the confidential information aside.*

This deliverable compiles the main breakthrough generated across four of the main documents generated in the scope of this WP.

1. D7.1 – PALAEMON Communications Platform (v2)
2. D7.3 – Uniform Data Exchange Modules – Interoperability Layer (v2)
3. D7.5 – Software Development and Integration Methodology (v2)
4. D7.7 – Test cases and overall system testing results (v2)

As a matter of fact, whereas the latest document is a public document [1], the other three do have a confidential nature and we cannot explicitly reference them.

Before that, it is worth introducing before the main objectives pursued by this Work Package 7 (PALAEMON Integrated System and Technology Validation Trials) we are closing with this last deliverable. For the sake of illustration, we have abridged in Figure 1 the main challenges postulated in the official (i.e., Grant Agreement) description of the WP.

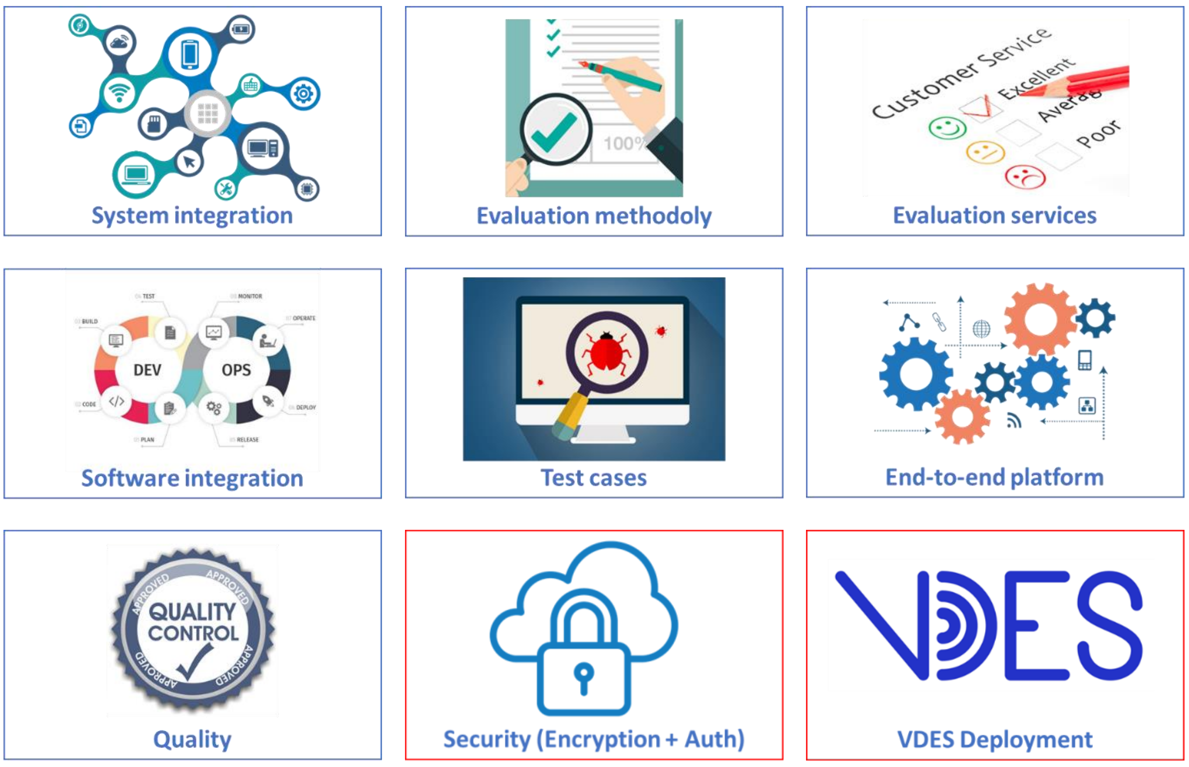


Figure 1. Work Package 7 Objectives in a nutshell

Basically, WP7 gathers all the developments carried out across the core technical ICT-ish WPs (WP3 – PALAEMON Intelligence Framework, WP5 – PALAEMON on-board mustering tools and services, WP6- PALAEMON Back-End Infrastructure). From this amalgam of devices and services and following the PALAEMON Reference Architecture defined in WP2 (Use Case Driven Requirements Engineering and Architecture), namely in D2.7 (PALAEMON

Architecture - v2) [2], we cope with the integration into a standalone communications platform, whose main objective is to bring about a smart evacuation process in the unlikely event of an incident, going beyond current evacuation operations. We have carried out all this integration process as part of T7.1 (PALAEMON Communications Platform).

It goes without saying that interoperability is one of the most common war horses when dealing with multi-owner solutions, which, by default, use proprietary information models to exchange information. To solve this lack of compatibility, we have relied on a timely ETSI (European Telecommunications Standardization Institute) standard, NGSI-LD [3], that guarantees that the data format follows common principles. This work was carried out as part of T7.3 (Uniform Data Exchange Modules – Interoperability Layer).

We have followed some timely and widespread software development and integration approaches, yielding an integrated end-to-end system that guarantees a reliable operation handling an evacuation. With this agreement, all components have been developed according to a handful of principles (e.g., open-source, micro-services, orchestration, etc.), ending up to not-so-painful final integration and testing stages. Alongside the breakthrough carried out from an ICT standpoint, all the platform operation and operation hinges around the concept of evacuation process management.

Finally, when the platform was ready, it is utmost necessary to perform a thorough validation campaign that evaluates all the possible combinatory of events, assessing that the whole PALAEMON system is reliable and can be ported to a real infrastructure.

We have to mention two tasks as well that are out of the scope of this report; however, they are essential parts within the whole list of objectives. On the one hand, the utilization of secure and safe communication protocols and interfaces, making sure that all the information that travels across the system is not compromised. Besides, we use personal data that must comply GDPR (General Data Protection Regulation) standards [4]. As a matter of fact, all this belongs to T7.2 (Encryption and Authentication Mechanisms). In addition, the design, manufacturing, development and testing of three prototypes of VDES (VHF Radio Exchange System) transceivers was framed within T7.4 (VDES Deployment).

The document is structured as follows. Section 2 presents the main aspects of the so-called PALAEMON Communications Platform. Section 3 outlines the information model we have come up with to guarantee that data sources and services are interoperable and can seamlessly speak with each other with no need of further translation or mapping. Section 4 nails down the software development and integration methodology defined during the initial phase of the project. Section 5 compiles the main validation results obtained after testing the whole platform in a cloud-based environment. Finally, Section 6 closes the document (and WP7) and bridge to WP8's activities (PALAEMON Application Field Trials, Evaluation and Outcomes), where PALAEMON System will be deployed and assessed over a real scenario (i.e., ANEK's Elyros Ro-Pax Ferry).

1.1 WP7 Relationship with other WPs

Figure 2 outlines the dependencies and connections between WP7 (PALAEMON Integrated System and Technology Validation Tool), within a red frame in the diagram, and the other Work Packages that shape the overall PALAEMON project. From a top-down approach, the process started in WP2 (Use Case Driven Requirements Engineering and Architecture), where, as main outputs, we came up with a thorough list of technical and non-technical requirements, beside a two-version approach of the so-called PALAEMON Reference Architecture. In parallel to this, and directly fed from WP2's breakthrough, we have the four core technical WPs, split into four different categories or complementary research areas: WP3 (PALAEMON Intelligence Framework – AI Services and Algorithms) brings an intelligent layer on top of raw data and traditional services. WP4 (PALAEMON Mass Evacuation Vessel), strives for the creation of a radical streamlining of Massive Evacuation Vessels (MEVs). WP5 (PALAEMON on-board mustering tools and services) focuses on the adoption of new and heterogeneous information sources (e.g., sensors, devices) and services to complement the shipboard legacy systems and help improve the evacuation procedures. WP6 (PALAEMON Back-End Infrastructure) aims to provide the core of the PALAEMON system, with modules and services that will be intensively used by the rest of the software modules.

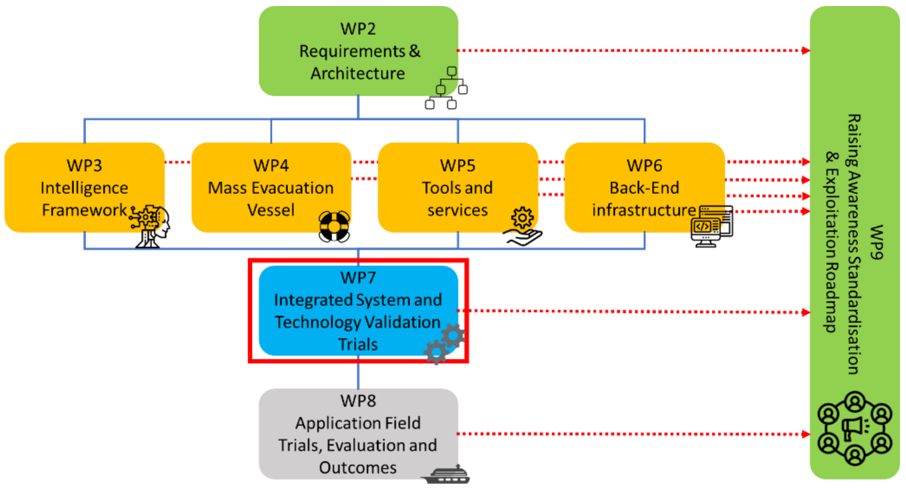

WP2
Requirements & Architecture
WP3
Intelligence Framework
WP4
Mass Evacuation Vessel
WP5
Tools and services
WP6
Back-End infrastructure
WP7
Integrated System and Technology Validation Trials
WP8
Application Field Trials, Evaluation and Outcomes
WP9
Raising Awareness Standardisation & Exploitation Roadmap

Figure 2. WP7 Dependencies with other WPs

All this "standalone" breakthrough funnels into WP7, responsible for, on the one hand, integrating all the different pieces and making them work altogether. Moreover, it is utmost important to assess that the whole system behaves as expected, so intensive validation trials will be carried out. As a direct continuation of this WP, WP8 (PALAEMON Application Field Trials, Evaluation and Outcomes) takes the integrated platform produced and tested throughout WP7's tasks and proceed to evaluate them on a real cruise ship (i.e., ANEK's Elyros), where different evacuation scenarios will be subject of analysis.

At last, it goes without saying that WP9 (Raising Awareness, Standardisation & Exploitation) reaps all the outcomes from all the previous work packages, aiming the increase the visibility of the work done throughout the project and paving the way to a successful commercial exploitation.

2 PALAEMON Communications Platform overview

As a direct continuation of the PALAEMON Reference Architecture we carried out in D2.7 (PALAEMON Architecture v2) [2], the logical connection of components gave rise to three different views or representations of the whole system.

2.1 Deployment view

Figure 3 represents the so-called “deployment view” of the PALAEMON Communications Platform. Said in other words, the physical infrastructure on which the different software modules are executed.

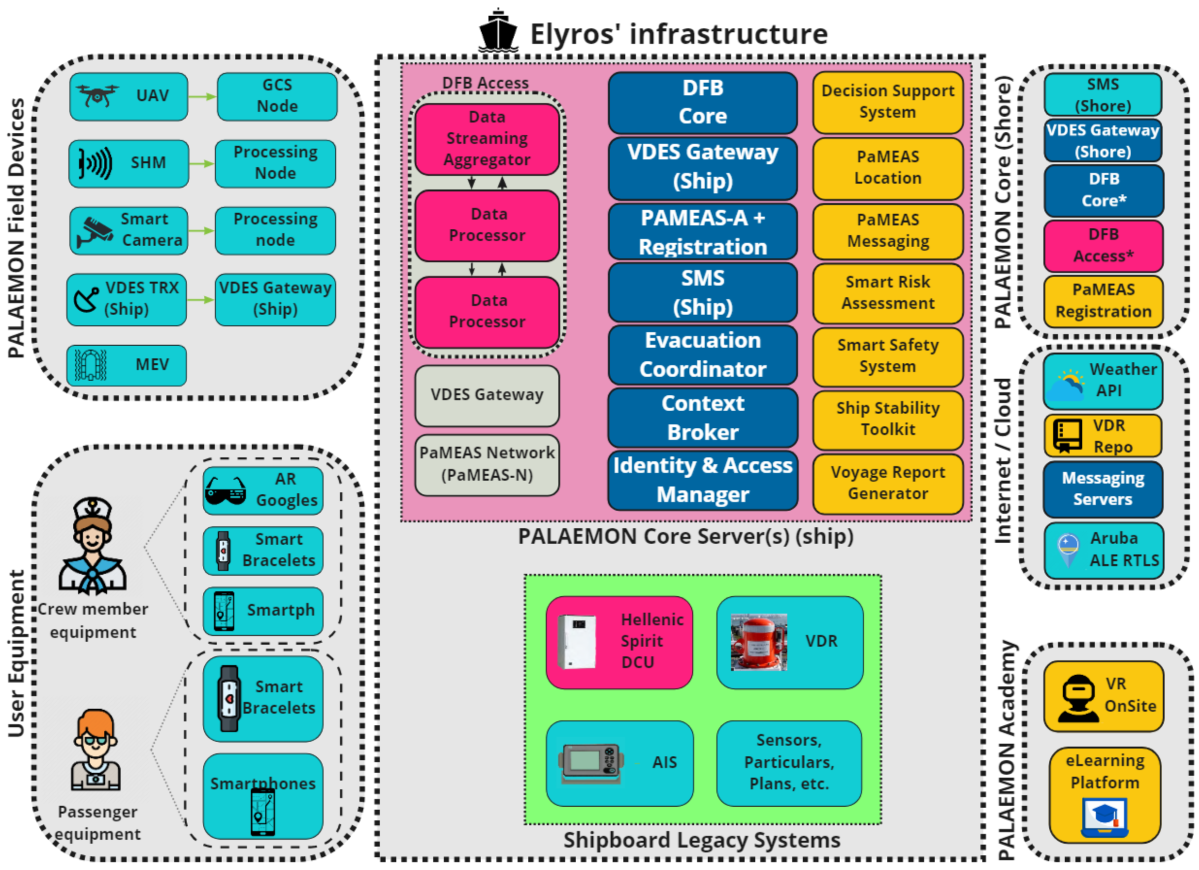


Figure 3. PALAEMON Communications Platform Deployment View

According to the above figure, we can differentiate up to six different host categories. In the next lines we will briefly describe the rationale behind each one.

- **Ship’s Infrastructure (Legacy)**. Modern (large) passenger ships must incorporate several sensors that permit a (remote) monitoring of all the underlying systems, in order to prevent/support during emergency situations.
- **PALAEMON Field Devices**. We introduce a number of new devices and technologies that bring additional information to enhance the contextual information and awareness before or during an incident and a (likely) subsequent evacuation process.
- **User equipment**. PaMEAS Services contribute to the (individual) real-time location of the passengers; which also receive notifications and recommendations of the best routes they should take to reach the closer/safer muster station. In addition, smart bracelets come to replicate the same location/routing features (with a simplified Human-Machine interface), adding the possibility as well of automatically detect

people falls (manually trigger an alarm via pressing a specific button). For the crew, they will wear a pair of AR goggles that display mission-critical information.

- **PALAEMON Core (Ship).** The heart of the Communications Platform is composed of a handful of software components that are installed and deployed over a traditional hardware infrastructure, that is, in layman's terms, one or various physical servers that host a group of configured Virtual Machines (VMs).
- **PALAEMON Core (Shore). A**shore extension of the main platform, some of the services will be replicated and deployed on a "lightweight" version of the setup we carry out aboard.
- **Internet / Cloud.** Some components do require a sporadic[1] Internet connection to extract/exchange information with external services.
- **PALAEMON Academy.** In the scope of the project, partners have made a thorough effort to prepare a dual training programme (based on the combination of Virtual Reality and a more traditional e-Learning platform).

2.2 Physical view

As we can observe in Figure 4, the various data sources and field devices lean on of a wide spectrum of physical (radio or wireless) interfaces when it comes to serve their information to the core of the PALAEMON system.

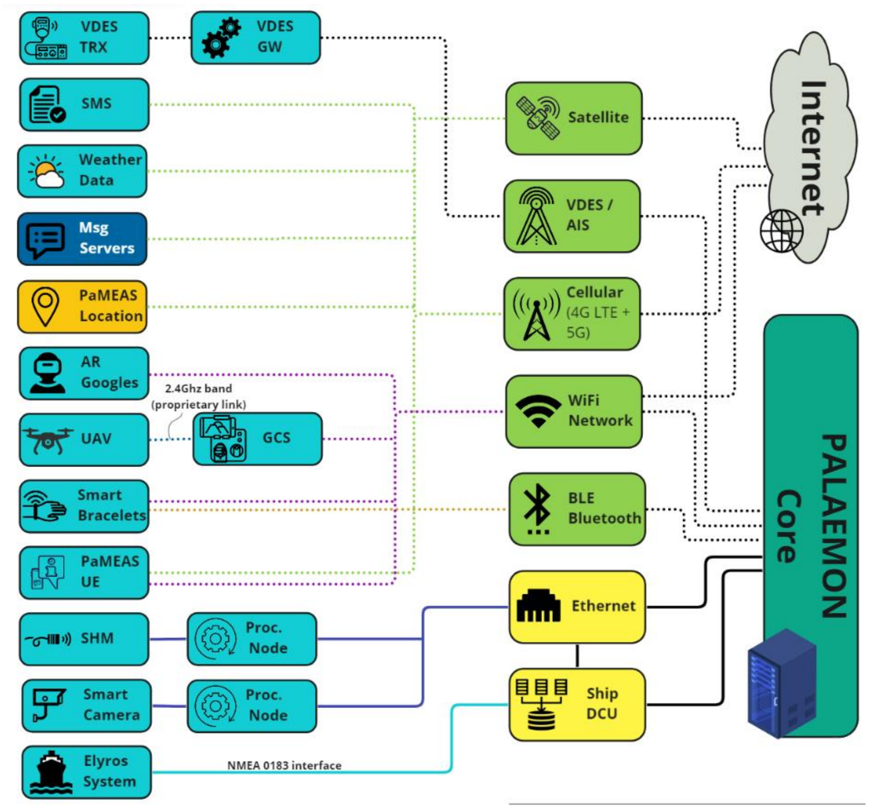


Figure 4. PALAEMON Components (physical interfacing)

[1] The system has been designed to minimize the dependency with external sources, making the system self-sufficient in case of emergency and potential Internet connection outage.

In the figure we can observe the different technologies (e.g., cellular, WiFi, Ethernet, Bluetooth…) that are used by each component to reach the central servers, where the core of the PALAEMON Communications Platform is running.

Framed under a maritime-specific realm, the utilization of a couple of VDES transceivers (designed and manufactured in the context of the project) allow a ship-to-shore communications[2].

It is worth stating that the system has been designed to be able to (mostly) work in a standalone mode, depending only on the inner communications (those ones that end or start on the PALAEMON Core). Whereas it is true that the system uses some Internet connections, its use during the evacuation process do not have a critical impact in case the connectivity is not available.

2.3 Security (i.e., Encryption and Authentication) at a glance

The PALAEMON Platform has to deal with critical situations, where a number of lives are at risk. It goes without saying that the system must guarantee a robust, secure and reliable operation throughout all the evacuation process. To that end, one of the key design patterns we have hooked at when creating the whole system hinged around the secure-by-design and private-by-design principles.

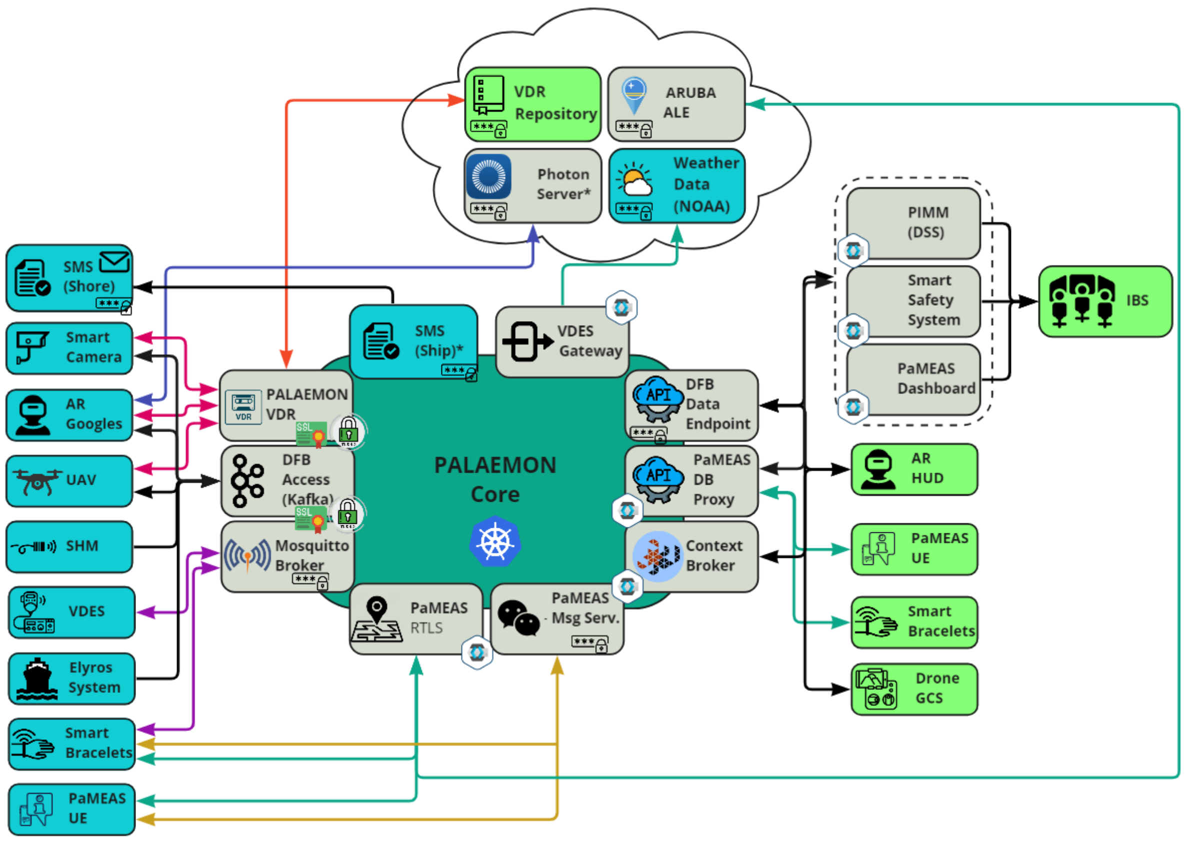


Figure 5. PALAEMON component connections and protection overview

[2] Though ship-to-ship and even ship-to-satellite links are also possible, they are out of the scope of PALEMON.

Figure 5 shows all the links among the different components of the PALAEMON Communications Platform. The next list enumerates and briefly outlines the security solutions we have followed to guarantee the security and privacy withing the PALAEMON Communications Platform:

- **PALAEMON Core (Kubernetes intra-cluster protection).** The utilization of Kubernetes [5] has a twofold purpose: 1- automate the deployment, scaling and management/orchestration of all the services; 2- harness it built-in intra-cluster protection mechanisms.
- **Secure Sockets Layer (SSL) certificates + Transport Layer Security (TLS).** Disregarding all the technical details, after the server confirms the identity of the client, a secure session is initiated, and the information is encrypted and protected. In fact, the central communication and data streaming component of DFB, based on an Apache Kafka broker, is configured with an Access Control List (ACL), where we have specified the list of topics that each user/component can read from or write to.
- **Keycloak Identity Management.** Keycloak is a separate server that “intercepts” external requests, check that they come from authorized users (for this it is compatible with several timely alternatives, as OAuth 2.0 [6] or OpenID Connect [7], for instance).
- **User and password.** This is a less sophisticated authentication method than Keycloak, but it still a very valid means when it comes to keep a restrictive access under control. In addition, it is a lightweight protocol that can be used in embedded devices (such as Smart Bracelets), so in some cases it could be the most suitable solution.
- **Mail.** Finally, there is a special case based on traditional e-mail service. This is the means that both SMS instances (i.e., ship to shore) have specified when it comes to update the fleet information ashore.

All the means we have taken are based on well-known and thoroughly tested open-source solutions that, altogether, give rise to the secure framework we targeted at the beginning of the project. Even more, we must also stress out that all the data travelling across these channels goes completely encrypted, hence external services cannot get access to the raw information by default.

3 PALAEMON Information model

This section presents the exercise of interoperability done across all information sources. We lean on a standardized framework, i.e., ETSI NGSI-LD [3], to make sure that all components understand each other, as data formats follow an homogeneous approach.

3.1 Data Fusion Bus Access

The PALAEMON Communications Platform hinges around the Data Fusion Bus. As we have already described in depth in former deliverables, like D2.6 [8], D2.7 [2], the focus was mainly on the ingestion and aggregation/combination of data, coming from multiple (and potentially heterogeneous) sources. In order to produce a common information model, we have defined the pipeline described in Figure 6.

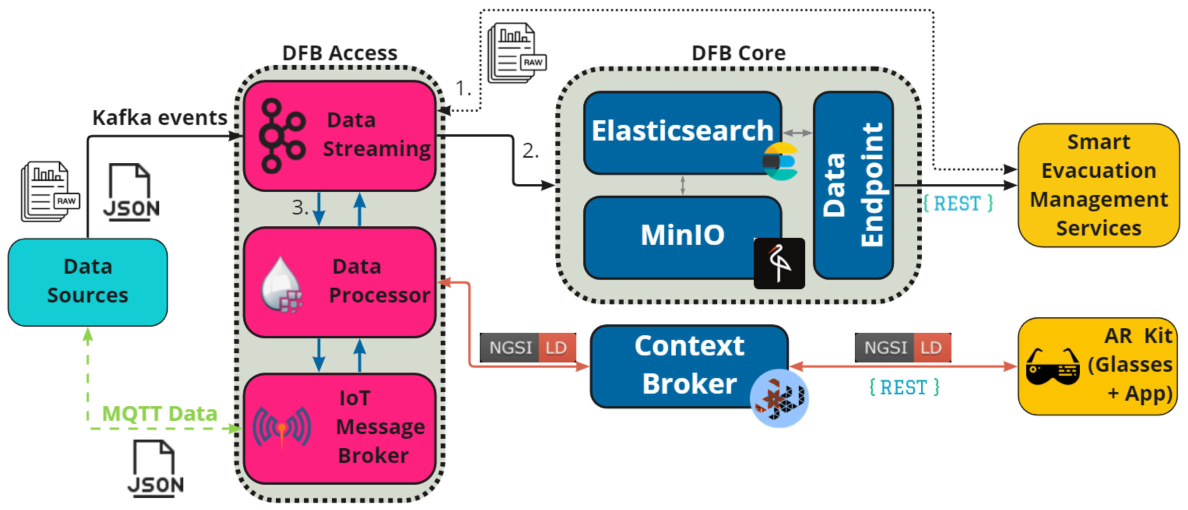


Figure 6. Data Fusion Bus interoperability flow.

At the left side, Data Sources send the raw information to DFB Access components, heading the Kafka Broker [9] (Data Streaming in the context of the picture). As main outputs, the Broker opens up to three different paths, whose starting points are explicitly marked in the figure:

1. Direct Kafka messages through well-known topics directly reach smart evacuation services. Thus, components subscribe to the topics they are interested in.
2. Raw data (as it comes from the sources) reaches DFB core, which mainly proceeds to store the information in the main database (Elasticsearch [10]).
3. There is also a direct communication between Kafka and Apache NiFi (i.e., Data Processor) [11], a so-called Extract, Transform and Load (ETL) tool that connects (i.e., subscribes) to the Kafka events. Basically, it gets the information as it is and transforms the data on-the-tly, handing the new format to the next component. As for the next stages, there are two possibilities: IoT Message Broker, based on Mosquitto [12] and a legacy NGSI-LD Context Broker, Scorpio [13].

3.2 Data transformation flow

One of the implicit advantages of using a platform like DFB is the flexibility it offers across its main components (i.e., Kafka/MQTT and NiFi). As we described in the last paragraph of the previous section, Apache NiFi is an ace up the sleeve that allows to straightforwardly define on-the-fly transformations, thus easily converting from raw data to standardized NGSI-LD context information. This way, components would be able to hook at this new data format if they want/need to. In fact, what can be called Data Transformation flow is displayed in Figure 7, where the raw inputs mainly come from Kafka or MQTT events, and after a transformation process (we will introduce how below), the outgoing NGSI-LD data is forwarded to a number of destinations (i.e., Apache Kafka, MQTT, Scorpio Broker, Elasticsearch and MinIO).

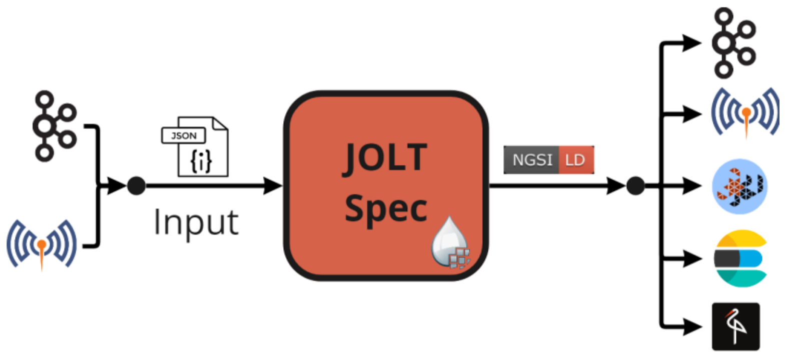
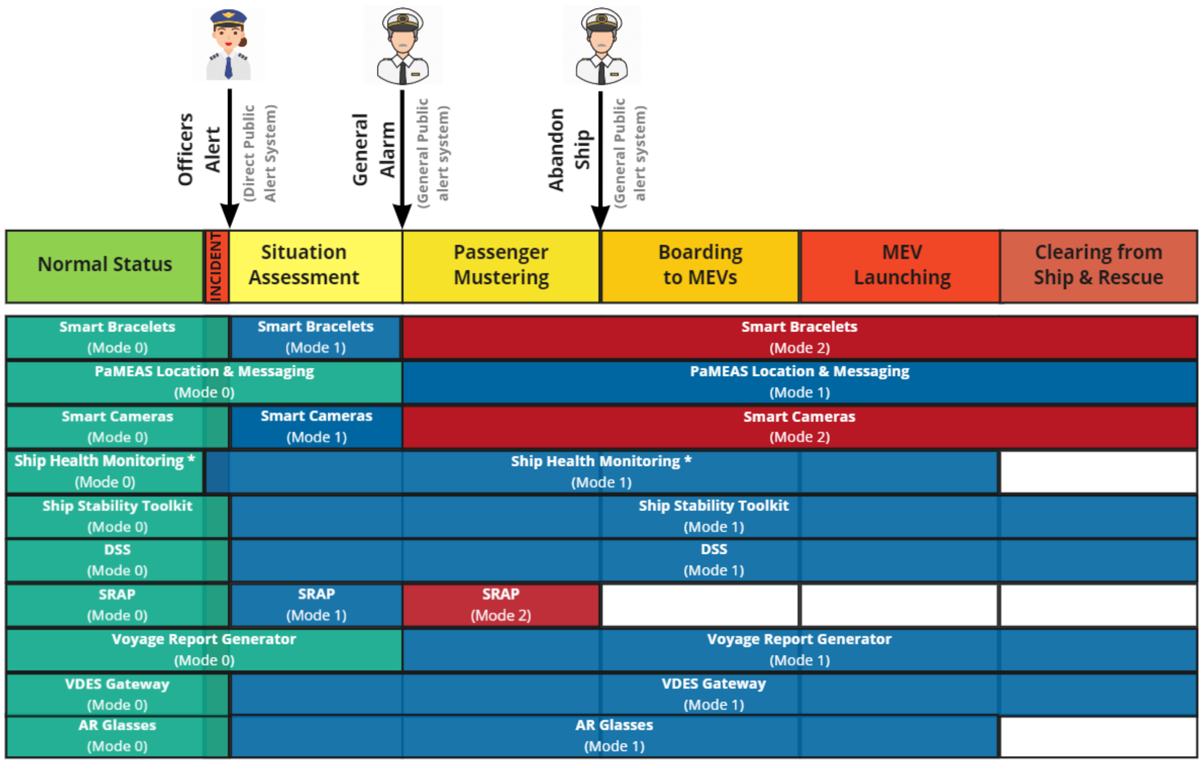


Figure 7. JOLT on-the-fly data transformations flow in PALAEMON.

3.3 Evacuation Operation Modes

Till this point, we have stated in numerous deliverables and reports that many (most) components do have a tight dependency with the ship evacuation status. Namely, they alter their operation mode according to the current evacuation phase. This behaviour spans from a normal situation, where the voyage elapses without any awkward situation, to the (in the worst case) moment in which the ship is abandoned. This chain of events is illustrated in Figure 8 (upper part of the picture), where we can observe the 6 phases.


Officers
Alert
(Direct Public Alert System)
General
Alarm
(General Public alert system)
Abandon
Ship
(General Public alert system)

Normal Status	INCIDENT	Situation Assessment	Passenger Mustering	Boarding to MEVs	MEV Launching	Clearing from Ship & Rescue
Smart Bracelets (Mode 0)	Smart Bracelets (Mode 0)	Smart Bracelets (Mode 1)	Smart Bracelets (Mode 2)	Smart Bracelets (Mode 2)	Smart Bracelets (Mode 2)	Smart Bracelets (Mode 2)
PaMEAS Location & Messaging (Mode 0)	PaMEAS Location & Messaging (Mode 0)	PaMEAS Location & Messaging (Mode 0)	PaMEAS Location & Messaging (Mode 1)	PaMEAS Location & Messaging (Mode 1)	PaMEAS Location & Messaging (Mode 1)	PaMEAS Location & Messaging (Mode 1)
Smart Cameras (Mode 0)	Smart Cameras (Mode 0)	Smart Cameras (Mode 1)	Smart Cameras (Mode 2)	Smart Cameras (Mode 2)	Smart Cameras (Mode 2)	Smart Cameras (Mode 2)
Ship Health Monitoring * (Mode 0)	Ship Health Monitoring * (Mode 1)	Ship Health Monitoring * (Mode 1)	Ship Health Monitoring * (Mode 1)	Ship Health Monitoring * (Mode 1)	Ship Health Monitoring * (Mode 1)	
Ship Stability Toolkit (Mode 0)	Ship Stability Toolkit (Mode 0)	Ship Stability Toolkit (Mode 1)	Ship Stability Toolkit (Mode 1)	Ship Stability Toolkit (Mode 1)	Ship Stability Toolkit (Mode 1)	Ship Stability Toolkit (Mode 1)
DSS (Mode 0)	DSS (Mode 0)	DSS (Mode 1)	DSS (Mode 1)	DSS (Mode 1)	DSS (Mode 1)	DSS (Mode 1)
SRAP (Mode 0)	SRAP (Mode 0)	SRAP (Mode 1)	SRAP (Mode 2)			
Voyage Report Generator (Mode 0)	Voyage Report Generator (Mode 0)	Voyage Report Generator (Mode 0)	Voyage Report Generator (Mode 1)	Voyage Report Generator (Mode 1)	Voyage Report Generator (Mode 1)	Voyage Report Generator (Mode 1)
VDES Gateway (Mode 0)	VDES Gateway (Mode 0)	VDES Gateway (Mode 1)	VDES Gateway (Mode 1)	VDES Gateway (Mode 1)	VDES Gateway (Mode 1)	VDES Gateway (Mode 1)
AR Glasses (Mode 0)	AR Glasses (Mode 0)	AR Glasses (Mode 1)	AR Glasses (Mode 1)	AR Glasses (Mode 1)	AR Glasses (Mode 1)	

Figure 8. Component operation modes vs. Evacuation Status Phase

4 Software Development and Integration Methodology (D7.5)

Typical software development workflows follow a number of phases, each covering a critical aspect of the software lifecycle. All the phases specified in this section must be overtaken in order to achieve a successful component development.

These phases do not necessarily need to be executed sequentially, they should be performed following an iterative approach, coming back to previous phases as many times as needed with the aim of reaching a stable and robust product.

4.1 Development phases

The development phases defined for the PALAEMON project are presented in Figure 9:

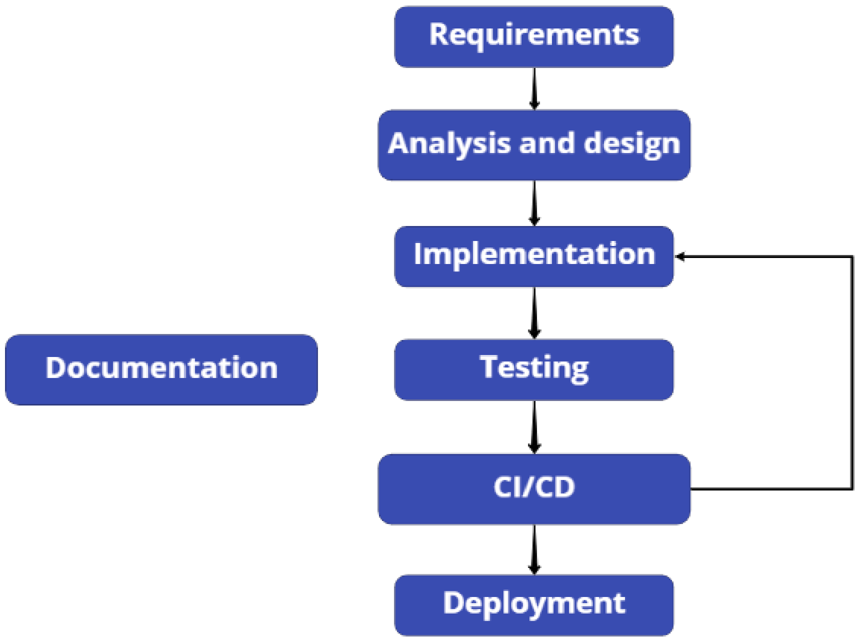


Figure 9. PALAEMON platform development phases

1. Requirements

For a software module to adequately perform its function, it is vital that its objectives are clear. For this reason, it is necessary to gather the requirements, pointing out the functionalities for each component, as well as its interfaces to other components and its needs regarding hardware and software. A first version to compile the requirements for the whole PALAEMON platform, and all its components individually has already been performed at D2.2 (First version of PALAEMON Requirement Capture Framework) [14]. As a matter of fact, the final compilation of the system requirements is gathered in D2.7 (PALAEMON Architecture v2) [2].

2. Analysis and design

This step is crucial for the following links in the development chain, since decisions taken at this point may be difficult to override at later stages (for example, changing the programming language may prove too cumbersome at a later point).

An accurate representation of the component flow can be achieved by using the 4+1 Architecture View Model [15], which can fully describe the component in all its different points of view (Logical, Process, Development, Physical, and Scenarios). Figure 10 represents the original representation of this popular design pattern.

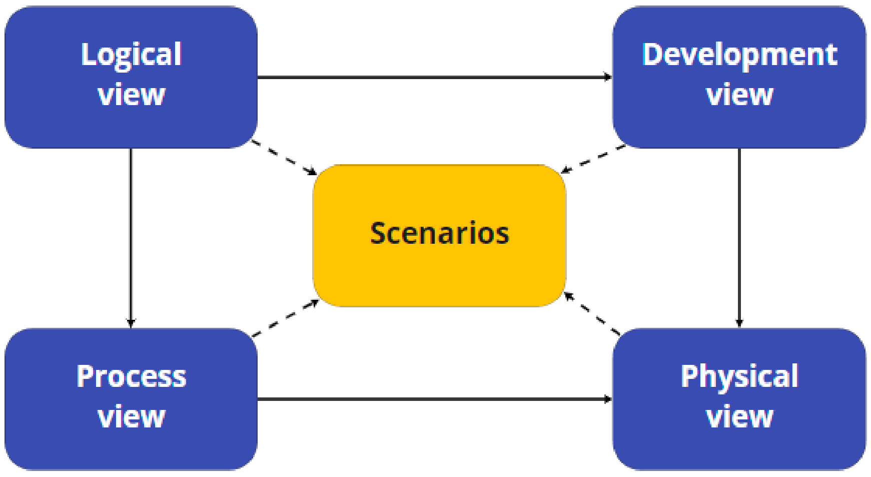


Figure 10. 4 + 1 Architecture Model view. Source: [15]

3. Implementation

This phase represents the actual software implementation of the component. At this stage, the developers will follow the design from the above stages in order to fulfil the requirements. This step can be iterated as many times as needed, until the required result and performance is achieved.

4. Testing

In order to ensure the software module works as expected and requirements are met, it is necessary to perform tests on the code. These tests will also prevent faulty code from being merged into the development branch and deployed to the production environment.

This step is best performed in parallel to the implementation phase, ensuring all produced and uploaded code passes a minimum quality check.

5. Continuous Integration/Continuous Delivery

The CI paradigm stablishes that code repositories need to be properly maintained, with frequent updates and automated building of artifacts, along with automated testing. This leads to an early finding of bugs and errors, which can be quickly solved and integrated into the codebase. Once the CI is configured, an automated way is made available to deploy the generated artifacts into the Testing and the Production (in the case of a stable version) environment. This way multiple environments can be managed, and versions can easily be tested or deployed.

6. Deployment

Once the software component has been implemented and tested, it needs to be deployed into the appropriate environment (Testing or Production). This process needs to take care of preparing the target machines with the necessary tools for the software to work (dependencies, databases, etc).

7. Documentation

This step proves to be one of the most important ones in the whole development process. Once a software component is implemented, it needs to be properly documented in order for users to be able to install it, configure it and be able to access to it. Typical documentation should contain references on how to:

- Install the component
- Configure the component
- Use the component
- Pass data to it
- Interpret the data outputted by the component

4.2 Source code versioning

It is of critical importance to make sure that the developed code is always up to date on the repository. This will make it an easier task to find errors and bugs on the code, as well as testing and generating artifacts. GitLab [16] provides a powerful tool to create issues and track the new and pending functionalities for each repository. This greatly helps following the progress in an agile manner, while allowing developers to plan ahead and communicate with each other.

The recommended way to work in this project is to create an issue for each functionality or found bug and associate it with a new branch that solves the task. This way, it is straightforward to track the progress of each task, as feedback from other developers can be provided in the form of comments and contributions to the issue.

Along with keeping the code updated, the documentation for the repository needs to be up to date as well. GitLab offers great tools for documenting code, such as Markdown files and Wikis that can host things like installation rules, APIs usage templates/guidelines or examples on how to make use of the component.

It is critical to keep the documentation up to date with the code versions, in order to maintain a stable and robust workflow. Functionality and bugfix branches should also update the documentation if changes in the API have been made.

4.3 Micro-service philosophy

One of the main shortcomings when dealing with software installations and deployments, is to make sure that all dependencies and libraries match the version being installed. A robust management of the configuration and generated files is also needed, in the case that multiple applications run on the same machine. For this reason, Docker [17] provides a way to encapsulate an application, together with all its dependencies and needed files, so that it can be easily deployed later.

The way Docker works is by running a container over the Operative System kernel that packages all the necessary items to run the application. This container always presents the same behaviour on every run, regardless of the actual hardware infrastructure. This means that containerized applications are portable and easily shareable over multiple infrastructures.

For the PALAEMON project, all applications that will run on server-side must be “Dockerized”, in order to achieve replicability over multiple environments, as well as to be easily deployed on every necessary instance.

All interfaces and instructions for deployment and use must be properly documented to ensure replicability.

- All services must be Dockerized and well documented for deployment
- Docker-compose/Kubernetes configuration files must be provided

4.4 Container orchestration

When deploying services, it is quite important to ensure availability and performance of said services at all times. One way to do this is by using Kubernetes [5] as an orchestration engine. Kubernetes allows system managers to closely control their hardware infrastructure, while at the same time manage the deployment of the services, ensure the scalability, availability and maintenance of the software components.

One of the main attributes for Kubernetes is its ability to keep track of all the deployed containers' status and to ensure their performance and availability at all times, by upscaling their hardware requirements or increasing the number of deployed containers.

A Kubernetes cluster is provided in order to be used to deploy the core of the PALAEMON platform, thus ensuring that all its components are fully operational at all times. This core will include the Data Fusion Bus, the Data Streaming Aggregator and the PALAEMON Evacuation Coordinator.

Another option when deploying microservices is the use of Docker Compose [19]. This tool manages the lifecycle of docker containers, by configuring its necessary variables and volumes, as well as exposing the proper ports needed by the service.

In order to manage the orchestration of services in the PALAEMON platform, partners are asked to supply the necessary docker-compose configuration files that deploy their components.

5 Test cases and overall system results

We summarize in this section the main outcomes of the assessment campaign we have carried out to validate that the whole PALAEMON system behaves as expected. As a direct continuation of the previous section, we describe here how the methodology proposed in the previous section has taken place throughout the development phase.

- **Requirement-based design**. In Deliverable D2.6 (PALAEMON Architecture v1) [8] we built a complete list of user-based (i.e., stakeholders') requirements. This compilation gave rise to the first and second versions of the PALAEMON Architecture.
- **Micro-service-oriented architecture**. Splitting the classical monolith into a number of independent software modules means that each partner could choose their programming framework (e.g., Python, C, Java, etc.). At the end, we only have to focus on the way these modules communicate with each other and how they save the information (e.g., databases, etc.), leaving all the rest to developers' choice.
- **Hybrid Kubernetes/Docker deployment**. This is a direct consequence of the use of micro-services instead of a monolithic solution. The deployment of the components become extremely simple and seamless. Moreover, the utilization of Kubernetes [5] over Docker [17] containers permits the orchestration of the most critical or complex components (e.g., databases, streaming brokers, etc.)
- **Inter-component communication**. One of the critical decisions we took is that, instead of specifying individual HTTP (HypeText Transfer Protocol) or RPC (Remote Procedure Call) interfaces for each component, we rely on[3] the main communications platform, Apache Kafka [9], the most widespread distributed event streaming platform.
- **Open-source nature**. One of the main perks of this type of collaborative projects is that partners can learn from each other. This opens the door to sharing the breakthrough achieved on the software development phase. In the context of PALAEMON, we have opted for GitLab [16] as the Version Control System (VCS) and Sonatype Nexus Manager [20] to manage all the binaries and artifacts created, i.e., Docker images, Python PIP packages, etc.
- **Continuous Integration / Continuous Deployment (CI/CD)**. Automating tasks permit developers to save tens of hours during the development process. Now, at the same time they save a (differential) copy of their source code, there are frameworks that seamlessly build all the infrastructure and redeploy the new and updated components.
- **Testing**. The main goal of this deliverable is to demonstrate that the PALAEMON Communication Platform is completely operative and, thus, WP8 receives a functional framework. We report throughout the document the main checks we have carried out to demonstrate that everything is in place to handle a real evacuation scenario.
- **Documentation**. There is no good development unless a good (or event better) documentation supports the source code. When it comes to replicate the conditions (e.g., initialization, configuration, expected operation, etc.), it is deemed necessary to have dedicated a fair amount on time writing down all this information.

It is worth describing here the actual infrastructure we have used to run all the tests. As a matter of fact, all the core services (e.g., Data Fusion Bus, etc.) are running on a cloud-based

[3] We can find some RESTful interfaces, but these are for secondary purposes.

ecosystem. During the real evacuation scenario (WP8), the idea is to migrate all the system onto an onboard hardware (i.e., server) infrastructure.

1. **PALAEMON Field Devices**: this category corresponds to external hardware (e.g., sensors, cameras, transceivers…) that bring information to the central system.
2. **User equipment**: similar to the previous case, but in this case a person (passenger/crew) uses/wears the device (Augmented Reality glasses, smart bracelets and smartphones).
3. **PALAEMON-01 (Elyros' emulated infrastructure)**. This is the cornerstone of the PALAEMON platform. Thought to run on a ship's premises (i.e., servers), key services run here. Communication (Kafka) brokers, databases and high-level services are executed in what we could assume as the system's mainframe.
4. **PALAEMON-02 (Shore emulated infrastructure)**. A lightweight version of the PALAEMON ship (main) system is replicated in an alike ashore environment, where different stakeholders (e.g., Port Authorities, Designated Person Ashore – DPA, etc.) can take part of the story.
5. **Internet/cloud services**: Some core components need to rely on external services to operate correctly. Hence, an Internet connection is required to have access to these.
6. **PALAEMON Academy**: Independent infrastructure that attends to passengers and training programme. Nothing to do with the actual evacuation process.

Taking into account that the purpose of this document is to offer an abstract that wraps up the main results of the WP, the reader shall refer to the original deliverable D7.6 (Test cases and overall system testing results) [1], which has a public dissemination level. There he/she will find all the results of the validation campaign.

6 Conclusions

This deliverable closes a journey that started more than three years ago, when we started collecting the stakeholders' requirements. This was the first stone on the definition of the PALAEMON Reference Architecture. From that first input, the platform has undergone several on-the-fly modifications and route re-calculations.

During the execution of the WP, we have gathered the breakthrough achieved across the core technical ICT (Information and Communication Technologies) Work Packages: WP3 - PALAEMON Intelligence Framework – AI Services and Algorithms, WP5 – PALAEMON On-board mustering tools and services and WP6 – PALAEMON Back-end Infrastructure. With all these software modules, we have given rise to a standalone framework, the PALAEMON Communications Platform that yields a modern vision to handle an evacuation process.

On top of this system, we have defined an information model, based on a timely ETSI standard, NGSI-LD, that guarantees that all components are able to understand each other. To do this, we have leaned on a popular ETL (Extract, Transform and Load) tool that transforms the raw data on-the-fly, adapts to this standard and forwards to the next phase in the pipeline.

It goes without saying that managing a development like this does need a predefined methodology. This document also presents the agreement we did when starting the definition phase in order to set the common bases when it comes to cope with software development.

A regular product needs a thorough validation phase before reaching e.g., the market. PALAEMON is not different and we have carried out an intensive assessment campaign to verify that the whole system works as expected.

Finally, once we have check that everything works as it should, the next stage consists in replicating the PALAEMON Communications Platform onboard ANEK's Elyros, where the whole system will serve to perform real evacuation scenarios.

7 References

[1] “PALAEMON Deliverable D7.6 - Test cases and overall system testing results (v2),” 2022.

[2] “PALAEMON Deliverable D2.7 - PALAEMON Architecture v2,” 2021.

[3] Cim, “GS CIM 009 - V1.2.2 - Context Information Management (CIM); NGSI-LD API,” 2020, Accessed: Nov. 23, 2021. [Online]. Available: https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx.

[4] GDPR, “General Data Protection Regulation (GDPR) – Official Legal Text,” *General Data Protection Regulation*, 2016. https://gdpr-info.eu/.

[5] “Production-Grade Container Orchestration - Kubernetes.” https://kubernetes.io/ (accessed Apr. 20, 2020).

[6] “OAuth 2.0 — OAuth.” https://oauth.net/2/.

[7] “OpenID Connect | OpenID.” https://openid.net/connect/.

[8] “PALAEMON Deliverable D2.6 - PALAEMON Architecture v1,” 2020.

[9] “Apache Kafka.” https://kafka.apache.org/ (accessed Jun. 02, 2021).

[10] “Elasticsearch: The Official Distributed Search & Analytics Engine | Elastic.” https://www.elastic.co/elasticsearch/ (accessed Jun. 18, 2021).

[11] “Apache NiFi.” https://nifi.apache.org/ (accessed Jun. 02, 2021).

[12] “Eclipse Mosquitto - An open source MQTT broker.” https://mosquitto.org/.

[13] “Scorpio Broker — ScorpioBroker documentation.” https://scorpio.readthedocs.io/en/latest/ (accessed May 26, 2021).

[14] “PALAEMON Deliverable D2.2 - First version of PALAEMON Requirement Capture Framework,” 2020.

[15] P. Kruchten, “Architectural Blueprints—The ‘4+1’ View Model of Software Architecture,” *arXiv*, vol. 12, no. November, pp. 42–50, 2020.

[16] “GitLab.org / GitLab · GitLab.” https://gitlab.com/gitlab-org/gitlab.

[17] “Docker. Accelerated, Containerized Application Development (homepage).” https://www.docker.com/.

[18] “Kubernetes.” .

[19] “Overview of Docker Compose | Docker Documentation.” https://docs.docker.com/compose/.

[20] “Sonatype Nexus Repository Manager (homepage).” https://www.sonatype.com/products/nexus-repository.