Project Nº: **FP7-614154**  
Brazilian Project Nº: **490084/2013-3**  
Project Acronym: **RESCUER**  

Project Title: **Reliable and Smart Crowdsourcing Solution for Emergency and Crisis Management**

Instrument: **Collaborative Project**  
European Call Identifier: **FP7-ICT-2013-EU-Brazil**  
Brazilian Call Identifier: **MCTI/CNPq 13/2012**

Deliverable D1.2.2  
System Architecture 2  

Due date of deliverable: PM 15  
Actual submission date: January 31, 2015

Start date of the project: **October 1, 2013 (Europe) | February 1, 2014 (Brazil)**  
Duration: **30 months**  
Organisation name of lead contractor for this deliverable: Fraunhofer IESE

<table>
<thead>
<tr>
<th>Dissemination level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PU</td>
<td>Public</td>
</tr>
<tr>
<td>PP</td>
<td>Restricted to other program participants (including Commission Services)</td>
</tr>
<tr>
<td>RE</td>
<td>Restricted to a group specified by the consortium (including Commission Services)</td>
</tr>
<tr>
<td>CO</td>
<td>Confidential, only for members of the consortium (including Commission Services)</td>
</tr>
</tbody>
</table>
Executive Summary
System Architecture

This document presents deliverable D1.2.2 (System Architecture 2) of project FP7-614154 | CNPq-490084/2013-3 (RESCUER), a Collaborative Project supported by the European Commission and MCTI/CNPq (Brazil). Full information of this project is available online at http://www.rescuer-project.org.

Deliverable D1.2.2 provides the results of Task 1.2 (System Architecture) for the first and second project iterations, summarizing the activities performed to define the architecture of the RESCUER system. The outcome is based on the input from the deliverables D1.1.2, D1.5.1, D2.1.2, D4.1.2, the RESCUER proposal Description of Work, and, most important, the direct communication with the end users and development partners.

It documents the main architectural drivers of the overall RESCUER system – e.g. the main business goals, key functional requirements, quality requirements and constraints. The architectural drivers are then reflected in the system design with appropriate architectural patterns. The scope and context of the system are made clear, and various perspectives (e.g., development perspective or deployment perspective, among others) are described as different views for different stakeholders. In addition, the architectural design decisions are documented.

RESCUER has a layered architecture. The overall system is designed in a modular way and it has several components. The components are organized into several layers – mobile application layer, data transport layer, data analysis layer, management layer and visualisation layer. For integrating these components, the integration layer provides the publish/subscribe mechanism. The components use this mechanism to have asynchronous communication among them.

Since this task is responsible for the overall platform, the individual components’ design decisions are left open to the respective project partners. The crosscutting quality requirements are broken down to address needs of specific individual components. The project partners will use those quality requirements as input while designing their own component. This approach will ensure the end-to-end quality of the RESCUER platform.

This is a living document. Since the requirements and the understanding about the domain evolve, the system architecture will be adapted accordingly. In the first iteration, the overall system architecture was developed with emphasis on information gathering from crowd sources and display of the analysed data. The second iteration improves the previous specification by emphasising follow-up interaction, and the third iteration will focus on crowd-steering. This document supports the understanding of the overall systems’ and should be used by project partners as a starting point to design their own components.
List of Authors
Taslim Arif – Fraunhofer
Tassio Vale – UFBA
Efraim Machado – UFBA
Paulo Jr – UFBA

List of Internal Reviewers
Laia Gasparin Pedraza – VOMATEC
Matthias Breyer – VOMATEC
Claudio Sant’Anna – UFBA
# Contents

1. Introduction ........................................................................................................................................ 7  
   1.1. Purpose of this Document ............................................................................................................ 7  
   1.2. Change Log ................................................................................................................................. 7  
   1.3. Approach to Create the System Architecture ............................................................................ 8  
   1.4. Partners’ Roles and Contributions ............................................................................................ 8  
   1.5. Document Overview .................................................................................................................. 8  

2. Conceptual Overview of the RESCUER System .............................................................................. 10  
   2.1. Mission ...................................................................................................................................... 10  
   2.2. Conceptual Architecture .......................................................................................................... 12  
   2.3. Scope and Iteration .................................................................................................................... 13  
   2.4. Locations and Stakeholders ..................................................................................................... 13  
       2.4.1. Stakeholders ....................................................................................................................... 13  
       2.4.2. Locations .......................................................................................................................... 15  

3. Architectural Drivers ......................................................................................................................... 16  
   3.1. Prioritisation ............................................................................................................................. 16  
   3.2. Business Goals ........................................................................................................................ 16  
   3.3. Key Functional Requirements .................................................................................................. 17  
   3.4. Devtime Requirements ........................................................................................................... 23  
   3.5. Integration Requirements ........................................................................................................ 24  
   3.6. Operation Requirements ........................................................................................................ 25  
       3.6.1. Operability ......................................................................................................................... 25  
       3.6.2. Evaluation and Demonstration Capability ...................................................................... 26  
       3.6.3. Installability ....................................................................................................................... 26  
   3.7. Runtime Quality Requirements ................................................................................................. 26  
       3.7.1. Availability ......................................................................................................................... 26  
       3.7.2. Robustness .......................................................................................................................... 27  
       3.7.3. Scalability ........................................................................................................................... 28  
       3.7.4. Safety .................................................................................................................................. 29  
       3.7.5. Upgradeability .................................................................................................................... 29  
       3.7.6. Auditability ......................................................................................................................... 29  
       3.7.7. Usability ............................................................................................................................. 29
Abbreviations ...................................................................................................................................... 82
Acronyms for the Modules to be developed ...................................................................................... 83
1. Introduction

1.1. Purpose of this Document

The RESCUER project aims at developing a smart and interoperable computer-based solution for supporting emergency and crisis management, with a special focus on emergencies in industrial areas and large-scale events. This deliverable provides the architectural foundations for the development and operation of the RESCUER platform. This document describes the architecture-significant requirements (ASR) for RESCUER. The system design is provided considering those ASRs of the domain. Each stakeholder sees the system from a different perspective; therefore, the architecture engagement purposes are described separately for different stakeholders, in the following paragraphs:

- **Development**: The project partners need to know the clear interface definition of their system components. They need to know the technologies and platforms to be used to realize their components. How the overall crosscutting quality attributes influences individual components needs to be clarified. Moreover, guidelines regarding the design of individual components are required in order to have a consistent and interoperable system at the end.

- **Deployment**: Another purpose of this document is to provide the deployment model that describes how the individual modules can be packaged and how it is mapped to different computing nodes and other resources. It is particularly important for the evaluation of the project because a functional demonstrator is mandatory at the end to evaluate the product.

- **Common Understanding**: Above all the main purpose of this document is to make the big picture of the RESCUER platform clear to all stakeholders. This is necessary to understand the overall system. Various gaps can be found out from the overall system diagram. Moreover, it can be used as a tool to discuss various points among the project partners and to the external world as well.

This document is a living document, which means it will continuously evolve during the three iterations of the project. It also covers currently known open issues, which might be elaborated in the next versions of this document.

1.2. Change Log

The changes performed in D1.2.1 (System Architecture 1) to give rise to D1.2.2 (System Architecture 2) are pointed out in the respective sections of this document. Due to the nature of this deliverable, it was necessary to describe the changes close to the models in their respective sections. In this way, stakeholders can easily find out what is new and what the changes with respect to the previous models are. The rationales are also described in case of necessity.
1.3. Approach to Create the System Architecture 2

The Fraunhofer’s ACES approach [1] was followed to define the overall system architecture for RESCUER. The steps are listed below:

- ASR Elicitation:
  - Various requirements documents, namely D1.1.2, D1.5.1, D2.1.2, D4.1.2 and the project proposal, were taken into consideration to generate the ASRs.
  - Further requirements were collected from discussions with the end-user and development organisations of RESCUER consortia.
  - Experiences of Fraunhofer from previous architectural projects were also revisited.
  - Input from standard quality models, like ISO 25010, were also taken into account.

  Finally, the requirements were validated with the partners and reviewers.

- Architecting the RESCUER System
  - Different views were selected to address different requirements and stakeholders;
  - Functional decomposition was created, based on the requirements.
  - Functional decomposition and deployment were, then, adjusted to address the quality requirements.

- Documenting the Architecture
  - Models were created in enterprise architect (EA).

Different views and models are categorized in this document to be understood and used by different partners and external stakeholders.

1.4. Partners’ Roles and Contributions

In the first project iteration, Fraunhofer was responsible for defining the system architecture and for writing the Deliverable D1.1.1 of Task 1.2 (System Architecture). VOMATEC, DFKI, FireServ and UPM supported Fraunhofer in identifying and reviewing the architectural drivers, constraints, models and technological solutions. UFBA supported Fraunhofer to describe the ASRs.

In the second project iteration, UFBA was responsible for evolving the system architecture previously proposed, by making compliance checking in individual components and writing the deliverable. Fraunhofer and UFBA performed such tasks in a joint work.

1.5. Document Overview

The remainder of this document is structured as follows.
- Chapter 2 contains the conceptual overview of the overall RESCUER system. It clarifies the mission and scope of the system.
- Chapter 3 presents the key architecturally significant requirements. Business drivers, key functional requirements, quality requirements and various constraints are presented as architectural scenarios.
- Chapter 4 provides fundamental architectural concepts. It presents the system context and the internal functional structure of the system.
- Chapter 5 describes the various aspects of the RESCUER system. The key functional perspective, the quality perspective, the development perspective, the deployment perspective, the usage perspective, and the task-assignment perspective are described in detail.
- Chapter 6 provides the design decisions list and shows the traceability between architectural significant requirements and design decisions.
- Chapter 7 provides the conclusion of this document.
2. Conceptual Overview of the RESCUER System

RESCUER is a crowd-sourced information solution for workforces, supporting forces, and visitors of large events or employees of industrial parks. It consists of a mobile application and a web application, among others. The mobile application supports information coming from the crowd, e.g. type of incident, severity, current situation (images, videos), etc. The Emergency Response Toolkit (ERT) solution together with other backend components do the data fusion and analysis and provide an appropriate visualisation for the command and control centre. Moreover, it is supposed to communicate with the workforces’ legacy systems and governmental organisations aiming to guide the crowd with better rescue strategies.

2.1. Mission

The main goal of RESCUER is to provide a system that is capable to collect real-time crowd-sourced data about an emergency situation and to analyse a huge amount of data in soft real-time to have a better picture in the command and control centre.

With the emergence of smart mobile devices, it is now possible to do crowd-sourcing more efficient than before. Nevertheless, in an emergency situation one needs a mechanism to collect data from the crowd without endangering their lives. The crowd must not be stressed with this additional work that they need to do to report an incident. This is one of the key challenges of RESCUER.

A huge amount of data is expected to be collected from the crowd. It is almost impossible for humans to both analyse texts, images, and videos, and generate a reliable view of the incident. Therefore, another goal of RESCUER is to analyse those data automatically as soon as data arrives, while ensuring reliability and displaying it in a way that staff in the command and control centre can immediately have a good understanding of the incident.

Figure 1 shows the mission of the RESCUER system. The RESCUER platform (shown as box) addresses the goals above-described. It is assumed that employees of industrial parks and visitors of large-scale events (b) will use the mobile crowd sourcing solution of RESCUER. Through this mobile solution, users will send incident reports to the backend server. Communication Infrastructure will receive data from the mobile solution. The data is then automatically analysed by the Data Analysis Solutions. The Emergency Response Toolkit (ERT) will provide visualisations of the analysed data in the command and control centre. It will support getting a quick overview of the emergency situation and prompt decision taking. ERT may also generate reliable messages for various stakeholders (c) to inform them about the situation. As advanced features, RESCUER steers the crowd and also coordinates the workforces (a) in the field.
• Exploiting the power of crowd-sourcing
• Using advanced techniques for data analysis and visualization
• Supporting mobility and providing context aware services

Figure 1: Mission of RESCUER platform
2.2. Conceptual Architecture

The main goal of the RESCUER system is to improve emergency management in the field. Figure 2 shows the conceptual architecture to address it. This architecture is divided in the following parts: the emergency environment comprised of its cause (e.g. fire, explosion or gas-leak), injuries, damages of property, among others; the Context Sensor component, responsible for sensing the environment; the Information Processing and Decision Making component, that receives the sensed data, analyses and visualizes them efficiently in the command and control centre; the command and control centre, which supports the decision making to take some actions to improve the situation; and the messages, sent back to the Effector component which does something physically on the emergency environment to improve the situation.

The components are described in the following points:

- **Context Sensors**: This conceptual component represents the stakeholders and systems that do sensing of the environment. Visitors in large-scale events, employees in the industrial parks, workforces, and supporting forces are the people who are truly on the spot during the incident. They are the people who can observe the situation and report it. Mobile device sensors can also sense the location and movement of the people on the spot.

Figure 2: RESCUER conceptual architecture
• **Effectors**: This conceptual component represents the stakeholders and systems that do something on the spot to handle the emergency situation in practice. The stakeholders can also be Effectors and take some actions based on the input from the command and control centre (C&C).

• **Information Processing and Decision Making (IPDM)**: This conceptual component represents the process and systems that do the off-site task. From the process perspective, it collects and analyses data from sensors and text messages, visualises the data efficiently, and plans, executes and monitors actions. To execute the plans it sends commands and messages to the Effectors.

• **Communication Channel**: This conceptual component represents the systems that bridge the communication between sensors and IPDM, and between effectors and IPDM. Smart mobile phone are the main communication channels selected in the RESCUER project.

### 2.3. Scope and Iteration

It is important to clearly understand the features and their respective project iterations. For the first iteration, the scope was information gathering from the mobile solution, analysis and visualisation of the gathered information in the command and control centre. This end-to-end solution is the heart of RESCUER and therefore part of the first phase. The follow-up interactions with the crowd are addressed in the second project iteration. Action plan and monitoring, crowd steering, workforce management, integration with the legacy workforces’ systems, and generation of reliable news will be addressed in the third iteration.

### 2.4. Locations and Stakeholders

To understand the overall system architecture, it is important to understand the locations and stakeholders. Figure 3 shows the stakeholders and organisations categorized into different locations.

#### 2.4.1. Stakeholders

From the architectural point of view, RESCUER stakeholders can be classified into several groups – namely the users, developers, operators and others (people inside and outside RESCUER). The developers and operators are the different project partners in the RESCUER context. In order to understand the architectural requirements, we have identified the key stakeholders (user group) from architectural point of view. The list of stakeholders with detailed description can be found in the deliverables D4.1.1 and D4.1.2 (Organisational Behaviour and Structures in Emergency Situations).

- **Civilians in large-scale events:**
  - General Civilians: These are the people who come to enjoy the event but do not have any professional skill that can be useful during an emergency situation.
• Civilians with Skills: These are the people who come to enjoy the event and have professional skills and experiences that are useful during emergency situations. For example, a visitor who is a doctor can be a great help in case of an emergency.

Figure 3: Locations and stakeholders (users)

• Employees of industrial parks: These are the people who work for a company in an industrial park.
• Workforces: Three types of workforces are usually in action during an emergency situation. They all have hierarchical structure – workforces working on the spot, group commanders managing the workforces on the spot, and the workforces on the Command and Control Centre (CCC) planning and monitoring everything. The types of workforces are:
  o Medical Service: The unit that is responsible for health related issues during an emergency situation.
  o Fire Service: The unit that is responsible for extinguishing fires and this sort of tasks during an emergency situation.
  o Police Service: The unit that is responsible for maintaining law and order during any event.
• Affected People: These are the people in the surrounding areas who are affected by the emergency situation.
• Command and Control Centre
  o On-site: The office that manages, monitors and controls the emergency operation on-site.
  o Off-site: The office that manages, monitors and controls the emergency operation off-site.
• Dispatchers
  o Medical Service: The officer that receives requests regarding medical teams and dispatches them.
  o Fire Service: The officer that receives requests regarding fire-service teams and dispatches them.
  o Police Service: The officer that receives requests regarding police forces and dispatches them.
• Authorities: Government offices/ministries that have the executive power or are accountable for maintaining a secure and safe public environment.
• Media: Agencies that collects news about the emergency situation and broadcasts them to the public.

2.4.2. Locations

The main locations that are relevant to understand the system are:
  • On-site-Workforces spot: It is the place where the incident took place. It is the core affected area.
  • On-site-Nearby area: It is a place that is very near to the incident, but not affected.
  • Off-site-Workforces: Offices where workforces work. It is far from the incident place.
  • Off-site-Others: It is far from the incident place. It can be government offices, media offices, and houses of the general public.
3. Architectural Drivers

This section presents the architecture significant requirements for the RESCUER platform. These drivers are derived from the system requirements and constraints, and act as input for later phases of architecting, namely making design decisions, structuring the system, and so on.

The key categories of architecture-significant requirements, defined in [1], are as follows:

- Business Goals
- Key-Functionality
- Devtime Requirements
- Integration Requirements
- Operation Requirements
- Runtime Quality Attributes
- Constraints

The architecture-significant requirements are not complete yet. They will evolve continuously during the project based on new understanding about the domain and its requirements.

3.1. Prioritisation

In the second iteration, the prioritisation is focused on addressing requirements related to follow-up interaction and making the compliance checking between the proposed architecture from deliverable D1.2.1 and individual components.

Additionally, the intention is to consider all known requirements in a way that their later incorporation does not lead to major architectural modifications. Thus, architectural considerations already go beyond the first and second project iterations, depending on the topic and its impact on the overall architecture.

3.2. Business Goals

- RESCUER aims to create a smart and interoperable computer-based solution for supporting emergency and crisis management, with a special focus on incidents in industrial areas and on large-scale events in Europe and Brazil;
- RESCUER aims to increase both time and effort efficiency in the following ways:
  - RESCUER aims provide improved time to collect information regarding an emergency situation;
  - RESCUER aims to minimize time and effort to analyse emergency data;
  - RESCUER aims to provide improved and reliable news to different stakeholders in the shortest possible time;
  - RESCUER aims to do context aware interaction with different stakeholders;
  - RESCUER aims to minimize collaboration effort among various workforces; and
  - RESCUER aims to efficiently manage the emergency by smart crowd steering and effective management of workforces on the spot.
RESCUER wants to demonstrate the power of crowdsourcing, smart mobile devices and automatic analysis techniques of multimedia data for emergency management.

3.3. Key Functional Requirements

The elicited architecture-significant requirements (ASR) related to the functional requirements are described in the tables 1, 2 and 3. Table 1 shows those realized during the first iteration of the RESCUER project, Table 2 shows the ones to be realized in the second iteration and Table 3 shows requirements for the third iteration of the project.

<table>
<thead>
<tr>
<th>ASR ID</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASR.FUNCTIONALITY.01</td>
<td>Sensor data recording of mobile devices of visitors, employees and workforces</td>
</tr>
<tr>
<td>ASR.FUNCTIONALITY.02</td>
<td>User interaction data collection</td>
</tr>
<tr>
<td>ASR.FUNCTIONALITY.03</td>
<td>Multimedia data collection</td>
</tr>
<tr>
<td>ASR.FUNCTIONALITY.04</td>
<td>Sending sensor data to the backend</td>
</tr>
<tr>
<td>ASR.FUNCTIONALITY.05</td>
<td>Sending user interaction data to the backend</td>
</tr>
<tr>
<td>ASR.FUNCTIONALITY.06</td>
<td>Sending multimedia data to the backend</td>
</tr>
<tr>
<td>ASR.FUNCTIONALITY.07</td>
<td>Receiving sensor data at the backend</td>
</tr>
<tr>
<td>ASR.FUNCTIONALITY.08</td>
<td>Receiving user interaction data at the backend</td>
</tr>
<tr>
<td>ASR.FUNCTIONALITY.09</td>
<td>Receiving multimedia data at the backend</td>
</tr>
<tr>
<td>ASR.FUNCTIONALITY.10</td>
<td>Feedback on data received from the mobile solution</td>
</tr>
<tr>
<td>ASR.FUNCTIONALITY.12</td>
<td>Automatic analysis of sensor data</td>
</tr>
<tr>
<td>ASR.FUNCTIONALITY.13</td>
<td>Automatic analysis of user interaction data</td>
</tr>
<tr>
<td>ASR.FUNCTIONALITY.14</td>
<td>Automatic analysis of multimedia data</td>
</tr>
<tr>
<td>ASR.FUNCTIONALITY.16</td>
<td>Combined analysis of individual analysis results</td>
</tr>
<tr>
<td>ASR.FUNCTIONALITY.17</td>
<td>Generating report ID</td>
</tr>
<tr>
<td>ASR.FUNCTIONALITY.19</td>
<td>Static plan and facilities visualisation</td>
</tr>
<tr>
<td>ASR.FUNCTIONALITY.20</td>
<td>Visualisation of analysed data</td>
</tr>
<tr>
<td>ASR.FUNCTIONALITY.21</td>
<td>Visualisation of the logs behind the analysis results</td>
</tr>
<tr>
<td>ASR.FUNCTIONALITY.22</td>
<td>Receiving messages from the RESCUER backend</td>
</tr>
<tr>
<td>ASR.FUNCTIONALITY.23</td>
<td>Sending profile information to the RESCUER backend</td>
</tr>
<tr>
<td>ASR.FUNCTIONALITY.24</td>
<td>Profile information collection and persistence</td>
</tr>
</tbody>
</table>
Table 2 - Architecture-significant requirements to be realized during the second project iteration

<table>
<thead>
<tr>
<th>ASR ID</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASR.FUNCTIONALITY.11</td>
<td>Sending/receiving SMS</td>
</tr>
<tr>
<td>ASR.FUNCTIONALITY.15</td>
<td>Automatic analysis of SMS</td>
</tr>
<tr>
<td>ASR.FUNCTIONALITY.18</td>
<td>Using ad-hoc P2P network</td>
</tr>
<tr>
<td>ASR.FUNCTIONALITY.25</td>
<td>Plan actions in the command and control centre</td>
</tr>
<tr>
<td>ASR.FUNCTIONALITY.26</td>
<td>Monitor actions</td>
</tr>
<tr>
<td>ASR.FUNCTIONALITY.27</td>
<td>Manage workforces</td>
</tr>
<tr>
<td>ASR.FUNCTIONALITY.28</td>
<td>Providing information to the press and the authorities</td>
</tr>
<tr>
<td>ASR.FUNCTIONALITY.29</td>
<td>Providing information to social media</td>
</tr>
<tr>
<td>ASR.FUNCTIONALITY.30</td>
<td>Follow up interaction</td>
</tr>
<tr>
<td>ASR.FUNCTIONALITY.32</td>
<td>Gathering data from map services</td>
</tr>
<tr>
<td>ASR.FUNCTIONALITY.33</td>
<td>Incident messages prioritisation</td>
</tr>
<tr>
<td>ASR.FUNCTIONALITY.34</td>
<td>Gathering data from weather information services</td>
</tr>
<tr>
<td>ASR.FUNCTIONALITY.35</td>
<td>Gathering data from traffic information services</td>
</tr>
<tr>
<td>ASR.FUNCTIONALITY.36</td>
<td>Incident reports prioritisation</td>
</tr>
<tr>
<td>ASR.FUNCTIONALITY.37</td>
<td>Gathering key places information</td>
</tr>
</tbody>
</table>

Table 3 - Architecture-significant requirements to be realized in third project iteration

<table>
<thead>
<tr>
<th>ASR ID</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASR.FUNCTIONALITY.31</td>
<td>Crowd steering</td>
</tr>
</tbody>
</table>

Requirements:

- **ASR.FUNCTIONALITY.01: Sensor data recording of mobile devices of visitors, employees and workforces**
  The RESCUER backend requests the mobile solution to record sensor data. Mobile solution records the data (GPS). The recording is done in the background of the mobile app. The user is not interrupted to do interaction with the mobile app while the sensor recording is done in the background.

- **ASR.FUNCTIONALITY.02: User Interaction data collection**
  Users of the mobile solution interact with the application with touches and gestures, and enter data. The mobile solution provides the interaction support and collects the user generated data (structured or unstructured text data).

- **ASR.FUNCTIONALITY.03: Multimedia data collection**
  Users of the mobile solution take photos or videos of the emergency situation through the mobile solution. The mobile solution captures meta-data (time and GPS, among others) along with the multimedia data.
• **ASR.FUNCTIONALITY.04: Sending sensor data to the backend**
  The mobile solution sends the recorded sensor data periodically as requested by the RESCUER backend. The user has the possibility to configure the system to stop sending the sensor reports.

• **ASR.FUNCTIONALITY.05: Sending user interaction data to the backend**
  The mobile solution sends the user interaction data together with meta-information (profile information, timestamp, and GPS) to the RESCUER backend. The mobile solution sends partial reports (whenever the user clicks some button) or incomplete report in case the user does not interact anymore for some period of time.

• **ASR.FUNCTIONALITY.06: Sending multimedia data to the backend**
  The mobile solution sends the multimedia data together with the meta-information to the RESCUER backend. Sending multimedia reports is a resource-consuming task; therefore it should run in the background and not interrupt the user interaction or any other relevant report sending to the backend.

• **ASR.FUNCTIONALITY.07: Receiving sensor data at the backend**
  The RESCUER backend is responsible for setting up the appropriate mechanism to receive sensor data. RESCUER backend receives the sensor data sent from mobile apps. This type of data is then made available for analysis.

• **ASR.FUNCTIONALITY.08: Receiving user interaction data at the backend**
  The RESCUER backend receives the user interaction data together with meta-information (profile information, timestamp, and GPS) from the mobile apps. The RESCUER backend is responsible for setting up the appropriate mechanism to receive user interaction data. This type of data is then made available for analysis.

• **ASR.FUNCTIONALITY.09: Receiving multimedia data at the backend**
  The RESCUER backend receives multimedia (image and video) data together with meta-information (profile information, timestamp, and GPS) from the mobile apps. The RESCUER backend is responsible for setting up the appropriate mechanism to receive multimedia data. This type of data is then made available for analysis and emergency response toolkit.

• **ASR.FUNCTIONALITY.10: Feedback on data received from the mobile solution**
  Users of the mobile solution see prompt feedback about the reports that they have sent. The feedback message is generated locally by the mobile app based on the code received from the RESCUER backend.
• **ASR.FUNCTIONALITY.11: Sending/receiving SMS**
  Eyewitnesses either using RESCUER mobile solution or not, sends SMS to a specific emergency number. The RESCUER backend is able to receive the SMS, which is made available for analysis and emergency response toolkit. In addition, the RESCUER backend sends SMS for the crowd to guide people using strategies for crisis mitigation.
  Related requirement in the D1.1.2 deliverable: FRGNR01.

• **ASR.FUNCTIONALITY.12: Automatic analysis of sensor data**
  The RESCUER backend analyses the sensor data collected from the mobile apps. The analysis result is updated within 1 minute.

• **ASR.FUNCTIONALITY.13: Automatic analysis of user interaction data**
  The RESCUER backend analyses the user interaction data which can be either free text or structured information or both. As partial or incomplete interaction reports can be sent from the mobile app, the RESCUER backend combines the partial reports and does the analysis. The analysis result is updated within 1 minute.

• **ASR.FUNCTIONALITY.14: Automatic analysis of multimedia data**
  The RESCUER backend automatically analyses the multimedia data (image and video). The multimedia data can have associated meta-data if the user captures the multimedia data through the app. However, in case the user sends multimedia data from the gallery (multimedia not created through RESCUER app), the RESCUER backend will not receive the meta-data.

• **ASR.FUNCTIONALITY.15: Automatic analysis of SMS**
  RESCUER analyses the SMS received from the eyewitnesses automatically. The SMS is associated with the user based on the phone number. The analysis result is produced within 1 minute and it is made available for further analyses or visualisations.

• **ASR.FUNCTIONALITY.16: Combined analysis of individual analysis results**
  RESCUER combines the individual analysis results. It means the individual analysis results (sensor analysis, text analysis, image analysis, video analysis, and SMS analysis) are combined to generate a higher-level analysis result. This aggregation of results is done automatically. Individual results should be converted into combined analysis result within 1 minute. The combined result is then made available for visualisation.

• **ASR.FUNCTIONALITY.17: Generating report ID**
  The RESCUER app generates a unique report ID and associates it to every report it sends to the backend. In case of partial reports in a session, it generates sub-report ID so that a complete report can be constructed in the backend side, if required.
• **ASR.FUNCTIONALITY.18: Using Ad-hoc P2P network**
  In case of low or no network availability, the RESCUER app uses the Ad-hoc P2P network to send the reports with only the most basic information (e.g. sensor and interaction) to the RESCUER backend. The switch between regular WiFi/3G and Ad-hoc P2P has to be done by the system automatically, without active user involvement.

• **ASR.FUNCTIONALITY.19: Static plan and facilities visualisation**
  RESCUER supports the visualisation of the emergency related static plans (building plan, evacuation plan, among others) in the command and control centre. This static information is uploaded at the beginning of the system operation (after installation) by the RESCUER operator.

• **ASR.FUNCTIONALITY.20: Visualisation of analysed data**
  RESCUER supports the visualisation of the analysed data in the command and control centre through the Emergency Response Toolkit. Users in the command and control centre can configure their respective dashboards based on their needs. They can see the crisis map, density of the crowd, position of the workforces, among others.

• **ASR.FUNCTIONALITY.21: Visualisation of the logs behind the analysis results**
  RESCUER supports the visualisation of the logs behind the analysis results in the command and control centre. Users in the command and control centre can investigate deep inside the logs that caused RESCUER to draw any specific analysis conclusion.

• **ASR.FUNCTIONALITY.22: Receiving messages from the RESCUER backend**
  The RESCUER backend sends messages to the RESCUER mobile solution (RESCUER apps). The RESCUER apps receive the messages and, if necessary, show them to the user. In case of administrative messages (for example: record sensor data), the RESCUER apps take the respective actions in the background.

• **ASR.FUNCTIONALITY.23: Sending profile information to the RESCUER backend**
  The RESCUER mobile solution sends the user profile information to the RESCUER backend whenever it is updated. The RESCUER mobile solution also sends key user profile information with every report it sends to the backend.

• **ASR.FUNCTIONALITY.24: Profile information collection and persistence**
  Eyewitnesses and other stakeholders on the spot use the RESCUER app to enter their profile information. The RESCUER app makes the profile information persistent.

• **ASR.FUNCTIONALITY.25: Plan actions in the command and control centre**
  The command and control centre uses the RESCUER platform to support him/her in creating a plan with respect to the current situation.
• **ASR.FUNCTIONALITY.26: Monitor actions**
  The command and control centre uses the RESCUER platform to monitor the status of his/her actions.

• **ASR.FUNCTIONALITY.27: Manage workforces**
  Staff in the command and control centre manages his/her workforces in the field through RESCUER. The user can deploy workforces to specific areas of the emergency situation, send them orders through messages, among others.

• **ASR.FUNCTIONALITY.28: Providing information to the press and the authorities**
  RESCUER can provide news and information for the press and the authorities. Staff in the command and control centre can use it to prepare final and reliable information. The information is sent to the stakeholders outside the area of the emergency situation.

• **ASR.FUNCTIONALITY.29: Providing information to social media**
  RESCUER provides important information about the emergency situation to relevant social media. Affected people or general public might be informed about the emergency situation in this way.

• **ASR.FUNCTIONALITY.30: Follow up interaction**
  The command and control centre interacts with the user of the RESCUER mobile solution to get more information about the situation. It asks context aware questions to the user and through conversation it extracts useful information. There is specific information to be requested depending on the stakeholder. Examples of additional information requested by the ERT to the specific stakeholders are:
  - The emergency brigade can provide information about products that caused the incident, if there is explosion/odour/environmental impact;
  - The civilian that is not in the incident area can inform if there are injured people, if they need support, etc.;
  - The supporting forces can provide information about fire severity, smoke direction, explosion reaction, etc.;
  - The workforces can send photos and videos of the incident.

• **ASR.FUNCTIONALITY.31: Crowd steering**
  RESCUER sends steering messages to the crowd to effectively relocate them. The information sent by the RESCUER platform is reliable and does not make the situation even worse.

• **ASR.FUNCTIONALITY.32: Gathering data from map services**
  One of the ERT visualisation metaphors is the map organization, displaying information according to geographic positions. The Toolkit requests data from map services to
provide such a visualisation. Even in case of map services crashes, the map data has to be available offline.
Related requirement in the D1.1.2 deliverable: FRETK18, FRETK05

- **ASR.FUNCTIONALITY.33: Incident messages prioritisation**
  The RESCUER should differentiate incident messages according to their importance calculated by the Data Analysis component (e.g. images with better resolution have higher priority compared to images low resolution images). Incoming important incident messages have to be highlighted in the ERT. Reports are clustered by profile and/or reliability.
  Related requirement in the D1.1.2 deliverable: FRGNR02, FRDAS02

- **ASR.FUNCTIONALITY.34: Gathering data from weather information services**
  The ERT requests data from weather services to provide information about current weather conditions.
  Related requirement in the D1.1.2 deliverable: FRETK12

- **ASR.FUNCTIONALITY.35: Gathering data from traffic information services**
  The ERT requests data from traffic information services to provide information about the current traffic situation. This information is provided only if such services are available.
  Related requirement in the D1.1.2 deliverable: FRETK17

- **ASR.FUNCTIONALITY.36: Incident reports prioritisation**
  The ERT allows the prioritisation of incident reports (important and normal reports). In addition, the system should classify the reports according to their reliability.
  Related requirement in the D1.1.2 deliverable: FRGNR03, FRETK22, FRETK23

- **ASR.FUNCTIONALITY.37: Gathering key places information**
  The ERT provides, whether it is available, the location of places/organisations relevant for a crisis scenario. Such key places are: hospitals, access roads, police stations and power stations. There are additional key places specific for industrial parks: gathering point, internal brigade’s position, surrounding communities, gas pipes and gas station. Similarly, the additional key places for large events are: rescue forces location, schools and central gas stations.
  Related requirement in the D1.1.2 deliverable: FRETK20

### 3.4. Devtime Requirements

- **ASR.DEVTIME.01: Documentation of design and code**
  The RESCUER project is developed in multiple iterations. Moreover, the project might be extended beyond the current time frame and individual partners may enhance their
components afterwards. Therefore the documentation of all design decisions and code is necessary to smoothly run the project for longer time.

- **ASR.DEVTIME.02: Distributed development**
  The development teams (project partners) should be able to build their module separately and in isolation. Therefore, for smooth integration afterwards, it is necessary to define clear interfaces and dependencies among the modules.

### 3.5. Integration Requirements

- **ASR.INTEGRATION.01: New components should be integrated to the RESCUER platform without much effort**
  Several components that are going to be built in the course of the project should be integrated without much effort. Integration of new components should cause less than 1 person day effort to adjust any component interacting with the new one.

- **ASR.INTEGRATION.02: Integration with social media**
  RESCUER might disseminate news and other important information through social media like Facebook or Twitter. Integration with popular social media should be addressed.

- **ASR.INTEGRATION.03: Integration among internal components**
  Each component should expose interfaces for the service they provide to others and they keep the internal complexities hidden. Integration among the system components are addressed during development time. Therefore no integration effort is required at runtime other than setting up deployment infrastructure and network connections.

- **ASR.INTEGRATION.04: Integration with workforces’ systems**
  The RESCUER platform might need to exchange information with operation forces’ legacy systems. An efficient integration mechanism has to be developed to integrate any operation force’s system. The integration mechanism has to be flexible enough to handle heterogeneity in data, protocol, network connection, and so on.

- **ASR.INTEGRATION.05: Uniformity of integration mechanism across components**
  The integration mechanism in RESCUER has to be uniform across components. The Integration Platform (Task 5.1) offers one mechanism to integrate all components. This approach is necessary to have consistency and easy understanding about the overall system.

- **ASR.INTEGRATION.06: Integration with utilities information services**
  The RESCUER has to provide interfaces to request information from map, weather and traffic information services.
3.6. Operation Requirements

3.6.1. Operability

Assumptions:

All mobile users install the app before the event. If some installations are done during the incident and those miss the administrative messages (request for sensor data recording) from the server, the risk is accepted. Nevertheless, they will be able to do other interactions like all other users.

Requirements:

- **ASR.OPERABILITY.01: Import of initial facility plans and so on**
  A new user wants to use the RESCUER platform and already has data about different static facility plans. The RESCUER backend provides an interface to import static data without manual adjustment of this data in RESCUER.

- **ASR.OPERABILITY.02: User ID generation**
  A user wants to use RESCUER mobile solutions and installs the app. The app generates a unique id, which can be used in all subsequent communication of this user with the RESCUER backend.

- **ASR.OPERABILITY.03: Easy installation**
  The installer (person) of the RESCUER backend should not need to be aware of the insides of each individual component. The configuration has to be kept to a bare minimum level.

- **ASR.OPERABILITY.04: Initial request of sensor data recording**
  Users of the RESCUER backend send the initial request for sensor data recording to the RESCUER apps. The RESCUER backend broadcasts the message to all RESCUER apps immediately.

- **ASR.OPERABILITY.05: Minimum operational effort**
  The RESCUER platform should run without (or with minimum) operational activities during runtime. Most of the activities for operating it smoothly should be done automatically.

- **ASR.OPERABILITY.06: Initial information request to the utilities information services**
  RESCUER must verify if map, weather and traffic information services are available for requests. For map services, the RESCUER has to execute alternative strategies to provide map and weather information if the services are not available.
3.6.2. Evaluation and Demonstration Capability

Requirements:

- **ASR.DEMO.01: Efficient preparation for evaluation and demonstration**
  Several evaluations are planned and a demo has to be given. The demo can be prepared with no more than 10 minute effort for configuration (effort for preparing the hardware is excluded). The demo can be reset to the start in less than 5 minutes.

3.6.3. Installability

Requirements:

- **ASR.INSTALLABILITY.01: Download and installation of the RESCUER app**
  A person who works in the industrial park or will attend a large-scale event, or any operation force’s member who has a duty in those places want to use the RESCUER mobile solution. The user can download and install the RESCUER app from the app stores of Apple or Google. When starting the app, the user may enter the profile information and other configuration information (not mandatory and can be entered afterwards too), and then the user can directly use the app.

3.7. Runtime Quality Requirements

3.7.1. Availability

Assumptions:

- There is not one single data centre that is operating in the RESCUER backend for all events and industrial parks. Rather, there will be independent backends.

- Thus, it is not expected that there is an equal load around the clock. Rather, non-working time might be quieter hours in an industrial park and it can be used as maintenance windows for planned down-times of the system. The large scale events also take place at predefined time slots and the rest of the time can be used for planned maintenance.

Requirements:

- **ASR.AVAILABILITY.01: 24/7 availability of the RESCUER backend**
  A user of RESCUER mobile solution interacts with his/her RESCUER app and thus the RESCUER app sends an emergency report to the RESCUER backend. In principle, the backend should be available 24 hours / 7 days, especially during the time span of large-scale events and during the working time of industrial parks.
A user of ERT works with his web-interface and thus sends requests to the RESCUER backend. The backend should be available during the above-mentioned time.

- **ASR.AVAILABILITY.02: 24/7 planned downtimes of the RESCUER backend**
  For maintenance reasons, the RESCUER backend can be made offline when there is no large-scale event or it is not a working hour in the industrial park.

- **ASR.AVAILABILITY.03: Unplanned downtimes of the backend due to hardware failure**
  A hardware failure occurs in the backend (loss of one of the server machines of RESCUER backend). This results in an unavailability of the backend for the users of the RESCUER mobile solution or the ERT during 1 minute at most.

- **ASR.AVAILABILITY.04: Unplanned downtimes of the backend due to software failure**
  A software failure occurs in the backend (due to errors in the code or other issues). This results in an unavailability of the backend for the users of the RESCUER mobile solution or the ERT during 1 minute at most.

### 3.7.2. Robustness

**Requirements:**

- **ASR.ROBUSTNESS.01: Robustness against arbitrary user navigation**
  A user of the RESCUER mobile solution interacts with his/her RESCUER app through multiple, unintended gestures and touches in a quick fashion. The app is robust and does not crash.

- **ASR.ROBUSTNESS.02: Robustness against arbitrary user input**
  A user of the RESCUER mobile solution interacts with his/her RESCUER app and enters arbitrary data in the input fields of the app. The app is robust and does not crash.

  A user of RESCUER ERT interacts with his web-interface and enters arbitrary data in the input fields. The web application is robust and does not crash.

- **ASR.ROBUSTNESS.03: Robustness against unstable network connections**
  A user of the RESCUER mobile solution interacts with his/her RESCUER app and experiences an unstable network connection with low bandwidth. The app is robust and does not crash. No data is lost and eventually is sent to the RESCUER backend.

- **ASR.ROBUSTNESS.04: Robustness against no network connections**
  A user of the RESCUER mobile solution interacts with his/her RESCUER app and experiences a network outage. The app is robust and does not crash. Basic emergency reports (sensor data) can still be sent through Ad-hoc p2p network. Any other data is not
lost and eventually is sent to the RESCUER backend, once the network connection comes back.

- **ASR.ROBUSTNESS.05: Robustness against crashes of mobile application**
  A user of the RESCUER mobile solution interacts with his/her RESCUER app and the app crashes. The app is able to restart its operation from where it crashed. It means that the app persists all administrative messages from the server, all partial reports and profile information. Whenever the app starts, it starts working based on the last saved administrative messages.

- **ASR.ROBUSTNESS.06: Robustness against ERT crashes**
  A user of the RESCUER Emergency Response Toolkit has massive interaction with his/her system during a crisis situation. The app is robust and does not crash. Considering the loose coupling of the system, damaged components should be fixed and the system has to continue providing information to the command centre.

### 3.7.3. Scalability

**Assumptions:**

The number of industrial parks or large-scale events covered by one installation of the RESCUER backend can increase or decrease based on the region.

**Requirements:**

- **ASR.SCALABILITY.01: Initial load during first evaluation**
  The RESCUER backend is running for 100-200 test users, in an initial version for the first evaluation. It is covering either one event or one industrial park scenario.

- **ASR.SCALABILITY.02: Scaling for large number of apps**
  The number of users of the RESCUER app or ERT can increase. The backend has to scale in a way that it does not need to compromise its performance. In addition to the increment in number of users, the multimedia data (image and video) can also increase.

- **ASR.SCALABILITY.03: Scaling over large number of events or industrial parks**
  The RESCUER backend is intended to be evaluated in one event or one industrial park in Europe or in Brazil. Later, the solution is intended to be offered in multiple events. The current assumption is that backends for different events can be independently operated, which means they do not share data. However, in real world, this assumption might not hold and the backend might need to scale up to cover events simultaneously.
3.7.4. Safety

Requirements:

- **ASR.SAFETY.01: Mobile user is not endangered**
  An eyewitness is using the RESCUER mobile solution. The RESCUER app should not add stress so that his/her life is endangered. If the user is guided by the app, it always gives reliable and correct information.

3.7.5. Upgradeability

Requirements:

- **ASR.UPGRADEABILITY.01: No data loss during app upgrade**
  A user installs a new version of the RESCUER app. All data that was previously entered by the user (for example, profile information) has to be persisted. The data will be available in the new version too.

3.7.6. Auditability

Requirements:

- **ASR.AUDITABILITY.01: Backend logging**
  Auditors can observe the logs that register RESCUER news and other analysis results. The RESCUER backend should log every report collected from a RESCUER app. It should also log the mapping between analysis results and incident reports.

3.7.7. Usability

Requirements:

- **ASR.USABILITY.01: Simple user interface for the mobile app**
  As RESCUER is meant to be used in emergency situations, its mobile app user interface should not have any information that is not relevant to the current user profile. The user must be confronted with fully-automatic information gathering mechanisms before the semi-automatic and the interaction-based information gathering (if possible). Fully-automated information gathering has to be triggered by a single interaction (i.e., pressing a single button). Self-descriptive options offer the possibility to rapidly describe the type of emergency using known metaphors. If needed, the user has to have the possibility to specify arbitrary information about the situation and its context.
• **ASR.USABILITY.02: User should receive immediate response**
  The process running at the RESCUER platform should avoid freezing the main thread of
  the application. Each user interaction must generate an indication to the user that the
  user’s input it is being processed.

• **ASR.USABILITY.03: Integrated with large event app**
  The RESCUER app or part of it can be integrated within the main app for a large-scale
  event.

• **ASR.USABILITY.04: Customizable user interface for the ERT**
  As RESCUER is to be used in emergency situations, its user interface should not have any
  information that is not relevant to the workforces, providing only relevant information
  for crisis management.

### 3.7.8. Variability

Requirements:

• **ASR.VARIABILITY.01: Multiple existing systems of workforces**
  The RESCUER platform should be able to communicate with existing systems used by the
  workforces (e.g. police, fire and rescue services), whether they are able for integration.

• **ASR. VARIABILITY.02: Adaptability for EU and Brazil environments**
  The RESCUER platform should be adaptable to the European and Brazilian environments.

• **ASR. VARIABILITY.03: Adaptability for large-scale event and industrial park environments**
  The RESCUER platform should be adaptable to emergency situations placed at large-scale
  events and industrial parks.

• **ASR. VARIABILITY.04: Multiple mobile platforms**
  The RESCUER app should be able to run in different mobile operational systems.

• **ASR. VARIABILITY.05: Multiple social media**
  RESCUER news can be published in different social media. The system should not be
  bound to any specific social media.

• **ASR. VARIABILITY.06: Multiple web interfaces**
  RESCUER should support multiple types of clients (laptops, tablets of various platforms)
  in the command and control centre.
ASR. VARIABILITY.07: Multiple types of users for mobile solutions
RESCUER should support various types of users through its mobile solution. For example, in large scale events, it can be visitors and workforces. Visitors can be further classified into two groups – the experienced users and general users. Workforces can also be classified into three groups – medical service, fire service, and police. Moreover, each operation force has two types of users – the commander and the general staff.

ASR. VARIABILITY.08: Multiple types of users for web interface
RESCUER should support the multiple types of stakeholders in the command and control centre. Each stakeholder should have different views of the emergency situation based on his or her role.

3.7.9. Credibility
Requirements:

ASR.RELIABILITY.01: Emphasis should be given on professionals
The information provided by operation forces’ members and users with professional knowledge that can be useful in an emergency situation must have a higher level of credibility.

ASR.RELIABILITY.02: No wrong conclusion about the situation
The information generated by the system must not lead to a miss conclusion about the current situation.

ASR.RELIABILITY.03: Reliability of data
The data received by the application should be classified by its reliability. The reliability level must be taken in account while analysing the data.

3.7.10. Performance
Requirements:

ASR.PERFORMANCE.01: Response time for user interaction
Someone is using the RESCUER app. The response time for each interaction can be at maximum 1 second. Background threads, as for example sensor data recording or multimedia data sending, should not interrupt the user interaction. The threads are prioritised based on the importance of the task.

ASR.PERFORMANCE.02: Sending partial user interaction data
In order to analyse the user data quickly and to get better understanding of the emergency situation as fast as possible, partial reports are sent to the server. If the user
is idle for 15 seconds, the mobile solution should automatically send the partial/incomplete report. In addition, the information obtained from each user interaction is sent directly.

- **ASR.PERFORMANCE.03: End-to-end response time**
  RESCUER updates the analysis result and current visualisation within 2 minutes from the moment it receives a user report from a mobile device.

- **ASR.PERFORMANCE.04: Performance on visualisation**
  A user is using the ERT and changes the view. RESCUER takes less than 3 seconds to generate his custom defined view.

- **ASR.EFFICIENCY.05: Working in the low power mode**
  A user is using the RESCUER app and the mobile device lacks power or network. The app suspends the bulky tasks (image, video uploading) and saves power for more important reports.

### 3.7.11. Security

**Requirements:**

- **ASR.SECURITY.01: Separation of views for different ERT users**
  The application interface must adapt itself according to the logged user’s profile.

- **ASR.SECURITY.02: Overall data privacy**
  The system has to manage private and confidential information (e.g. only authorised people will have access to it).

- **ASR.SECURITY.03: Log browsing in ERT**
  Actions taken by users of the ERT must be recorded for auditing. Internal threads must be also able to log information about performed tasks.

- **ASR.SECURITY.04: Logging of analysis**
  Data analysis modules must log information about received data and data after analysis processing.

- **ASR.SECURITY.05: User should be able to delete his information and nothing will be used afterwards**
  There should be a function that enables the user to delete its personal information.

- **ASR.SECURITY.06: Data processing logging**
Data analysis modules must log information about data received and data analysis processing.

3.8. Constraints

Constraints:

- **ASR.CONSTRAINTS.01: Individual partners should be able to use (commercialize) their components separately**
  This constraint imposes that each module of the solution should have its interface and responsibilities well defined, so they can be individually reused.

- **ASR.CONSTRAINTS.02: Individual partners should be able to use their preferred technologies (on which they are skilled at) to develop their own components**
  Each partner has expertise in different technologies. Therefore, it is not realistic to use the same technologies for every component.

4. Key Architectural Concepts

4.1. Context Delineation

Figure 4 shows the RESCUER system and the external systems and stakeholders around it.

4.2. Internal Structure

The overall RESCUER system is divided into several layers. Figure 5 shows the layers of the RESCUER system. Figure 6 shows the components inside the layers.

- **Mobile Application Layer**: This layer is responsible for collecting sensor data, user interaction data, multimedia data (image, video), and unstructured text report from the users. The users are the visitors or employees and the workforces on the spot. This layer also shows the guidance and commands that come from the command and control centre.

- **Ad-hoc Network Layer**: This layer is responsible for providing alternative communication channel in case of lack of Internet access (Wi-Fi, mobile data network).

- **Data Transport Layer**: This layer is responsible for transporting data between the RESCUER mobile solution and the RESCUER backend. Transportation means collecting data from the RESCUER apps as well as forwarding data to them.

- **Data Analysis Layer**: This layer is responsible for analysing various kinds of data collected from the mobile applications independently and for doing a combined analysis.
• **Management and Dispatching Layer**: This layer is responsible for managing the crowd and Workforces. It helps to plan a set of actions based on the current situation. It prepares reliable news and information for various stakeholders.

• **Visualisation Layer**: This layer is responsible for showing analysed data, static emergency plans, as well as the status of the action decided by the command and control centre.

![Figure 4: Context delineation](image)
Figure 5: Layering concept
Figure 6: Layers including functional components
4.3. **Client Server Separation**

The RESCUER platform can be separated in client-server fashion (Figure 7). This separation is done considering the overall RESCUER platform. Individual component might have also client-server separation within the component.

- **Client Layer**: Various users of RESCUER platform use the RESCUER Mobile Solution (MS) and/or the RESCUER Emergency Response Toolkit (ERT) as client components to interact with the RESCUER platform. In case of large-scale events, visitors and members of the workforces will use two different MS to report about an incident and to get feedback from RESCUER.
backend. In similar fashion, in the scenario of industrial parks, employees and members of the workforces will use different MS to communicate with the RESCUER backend. On the other hand users in the command and control centre will use the ERT to know about the current state of the emergency scenario and to communicate news and guidance to appropriate stakeholders.

- **Server Layer:** The Server layer is responsible for gathering data from the various MS and for appropriately analysing data in order to provide a good visualisation of the emergency scenario. It also provides integration mechanisms to communicate with external systems, social media, among others. The RESCUER backend is the component that does all those activities. Figure 8 shows the internal structure of RESCUER Backend. It mainly consists of three main layers. The Data Transport layer helps to receive and send messages to the mobile applications and other external systems. The Data Analysis layer is responsible for analysing different reports collected from the mobile applications. The Integration layer is responsible for integrating components built inside RESCUER, but also responsible for the integration with the legacy systems of the operation forces. In addition, this layer stores shared data required by the components.

![Figure 8: RESCUER backend](image)

- **Ad-hoc Network Layer:** This layer is responsible for providing network in case of limited or no network is available at the place of the emergency situation. However, due to technical constraints, this layer does not provide enough bandwidth and thus does not allow all sorts of communication between the Client and Server layers. In any case, MS should be able to send sensor data to the server.
4.4. Conceptual Component Realisation

In Figure 2, the conceptual architecture is shown and several conceptual components are identified. On the other hand, Figure 6 provides the functional decomposition of the RESCUER platform and several functional components are identified. The mapping between these two types of components is described below.

- Communication Channel: Communication Channel consists of the hardware systems and software components responsible for establishing communication between the Sensors/Effectors of the emergency environment and the Information Processing and Decision Making component. The software components provided by RESCUER are summarized in Figure 9. This figure shows the layers and components that actually realize the Communication Channel. Visitors of large-scale events, employees of industrial park or workforce would use the RESCUER map to indicate the location of the incident in the report. The Ad-hoc P2P network is also necessary to establish communication to the RESCUER Backend. Finally, different sending/receiving components are required to send/receive data of different types.

- Information Processing and Decision Making (IPDM): It consists of the hardware and software components required for information processing and visualisation and user interaction at the server side. Figure 10 shows the software components provided by RESCUER to fulfil the responsibilities. It shows the layers and components that realize the IPDM. The four main layers are the data analysis layer, the decision making layer, the visualisation layer, and the integration layer. The data analysis layer is responsible for analysing different types of data collected from the MS. The decision making layer is responsible for planning several actions (e.g. interacting with the crowd, interacting with the workforces, external communication, reminders and alerts, etc.). The visualisation layer is responsible from providing easy, understandable and comprehensive status of the emergency scenario in soft real time. One more layer is required, the integration layer, mainly for technical purposes. It helps to provide integration among the components built inside RESCUER, but also to provide integration with the legacy systems of the workforces. Furthermore, it stores the data (text, images, videos and configuration data) that is shared across the component boundary.
Figure 9: Communication Channel
Figure 10: Information Processing and Decision Making
5. Perspectives

This chapter presents the RESCUER platform from different perspectives. The perspectives described below are the data perspective, the sub-systems perspective (each sub-system also has various perspectives), the development perspective, the deployment perspective, and the usage perspective.

5.1. Data Perspective

- Report Data Model: Figure 11 shows the data model for the different reports sent from the MS to the RESCUER backend. One Report consists of one or more Context Data and one or more Report Data. Context Data can be further refined into four main categories namely the sensor context data, user context data, application context data, and session context data.
  - Context Data: The data that captures the context of the report. It is the meta-information that helps to understand the origin and the situation when the report was sent. Data analysis components will make use of this context information to apply the quality model, reliability checking, and report aggregation, among others.
    - Sensor Context Data: It consists of the GPS and timestamp data collected from the mobile device when the report is sent.
    - User Context Data: It consists of the profile information of the user (e.g. type of user, skill, experience level, etc.).
    - Application Context Data: it means the position of the hierarchical UI structure from where the user is sending the report.
    - Session Context Data: Users might send reports at different points in time. Session context data is the logical information used to divide these reports into different logical chunks. The actual attributes for session context data will be defined in the next iteration. The identifier (id) of the report and sub-report id can be used to represent a session.
  - Report Data: It is the actual data that is generated by the user or the mobile app and it is sent to the RESCUER backend.
    - Interaction Data: The predefined buttons, combo-boxes, and other selections that the user selects or clicks. This data is structured by nature.
    - Unstructured Text Data: The free text that the user writes.
    - Image: Image taken by the user regarding the emergency situation.
    - Video: Video taken by the user regarding the emergency situation.
    - Image Meta Data: It consists of the GPS, timestamp and other meta-information of the image collected when it is taken.
    - Video Meta Data: It consists of the GPS, timestamp and other meta-information of the video collected when it is recorded. It can be either per frame information or information collected at the beginning only. Further information is calculated based on the information available in the video header.
Report Type: Various types of reports sent from the MS are described below. The reports have hierarchical structure. It means higher level reports may have the lower level report data as optional.

- **Sensor**: This is the most basic report. This report consists of sensor data only. All other reports also have this data.
- **Interaction**: This report consists of the interaction data and at least the sensor context data is included.
- **Text**: This report consists of free text, but it might include interaction data. Moreover, it includes at least sensor data as context information.
- **Multimedia**
  - **Image**: This report consists of one image. Optionally it can have image meta-data, which can also have all information included in the text report.
  - **Video**: This report consists of one video. Optionally it can have video meta-data. It can also have all data included in the text report.

- **Topic Model**: Figure 12 shows the various topics that are used to have communication among the components built inside RESCUER (further information is available in the D1.5.2 deliverable). Individual components are supposed to publish and subscribe to different topics based on their needs. The topics are described below.
  - **Profile Topic**: This topic holds the data about the profile of the mobile user. It consists of user-id, type of user, skills, among others.
  - **Incident Report Topic**:
    - **Sensor Topic**: It corresponds to the sensor report.
    - **Interaction Topic**: It corresponds to the interaction report.
    - **Unstructured Text Topic**: It corresponds to the text report.
    - **Image Topic**: It corresponds to the image report.
    - **Video Topic**: It corresponds to the video report.
    - **SMS Topic**: This topic holds the data of an individual SMS sent by a mobile user.
  - **Server Message Topic**: it covers only the administrative messages from the server (sensor data recording request). In the later iteration, this topic will be refined and it can be used for crowd steering.
  - **Analysis Report Topic**: The following topics correspond to various analysis results on the data collected from the MS. Combined analysis is the only exception – it corresponds to the results of a combined analysis of the individual analysis.
    - **Sensor Analysis Result Topic**
    - **Interaction Data Analysis Result Topic**
    - **Text Analysis Result Topic**
    - **Image Analysis Result Topic**
    - **Video Analysis Result Topic**
    - **Combined Analysis Result Topic**
Figure 11: Report Data model
Figure 12: Integration platform topics
5.2. Sub-Systems Perspective

Figure 14 shows the different sub-systems of the overall RESCUER platform. Figure 15 shows the mapping between those sub-systems and the functional components.

The sub-systems perspectives for the different RESCUER components (mobile solution, communication infrastructure, data analysis, emergency response toolkit and integration platform) are explained in the respective sub-sections (5.2.1 to 5.2.5). For each component sub-systems perspective, four views are presented: conceptual view, context/data view, development/technology view and deployment view.
Figure 14: Sub-systems view
Figure 15: Mapping of sub-systems to functional components
5.2.1. Mobile Solutions

- Conceptual View:

Figure 16: Conceptual view of the RESCUER Mobile Solution
Figure 17: Data flow and context of the RESCUER Mobile Solution
Development and Technology View:

Figure 18: Development and technology of the RESCUER Mobile Solution
• Deployment View:

Figure 19: Deployment of the RESCUER Mobile Solution
5.2.2. Communication Infrastructure

- Context and Data View:

Figure 20: Context and data flow of the Communication Infrastructure
Development View:

Figure 21: Development of the Communication Infrastructure
• Deployment View:

Figure 22: Deployment of the Communication Infrastructure
5.2.3. Data Analysis

- Context and Data View:

Figure 23: Context and data flow of the Data Analysis Solutions

56
Figure 24: Development of the Data Analysis Solutions
Deployment View:

Figure 25: Deployment of the Data Analysis Solutions
5.2.4. Emergency Response Toolkit

- Context and Data View:

Figure 26: Context and data flow of the Emergency Response Toolkit
Development View:

Figure 27: Development of the Emergency Response Toolkit
5.2.5. Integration Platform

The Integration Platform is responsible for integrating the components developed inside RESCUER (in different work packages) or outside RESCUER (i.e. the existing systems of workforces). Different views of the integration platform are described below.

- Context and Data View: Figure 29 shows the context delineation of the Integration Platform. The Integration Platform consists of two entities, the Integration Platform Connector and the BLOB Storage Component. The figure shows the components that are connected with the Integration Platform and the direction of data flow among them. Almost all components built inside RESCUER are integrated through the Integration Platform. ERT, five analysis components, Sensor Data Receiver, SMS Receiver, and the Mobile Solution are connected with each other through the Integration Platform.

- Development View: Figure 30 shows the development perspective of the Integration Platform and surrounding components. RabbitMQ\(^1\) will be used to realize the Integration Platform, and Cloud BLOB Service will be used to realize the BLOB Storage. The modules of the surrounding components publish and subscribe appropriate topics to be integrated.

---

\(^1\) http://www.rabbitmq.com/
Figure 29: Context and data flow of the Integration Platform
Figure 30: Development of the Integration Platform
• Deployment View:

Figure 31: Deployment of the Integration Platform
5.3. Development Perspective

Figure 32 shows the development perspective of the overall RESCUER system. The main points are summarized below.

- The Visualisation Layer and Decision Making Layer (including all components inside) are realized by the Emergency Response Toolkit Module. The ERT Module is a web application that can be accessed by devices like tablets or laptops used by the command centre staff.
- The five analysis components will result into five respective modules. This modularisation is required in case of partial deployment of the RESCUER system depending on the context of evaluation.
- The Sensor Data Receiver and SMS Receiver components will result into their respective modules. Social Media Connector and External Systems Connector are out of scope for the first project iteration.
- The User-interaction Data Receiver Component is realized by five different topics of RabbitMQ. The Server Message Topic of RabbitMQ realizes the Data Sender Component.
- RabbitMQ realizes integration Platform. BLOB Storage is realized by an appropriated cloud service.
- Ad-hoc P2P Network will result into an Ad-hoc P2P Network library which will be used by the Mobile Solutions to establish a network.
- The RESCUER Mobile Solution realizes Mobile Application Module, which is a mobile app, used by various stakeholders on the spot of an emergency situation.
Figure 32: Development of RESCUER
Figure 33 shows the various technologies used to realize the overall RESCUER system.
5.4. Deployment Perspective

Figure 34 shows the deployment of the overall RESCUER system. The main points of the deployment are described below.

- Users in the command and control centre will use a Client Device (e.g. tablet, laptop) to access the Emergency Response Toolkit web application. The network connection between the client device and the web application can be Internet, local intranet, or 3G.
- The Emergency Response Toolkit web application is deployed into a web application server that is running in the Emergency Response Toolkit Server. ERT is connected with the Integration Server through Internet, local intranet, or 3G. The network connection between ERT and Cloud BLOB Service is Internet or 3G.
- Five analysis modules namely Sensor Analysis Module, Image Analysis Module, Text Analysis Module, Video Analysis Module and Combined Analysis Module are deployed separately in their respective servers. Those are connected with the Integration Server through Internet, intranet, or 3G. The analysis modules that require connection with Cloud BLOB Service are connected through Internet or 3G.
- The Sensor Data Receiver Module is deployed into the Mongo DB Cluster. Mobile devices are connected with the Mongo DB Cluster through Wi-Fi, 3G, or Ad-hoc P2P Network. Mongo DB Cluster is connected with the Integration Server through Internet, local intranet, or 3G.
- The SMS Receiver Module is deployed into the SMS Receiver Server and it is connected with the Integration Server through Internet, intranet, or 3G.
- Mobile Solutions for various stakeholders are deployed into Android or iOS devices. Mobile devices are connected with each other through local Wi-Fi. Mobile devices are connected with the Integration Server and Cloud BLOB Service through Wi-Fi or 3G. With the SMSC the connection is through mobile network (GSM/CDMA etc.). Mobile devices are connected with the Mongo DB Cluster through Wi-Fi, 3G, or Ad-hoc P2P Network.
- The Integration Platform RabbitMQ is deployed into the Integration Server. Internet or local intranet connects it with all other computing nodes hosting RESCUER components. Cloud BLOB Service is a BLOB storage provided by a cloud provider. All components require that multimedia data can be connected through Internet with the BLOB Storage Service.
5.5. Usage Perspective

Figure 35 shows different components that will be used in different locations during an emergency situation. The locations are primarily divided into two types – on-site and off-site. On-site is then further divided into two regions – on-spot area and the area nearby. In the on-spot area, various types of users will use the RESCUER Mobile Solution. In the nearby area the command and
control centre will use the ERT. The off-site region is divided into two groups based on the stakeholder class – the workforces and the rest. Workforces would use the Emergency Response Toolkit in their dispatcher and off-site command and control centres. On the other hand, all other stakeholders would receive periodic RESCUER news from the RESCUER platform.

5.6. Task Distribution

5.6.1. MTM

MTM is responsible for implementing the mobile solution. The mobile solution is composed of mobile applications for visitors, employees and workforces (general members and group commanders). These mobile applications should run in both Android and iOS. MTM needs to integrate the mobile solution with the two libraries provided by DFKI, namely Ad-hoc P2P Library and Sensor Recording Library. They also need to integrate their solution with the large-scale event app. Moreover, their solution should interface with the Integration Platform for publishing and subscribing messages. Finally, the mobile solution should also be able to upload multimedia to BLOB Storage.

MTM is also responsible for building the BLOB Storage service that is responsible for storing multimedia data.
5.6.2. DFKI

DFKI is responsible for developing five modules. They will provide the Ad-hoc Network Module and the Sensor Recorder as libraries to be integrated with the Mobile Solution. DFKI is also responsible for providing Sensor Data Receiver Module, Sensor Analysis Module, and SMS Receiver Module. All these solutions have to be integrated with the Integration Platform. It means they all need to be accompanied with respective AMQP clients. Moreover, the SMS Receiver Module has to interface with the SMS gateway of the mobile network providers.

In addition, DFKI might need to take care of the User-Interaction Data Receiver and Data Sender components. Currently it is planned that RabbitMQ will be used to receive and send data to the mobile solution. In case this approach fails, DFKI will provide solutions for those.

5.6.3. VOMATEC

VOMATEC is responsible for building the Emergency Response Toolkit, Combined Analysis Module, and Integration Platform. For the Emergency Response Toolkit and Combined Analysis Module, they should use AMQP clients of their chosen technologies. In order to get multimedia data from the Cloud BLOB Service, they should use the protocol of the cloud service provider.

5.6.4. UPM

UPM is responsible for building the Video Analysis Module. In order to interact with the Integration Platform, UPM should use an AMQP client of their chosen technology. For getting videos from the Cloud BLOB Service, they should use the protocol of the cloud service provider.

5.6.5. USP

USP is responsible for building the Image Analysis Module. In order to interact with the Integration Platform, USP should also use an AMQP client. For getting images from the Cloud BLOB Service, they should also use the protocol of the cloud service provider.

5.6.6. UFBA

UFBA is responsible for building connectors to social media and external legacy systems of the workforces. For the social media, the connector will be one-way. It means just periodically uploading necessary information to be viewed by the respective stakeholders. For the external legacy systems, the requirements are still to be refined in the subsequent project iteration.
### Table 4: Development task allocation among partners

<table>
<thead>
<tr>
<th>Partner</th>
<th>Modules to build</th>
</tr>
</thead>
<tbody>
<tr>
<td>MTM</td>
<td><em>Mobile Solutions (MS), Cloud BLOB Service (BLOB)</em></td>
</tr>
<tr>
<td>DFKI</td>
<td><em>Ad-hoc P2P Network (ADHOC), Sensor Data Recorder (SDR), Sensor Data Receiver (SEN_RECV), Sensor Data Analysis (SDA), SMS Data Receiver (SMS_RECV), Text Analysis (TA)</em></td>
</tr>
<tr>
<td>VOMATEC</td>
<td><em>Emergency Response Toolkit (ERT), Combined Analysis (CA), Integration Platform (IP)</em></td>
</tr>
<tr>
<td>UPM</td>
<td><em>Video Analysis (VA)</em></td>
</tr>
<tr>
<td>USP</td>
<td><em>Image Analysis (IA)</em></td>
</tr>
<tr>
<td>UFBA</td>
<td><em>Social Media Connector (SMC), Legacy System Connector (LSC)</em></td>
</tr>
</tbody>
</table>
6. Traceability

This chapter presents a map between the architectural requirements to architectural design decisions to components. It shows how Devtime, Integration and Operation requirements, previously described (see chapter 3), are realized in this project.

<table>
<thead>
<tr>
<th>Architecture Significant Requirements</th>
<th>Realisation</th>
<th>Responsible Components</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Devtime Requirements</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ASR.DEVTIME.01: Documentation of design and code</td>
<td>This deliverable documents the overall system architecture. Individual project partners have been communicated to document their own design and code.</td>
<td>All</td>
</tr>
<tr>
<td>ASR.DEVTIME.02: Distributed development</td>
<td>System is designed in a modular way, and interfaces and data exchanges are made clear among them.</td>
<td>IP</td>
</tr>
<tr>
<td><strong>Integration Requirements</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ASR.INTEGRATION.01: New components should be integrated to the RESCUER platform without much effort</td>
<td>Generic integration mechanism publish-subscribe is used, which is not bound to any technology and provides asynchronous communication.</td>
<td>All, mostly IP</td>
</tr>
<tr>
<td>ASR.INTEGRATION.02: Integration with social media</td>
<td>Social Media Connector in the Data Transport layer is responsible for integration with the social media. This connector makes the overall RESCUER platform not tied to any social media. Realisation concepts will be built in next iterations.</td>
<td>SMC</td>
</tr>
<tr>
<td>ASR.INTEGRATION.03: Integration among internal components</td>
<td>Generic integration mechanism publish-subscribe is used, which is not bound to any technology and provides asynchronous communication.</td>
<td>IP</td>
</tr>
<tr>
<td>ASR.INTEGRATION.04: Integration with workforces’ systems</td>
<td>External System Connector in the Data Transport layer is responsible for integration with the workforces’ systems. Actual integration concepts will be built in the next project iteration.</td>
<td>LSC</td>
</tr>
<tr>
<td>ASR.INTEGRATION.05: Uniformity of integration mechanism across components</td>
<td>Generic integration mechanism publish-subscribe</td>
<td>IP</td>
</tr>
</tbody>
</table>
## Architecture Significant Requirements

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Realisation</th>
<th>Responsible Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>is used, which is not bound to any technology and provides asynchronous communication.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**ASR.INTEGRATION.06: Integration with utilities information services**

The system should provide mechanisms to communicate with different interfaces of the information services. It should perform such interactions based on RESTful principles.

**ERT**

### Operation Requirements

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Realisation</th>
<th>Responsible Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static Plan and Facilities Visualizer components provides interface to upload this data and visualize them.</td>
<td></td>
<td>ERT</td>
</tr>
</tbody>
</table>

**ASR.OPERABILITY.01: Import of Initial facility plans and so on**

Once the RESCUER app is installed, it generates a user id and uses this id to communicate with the backend.

**MS**

**ASR.OPERABILITY.03: Easy installation**

The app can be installed from the app-stores and do not need any configuration except optional profile information.

**MS**

**ASR.OPERABILITY.04: Initial request of sensor data recording**

ERT provides the interface to generate the sensor recording request.

**ERT**

**ASR.OPERABILITY.05: Minimum operational effort**

Components mostly do automatic analysis, visualisation and no special human intervention is needed to operate the system.

**All**

**ASR.OPERABILITY.06: Initial information request to the utilities information services**

The ERT need to check whether the utilities information services are available before requesting data.

**ERT**

**ASR.DEMO.01: Efficient preparation for evaluation and demonstration**

The evaluation partners will create test data and environment.

**All**

**ASR.INSTALLABILITY.01: Download and installation of RESCUER app**

The RESCUER app will be made available in the respective app stores and can be downloaded and installed directly from there.

**MS**

### Availability

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Realisation</th>
</tr>
</thead>
<tbody>
<tr>
<td>All backend components will be running in the server which</td>
<td>All</td>
</tr>
<tr>
<td>Architecture Significant Requirements</td>
<td>Realisation</td>
</tr>
<tr>
<td>---------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>provides 99.99% availability per month. But in real commercial deployment, multiple replicated servers would be required to ensure the expected availability.</td>
<td></td>
</tr>
<tr>
<td>ASR.AVAILABILITY.02: 24/7 planned downtimes of the RESCUER backend</td>
<td>Backend components and infrastructure operators will perform their maintenance task during the planned downtimes.</td>
</tr>
<tr>
<td>ASR.AVAILABILITY.03: Unplanned downtimes of the backend due to hardware failure</td>
<td>Additional free servers will be kept ready. In case any server fails, other can be started within minutes.</td>
</tr>
<tr>
<td>ASR.AVAILABILITY.04: Unplanned downtimes of the backend due to software failure</td>
<td>Additional free servers and fresh software copy will be kept ready. In case any backend component fails, other can be started within minutes. Software problems will be immediately reported to the development team.</td>
</tr>
<tr>
<td>ASR.ROBUSTNESS.01: Robustness against arbitrary user navigation</td>
<td>Elaborate user interfaces with well-defined components to avoid wrong interactions with the ERT.</td>
</tr>
<tr>
<td>ASR.ROBUSTNESS.02: Robustness against arbitrary user input</td>
<td>Each input from the users will be validated before processing.</td>
</tr>
<tr>
<td>ASR.ROBUSTNESS.03: Robustness against unstable network connections</td>
<td>MS will use Ad-hoc P2P network in case of unstable network. Or at least it will store the messages generated during that period of time to send them afterwards.</td>
</tr>
<tr>
<td>ASR.ROBUSTNESS.04: Robustness against no network connections</td>
<td>MS will use Ad-hoc P2P network in case of no network connections are available.</td>
</tr>
<tr>
<td>ASR.ROBUSTNESS.05: Robustness against crashes of mobile application</td>
<td>MS will persist all relevant information. In case it crashes, once restarted, it should be able to start smoothly based on the persisted data.</td>
</tr>
<tr>
<td>ASR.ROBUSTNESS.06: Robustness against ERT crashes</td>
<td>ERT will also persist relevant information and, in case it</td>
</tr>
<tr>
<td>Architecture Significant Requirements</td>
<td>Realisation</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>crashes, it should be able to start smoothly based on the persisted data.</td>
<td></td>
</tr>
</tbody>
</table>

### Scalability

<table>
<thead>
<tr>
<th>ASR.SCALABILITY.01: Initial load during first evaluation</th>
<th>One server machine per component will be sufficient for the first evaluation scenarios.</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASR.SCALABILITY.02: Scaling for large number of apps</td>
<td>The required backend components need to be installed in multiple servers. Appropriate queuing mechanism will be established to parallelize the processing. Detailed concepts might be built in the next project iteration.</td>
<td>All</td>
</tr>
<tr>
<td>ASR.SCALABILITY.03: Scaling over large number of events or industrial parks</td>
<td>Messages will be augmented with event id to distinguish the messages of an event from messages of another event. Each individual component should process the individual events separately.</td>
<td>All</td>
</tr>
</tbody>
</table>

### Safety

<table>
<thead>
<tr>
<th>ASR.SAFETY.01: Mobile user is not endangered</th>
<th>MS will be flexible and allow sending of partial reports.</th>
<th>MS, ERT</th>
</tr>
</thead>
</table>

### Upgradeability

<table>
<thead>
<tr>
<th>ASR.UPGRADEABILITY.01: No data loss during app upgrade</th>
<th>Profile and any other necessary configuration will be persisted and during upgrades still be kept persisted.</th>
<th>MS</th>
</tr>
</thead>
</table>

### Auditability

<table>
<thead>
<tr>
<th>ASR.AUDITABILITY.01: Backend logging</th>
<th>All analysis components log every individual raw message.</th>
<th>All</th>
</tr>
</thead>
</table>

### Usability

<table>
<thead>
<tr>
<th>ASR.USABILITY.01: Simple user interface for the mobile app</th>
<th>The UI has to be really simple and should avoid all kinds of fancy complex interactions.</th>
<th>MS, ERT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASR.USABILITY.02: User should receive immediate response</td>
<td>MS should be responsive and, show messages to the user as soon as it gets feedback from the server.</td>
<td>MS</td>
</tr>
<tr>
<td>ASR.CONSUMABILITY.03: Integrated with</td>
<td>It will be explored in the last</td>
<td>MS</td>
</tr>
</tbody>
</table>
## Architecture Significant Requirements

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Realisation</th>
<th>Responsible Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>Large event app</td>
<td>Project iterations</td>
<td></td>
</tr>
<tr>
<td>ASR.USABILITY.04: Customizable user interface for the ERT</td>
<td>The UI should provide customizations for the different user profiles and needs. The user interface should be dynamic (we chose it to be stored in the Amazon S3. The mobile solution will request information and populate the interface)</td>
<td>ERT</td>
</tr>
</tbody>
</table>

### Variability

<table>
<thead>
<tr>
<th>Variability</th>
<th>Realisation</th>
<th>Responsible Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASR.VARIABILITY.01: Multiple existing systems of workforce</td>
<td>External system connector should be configured and should have plugin mechanism to connect to available workforce legacy systems.</td>
<td>ERT, LSC</td>
</tr>
<tr>
<td>ASR.VARIABILITY.02: Adaptability for EU and Brazil environments</td>
<td>It will be explored in last project iteration.</td>
<td>All</td>
</tr>
<tr>
<td>ASR.VARIABILITY.03: Adaptability for large-scale event and industrial park environments</td>
<td>It will be explored in last project iteration.</td>
<td>All</td>
</tr>
<tr>
<td>ASR.VARIABILITY.04: Multiple mobile platforms</td>
<td>Two versions (iOS and Android) are being developed.</td>
<td>MS</td>
</tr>
<tr>
<td>ASR.VARIABILITY.05: Multiple social media</td>
<td>Social media connector should be configured and should have plugin mechanism to connect to different social media.</td>
<td>SMC</td>
</tr>
<tr>
<td>ASR.VARIABILITY.06: Multiple web interface</td>
<td>ERT will be developed in HTML5 and can be easily accessible through tablet or PC.</td>
<td>ERT</td>
</tr>
<tr>
<td>ASR.VARIABILITY.07: Multiple types of users for mobile solutions</td>
<td>MS will configure itself automatically based on the user category.</td>
<td>MS</td>
</tr>
<tr>
<td>ASR.VARIABILITY.08: Multiple types of users for web interface</td>
<td>It will be explored in last project iteration.</td>
<td>ERT</td>
</tr>
</tbody>
</table>

### Reliability

<table>
<thead>
<tr>
<th>Reliability</th>
<th>Realisation</th>
<th>Responsible Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASR.RELIABILITY.01: Emphasis should be given on professionals</td>
<td>Users are categorized into different groups. Analysis components will take into consideration the credibility of each group during analysis.</td>
<td>TA, IA, VA, CA, ERT</td>
</tr>
<tr>
<td>ASR.RELIABILITY.02: No wrong conclusion about the situation</td>
<td>Multiple analysis components will be crosschecked to identify wrong calculations.</td>
<td>TA, IA, VA, CA, ERT</td>
</tr>
<tr>
<td>ASR.RELIABILITY.03: Reliability of data</td>
<td>Reliability of data will be</td>
<td>TA, IA, VA, CA,</td>
</tr>
<tr>
<td>Architecture Significant Requirements</td>
<td>Realisation</td>
<td>Responsible Components</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>increased by aggregating results of multiple analysis components and multiple sources of same input.</td>
<td>ERT</td>
<td>ERT</td>
</tr>
</tbody>
</table>

**Performance**

<table>
<thead>
<tr>
<th>ASR.PERFORMANCE.01: Response time for user interaction</th>
<th>UI logics are local to the client component to be faster.</th>
<th>MS, ERT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASR.PERFORMANCE.02: Sending partial user-interaction data</td>
<td>If the user stops interacting with MS while preparing some incident report, MS will automatically send the partial report. It will also send partial reports periodically in case of long interactions.</td>
<td>MS</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ASR.PERFORMANCE.03: End-to-end response time</th>
<th>All components should be compliant with the individual latency requirements.</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASR.PERFORMANCE.04: Performance on visualisation</td>
<td>Look for efficient rendering technologies.</td>
<td>ERT</td>
</tr>
<tr>
<td>ASR.EFFICIENCY.05: Working in the low power mode</td>
<td>MS will identify the low power mode and will stop doing high resource consuming tasks (by default).</td>
<td>MS</td>
</tr>
</tbody>
</table>

**Security**

<table>
<thead>
<tr>
<th>ASR.SECURITY.01: Separation of views for different ERT users</th>
<th>It will be explored in the last project iteration.</th>
<th>ERT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASR.SECURITY.02: Overall data privacy</td>
<td>It will be explored in the last project iteration.</td>
<td>All</td>
</tr>
<tr>
<td>------------------------------------------------------------</td>
<td>--------------------------------------------------</td>
<td>-----</td>
</tr>
<tr>
<td>ASR.SECURITY.03: Log browsing in ERT</td>
<td>ERT will provide the interface to browse through the logs behind every analysis output it shows in the dashboard.</td>
<td>ERT</td>
</tr>
<tr>
<td>ASR.SECURITY.04: Logging of analysis</td>
<td>All analysis components will log every raw message it uses to come to some conclusion. The analysis results will always be associated with the raw messages from the MS.</td>
<td>TA, IA, VA, CA</td>
</tr>
<tr>
<td>ASR.SECURITY.05: User should be able to delete his information and nothing will be used afterwards</td>
<td>The message about the user’s wish will be immediately broadcasted to all components. Individual components will delete the data related to this user.</td>
<td>All</td>
</tr>
<tr>
<td>Architecture Significant Requirements</td>
<td>Realisation</td>
<td>Responsible Components</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>ASR.SECURITY.06: News should be anonymous</td>
<td>News does not contain information directed to any particular user. The RESCUER platform will not send news directly; a human user will check any message and then publish it for broadcasting.</td>
<td>ERT</td>
</tr>
</tbody>
</table>
7. Conclusions

This deliverable proposes the overall system architecture for the RESCUER platform. This architecture was created based on several requirement documents as well as direct communication with the development and operation partners. This deliverable documents the architectural significant requirements. It proposes a functional decomposition of the system taking into consideration the quality requirements. Key architectural concepts are elaborated – the context delineation of the system, scope of the system, client-server separation, among others. Various perspectives of the system are then shown as different views for different stakeholders. The main perspectives in this deliverable are the sub-systems perspective, data perspective, development perspective, deployment perspective, and task distribution perspective. Crosscutting quality requirements are broken down into requirements on the level of individual components. Individual component developers need to consider those requirements when designing their component in order to ensure end-to-end quality guarantee for RESCUER.

In this second iteration of Task 1.2 (System Architecture), we improved the system architecture for the RESCUER platform proposed in deliverable D1.2.1. This refinement was based on several requirements documents (D1.1.2, D2.1.2 and D4.1.2) from the second project iteration as well as implementation feedback from the development and operation partners from the first iteration. The outcome of this second iteration of Task 1.2 should be used as a guideline by the development and operation project partners when implementing the follow up interaction and, when necessary, for adapting the individual components developed in the first project iteration. Furthermore, it can be used by other stakeholders to understand the general internal views of the RESCUER system. In the last project iteration the emphasis will be on crowd-steering and communication to external organizations. As it has been done so far, the proposed architecture will continue to be refined in the last project iteration based on the increasingly better understanding of the requirements and of the domain.
References

Glossary

**Command and Control Centre** Group of people and tools assigned to evaluate risks and make decisions in an emergency and/or crisis in an industrial area or at a large-scale event, usually at the same physical place.

**Communication Infrastructure** Component of the RESCUER platform whose goal is to support the information flow between the crowd and the command centre.

**Data Analysis Solutions** Component of the RESCUER platform whose goals are 1) fusing similar data coming from different eyewitnesses, 2) analysing photos, videos, and text messages in order to extract information such as the type of incident, the position and dimensions of the affected area, people density, surrounding sources of further danger, evacuation routes, and possible approach routes for the formal responders.

**Emergency** Critical situations caused by incidents, natural or man-made, that require measures to be taken immediately to reduce their adverse consequences to life and property.

**Emergency Response Toolkit** Component of the RESCUER platform whose goals are to: 1) get contextual information about the emergency, 2) ask eyewitnesses and formal responders for relevant missing information, 3) give instructions to eyewitnesses, first responders and potentially affected people or companies, and 4) communicate the emergency to the media, public authorities, and the general public in a context-aware way. The emergency response toolkit is meant to be used primarily by the command and control centre staff.

**Mobile Crowdsourcing Solution** Component of the RESCUER platform whose goal is to support eyewitnesses and formal responders in providing the command and control centre with information about an emergency situation, taking into account the different smartphones that might be used and how people interact with smartphones under stress.

Abbreviations

**RESCUER** Reliable and Smart Crowdsourcing Solution for Emergency and Crisis Management

**UI** User Interface

**ASR** Architecture Significant Requirements

**C&C** Command and Control Centre

**ERT** Emergency Response Toolkit

**ACES** Architecture Centric Engineering Solutions

**ADF** Architecture Decomposition Framework

**SMSC** Short Message Service Centre

**CCC User** Command and Control Centre User
Acronyms for the Modules to be developed

**MS** Mobile Solution
**SDR** Sensor Data Recorder
**ADHOC** Ad-hoc P2P Network
**SEN_RECV** Sensor Data Receiver
**SMS_RECV** SMS Receiver
**IP** Integration Platform
**BLOB** Blob Storage
**SDA** Sensor Data Analysis
**TA** Text Analysis
**IA** Image Analysis
**VA** Video Analysis
**CA** Combined Analysis
**ERT** Emergency Response Toolkit