RESCUER

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 D1.2.1
System Architecture 1

Due date of deliverable: PM 8
Actual submission date: June 15, 2014

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 (Fraunhofer)

Dissemination level		
PU	Public	✓
PP	Restricted to other program participants (including Commission Services)	
RE	Restricted to a group specified by the consortium (including Commission Services)	
CO	Confidential, only for members of the consortium (including Commission Services)	

Executive Summary

System Architecture

This document presents deliverable D1.2.1 (System Architecture 1) of project FP7-614154 | CNPq-490084/2013-3 (RESCUER), a Collaborative Project supported by the European Commission and MCTI/CNPq (Brazil). Full information on this project is available online at http://www.rescuer-project.org.

Deliverable D1.2.1 provides the results of Task 1.2 (System Architecture) for the first project iteration. Therefore it summarizes the activities performed to define the architecture of the RESCUER system. The outcome is based on the input of several requirements deliverables D1.1.1, D2.1.1, D4.1.1, RESCUER project proposal and most importantly communication with the user and development partners directly.

It documents the main architectural drivers of the overall RESCUER system – i.e. 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 (for example, development perspective, deployment perspective, among others) are described as different views for different stakeholders. The architectural design decisions are documented.

RESCUER has a layered architecture. The overall system is designed in a modular way and 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. Components use this mechanism to have asynchronous communication among them.

As this task is responsible for the overall platform, the individual components' design decisions are left open to the respective project partners. The cross-cutting quality requirements are broken down on an individual component level. 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 document is a living document. As the requirements evolve and understanding about the domain matures, the system architecture will be adapted accordingly. In this first iteration the overall system architecture is presented with emphasis on information gathering from crowd sources and display of the analysed data. The next two iterations will emphasis on follow-up interaction and crowd-steering. This document will be helpful to understand the overall system's big-picture and it should be used by project partners as a starting point to design their own components.

List of Authors

Taslim Arif – Fraunhofer

Paulo Jr – UFBA

List of Internal Reviewers

Tobias Franke – DFKI

Silas Graffy – VOMATEC

Karina Villela – Fraunhofer

RESCUER

Contents

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 incidents in industrial areas and on 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. As the system is viewed differently by each stakeholder, the architecture engagement purposes are described for different stakeholders in the following paragraphs separately.

- *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 cross-cutting 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 as well because a running 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 intended to be a working document, which means it will continuously evolve in 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. Approach to Create System Architecture

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

- ASR Elicitation:
    - Various requirements documents namely D1.1.1, D2.1.1, D4.1.1 and project proposal are taken into consideration to generate ASRs.

- Further requirements are collected from the discussions with the development and user organisations of the consortia.
- Experiences of Fraunhofer from previous architectural projects are also revisited.
- Input from standard quality models like ISO 25010 are also taken into account.

Finally the requirements are validated and finalized after discussion with the partners in the First European Consortium meeting in Kaiserslautern.

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

- Documenting the Architecture
  - Models are created in enterprise architect (EA).
  - Different views, models are categorized in this document to be understood and used by different partners and external stakeholders.

1.3. Partners' Roles and Contributions

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

1.4. 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 conclusion of this document.

2. Conceptual Overview of the RESCUER System

RESCUER is a crowd-sourced information system solution for operational forces and visitors of large events or employees of industrial parks. It consists of a mobile app and a portal application using web technologies. The mobile app will support the information collection like type of incident, severity, current situation (images, videos), and so on. It will be used by the operational forces' staffs located at the premises and visitors or employees. The portal application and some backend components will 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 operational forces' legacy systems, Govt. organisations, and guide the crowd.

2.1. Mission

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

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

Huge amount of data is expected to be collected from the crowd. It is almost impossible for humans to both analyse these data - consisting of 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 have a good understanding of the incident immediately.

Figure 1 shows the mission of the RESCUER system. The RESCUER platform (shown as box) is supposed to address the goals described above. The system should realize the following concept to achieve those goals. It is assumed that visitors of large-scale event will use the mobile crowd sourcing solution of RESCUER. Through this app the visitors including the operational forces will send the incident reports to the backend server. Communication Infrastructure will do the job of receiving data from the apps. 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 will also generate reliable and periodic messages for various stakeholders (general public, authorities, media, and so son) to inform them about the situation. As advanced features RESCUER will steer the crowd and also manage the operational forces in the field.

- Exploiting the power of crowd-sourcing
- Using advanced techniques for data analysis and visualization
- Supporting mobility and providing context aware services

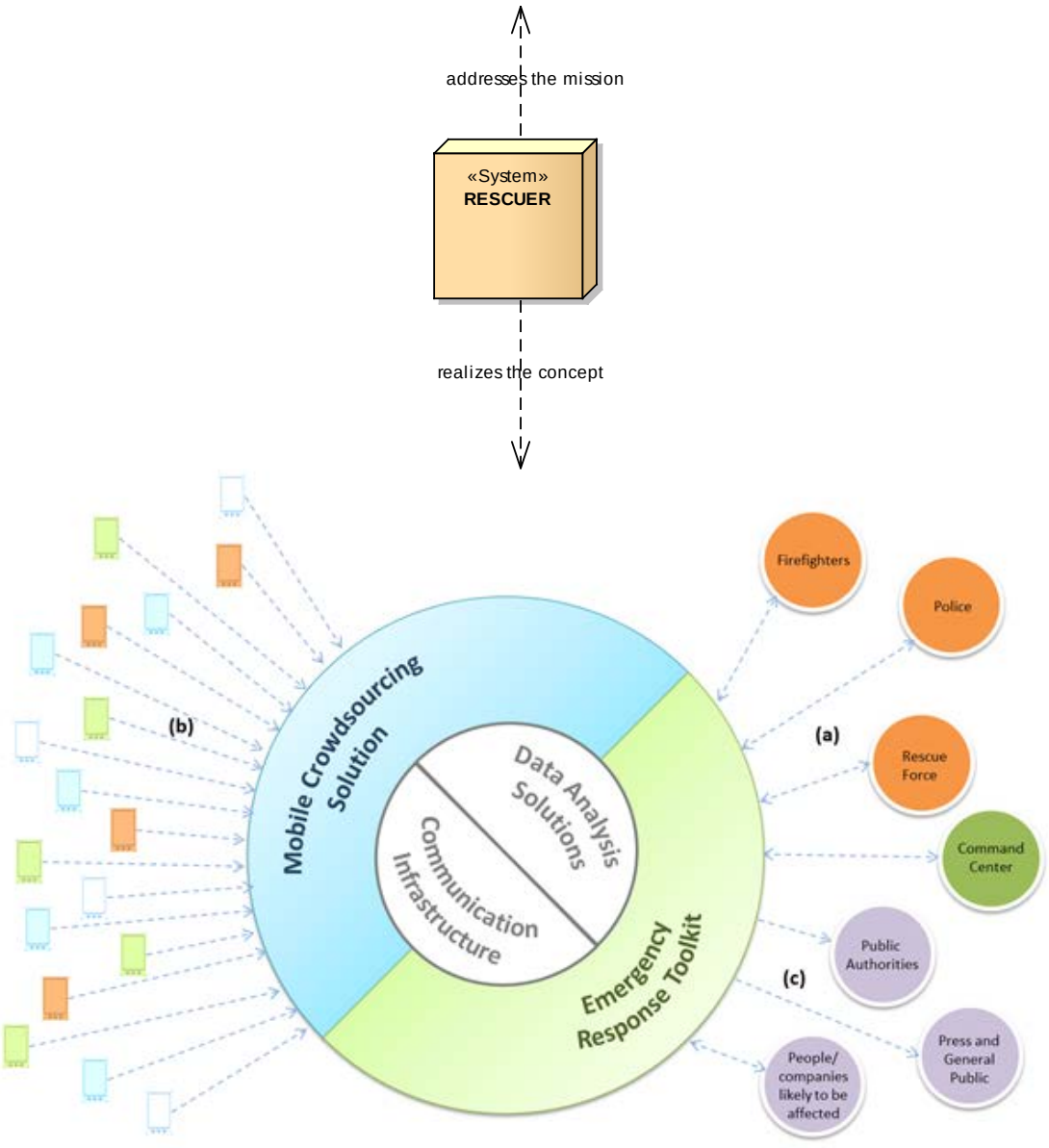

Figure 1: Mission of RESCUER platform

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 taking this goal into consideration. We have the emergency situation comprising of cause of emergency (fire/ explosion/ gas-leak etc.), injuries, damages of property, and so on. The *Context Sensor* component is responsible for sensing the environment. The *Information Processing and Decision Making* component receives the sensed data, analyses them and visualizes them efficiently in the command and control centre. The command and control centre decides on some actions to improve the situation. Messages are sent back to the *Effector* component which does something physically on the emergency environment to improve the situation. The components are described in detail below.

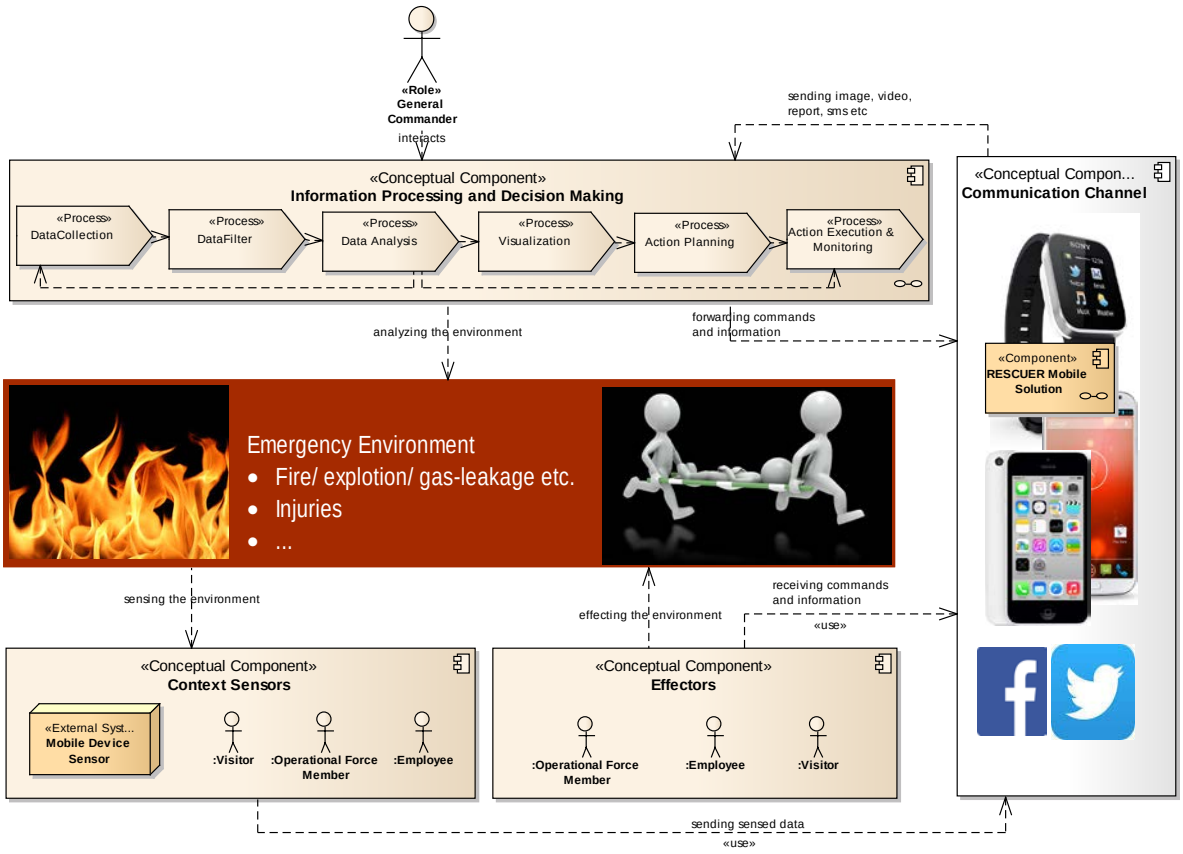

Figure 2: RESCUER conceptual architecture

- *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, operational forces' members are the people who are truly on the spot during the incident. They are the people who can observe the situation and report. Mobile device sensors can also sense the location and movement of the people on the spot.
- *Effectors*: This conceptual component represents the stakeholders and systems that do something on the spot to handle the emergency situation in practice. The sensing

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, 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 and smart watches are the main communication channels selected in the RESCUER project.

2.3. Scope and Iteration

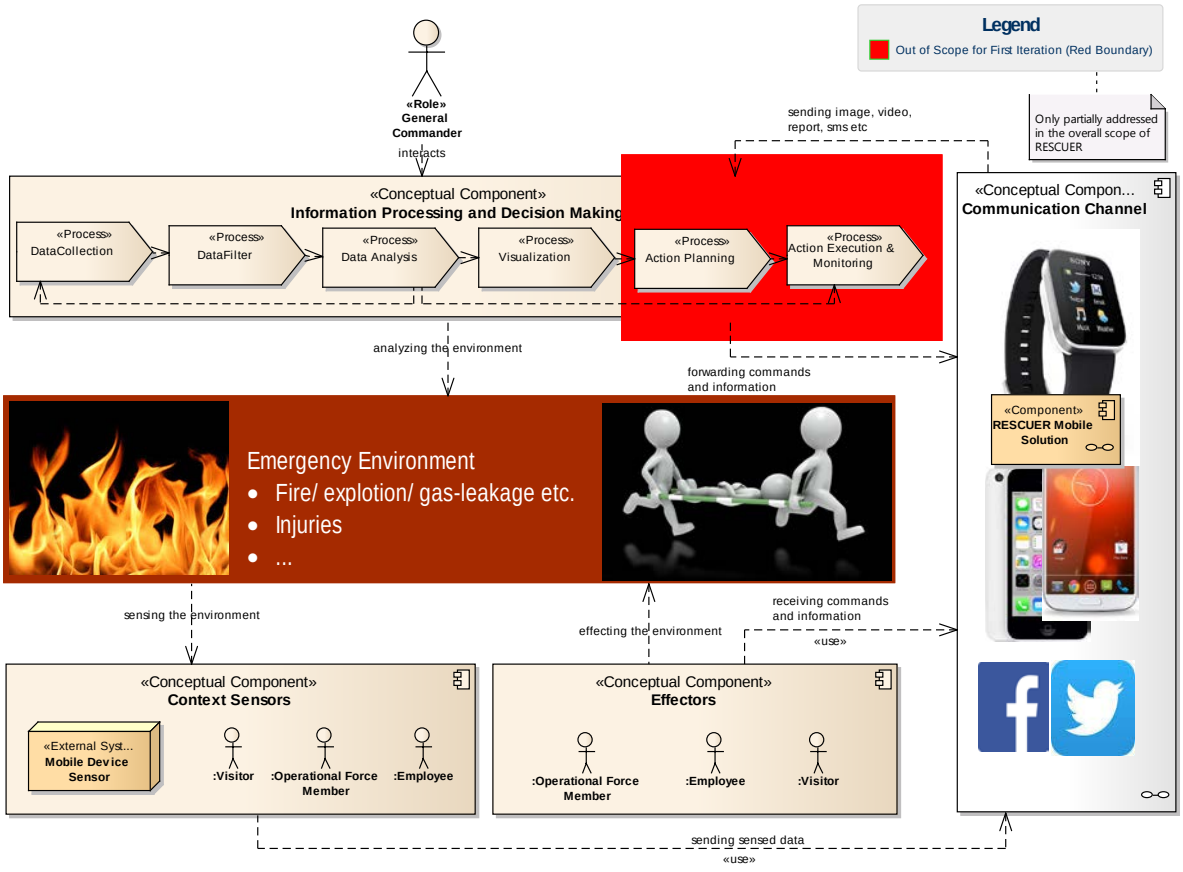

Figure 3: Scope of the first project iteration

As this project is planned to have three iterations, it is important to clearly understand what features are part of the first iteration and what are for the next iterations. For the first iteration the scope is 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. Action plan and monitoring, follow-up interactions with the

crowd, crowd steering, workforce management, integration with the legacy operational forces' systems, and generation of reliable news will be addressed in the subsequent phases.

2.4. Locations and Stakeholders

To understand the overall system architecture, it is important to understand the locations and stakeholders. Figure 4 shows the stakeholders and organisations categorized into different locations.

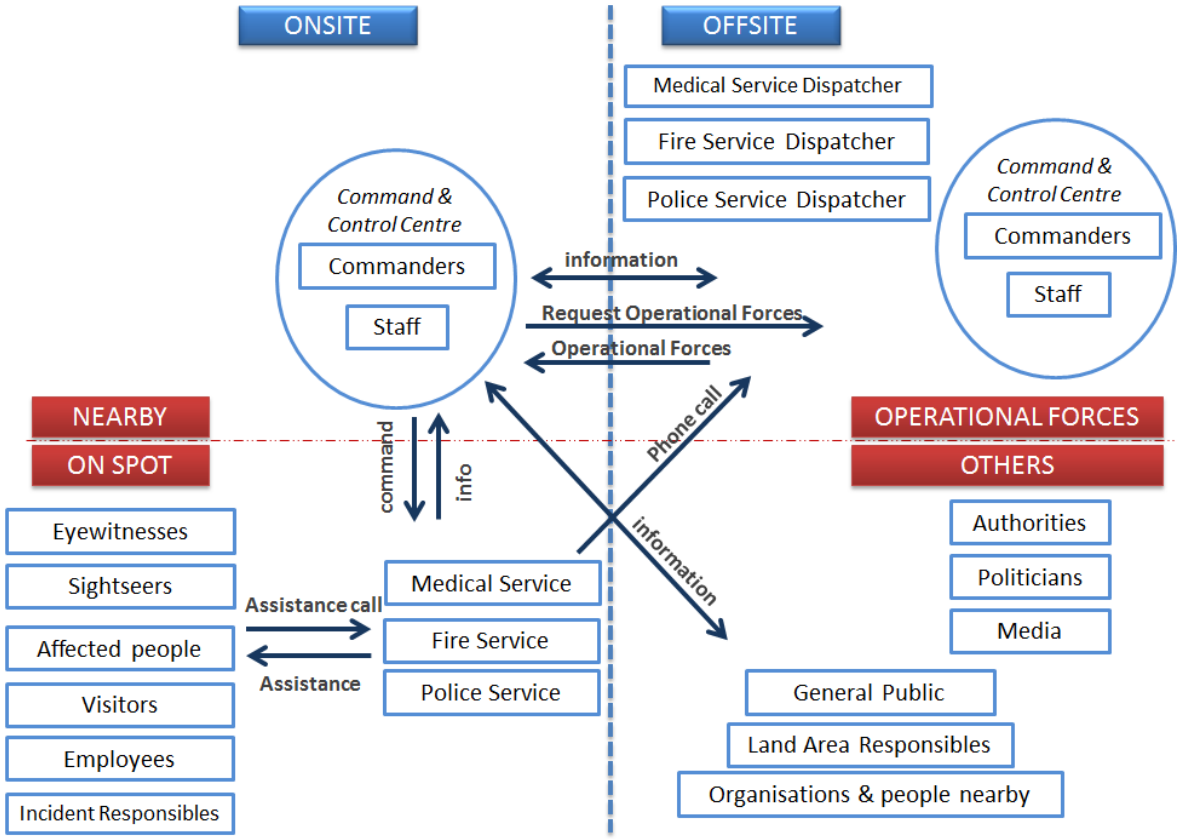

Figure 4 : Locations and stakeholders (users)

2.4.1. Stakeholders

From the architectural point of view the 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 all stakeholders with detailed description can be found in deliverable D4.1.1 (Organisational Behaviour and Structures in Emergency Situations).

- Visitors of large scale events:
    - General Visitors: 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.
    - Visitors 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 emergencies.
- Employees of industrial parks: These are the people who work for different companies in an industrial park.
- Operational forces: Three types of operation forces 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 a general commander plan and monitor everything from the command and control centre. The types of operational forces are:
    - Medical Service: The unit that is responsible for health related issues during an emergency situation.
    - Fire Service: The unit that is responsible for extinguishing fires and this sort of tasks during an emergency situation.
    - 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
    - On-site: The office that manages, monitors and controls the emergency operation on-site. General Commanders of operational forces hold the office.
    - Off-site: The office that manages, monitors and controls the emergency operation off-site. General Commanders of operational forces hold the office.
- Dispatchers
    - Medical Service: The officer that receives requests regarding medical teams and dispatches them.
    - Fire Service: The officer that receives requests regarding fire-service teams and dispatches them.
    - 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: News agencies that collects news about the emergency situation and broadcasts it to the public.

2.4.2. Locations

The main locations that are relevant to understand the system are:

- On-site-Op 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-Operational Forces: Offices where operational forces 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 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 its first version, this prioritisation distinguishes between the requirements that have to be realized by the end of the first project iteration in July 2014 and later requirements (for the next two iterations). All requirements for the first iteration should be addressed, but maybe not with full functionality. Functionalities will be enhanced in the next iterations.

For the first project iteration, a first production system has to be built, which allows operating the system for the first evaluation in experimental settings.

However, 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 project iteration, 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 to minimize 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 operational forces.
  - RESCUER aims to efficiently manage the emergency by intelligent crowd-steering and effective management of operational forces on the spot.
- RESCUER wants to demonstrate the power of crowd-sourcing, 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 requirement are described in Tables 1 and 2. Table 1 shows those to be realized during the first iteration of the RESCUER project and Table 2 shows the ones to be realized in the next iterations of the project. This deliverable describes all the requirements presented in Table 1 and 2.

Table 1: Architecture-significant requirements to be realized during the first project Iteration

ASR ID	Title
ASR.FUNCTIONALITY.01	Sensor data recording of mobile devices of visitors, employees and operational forces
ASR.FUNCTIONALITY.02	User interaction data collection
ASR.FUNCTIONALITY.03	Multimedia data collection
ASR.FUNCTIONALITY.04	Sending sensor data to the backend
ASR.FUNCTIONALITY.05	Sending user interaction data to the backend
ASR.FUNCTIONALITY.06	Sending multimedia data to the backend
ASR.FUNCTIONALITY.07	Receiving sensor data at the backend
ASR.FUNCTIONALITY.08	Receiving user interaction data at the backend
ASR.FUNCTIONALITY.09	Receiving multimedia data at the backend
ASR.FUNCTIONALITY.10	Feedback on data received from the mobile solution
ASR.FUNCTIONALITY.12	Automatic analysis of sensor data
ASR.FUNCTIONALITY.13	Automatic analysis of user interaction data
ASR.FUNCTIONALITY.14	Automatic analysis of multimedia data
ASR.FUNCTIONALITY.16	Combined analysis of individual analysis results
ASR.FUNCTIONALITY.17	Generating report ID
ASR.FUNCTIONALITY.19	Static plan and facilities visualisation
ASR.FUNCTIONALITY.20	Visualisation of analysed data
ASR.FUNCTIONALITY.21	Visualisation of the logs behind the analysis results
ASR.FUNCTIONALITY.23	Receiving messages from the RESCUER backend
ASR.FUNCTIONALITY.24	Sending profile information to the RESCUER backend
ASR.FUNCTIONALITY.25	Profile information collection and persistence

Table 2 - Architecture-significant requirements to be realized in subsequent project iterations

ASR ID	Title
ASR.FUNCTIONALITY.11	Receiving SMS
ASR.FUNCTIONALITY.15	Automatic analysis of SMS
ASR.FUNCTIONALITY.18	Using ad-hoc P2P network
ASR.FUNCTIONALITY.22	Visualising messages from the backend
ASR.FUNCTIONALITY.26	Collaboration among RESCUER users
ASR.FUNCTIONALITY.27	Plan actions in the command and control centre
ASR.FUNCTIONALITY.28	Monitor actions
ASR.FUNCTIONALITY.29	Manage workforces
ASR.FUNCTIONALITY.30	Providing news
ASR.FUNCTIONALITY.31	Providing information to social media
ASR.FUNCTIONALITY.32	Follow up interaction
ASR.FUNCTIONALITY.33	Crowd steering

Requirements:

- ***ASR.FUNCTIONALITY.01: Sensor data recording of mobile devices of visitors, employees and operational forces***

  The RESCUER backend requests the mobile solution to record sensor data. Mobile solution records the data (GPS for the first iteration). 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 photo or video of the emergency situation through the mobile solution. The mobile solution captures meta-data (time, 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 and 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.

- ***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: 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. This SMS is then made available for analysis.

- ***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 data 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 during a session. 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 most basic reports (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. General Commanders and other staffs in the command and control centre can configure their respective dashboards based on their needs. They can see the crisis map, position of the affected people, position of the operational forces, 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. General Commanders and other staffs in the command and control centre can investigate deep inside the logs that caused RESCUER to draw any specific analysis conclusion.

- ***ASR.FUNCTIONALITY.22: Visualising messages from the backend***

  RESCUER supports the visualisation of the analysed data in the command and control centre. General Commanders and other staffs in the command and control centre can configure their respective dashboards based on their needs.

- ***ASR.FUNCTIONALITY.23: 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.24: 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.25: 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.26: Plan actions in the command and control centre***

  The General Commander in 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.27: Monitor actions***

  The General Commander in the command and control centre uses the RESCUER platform to monitor the status of his actions.

- ***ASR.FUNCTIONALITY.28: Manage workforces***

  Staff in the command and control centre manages his/her workforces in the field through RESCUER. He can deploy workforces to specific areas of the emergency situation, send them orders through messages, among others.

- ***ASR.FUNCTIONALITY.29: Providing news***

  RESCUER generates periodic news and information for the media and authorities. Staff in the command and control centre can use it to prepare final news. The news is sent to the stakeholders outside the area of the emergency situation.

- ***ASR.FUNCTIONALITY.30: 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.31: 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. This is done by understanding the situation of the user without adding stress.

- ***ASR.FUNCTIONALITY.32: Crowd steering***

  RESCUER sends steering messages to the crowd to effectively relocate it. The information sent by the RESCUER platform is reliable and does not make the situation even worse.

3.4. Devtime Requirements

- ***ASR.DEVTIME.01: Documentation of design and code***

  The RESCUER project is intended to be developed in multiple iterations. Moreover, the project might be extended beyond the current time frame and individual partners also want to 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 operational forces' 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 without much effort. The integration mechanism has to be flexible enough to handle heterogeneity in data, protocol, network connection, and so on.

- ***ASR.INTEGRATION.05: Integration of mobile device with smart watch***

  The users of RESCUER mobile solution should be able to use some basic functionalities through smart watches as well. Their smart phones and smart watches have to be integrated so that the users can have a seamless experience with the Rescuer mobile solution.

- ***ASR.INTEGRATION.06: 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.

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 be need to be aware of the insides of each individual components. 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.

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 duty in those places want to use the RESCUER mobile solution. S/He can download and install the RESCUER app from the app stores of Apple or Google. When starting the app, he may enter the profile information and other configuration information (not mandatory and can be entered afterwards too), and then s/he 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 it 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.03: 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.

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 in an initial version for 100-200 test users in 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 emergency reports.

3.7.7. Usability

Requirements:

- ***ASR.USABILITY.01: Simple user interface***
  As RESCUER is to be used in emergency situations, its user interface should not have any information that is not relevant to the current user profile.

- ***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.

3.7.8. Variability

Requirements:

- ***ASR.VARIABILITY.01: Multiple existing systems of operational forces***

  The RESCUER platform should be able to communicate with existing systems used by the operational forces (e.g. police, fire and rescue services).

- ***ASR. VARIABILITY.02: Interoperability in EU and Brazil***

  The RESCUER platform should be adaptable to the European and Brazilian environments.

- ***ASR. VARIABILITY.03: Interoperability in large-scale event and in industrial park***

  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 interface***

  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 operational forces. Visitors can be further classified into two groups – the experienced users and general users. Operational forces 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 users in Emergency Response Toolkit***
  The application interface must adapt itself according to the logged user's profile.

- ***ASR. SECURITY.02: Data of a user should not be shared with others***
  Private user information, such as login and password, must not be visible to other users.

- ***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: News should be anonymous***
  Data analysis modules must log information about data received and data analysis processing.

- ***ASR. SECURITY.07: Overall data security***
  There should be a function that enables the user to delete its personal information.

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 same technologies for every component.

4. Key Architectural Concepts

4.1. Context Delineation

Figure 5 shows the RESCUER system and the external systems and stakeholders around it.

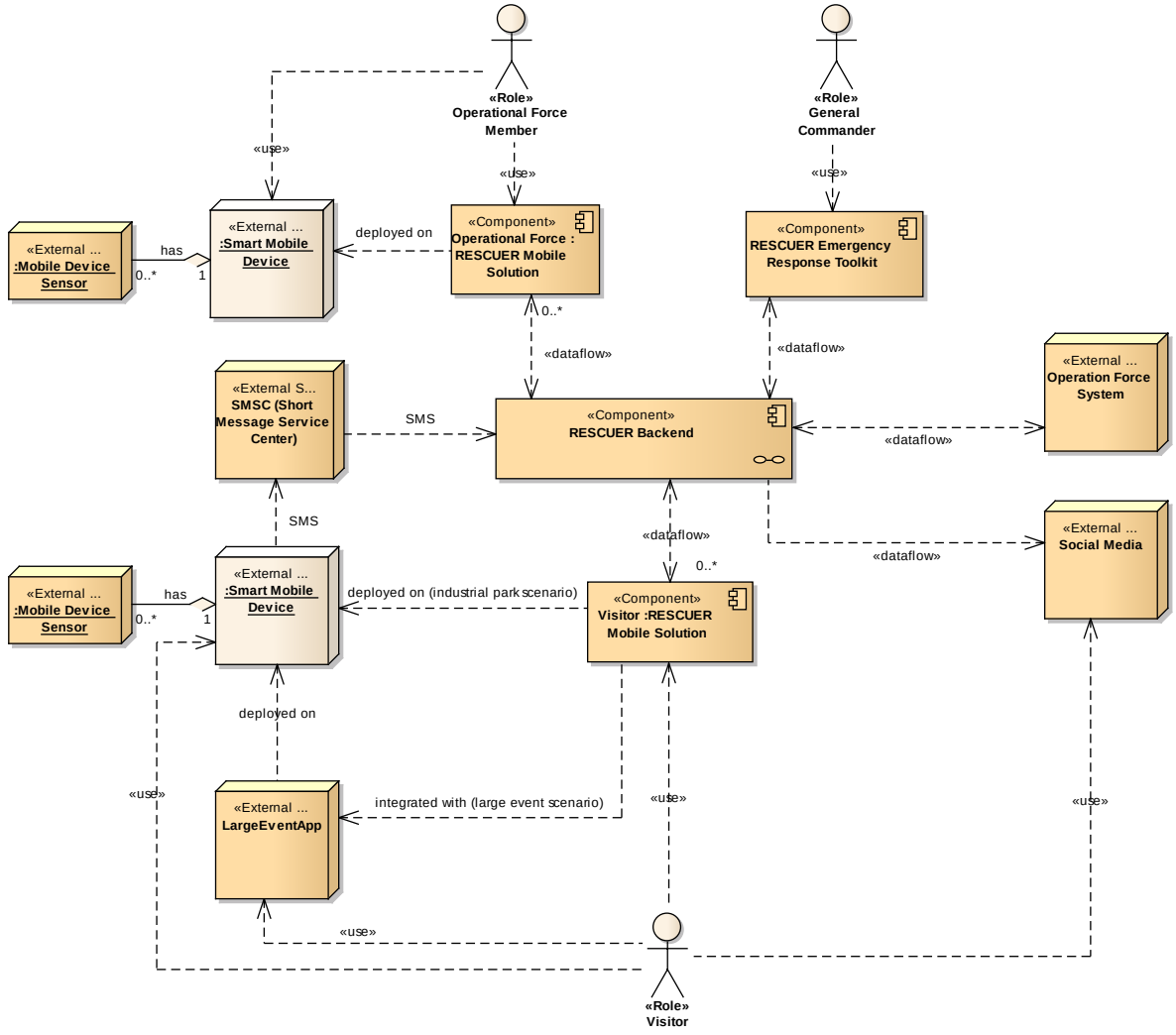

Figure 5 : Context delineation

4.2. Internal Structure

The overall RESCUER system is divided into several layers. Figure 6 shows the layers of the RESCUER system. Figure 7 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 and the staff of different operational forces working on the spot. This layer also shows the guidance and commands that comes 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 apps 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 then doing a combined analysis.
- *Management and Dispatching Layer*: This layer is responsible for managing the crowd and operational forces. 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.

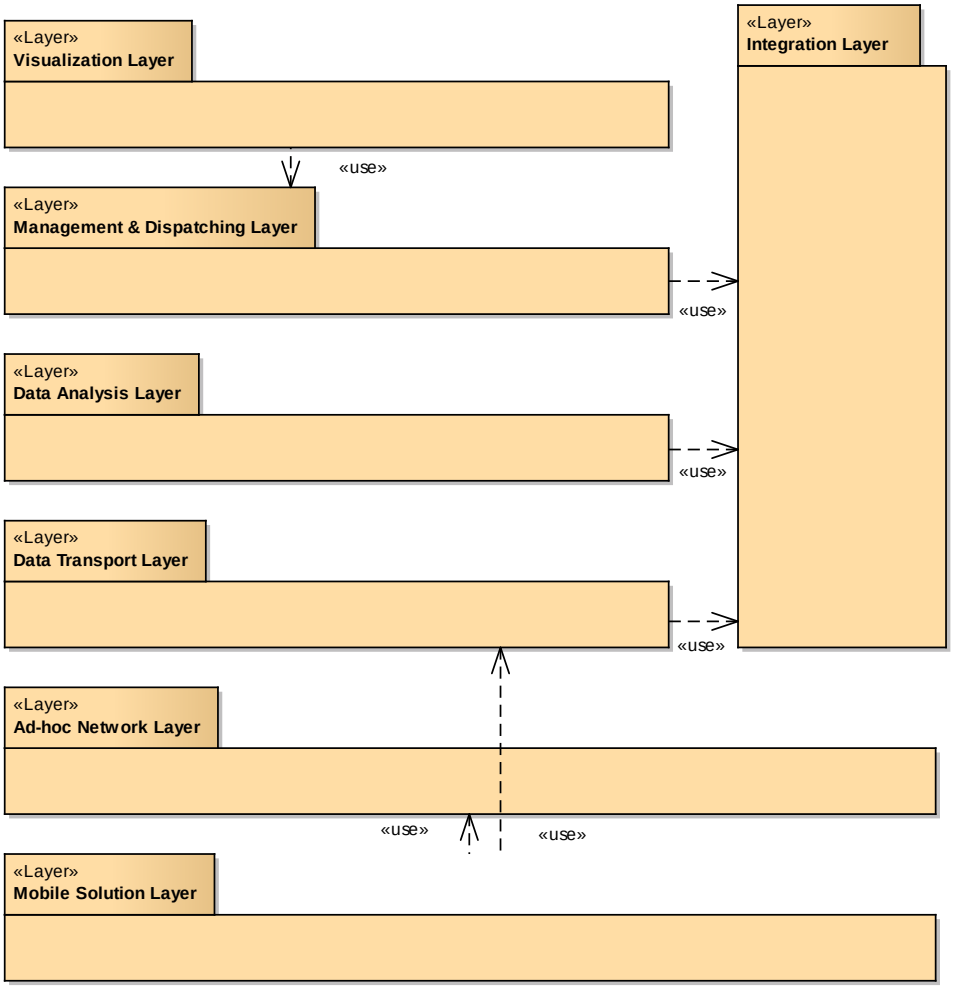

Figure 6 : Layering concept

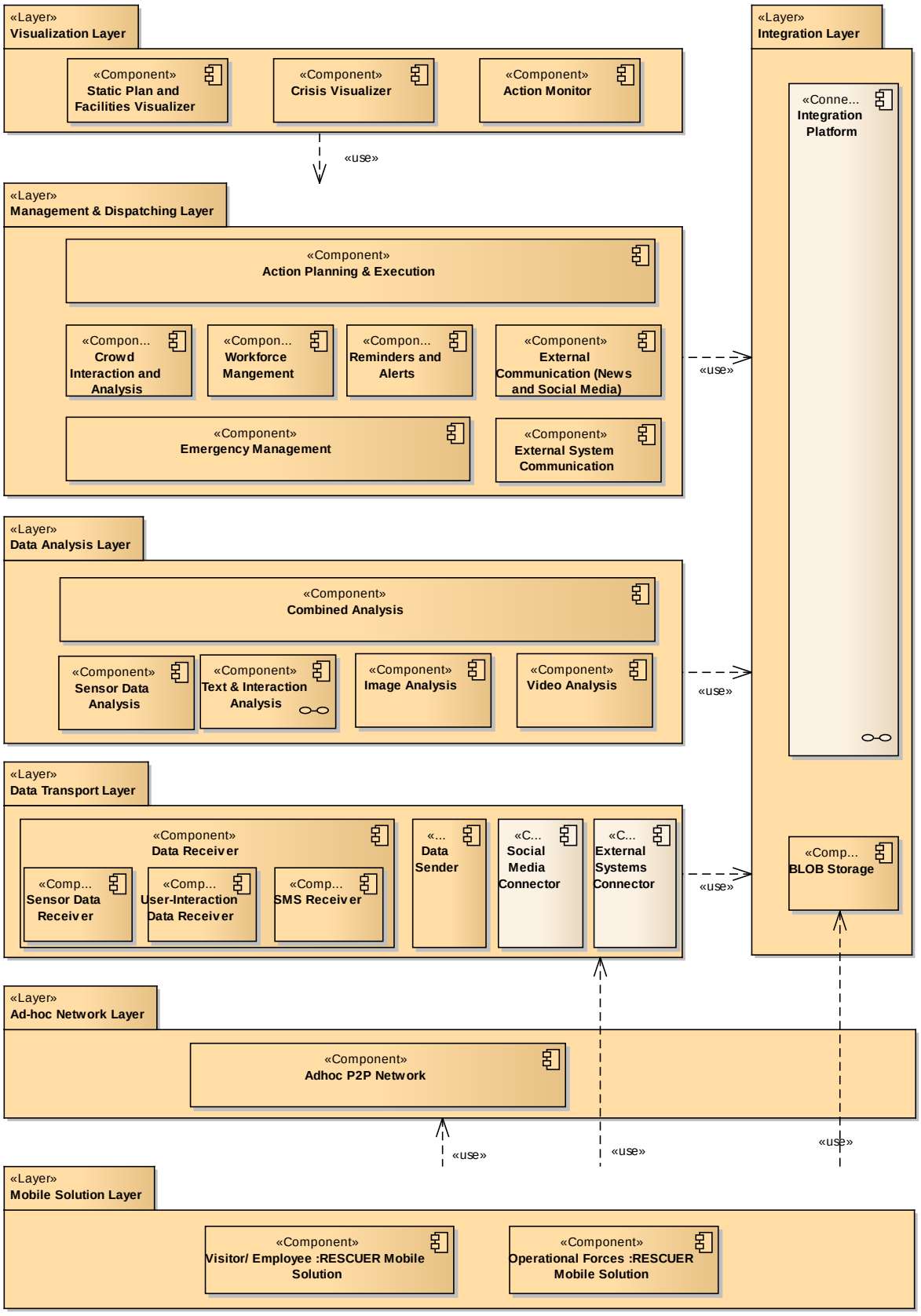

Figure 7 : Layers including functional components

4.3. Client Server Separation

The RESCUER platform can be separated in client-server fashion (Figure 8). This separation is done considering the overall RESCUER platform. Individual component might have also client-server separation within the component.

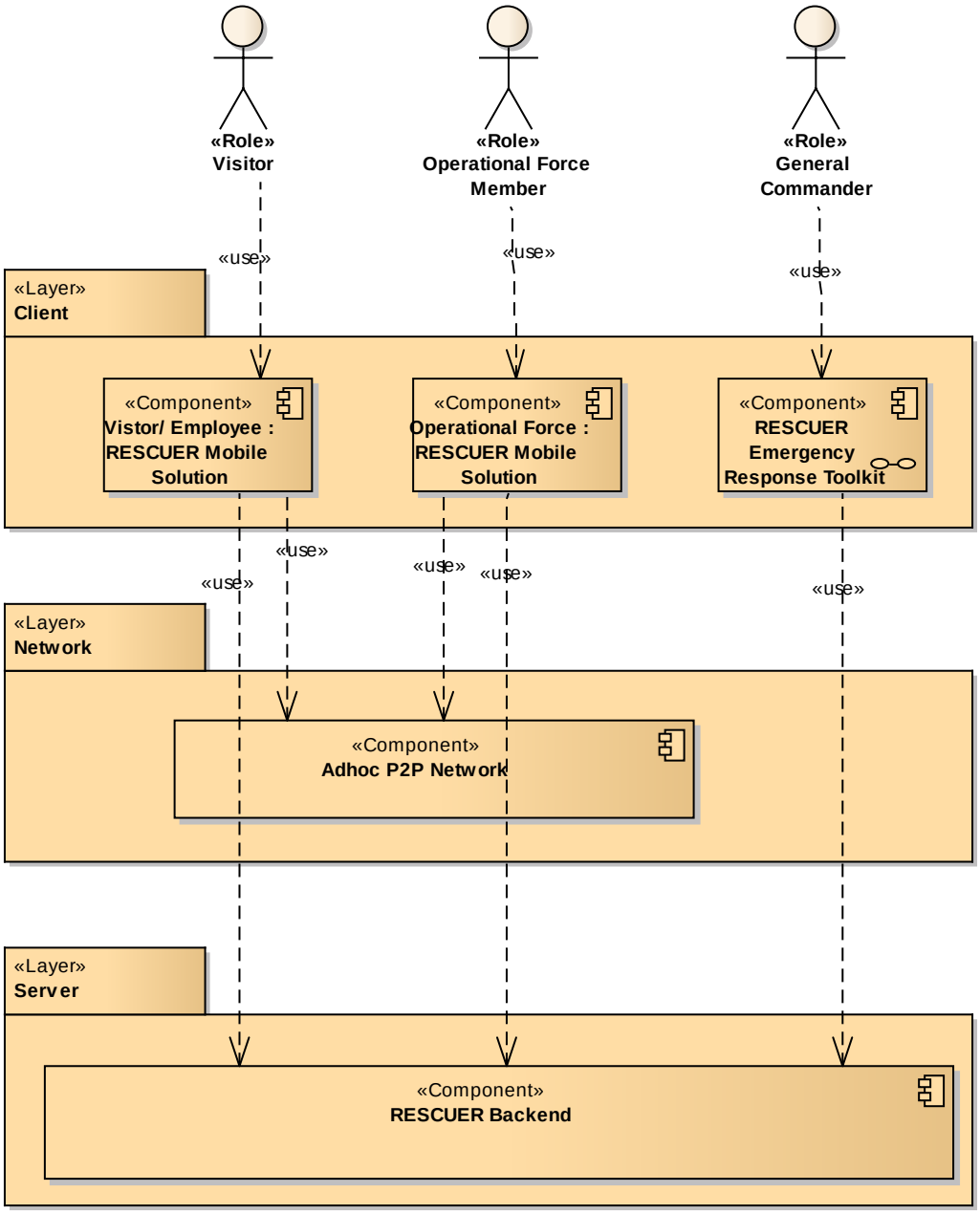

Figure 8 : Client server separation

- 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 operational

forces 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 operational forces will use different MS to communicate with the RESCUER backend. On the other hand General Commanders in the command and control centre will use the EMT 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 9 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.

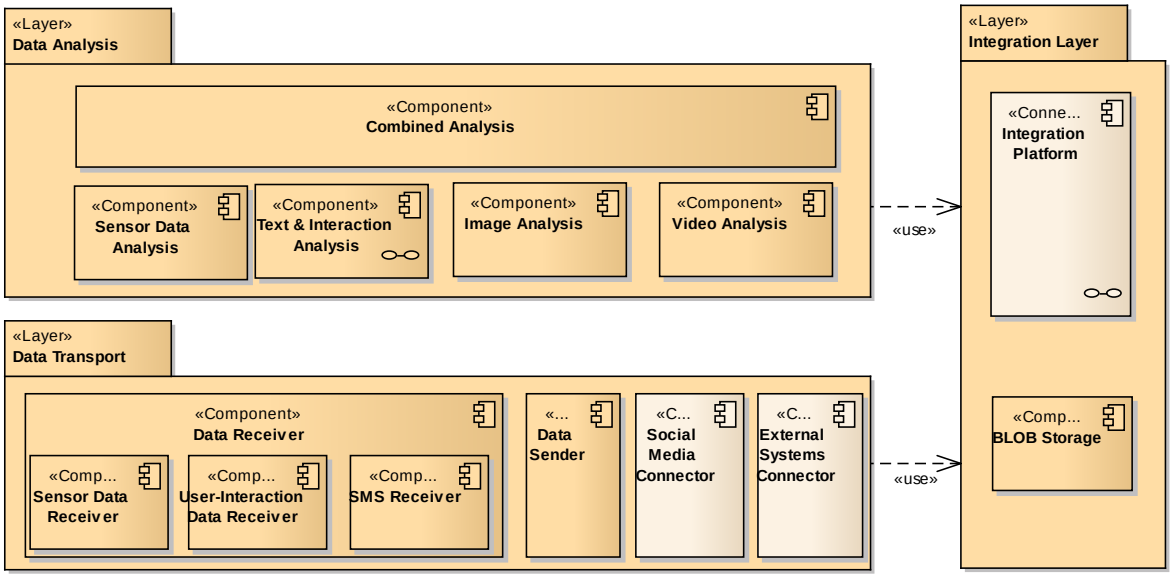

Figure 9 : RESCUER backend

- 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 7 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 10. This figure shows the layers and components that actually realize the Communication Channel. Visitors of large-scale events, employees of industrial park or operation forces would use the RESCUER map to report the incident. 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 11 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 various actions and executing those actions. For example interacting with the crowd, interacting with the operational forces, external communication, reminders and alerts, and so on. 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 operational forces. Furthermore, it stores the data that is shared across the component boundary.

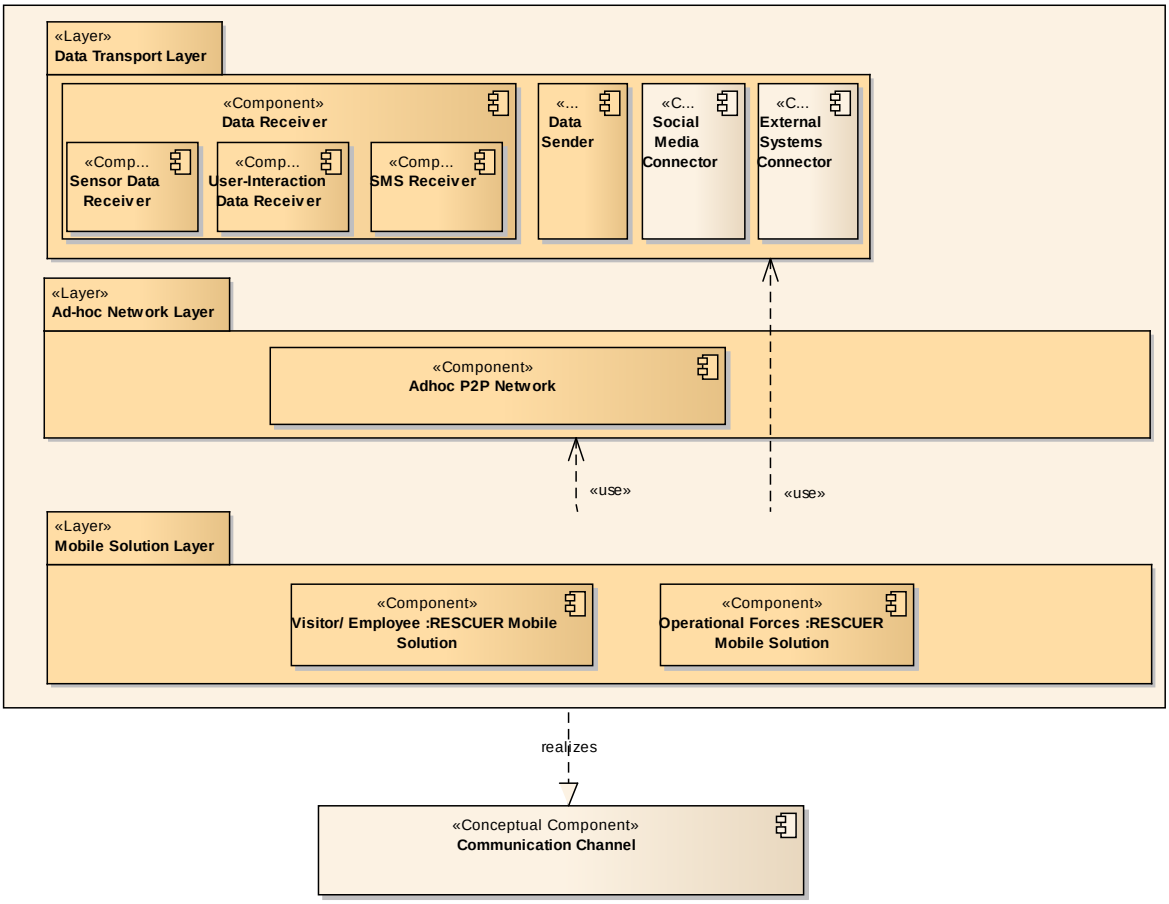

Figure 10 : Communication Channel

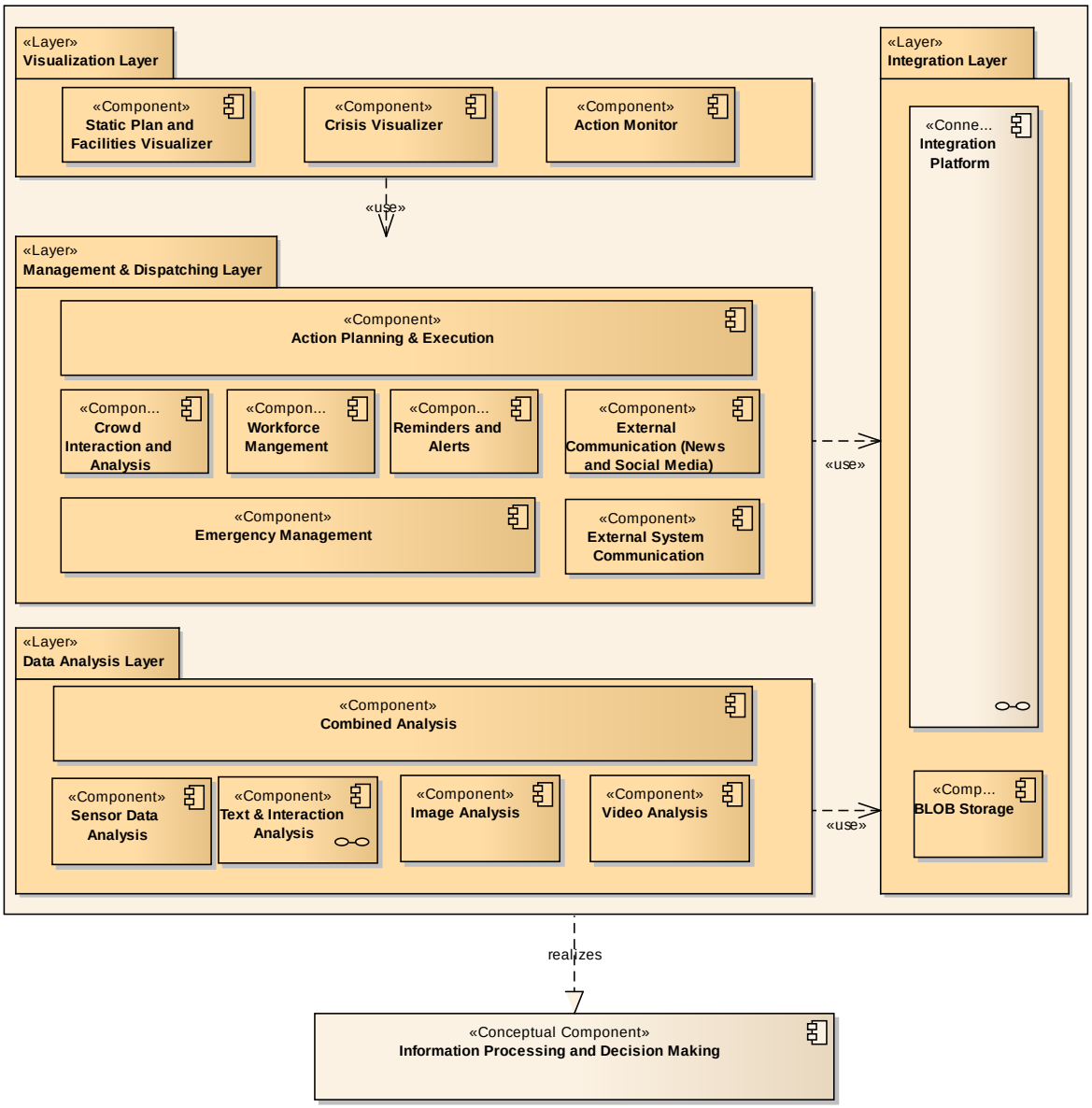

Figure 11 : Information Processing and Decision Making

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 12 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, report aggregation and so on.
        - 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 (type of user, skill, experience level, and so on).
        - Application Context Data: It consists of the application context of the mobile application that triggers the report sending. In simple words, 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 point 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 is to be defined yet. Report id 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 and 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. It can also have all data 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 13 shows the various topics that are used to have communication among the components built inside RESCUER. 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: In the first project iteration, it covers only the administrative messages from the server (sensor data recording request). In the later iterations, this topic will be refined and it can be used for follow-up interactions and 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

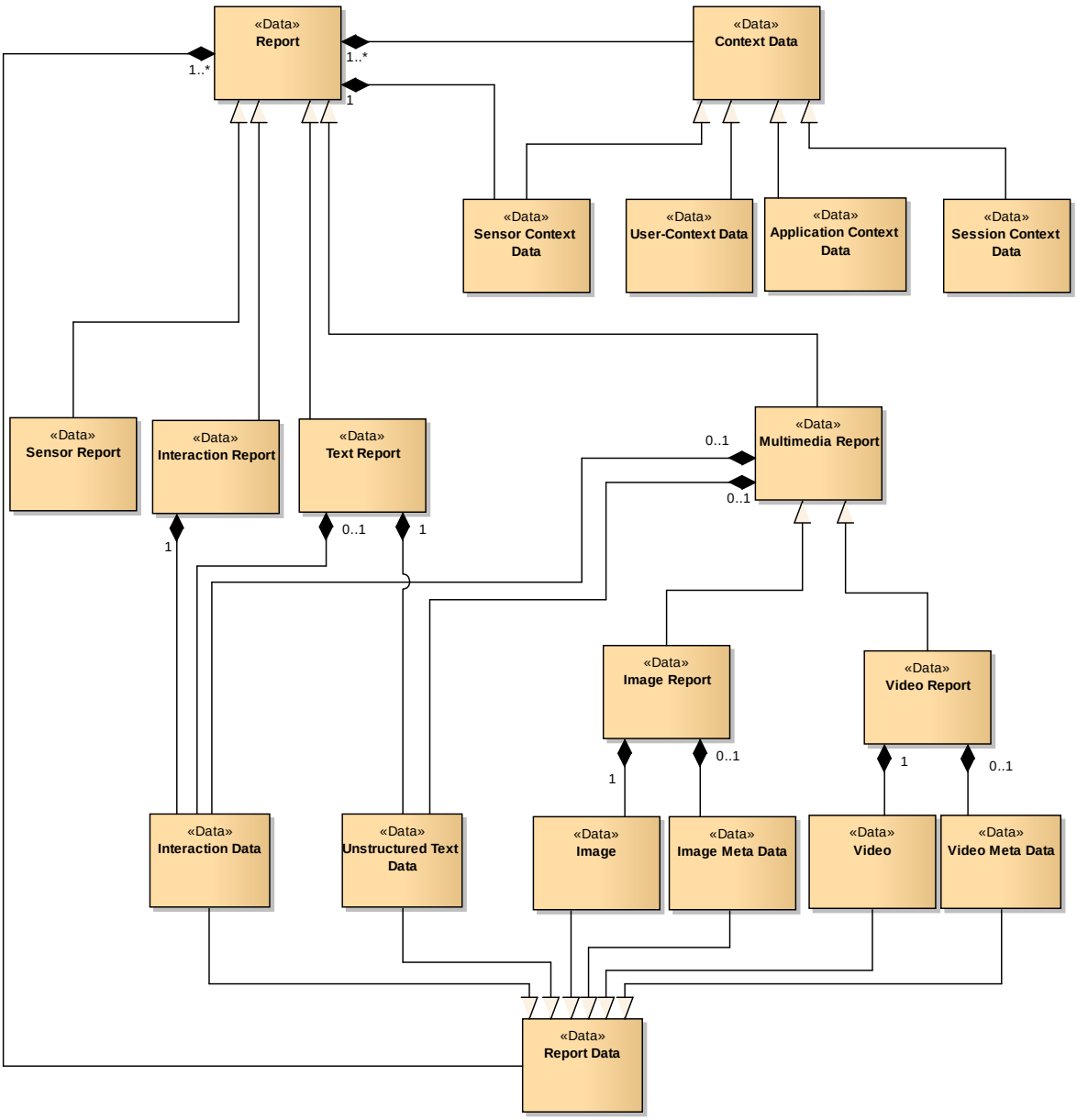

Figure 12 : Report data model

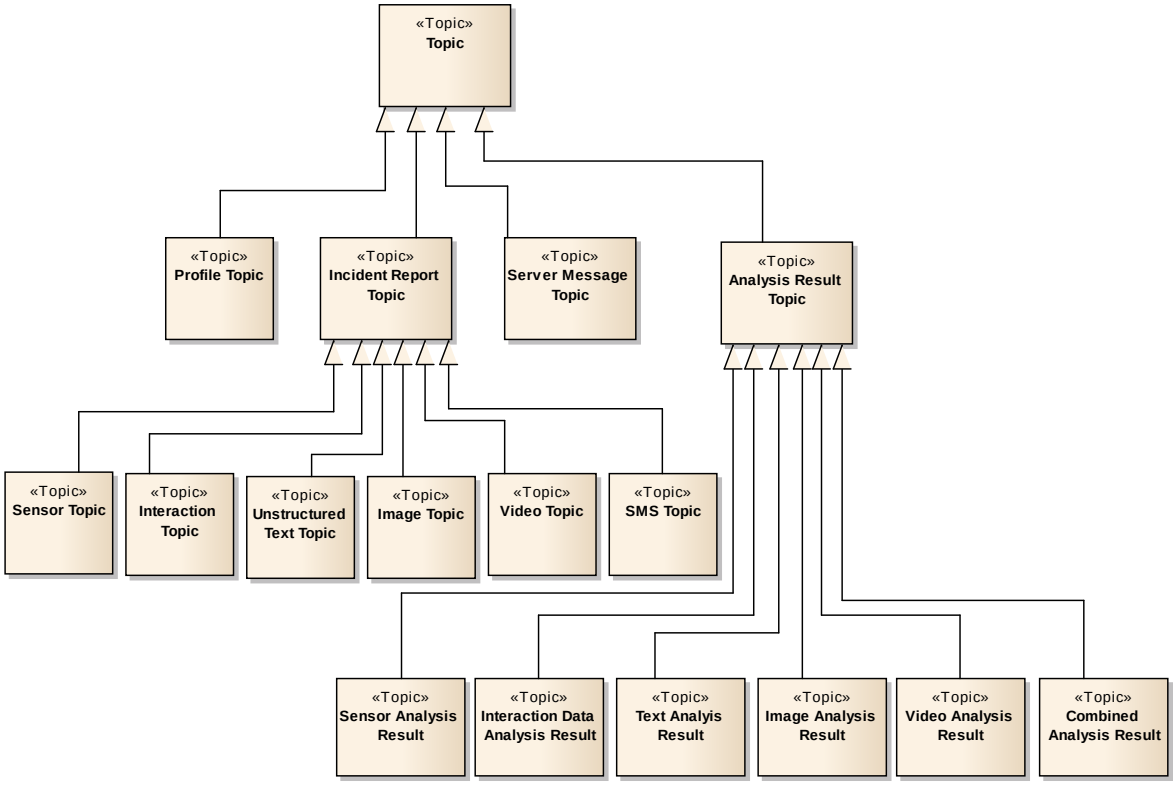

Figure 13 : Integration platform topics

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.

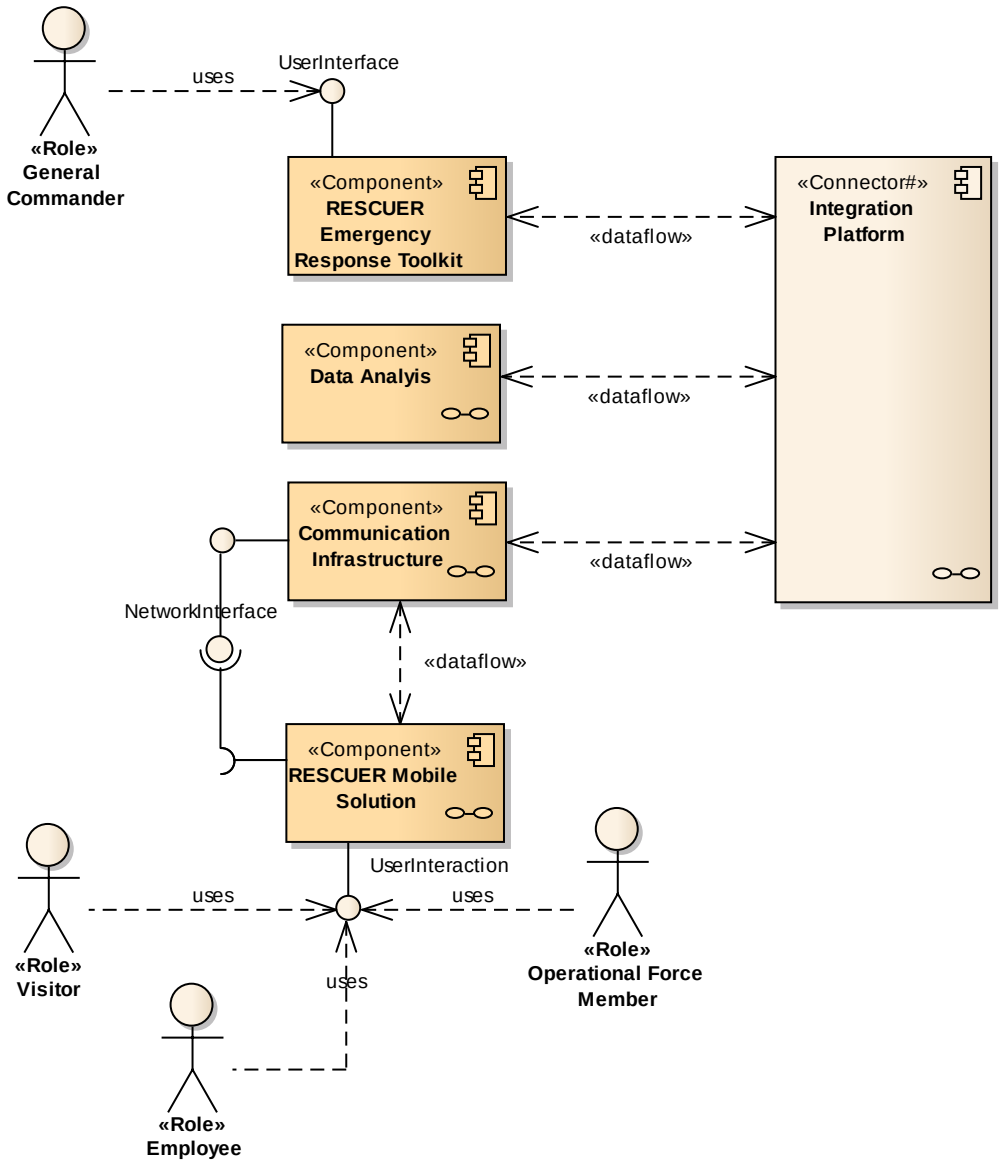

Figure 14 : Sub-systems view

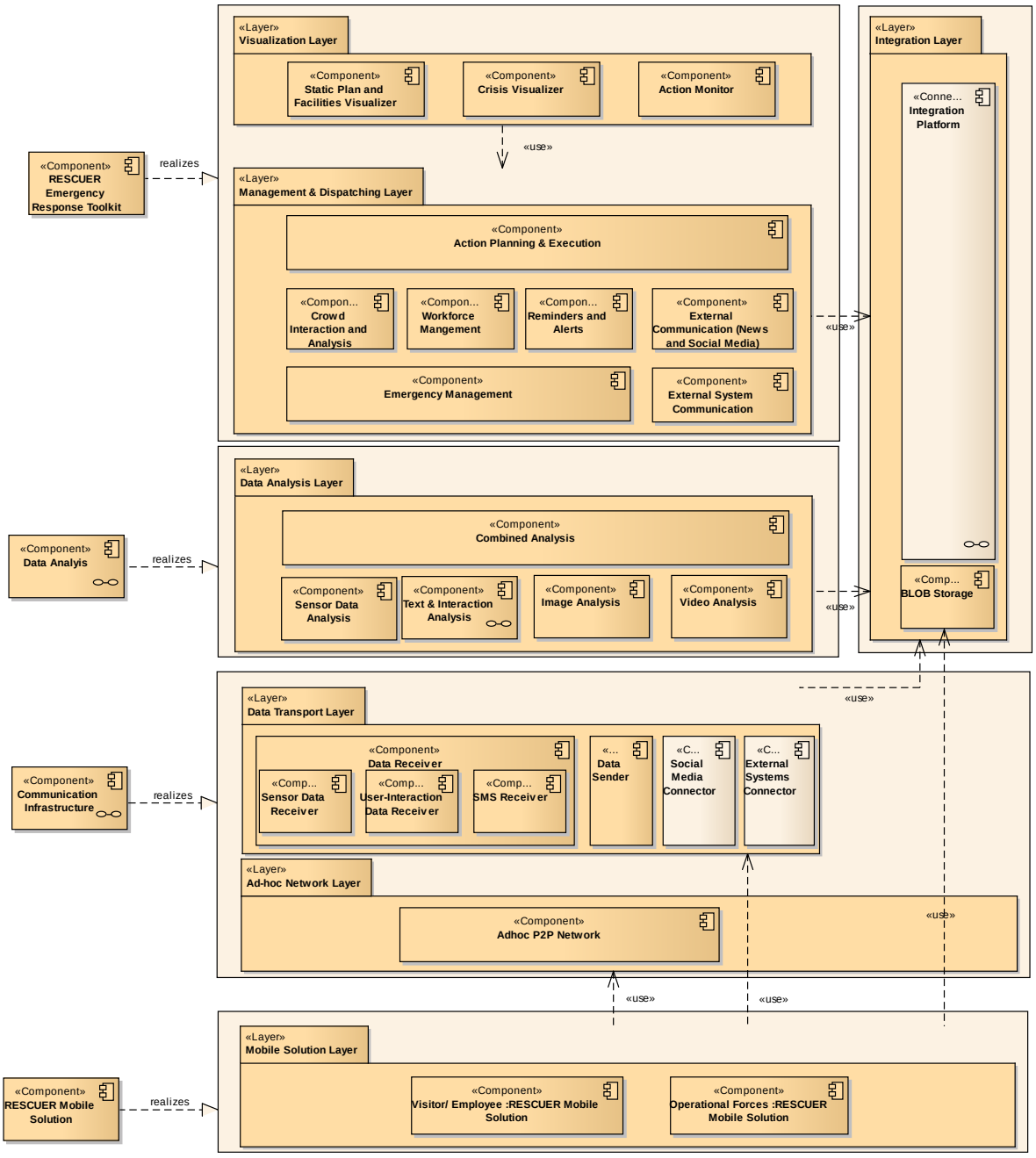

Figure 15 : Mapping of sub-Systems to functional components

5.2.1. Mobile Solutions

- Conceptual View:

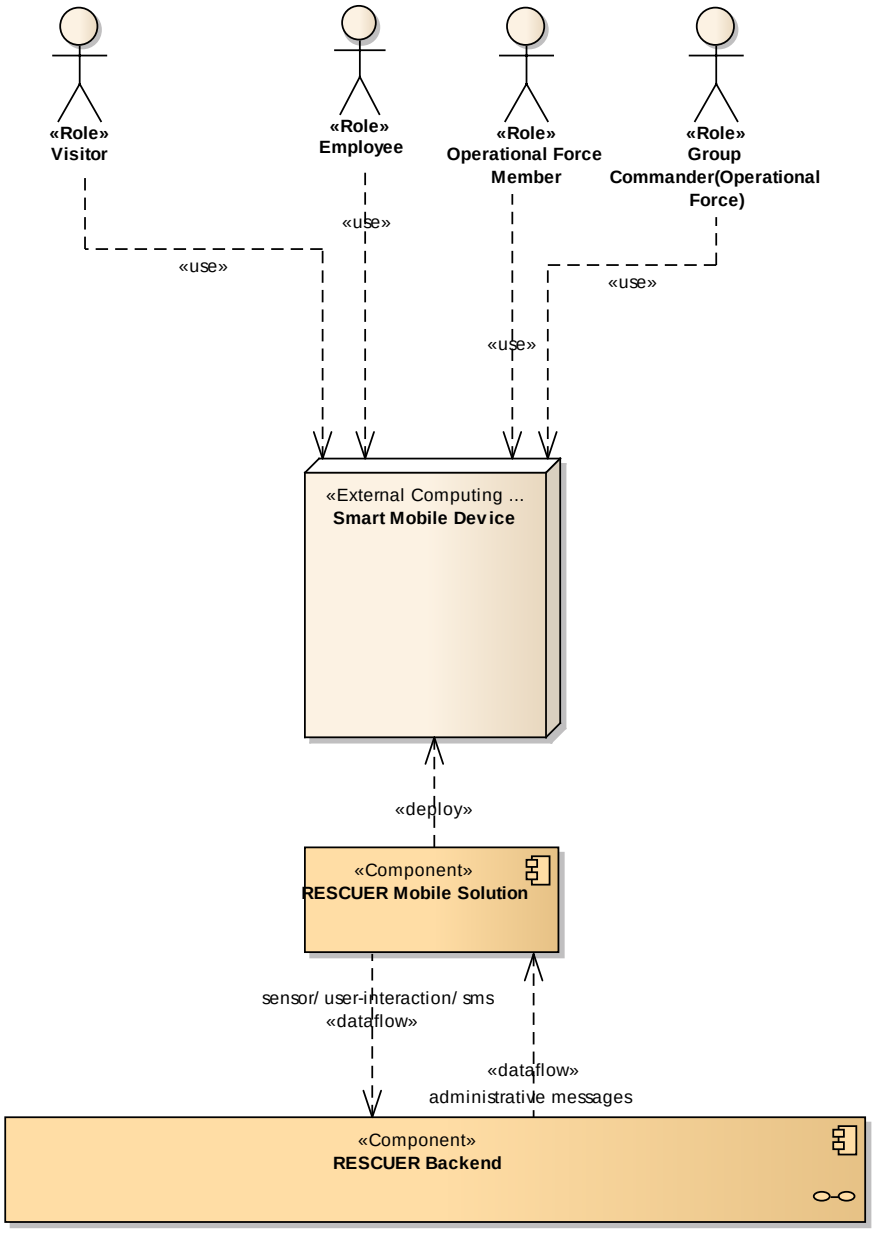

Figure 16 : Conceptual view of the RESCUER Mobile Solution

- Context and Data View:

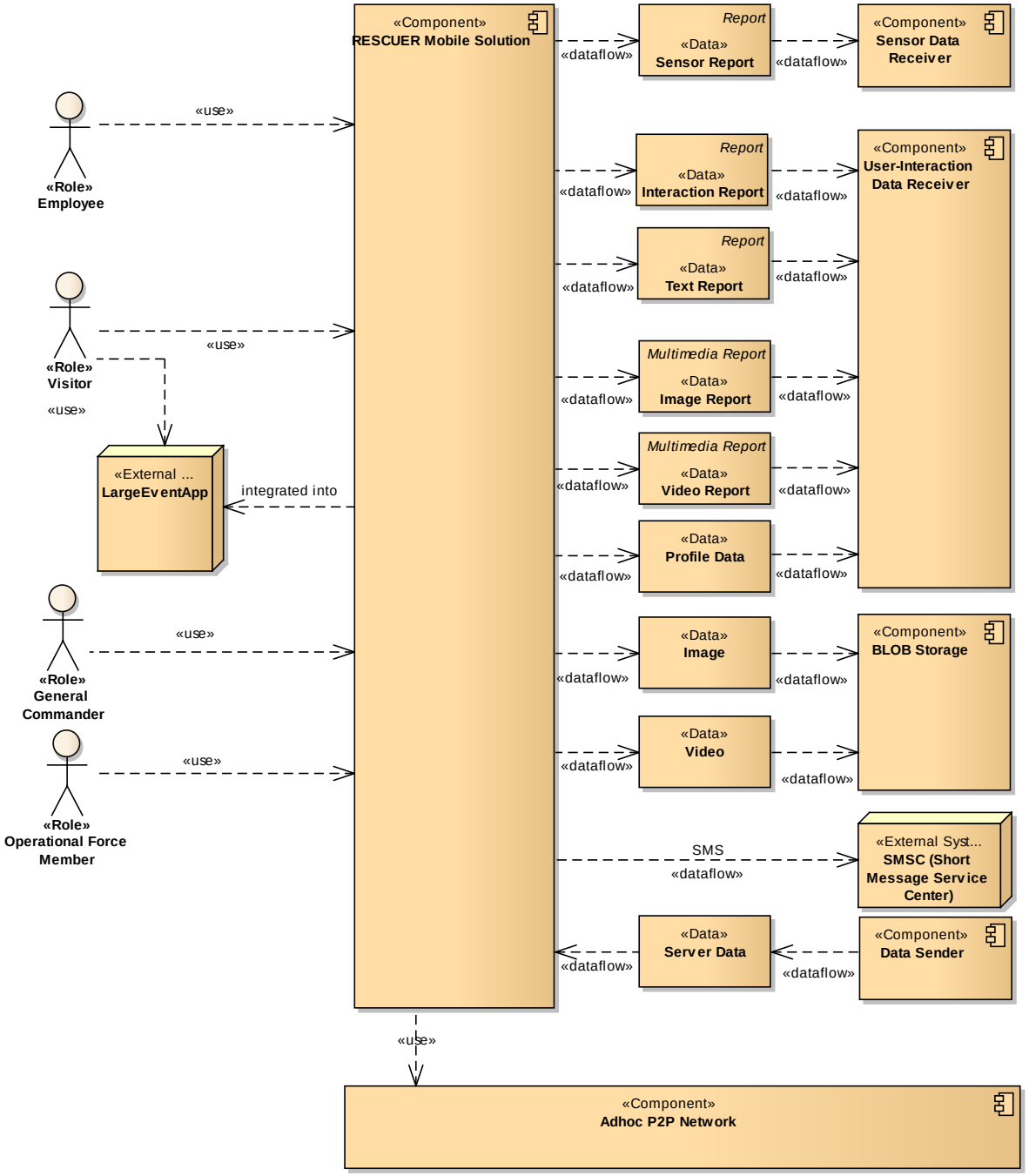

Figure 17 : Data flow and context of the RESCUER Mobile Solution

- Development and Technology View:

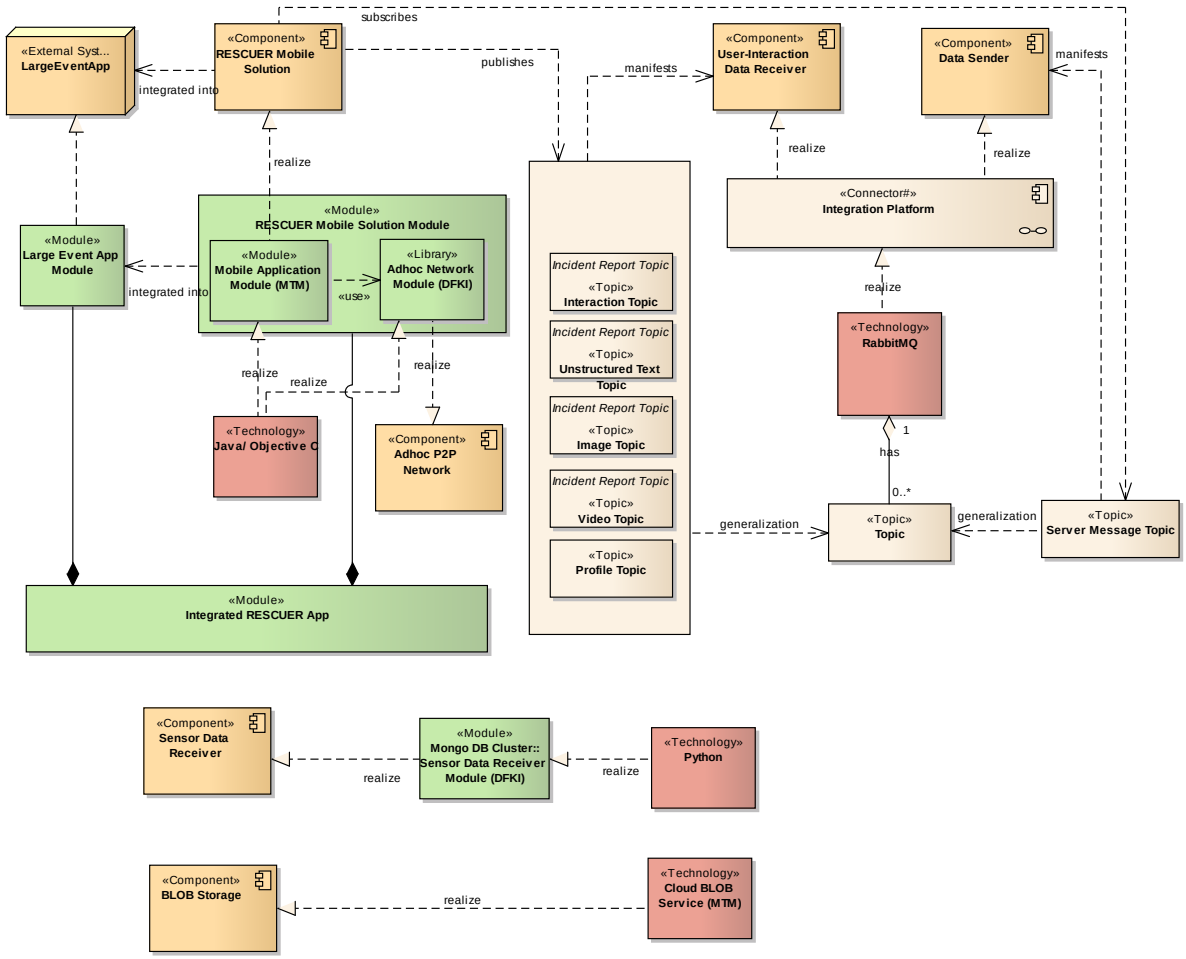

Figure 18 : Development and technology of the RESCUER Mobile Solution

- Deployment View:

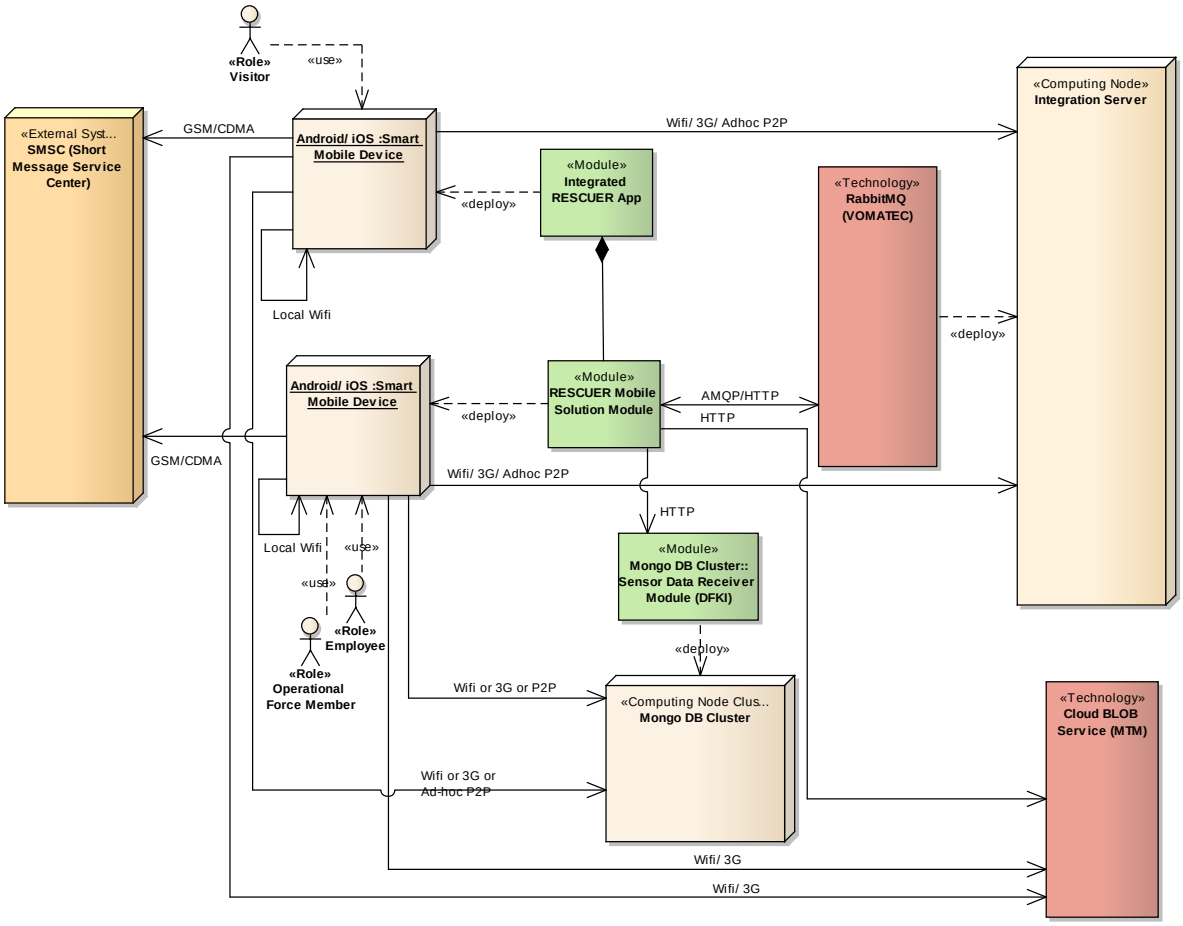

Figure 19 : Deployment of the RESCUER Mobile Solution

5.2.2. Communication Infrastructure

- Context and Data View:

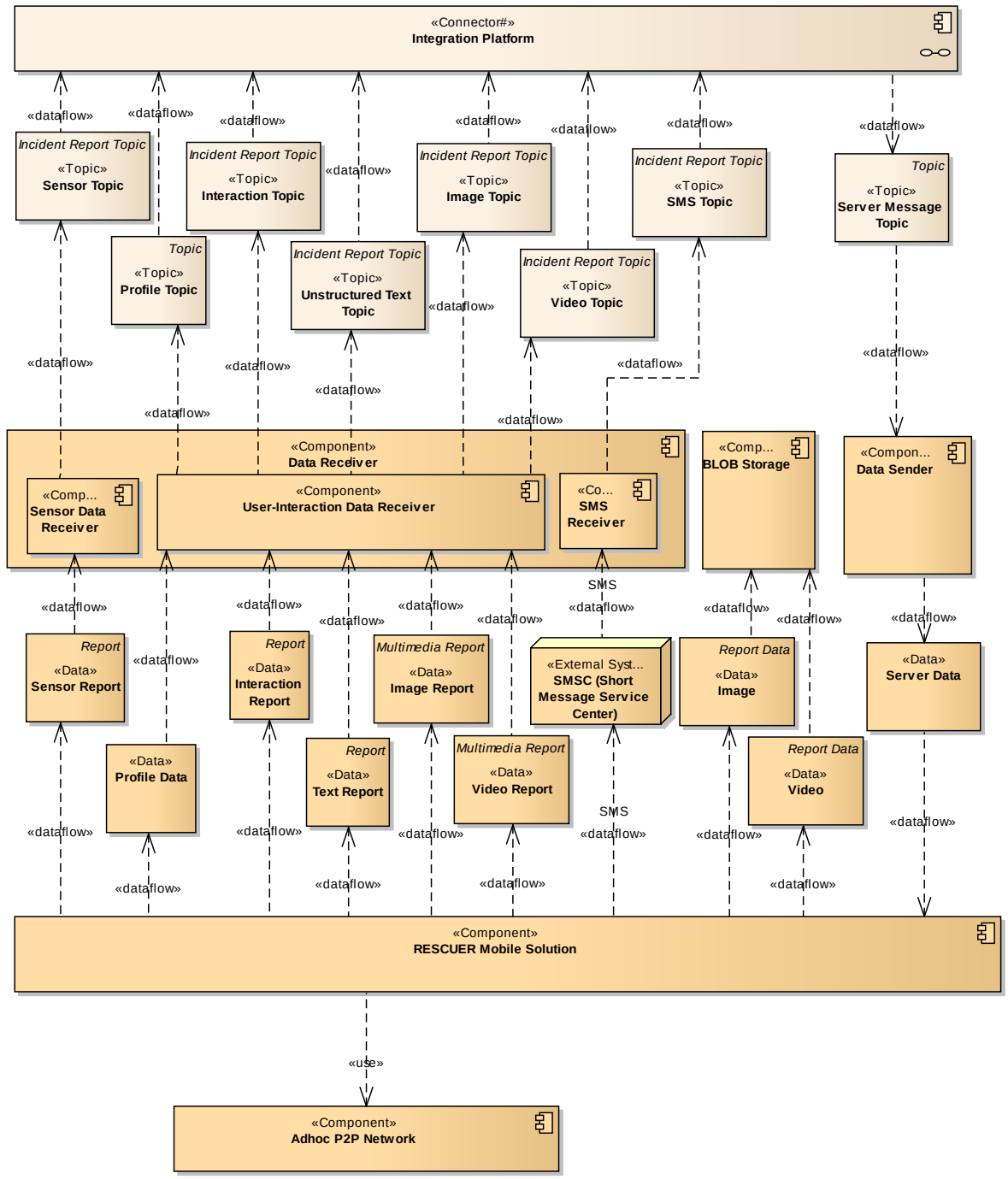

Figure 20: Context and data flow of the Communication Infrastructure

- Development View:

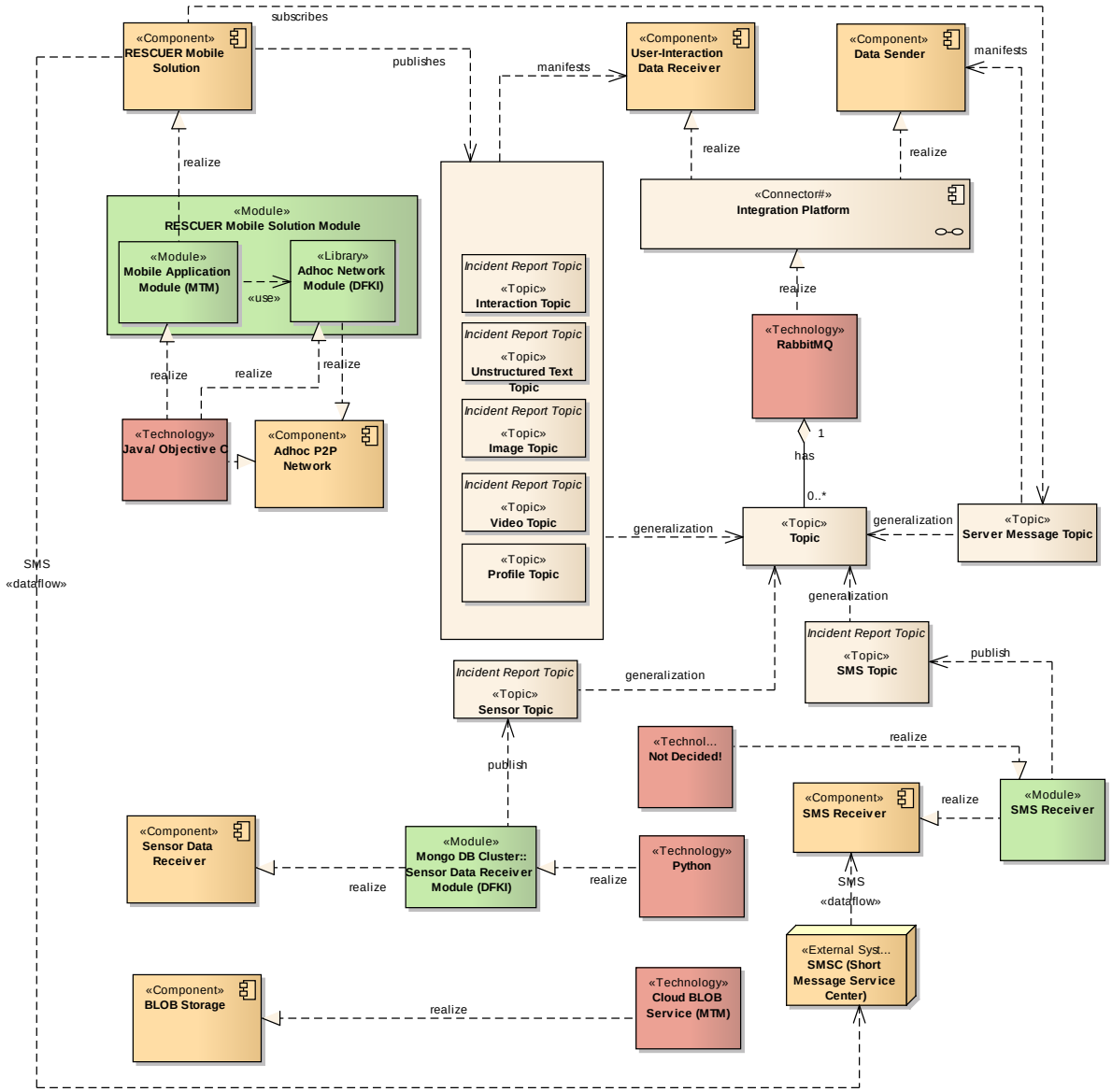

Figure 21 : Development of the Communication Infrastructure

- Deployment View:

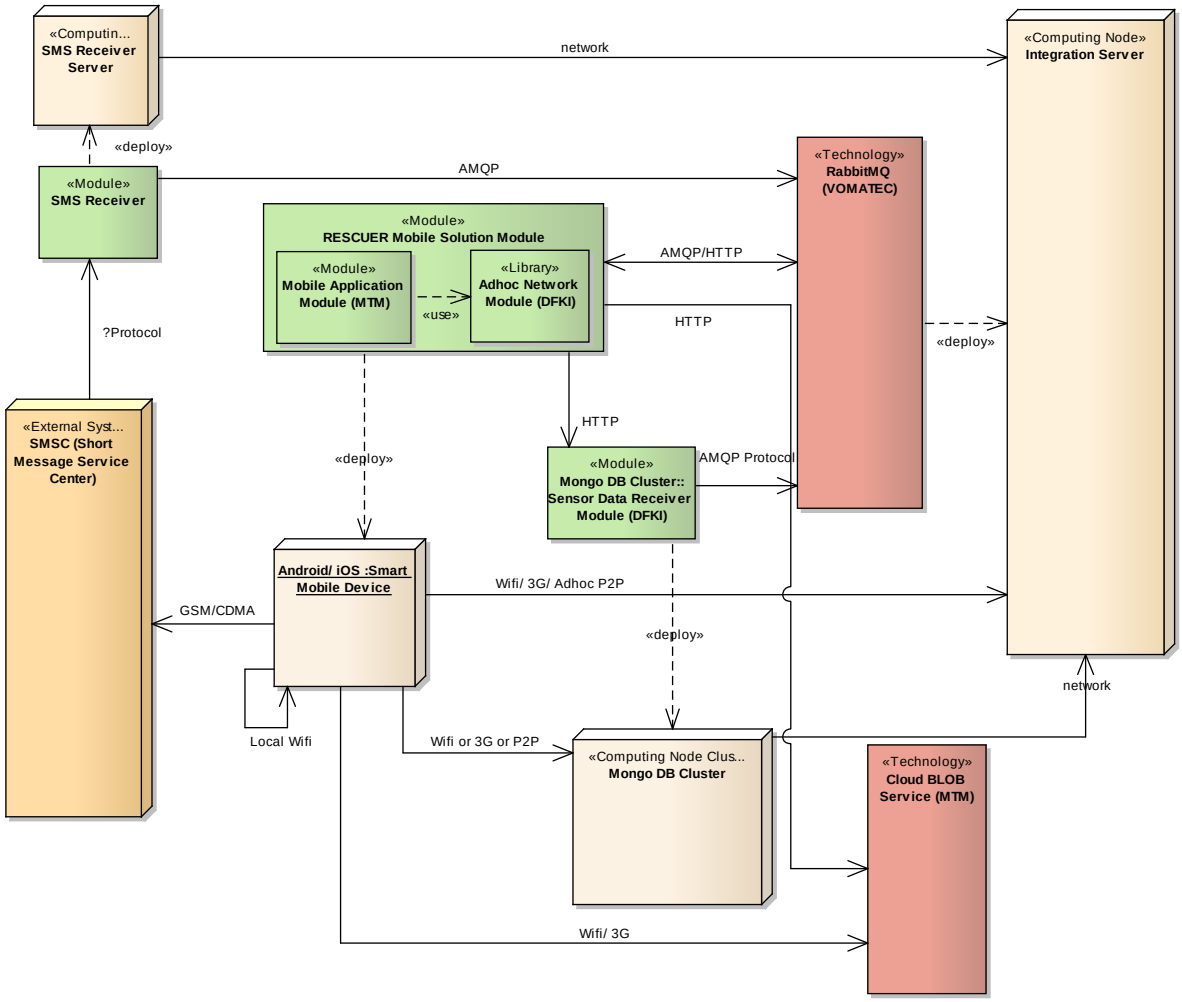

Figure 22 : Deployment of the Communication Infrastructure

5.2.3. Data Analysis

- Context and Data View:

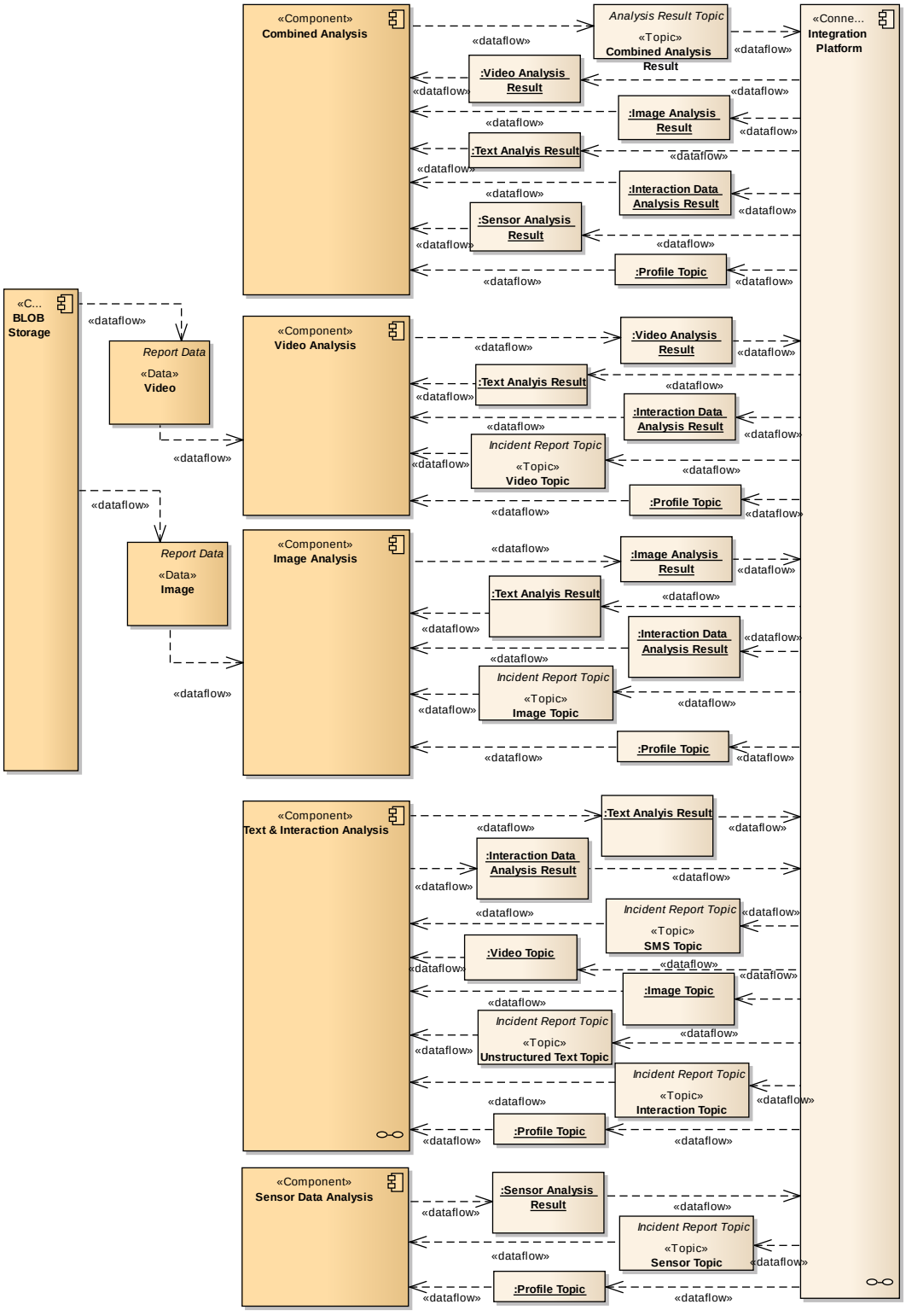

Figure 23 : Context and data flow of the Data Analysis Components

- Development View:

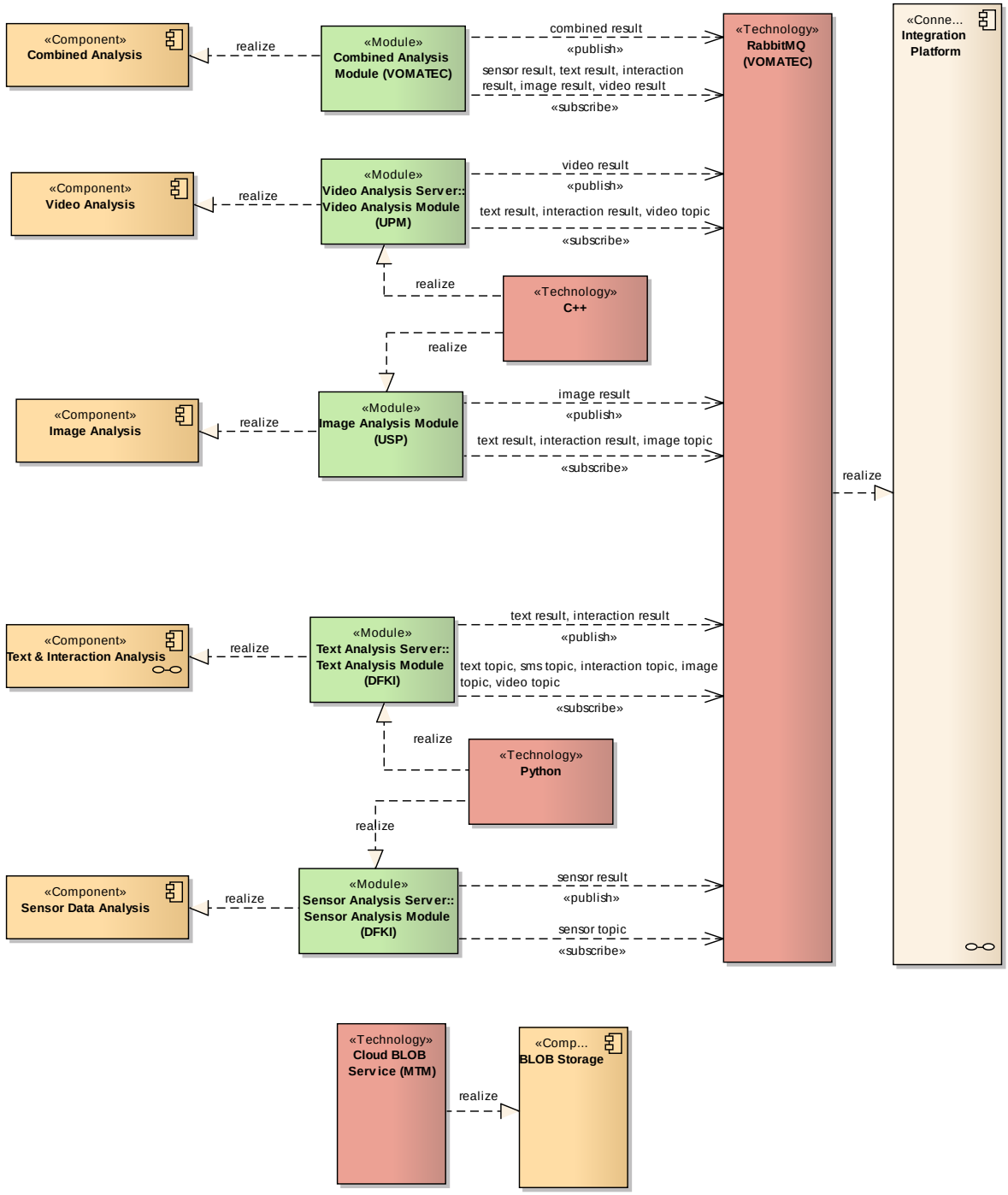

Figure 24: Development of the Data Analysis Components

- Deployment View:

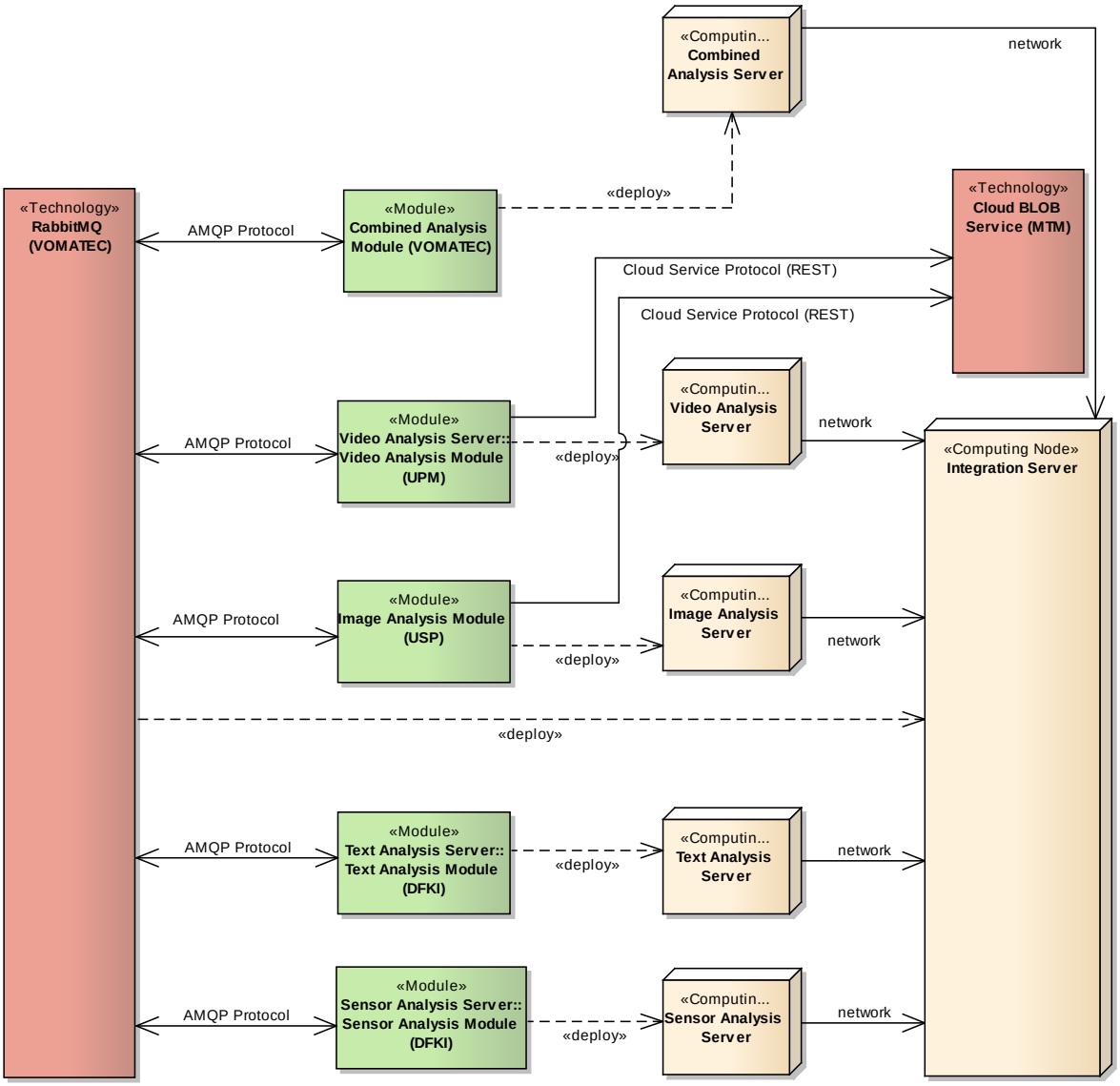

Figure 25 : Deployment of the Data Analysis Components

5.2.4. Emergency Response Toolkit

- Context and Data View:

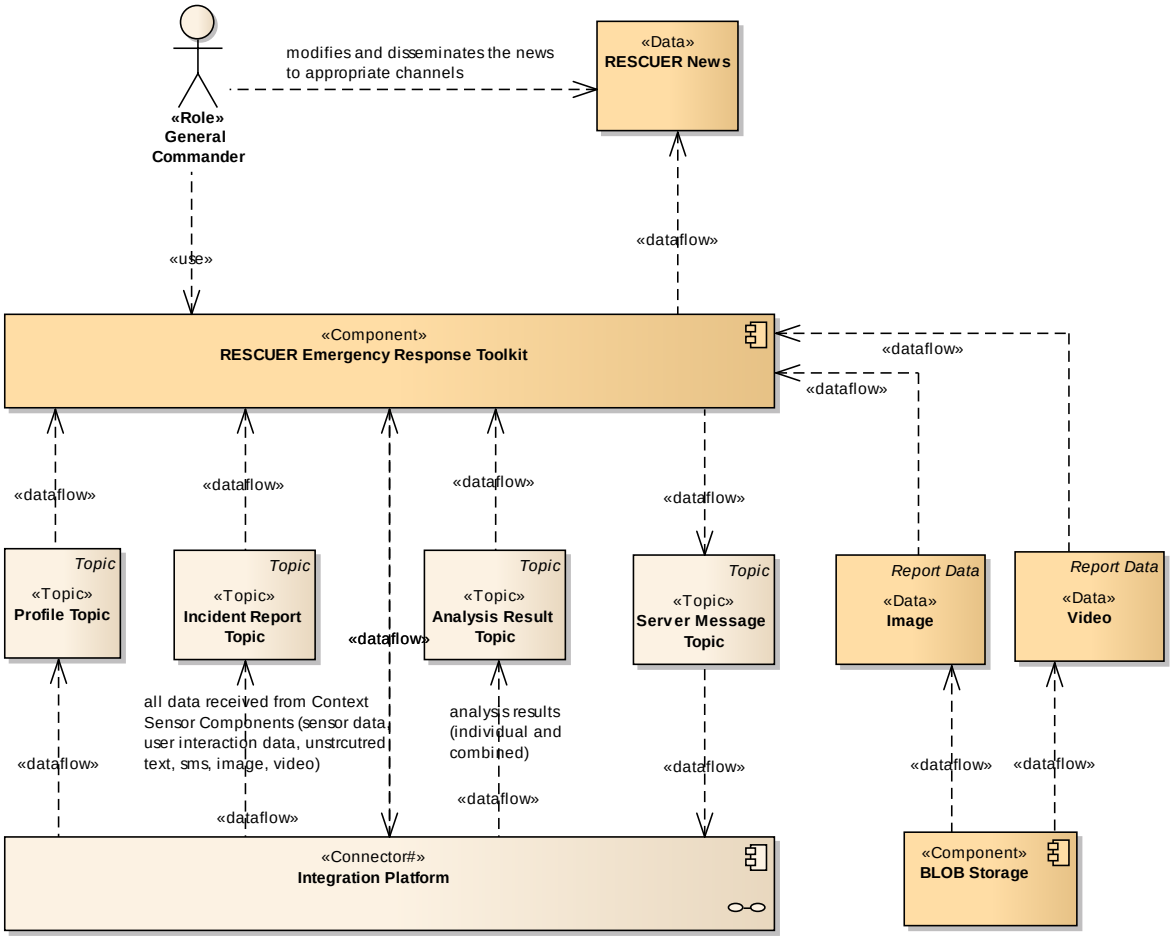

Figure 26 : Context and data flow of the Emergency Response Toolkit

- Development View:

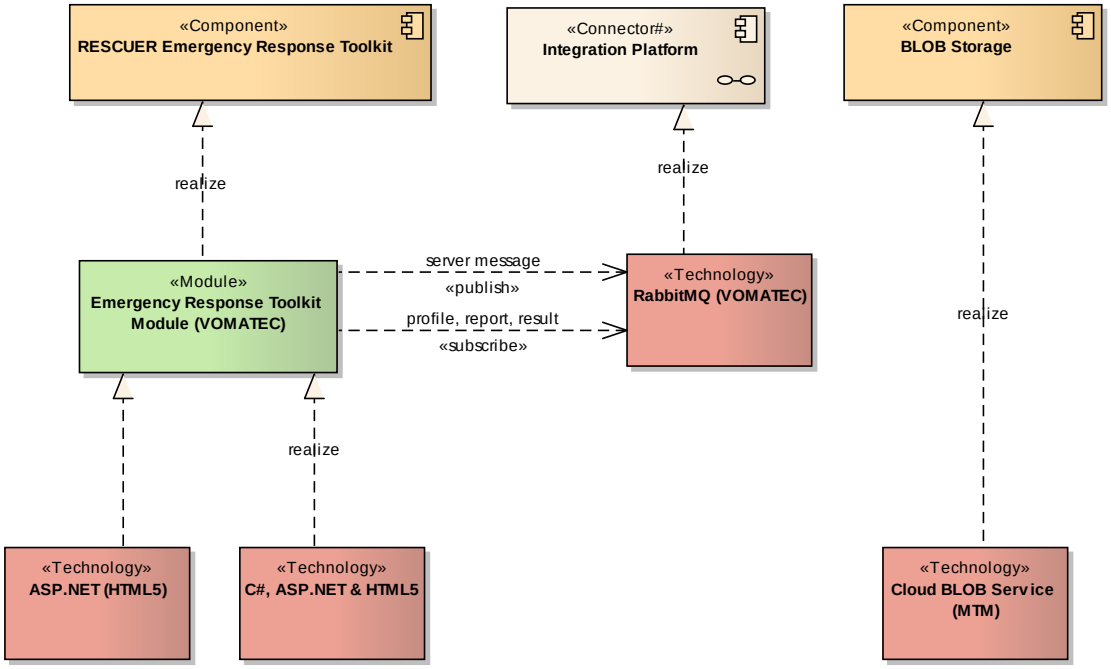

Figure 27 : Development of the Emergency Response Toolkit

- Deployment View:

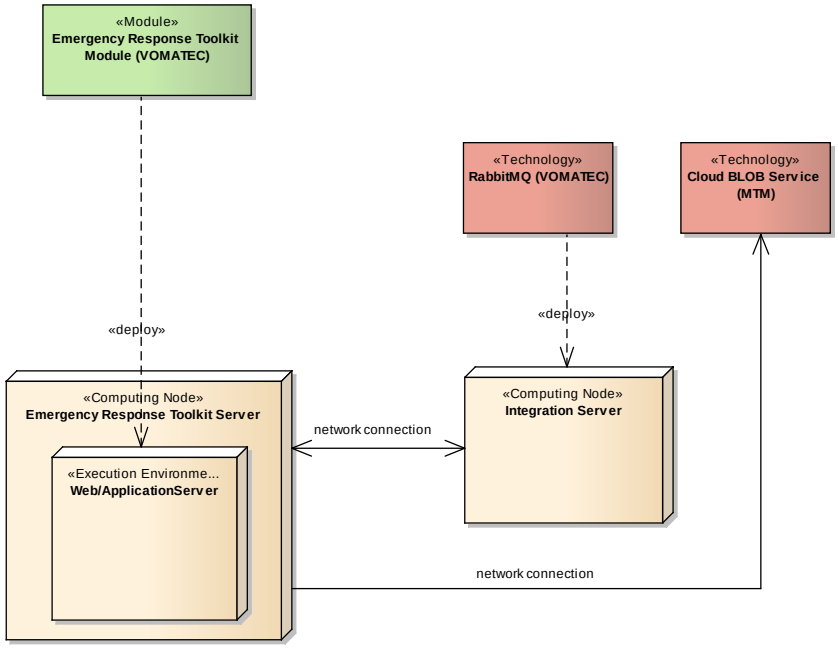

Figure 28 : Deployment of the Emergency Response Toolkit

5.2.5. Integration Platform

The Integrated Platform is responsible for integrating the components developed inside RESCUER (in different work packages) or outside RESCUER (i.e. the existing systems of operational forces). 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 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 will publish and subscribe appropriate topics to be integrated.

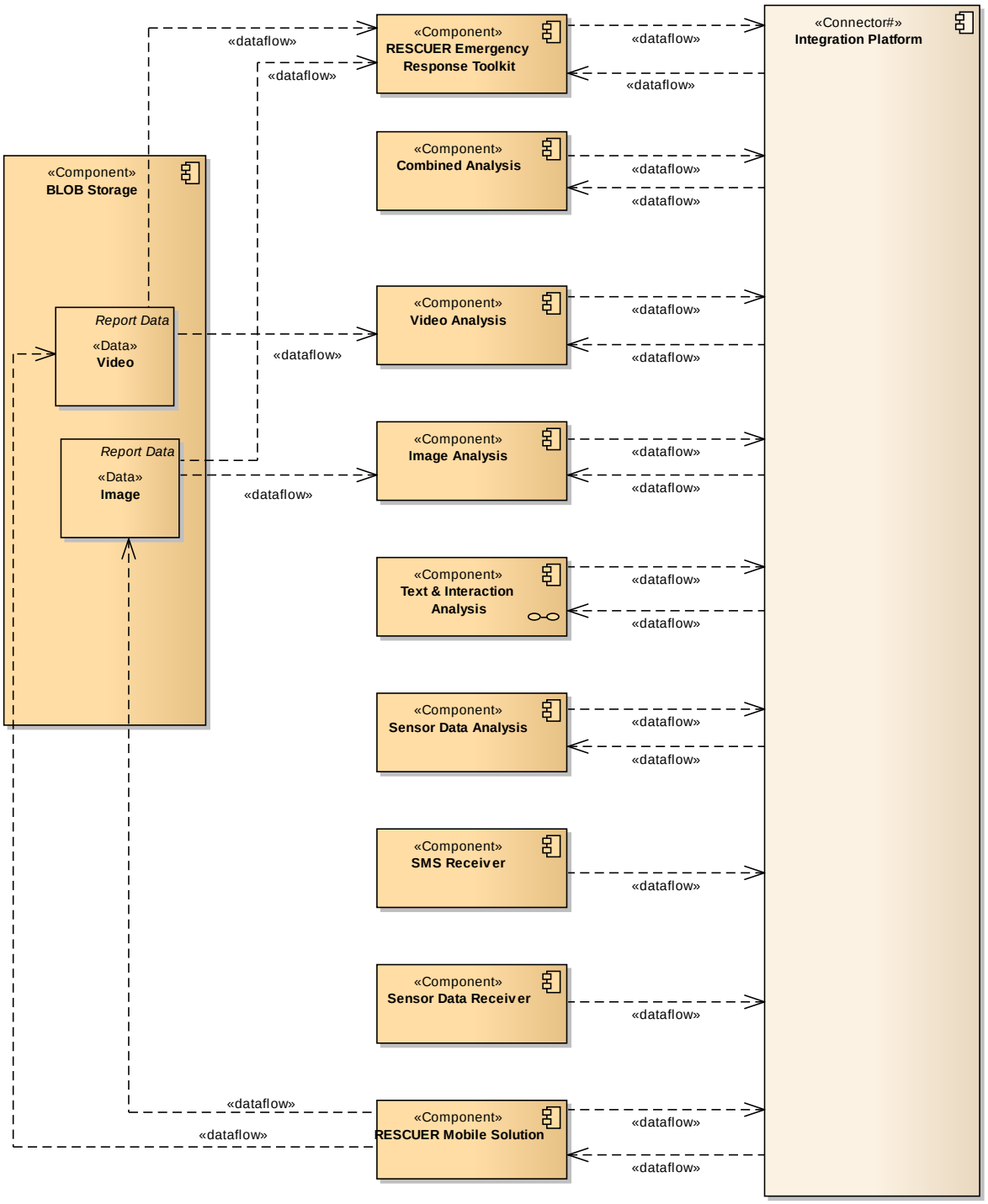

Figure 29 : Context and data flow of the Integration Platform

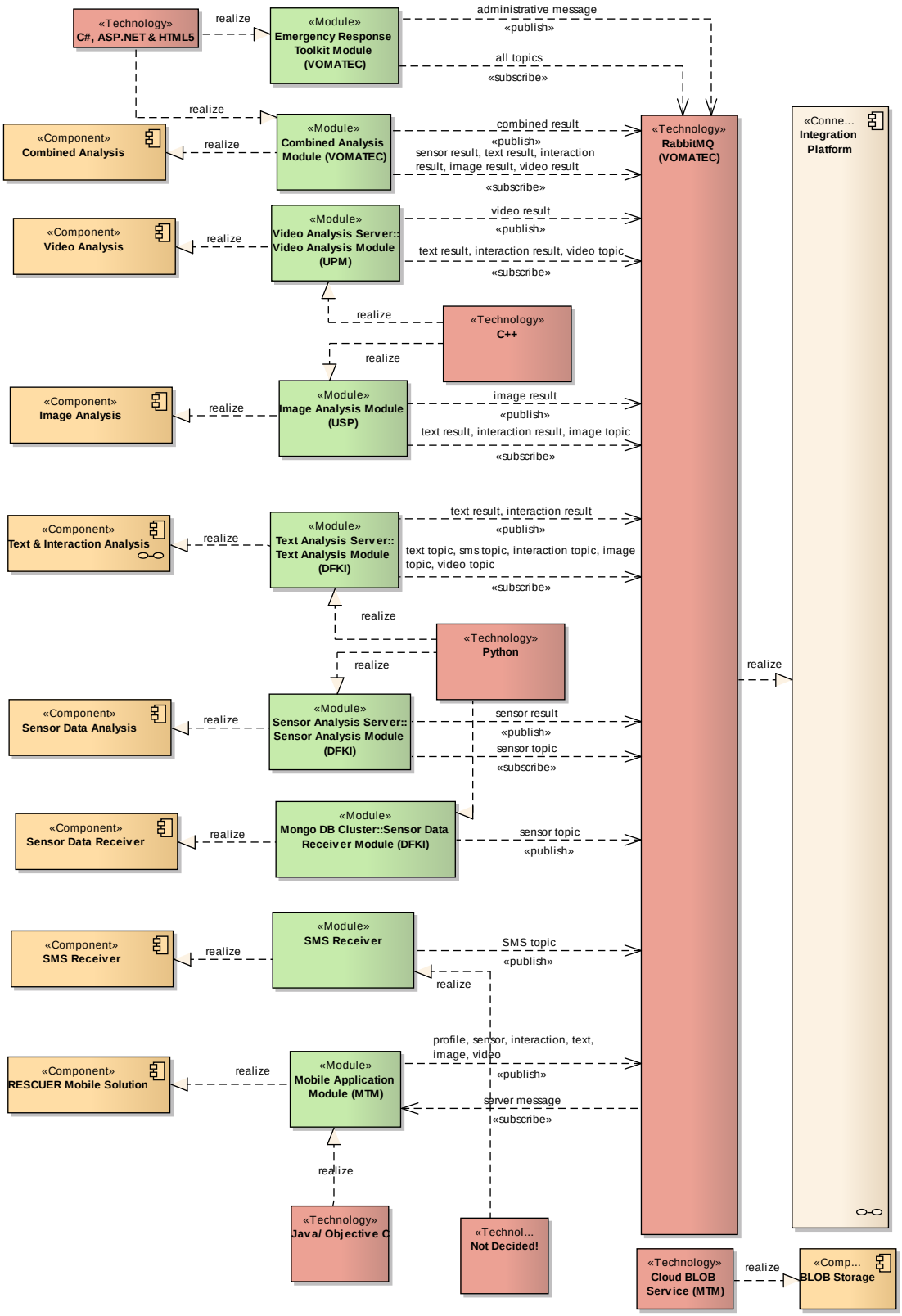

Figure 30 : Development of the Integration Platform

- Deployment View:

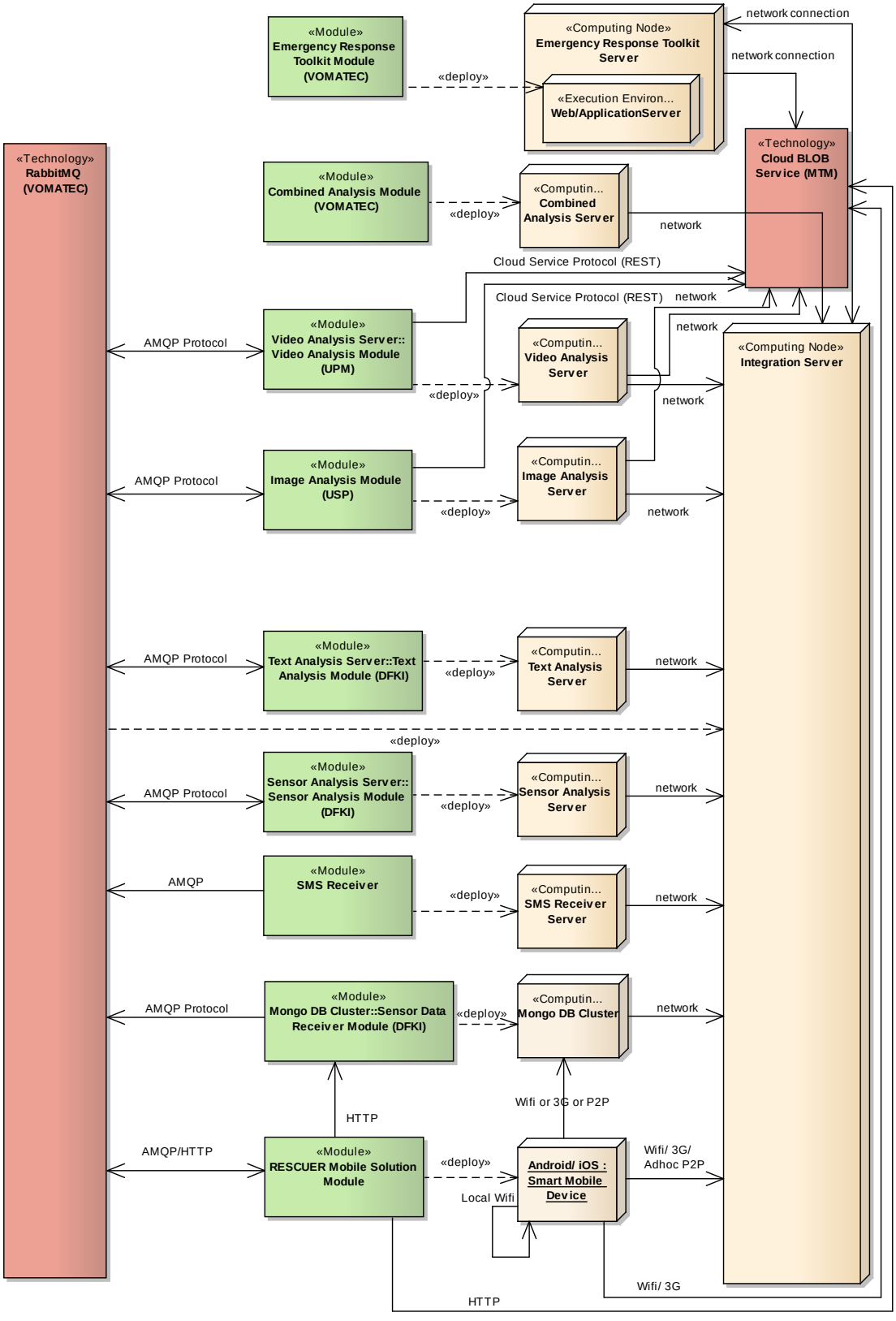

Figure 31 : Deployment of the Integration Platform

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 will be a web application which can be accessed by 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 Data Sender Component is realized by the Server Message Topic of RabbitMQ.
- Integration Platform is realized by RabbitMQ. 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 will be realized Mobile Application Module which is a mobile app used by various stakeholders on the spot of an emergency situation.

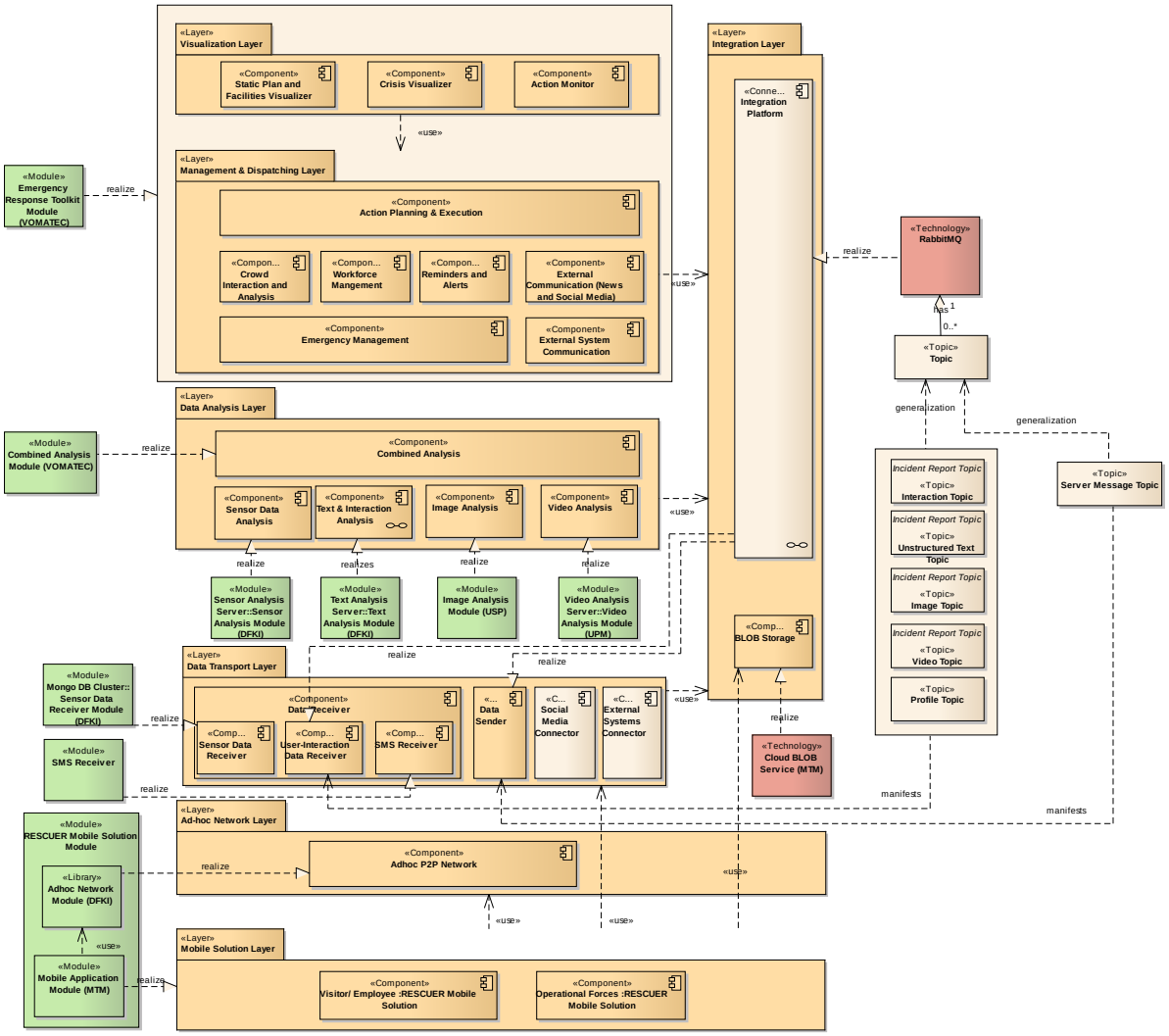

Figure 32 : Development of RESCUER

Figure 33 shows the various technologies used to realize the overall RESCUER system.

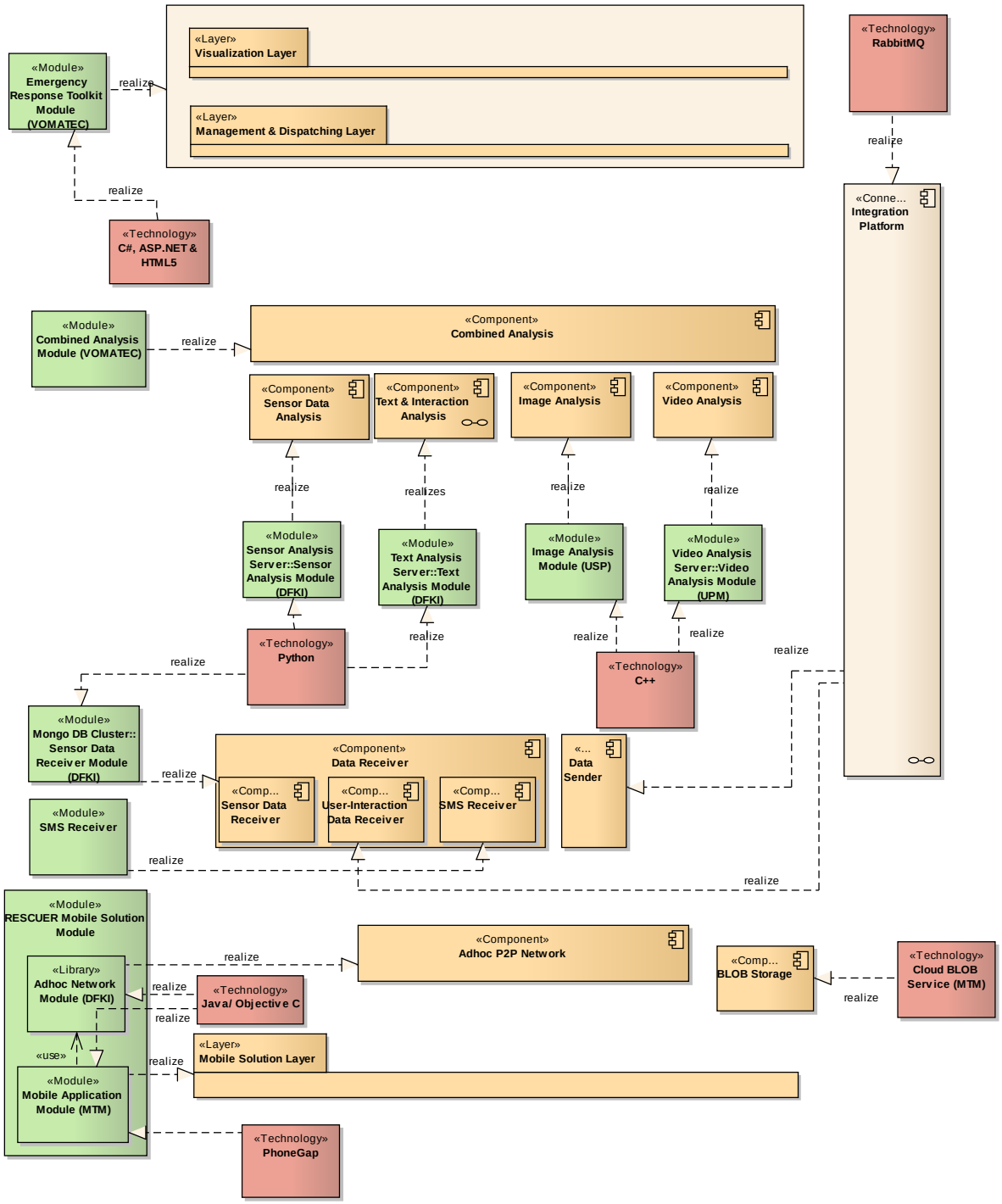

Figure 33 : Technology used in RESCUER

5.4. Deployment Perspective

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

- General Commanders in the command and control centre will use a Client Device (tablet, laptop) to access the Emergency Response Toolkit web application. The network connection between the client device and the web application can be either internet, local intranet, or 3G.
- The Emergency Response Toolkit web application is deployed into a web application server which 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 the 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. It is either connected by internet or local intranet 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.

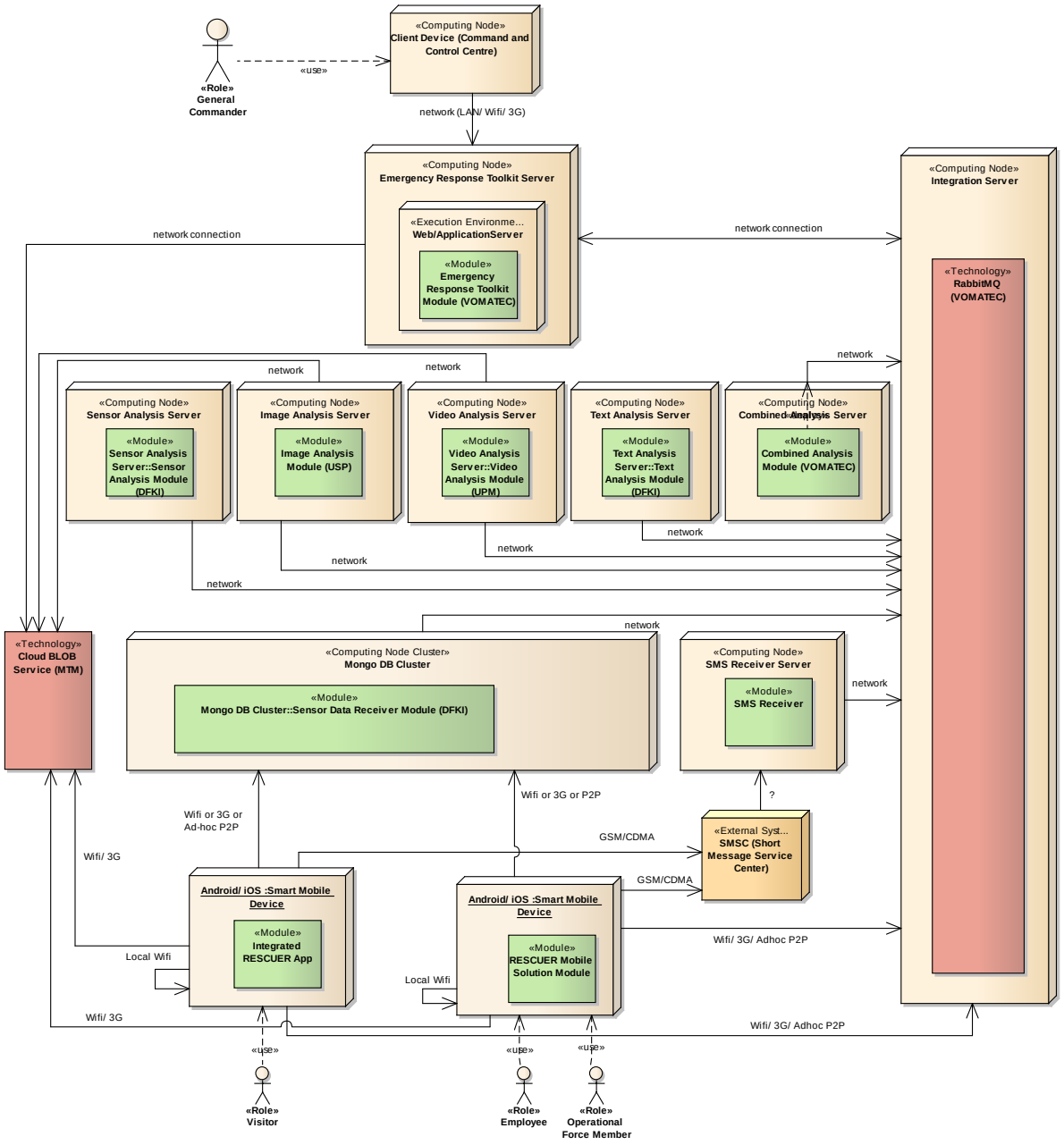

Figure 34 : Deployment of RESCUER System

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, the RESCUER Mobile Solution will be used by various types of users. In the nearby area the Emergency Response Toolkit will be used by the command and control centre. The off-site region is divided into two groups based on the stakeholder class – the operational forces and the rest. Operational forces

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.

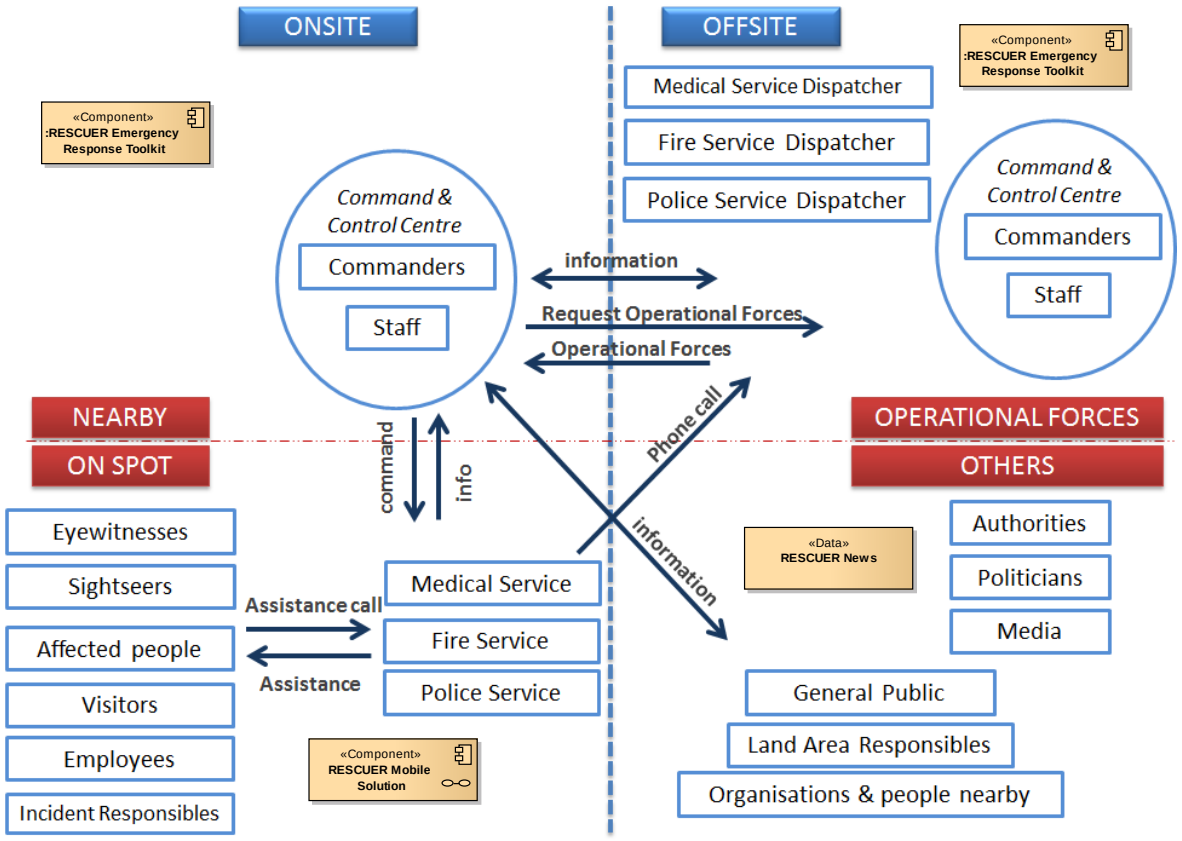

Figure 35 : RESCUER usage perspective

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 operational forces (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 RabbitMQ 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 5 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.

VOMATEC is also responsible for building connectors to social media and external legacy systems of the operational forces. 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 iterations.

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.

Table 3 : Development task allocation among partners

Partner	Modules to build
MTM	*Mobile Solutions (MS), Cloud BLOB Service (BLOB)*
DFKI	*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)*
VOMATEC	*Emergency Response Toolkit (ERT), Combined Analysis (CA), Integration Platform (IP), Social Media Connector (SMC), Legacy System Connector (LSC)*
UPM	*Video Analysis (VA)*
USP	*Image Analysis (IA)*

5.7. Design Decisions

Architecture Significant Requirements	Realisation	Responsible Components
Devtime Requirements		
ASR.DEVTIME.01:Documentation of design and code	Overall system architecture is being documented by this deliverable. Individual project partners have been communicated to document their own design and code.	All
ASR.DEVTIME.02:Distributed development	System is designed in a modular way, and interfaces and data exchanges are made clear among them.	IP
Integration Requirements		
ASR.INTEGRATION.01:New components should be integrated to the RESCUER platform without much effort	Generic integration mechanism publish-subscribe is used which is not bound to any technology and provides asynchronous communication.	All, mostly IP
ASR.INTEGRATION.02:Integration with social media	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.	SMC
ASR.INTEGRATION.03:Integration among internal components	Generic integration mechanism publish-subscribe is used which is not bound to any technology and provides asynchronous communication.	IP

Architecture Significant Requirements	Realisation	Responsible Components
ASR.INTEGRATION.04:Integration with operational forces' systems	External System Connector in the Data Transport layer is responsible for integration with the operational forces' systems. Actual integration concepts will be built in the next project iterations.	LSC
ASR.INTEGRATION.05:Integration of mobile device with smart watch	Integration concepts will be built in the next project iterations.	MS
ASR.INTEGRATION.06:Uniformity of integration mechanism across components	Generic integration mechanism publish-subscribe is used which is not bound to any technology and provides asynchronous communication.	IP
Operation Requirements		
ASR.OPERABILITY.01:Import of Initial facility plans and so on	Static Plan and Facilities Visualizer components provides interface to upload this data and can visualize them.	ERT
ASR.OPERABILITY.02:User ID generation	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.DEMO.01:Efficient preparation for evaluation and demonstration	Test data and environment will be created by the evaluation partners.	All
ASR.INSTALLABLILITY.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

Architecture Significant Requirements	Realisation	Responsible Components
Availability		
ASR.AVAILABILITY.01:24/7 availability of the RESCUER backend	All backend components will be running in the server which provides 99.99% availability per month. But in real commercial deployment, multiple replicated servers would be required to ensure the expected availability.	All
ASR.AVAILABILITY.02:24/7 planned downtimes of the RESCUER backend	Backend components and infrastructure operators will perform their maintenance task during the planned downtimes.	All
ASR.AVAILABILITY.03:Unplanned downtimes of the backend due to hardware failure	Additional free servers will be kept ready. In case any server fails, other can be started within minutes.	All
ASR.AVAILABILITY.04:Unplanned downtimes of the backend due to software failure	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.	All
Robustness		
ASR.ROBUSTNESS.01:Robustness against arbitrary user navigation		MS, ERT
ASR.ROBUSTNESS.02:Robustness against arbitrary user input	Each input from the users will be validated before processing.	MS, ERT
ASR.ROBUSTNESS.03:Robustness against unstable network connections	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.	ADHOC, MS
ASR.ROBUSTNESS.04:Robustness against no network connections	MS will use Ad-hoc P2P network in case of no network connections are available.	ADHOC, MS
ASR.ROBUSTNESS.05:Robustness against crashes of mobile application	MS will persist all relevant information. In case it crashes, once restarted, it should be able to start smoothly based on the persisted data.	MS

Architecture Significant Requirements	Realisation	Responsible Components
Scalability		
ASR.SCALABILITY.01:Initial load during first evaluation	One server machine per component will be sufficient for the first evaluation scenarios.	All
ASR.SCALABILITY.02:Scaling for large number of apps	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 iterations.	All
ASR.SCALABILITY.03:Scaling over large number of events or industrial parks	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.	All
Safety		
ASR.SAFETY.01:Mobile user is not endangered	MS will be flexible and allow sending of partial reports.	MS, ERT
Upgradeability		
ASR.UPGRADEABILITY.01:No data loss during app upgrade	Profile and any other necessary configuration will be persisted and during upgrades still be kept persisted.	MS
Auditability		
ASR.AUDITABILITY.01:Backend logging	All analysis components log every individual raw messages.	All
Usability		
ASR.USABILITY.01:Simple user interface	The UI has to be really simple and should avoid all kinds of fancy complex interactions.	MS, ERT
ASR.USABILITY.02:User should receive immediate response	MS should be responsive and, show messages to the user as soon as it gets feedback from the server.	MS
ASR.CONSUMABILITY.03:Integrated with large event app	It will be explored in later project iterations.	MS

Architecture Significant Requirements	Realisation	Responsible Components
Variability		
ASR.VARIABILITY.01:Multiple existing systems of operational force	External system connector should be configured and should have plugin mechanism to connect to any operation force's legacy system.	ERT, LSC
ASR.VARIABILITY.02:Interoperability in EU and Brazil	It will be explored in later project iterations.	All
ASR.VARIABILITY.03:Interoperability in large-scale event and in industrial park	It will be explored in later project iterations.	All
ASR.VARIABILITY.04:Multiple mobile platforms	Two versions (iOS and Android) will be developed.	MS
ASR.VARIABILITY.05:Multiple social media	Social media connector should be configured and should have plugin mechanism to connect to any social media.	SMC
ASR.VARIABILITY.06:Multiple web interface	ERT will be developed in HTML5 and can be easily accessible through tablet or PC.	ERT
ASR.VARIABILITY.07:Multiple types of users for mobile solutions	MS will configure itself automatically based on the user category.	MS
ASR.VARIABILITY.08:Multiple types of users for web interface	It will be explored in later project iterations.	ERT
Reliability		
ASR.RELIABILITY.01:Emphasis should be given on professionals	Users are categorized into different groups. Analysis components will take into consideration the credibility of each group during analysis.	TA, IA, VA, CA, ERT
ASR.RELIABILITY.02:No wrong conclusion about the situation	Multiple analysis components will be cross-checked to identify wrong calculations.	TA, IA, VA, CA, ERT
ASR.RELIABILITY.03:Reliability of data	Reliability of data will be increased by aggregating results of multiple analysis components and multiple sources of same input.	TA, IA, VA, CA, ERT

Architecture Significant Requirements	Realisation	Responsible Components
Performance		
ASR.PERFORMANCE.01:Response time for user interaction	UI logics are local to the client component to be faster.	MS, ERT
ASR.PERFORMANCE.02:Sending partial user-interaction data	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.	MS
ASR.PERFORMANCE.03:End-to-end response time	All components should be compliant with the individual latency requirements.	All
ASR.PERFORMANCE.04:Performance on visualisation	Probable next query result will be prepared to allow prompt answer.	ERT
ASR.EFFICIENCY.05:Working in the low power mode	MS will identify the low power mode and will stop doing high resource consuming tasks (by default).	MS
Security		
ASR.SECURITY.01:Separation of views for different users in ERT	It will be explored in next project iterations.	ERT
ASR.SECURITY.02:Data of a user should not be shared with others	MS should only receive the message it is targeted to. ERT should not provide information of a user in other users' messages.	All
ASR.SECURITY.03:Log browsing in ERT	ERT will provide the interface to browse through the logs behind every analysis output it shows in the dashboard.	ERT
ASR.SECURITY.04:Logging of analysis	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.	TA, IA, VA, CA
ASR.SECURITY.05:User should be able to delete his information and nothing will be used afterwards	The message about the user's wish will be immediately broadcasted to all components. Individual components will delete the data related to this user.	All

Architecture Significant Requirements	Realisation	Responsible Components
ASR.SECURITY.06:News should be anonymous	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.	ERT
ASR.SECURITY.07:Overall data security	It will be explored in later project iterations.	All

6. Conclusion

In this deliverable we propose the overall system architecture for the RESCUER platform. This architecture was created based on several requirements 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. Cross-cutting quality requirements are broken down into individual component level requirements. 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 first iteration of Task 1.2 (System Architecture), the deliverable provides the big-picture of the overall RESCUER platform. It focuses on mobile crowdsourcing data collection, analysis and display of relevant results in the command and control centre. In the next project iterations the emphasis will be on follow-up interactions with the crowd and crowd-steering. The outcome of this first iteration of Task 1.2 should be used as a guideline by the development and operation project partners when designing their own individual components. Furthermore, it can be used by other stakeholder to understand the general internal views of the RESCUER system. As a next step, the proposed architecture will be refined based on the increasingly better understanding of the requirements and of the domain.

Bibliography

[1] Keuler, Thorsten; Knodel, Jens; Naab, Matthias, Architecture-centric software and systems engineering. Fraunhofer ACES: Architecture-Centric Engineering Solutions, IESE-Report, 079.11/E, 2011 (http://publica.fraunhofer.de/dokumente/N-186361.html).

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

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 Enterprise Response Toolkit