European Project Nº: FP7-614154
Brazilian Project Nº: CNPq-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 D1.5.1
Integrated Platform Demonstrator 1

Due date of deliverable: PM10
Actual submission date: January 21, 2015

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

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

Integrated Platform Demonstrator 1

This document presents deliverable D1.5.1 (Integrated Platform Demonstrator 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 D1.5.1 provides the results of Task 1.5 (Integrated Platform Demonstrator) that are concerned with the design and implementation of an integration platform for all components developed in the RESCUER project as well as their communication patterns and interfaces.

Therefore, an integration platform that follows the hub-and-spoke topology was set up using the industry standard message-oriented middleware RabbitMQ. Consequently, RESCUER components can publish and subscribe JSON messages based on certain topics. The publish/subscribe pattern leads to a complete decoupling of sending and receiving components.

This first iteration of Task 1.5 focuses on the information flow from the crowd, through the RESCUER mobile solution, to the RESCUER Emergency Response Toolkit I solution in the command and control centre. It also defines the topic exchanges and message contents. This information flow includes the reports sent actively by users as well as the passive information gathered through the mobile devices’ sensors.

The integration platform also considers the data analysis components, which subscribe for messages from the crowd and publish the analysis results. However, in this deliverable the data analysis components’ interaction is only considered superficially. In later iterations of Task 1.5 we will define in-depth the messages for the analysis results, and the information flow from the command centre to the crowd.

List of Authors

Silas Graffy - VOMATEC
Matthias Breyer - VOMATEC
Laia G. Pedraza - VOMATEC

List of Internal Reviewers

Tobias Franke – DFKI
Taslim Arif – Fraunhofer
Vaninha Vieira – UFBA
Contents

1. Introduction .................................................................................................................................... 4
   1.1. Purpose ................................................................................................................................... 4
   1.2. Partners’ Roles and Contributions .......................................................................................... 4
   1.3. Document Overview ............................................................................................................... 5
2. System Integration Approach ......................................................................................................... 6
   2.1. Functional Decomposition and Integration Platform Approaches ......................................... 6
   2.2. Building a Central Integration Platform .................................................................................. 6
   2.3. Excursus: Rabbit MQ ............................................................................................................... 7
   2.4. Gathering requirements to Integration Middleware .................................................................... 10
3. Prototype of RESCUER Integrated Platform Demonstrator .......................................................... 12
   3.1. Overview ............................................................................................................................... 12
   3.2. Definitions and Infrastructure ............................................................................................... 13
   3.3. Information Flow from the Crowd ........................................................................................ 13
   3.4. Information Flow from the Analysis Components ................................................................ 16
   3.5. Information Flow from the Command and Control Centre .................................................. 16
4. Conclusions and Future Work ....................................................................................................... 17
   Appendix A: Questionnaire for Information Needs and Offers ...................................................... 18
   Appendix B: Information Needs and Offers per Component .......................................................... 19
   Appendix C: Data Format for Crowd Density .................................................................................. 29
   Appendix D: Data Format for Incident Report .................................................................................. 30
1. Introduction

1.1. Purpose

The RESCUER project aims at developing a smart and interoperable computer-based solution for supporting emergency and crisis management by a crowdsourcing approach, with a special focus on incidents in industrial areas and on large-scale events. The idea is to enable the crowd (workers in industrial areas and visitors in large-scale events, etc.) to directly interact with the command and control centre, for example, by sending so-called ‘incident reports’ through a smartphone app. Both the concept and the objectives of the RESCUER project are described in the Description of Work (DoW), which is part of the EC Grant Agreement (Annex I).

The purpose of this document (D1.5.1: Integrated Platform Demonstrator 1) is to describe the prototype developed in Task 1.5: Integrated Platform Demonstrator, which aims at integrating the RESCUER components into one demonstrator.

The RESCUER Integrated Platform Demonstrator is responsible for ensuring the connection among the RESCUER components. The RESCUER components are:

- Mobile Crowdsourcing Solution, for actively sending incident reports and passively recording and sending crowd density data.
- Data Analysis Solution processes the incident reports created by the Mobile Crowdsourcing in order to extract information from them. This information is meant to support the dispatcher.
  - Text Analysis Module, analyses the text written by the crowd in the incident reports.
  - Image Analysis Module, analyses the pictures taken by the crowd and attached in the incident reports.
  - Video Analysis Module, analyses the videos recorded by the crowd and attached in the incident reports.
- Emergency Response Toolkit consists of a user interface (web portal) and a component that combines all incident reports and analysis results.

Each of these components needs to send and receive data to and from the others.

In the first iteration, Task 1.5: Integrated Platform Demonstrator covers information gathering from the crowd and sending it to the command and control centre. In addition, this deliverable is meant to be a living document and it will be modified and improved together with the prototype during next iterations.

1.2. Partners’ Roles and Contributions

VOMATEC was responsible for coordinating the elaboration of this document. VOMATEC also contributed with the implementation of the integration platform, that enables all components to exchange messages, and for the demonstrator that is used in the command and control centre, the Emergency Response Toolkit (ERTK, for details see D4.4.1 Emergency Response Toolkit 1).
1.3. Document Overview

The remainder of this document is structured as follows.

- Chapter 2 describes the general idea and the approach of RESCUER integration platform.
- Chapter 3 explains how the integration platform has been set up and the Integrated Platform Demonstrator has been developed.
- Chapter 4 describes the conclusions drawn after developing this first prototype, and analyses further steps.
2. System Integration Approach

The RESCUER system is composed of different modules built using different platforms. This chapter describes approaches to achieve the communication between modules and how the integration system satisfies RESCUER needs.

2.1. Functional Decomposition and Integration Platform Approaches

A functional decomposition of RESCUER’s overall feature set results in different components made by different project partners (see D1.2.1 System Architecture 1). To build up the overall Integrated Platform Demonstrator, those building blocks have to exchange information.

A naive approach would be that every component has a direct communication interface to every other component. For n components this point-to-point (or peer-to-peer) integration pattern would lead up to n*(n-1)/2 interfaces – and, even worse, each component might have its own technical and logical definition. This approach is not feasible, in practice, since it makes the overall system maintenance quite difficult, especially when components are changed, added or removed.

A better approach for an integration pattern is hub-and-spoke. This means that there is (usually) one central component (the hub) that has a communication interface to every other component (the spokes). No other communication interface is allowed. In this case, for n components there are only n interfaces. Every time a component is changed, added or removed a single communication interface (the one between the component and the hub) has to be adjusted. This approach simplifies the overall system maintenance and makes it easy to change and extend it.

2.2. Building a Central Integration Platform

A simple approach to build a hub-and-spoke based integration platform would be to design and implement a hub as a domain-aware central middleware. This middleware will provide an information exchange platform, and deal with crosscutting concerns (e.g., central data persistency, user roles and access control management, components management, among others). However, there are significant drawbacks of such a solution, both technical and from a project management point of view.

From a technical perspective, providing an own implementation makes it difficult to guarantee non-functional requirements like high availability and reliability, scalability, security (for example transparent transport layer encryption), support different client platforms and operating systems, and others. From a project management point of view, one is going to end up having a dependency for every component that has to be maintained by one partner. However, after a publicly funded research project like RESCUER ends, the partners may lack budget to maintain the components that other partners depend on. It makes the economic exploitation of the project results quite complicated, since a lot of bilateral contracts have to be signed.

To overcome the drawbacks for the hub implementation, an existing industry standard component can be used. In RESCUER we opt for this solution and chose the RabbitMQ\(^1\), a message-

\(^1\)https://www.rabbitmq.com/
oriented middleware, for that task. RabbitMQ was chosen mainly because of its interoperability, scalability and performance advantages. Other considered alternatives, including but not limited to, Open AMQ\(^2\), ejabberd\(^3\), Apache ActiveMQ\(^4\), Open Message Queue\(^5\), ZeroMQ\(^6\), MassTransit\(^7\), Microsoft BizTalk\(^8\), and Apache Camel\(^9\) have been discarded for different reasons. Some of them did not support all needed platforms and programming languages, others did not provide infrastructure features like transport layer security out of the box. Some systems mainly targeted on integrating legacy applications instead of implementing new ones. In some cases, also systems used communication protocols that imply disadvantages because they had a bad performance or target on other use cases. From the remaining systems RabbitMQ as a widely spread industry standard offering a lot of community and also commercial support gave an excellent impression.

In RabbitMQ, and most other message-oriented middleware solutions, every component in a system (the spokes) can send messages with a certain topic to the hub. On the other hand, each component in the system can subscribe to messages of one or more topics. Using topics instead of addressing messages to a certain receiver, we completely decouple senders and receivers of messages. A sender does not have to care if there is a receiver for its message or not – or if there might be several receivers. The sender just sets the topic of the message and pushes it to the middleware. On the other hand, a receiver of a message (subscriber of a topic) does not have to care if there is one (or more) sender of messages with the subscribed topic or not. This level of decoupling together with the usage of an industry standard message exchange protocol (AMQP, Advanced Message Queuing Protocol) makes it as easy as possible to introduce, remove and change components in the RESCUER Integrated Platform Demonstrator. In enterprise application integration, this approach usually is referred to as event-driven architecture, where sending a message is considered an event.

2.3. Excursus: Rabbit MQ

RabbitMQ is a message broker using the networking protocol AMQP; it enables the communication between different applications accepting and forwarding messages, containing binary data.

The basic concepts of RabbitMQ are producer, consumer and queue. A producer is an application that sends messages to the broker (producing messages). A consumer is an application that receives messages from the broker (consuming messages). When the messages arrive to the broker they are stored in queues. A queue is a buffer and it has no storage limit. A queue can be filled by several producers, and consumed by several consumers. The consumers, producers and the broker do not need to run in the same machine.

---

\(^2\) http://www.openamq.org/
\(^3\) https://www.ejabberd.im/
\(^4\) http://activemq.apache.org/
\(^5\) https://mq.java.net/
\(^6\) http://zeromq.org/
\(^7\) http://masstransit-project.com/
\(^8\) http://www.microsoft.com/en-us/server-cloud/products/biztalk/
\(^9\) http://camel.apache.org/
The queues and the messages may be *durable* or not. The property durable means that in case that the broker quits or crashes, a queue will not be lost if it is durable. The messages inside a durable queue will be also there if they are also durable. This is a property that can be chosen by the producer of the message.

The producers send messages to an *exchange*. The exchange is in charge of receiving the messages and route them pushing them into a queue, to several queues or discard it. Figure 1 shows a simple example of a message routing. A producer sends a message to RabbitMQ which is handled by the exchange. The exchange routes the message to a single queue. This queue is consumed by only one consumer, which gets this message.

The exchange and a queue are linked by a *binding*; which is the rule that the exchange follows to route the message. The rule that define what should be done with a message, the meaning of a binding, depends on the *exchange type*. The exchange types are: *direct*, *topic*, *headers* and *fanout*.

![Diagram of general message routing in RabbitMQ](https://www.rabbitmq.com/tutorials/amqp-concepts.html)

- **Direct**: the messages are sent with a routing key and they are pushed to queues bound to the exchange with exactly the same routing key (Figure 2).

---

8

---

10 https://www.rabbitmq.com/tutorials/amqp-concepts.html
Fanout: the messages are broadcasted to all queues bound to the exchange and it does not take into account any routing key (Figure 3).

Topic: the messages are routed to queues that have a pattern as routing key matching the topic of the message. The idea is to get messages from various publish/subscribe pattern variations.

Headers: the messages are routed following the attributes of the message header. The exchange ignores the routing key.
In RESCUER system, a message may be consumed by different modules for different purposes. Each module fetches the messages depending on different properties, this leads to different routing keys patterns. Therefore, the most suitable exchange for our case is the topic exchange. A header exchange would be possible as well, but this would lead to a higher complexity without adding value, since there is no requirement in RESCUER that is not met by the topic exchange type.

The topic exchange uses routing keys structured as a list of words separated by dots (e.g. “lazy.brown.dog”). These words are meant to reflect a property of a message, so the consumer can consume only the message they are interested for. The number of characters in such a phrase is limited to 255. Words can be substituted by the placeholders “*” or “#”. The placeholder “*” means whatever word, but exactly one word, and the placeholder “#” means whatever zero or more words, e.g. with routing key “#” all messages will be consumed. These placeholders are used to add flexibility using patterns, otherwise it would act like direct exchange. Figure 4 shows an example using patterns as routing keys.

Figure 4: Topic exchange using patterns as routing keys (Source: RabbitMQ Topic Exchange 11)

2.4. Gathering requirements to Integration Middleware

To define the topics and contents for the messages managed by senders and receivers, we need to know the information needed and provided by RESCUER components. A questionnaire was set up and answered for each RESCUER component’s vendor. The questionnaire is attached as appendix A and the answers are in appendix B.

The questionnaire was used to collect the information needs of each RESCUER component. Besides, it asks also for the information they offer. The information offered by each component defines which are the messages published by it. The information needs define the messages to be consumed. The information needed of all components also determine the definition of the routing keys. Several components might want the same message depending on different properties of this one.

An example would be: the mobile crowdsourcing solution publishes the messages created by the crowd; the text analysis module wants to consume these messages but only the ones containing text; the video analysis module only wants the ones containing video; the image analysis only wants the ones containing images; and the emergency response toolkit wants all of them – plus the

11 https://www.rabbitmq.com/tutorials/tutorial-five-python.html
analysis results afterwards. In this case, the interesting attributes for each component of messages coming from the crowd are known. This is explained deeply in Chapter 3.

The questionnaires asked also for the platform and the programming language used for the implementation of the module. These information has been considered in order to choose a message broker with client libraries available for all needed technology stacks as well as finding a suitable way to serialize and deserialize the message contents. Thus, they have been chosen taking into account that there are available libraries for both aspects (sending/receiving of messages and serialization/deserialization of their contents) for all of them.

Based on the received answers, a topic pattern was defined for RabbitMQ messages and a structure for their contents. To serialize the message contents, the project consortium agreed to use JSON (JavaScript Object Notation), encoded in UTF-8.
3. Prototype of RESCUER Integrated Platform Demonstrator

The RESCUER Integration platform is a messaging middleware broker that works with the Advanced Message Queuing Protocol (AMQP) to deal with the communication between the different components of the RESCUER platform. The integration platform implementation uses RabbitMQ, which offers a common platform to push and pull messages to all components using AMQP. This chapter describes how this approached is defined for the case of RESCUER system.

3.1. Overview

Figure 5 shows a high-level overview of the message exchange between RESCUER components. The integration middleware connects to all of the components but the other ones only connect with it. With arrows is described which information is published with arrow in direction to the integration middleware and with the other direction the consumed information.

In Figure 5, there is also shown a sub-component of the integration middleware, the BLOB Storage. This component assists RabbitMQ by storing binary large objects (BLOBs), the images and videos and making them accessible through HTTP. The messages transferred between the different components via RabbitMQ will make a reference to them by a URL. This helps to keep the message payloads transferred via RabbitMQ small.

![Figure 5: RESCUER components overview 1](image)

The first iteration of Task 1.5 focuses on the information gathering from the crowd (Mobile Crowdsourcing Solution) to the command and control centre (Emergency Response Toolkit). The
flow from the command centre to the crowd is subject of the next iterations. Therefore, the following sections describe the information flow from this perspective.

### 3.2. Definitions and Infrastructure

We use a single exchange of type ‘topic’ in RESCUER, described in previous chapter. The name of the global topic exchange, where messages have to be sent to, is “RESCUER_Global_Topic_Exchange”.

VOMATEC has set up a RabbitMQ server. Connection details will be sent by email to the project consortium, since this deliverable is public.

From a RabbitMQ point of view, message payload is just a byte array. Instead of passing bytes around, in RESCUER message payload is a JSON object. Details are defined in the following section. The message payload is encoded using UTF-8.

Messages could contain binary data (BLOB), for example images and videos sent from the crowd. Those BLOBs are not directly included as part of the message payload, so they are not uploaded to RabbitMQ. Instead, they are uploaded to a blob server, where the files can be accessed through HTTP GET. The blob server URL is placed in the JSON structure of the message payload.

UFBA has set up the blob storage server. Connection details will be sent by email to the project consortium, since this deliverable is public.

In general, message topics have the following pattern:
1. The topic starts with a constant for the subject (e.g., “crowdmessage”) followed by a point (.)
2. The topic continues with none, one or many subject-specific parameter values, separated by a dot (.). For example, “true.5.red”.

### 3.3. Information Flow from the Crowd

Information flow from the crowd can be separated in crowd sourcing and crowd sensing. Crowd sourcing means that people in the crowd actively send data using their mobile devices, the incident reports. Crowd sensing means that the people’s devices passively gather sensor information (e.g., position) and send it as a continuous stream to the Data Analysis module. This data can be used, for example, to calculate the crowd density.

Since the crowd sensing is a continuous data stream, a message-based approach may not be suitable. Furthermore, the Sensor Analysis Module can be accessed via HTTP GET for the current crowd density (overall output of the crowd sensing solution). The format of the crowd density data is defined in Appendix C, the URL is internal to the consortium and will be distributed via email by the DFKI. In this version of the RESCUER Integration Platform, we decided to use the polling approach, by which an interested component (e.g., the ERTK) can decide in what interval it wants to receive updates. This solution has a drawback that components coupling gets tighter since the interested component has to know where to get the data from. Therefore, in later iterations of this task, a new component will be introduced to act as an abstraction layer by polling the crowd density data and sending it to RabbitMQ with the topic “crowddensity:”.

13
The topic definition for a crowd message (e.g., incident report) actively sent from the mobile app to the Integration Platform is described below and its parameters are explained in Table 1:

**Message topic:** crowdmessage.<text>.<image>.<video>.<role>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>Indicates if the message contains text information</td>
<td>[true, false]</td>
</tr>
<tr>
<td>Image</td>
<td>Indicates if the message contains one or more images</td>
<td>[true, false]</td>
</tr>
<tr>
<td>video</td>
<td>Indicates if the message contains one or more videos</td>
<td>[true, false]</td>
</tr>
<tr>
<td>role</td>
<td>Indicates if the message contains role of sender</td>
<td>[visitor, employee, workforce]</td>
</tr>
</tbody>
</table>

By using this topic pattern the routing keys used by each component to publish and consume can be defined. Below, there are several examples.

The mobile app sending an incident report written by a workforce, which contains text and video but no images would do this by publishing a message with the topic “crowdmessage.true.false.true.workforce”.

The video analysis component could subscribe with the routing key “crowdmessage.*.true.*.*” or even “crowdmessage.*.true.#”. The first one specifies that there should be five words in the routing key, but only specifies two of them ignoring the value of the other ones. In the second case, it sets which one should be the first and the third word, that all values for the second word are valid and after the third word there is no restriction, it ignores whether there are more words or not and their values.

The ERTK subscribes to “crowdmessage.#”. In this case, ERTK consumes all queues bound to a topic with a first word “crowdmessage”.

The decision to include the sender role in the queue is due messages sent by a workforce might be processed with a higher priority (or by a different component). To do this in an elegant way, they can be filtered by using separate queues, when a component subscribe with a routing key specifying that the last word or the word at the fifth position should be “workforce”.

The content of a message for an incident report sent from the crowdsourcing solution is shown in Appendix D. The following structure is an example of a report containing:

- **reportTimestamp**: timestamp of when has the report been sent,
- **reportSender**: information about the user of the crowdsourcing solution who has send the report,
- **text**: text written in the report,
- **mediaPhoto**: data related to the picture attached to the report, it does not include the picture data but the URL where it has been uploaded,
- **mediaVideo**: data related to the video attached to the report, it does not include the video data but the URL where it has been uploaded,
- **keyword**: type of incident that crowdsourcing solution user wants to report,
• **reportPosition**: geographical information of report, and
• **reportIdentifier**: is a unique identifier in order to later on link all data related to the same report.

Example of an Incident Report message content:

```json
{
    "reportTimestamp":"2014-07-04T12:48:06.4147832+02:00",
    "reportSender":{
        "personRole":1, /* 1=visitor, 2=employee, 3=workforce */
        "personPosition":{
            "latitude":47.374275322995,
            "longitude":8.5428106828621733,
            "sequenceNumber":0,
            "altitude":0.0
        }
    },
    "text":"Injured people Explosion",
    "mediaPhoto":{
        "mediaPath":"http://exampleURL.com/images/fire3.jpg", /* URL of blob storage server, see section 3.3 */
        "mediaIdentifier":"1132451",
        "size":null,
        "quality":null,
        "type":"image/png",
        "mediaTimestamp":"2014-07-04T09:43:00.4147832-03:00",
        "location":{
            "latitude":47.374275322995,
            "longitude":8.5428106828621733,
            "sequenceNumber":0,
            "altitude":0.0
        }
    },
    "mediaVideo":{
        "mediaPath":"http://exampleURL.com/data/videos/massa.mp4", /* URL of blob storage server, see section 3.3 */
        "mediaIdentifier":"1132453",
        "size":null,
        "quality":null,
        "type":"video/mp4",
        "mediaTimestamp":"2014-07-04T09:43:00.4147832-03:00",
        "location":{
            "latitude":47.374275322995,
            "longitude":8.5428106828621733,
            "sequenceNumber":0,
            "altitude":0.0
        },
        "accelerometer":0.0,
        "gyroscope":0.0
    }
}
```

"keyword":"Fire",
"reportPosition":{
    "latitude":47.374275322995,
    "longitude":8.5428106828621733,
    "sequenceNumber":0,
    "altitude":0.0}
3.4. Information Flow from the Analysis Components

The components that analyse text, image and video data will subscribe to “crowdmessage” topics to get all raw data from the crowd. After the analysis they will publish their results using an equivalent topic pattern. The role of the original sender is just passed through the analysis component, again to enable subscribers to filter and prioritise based on the sender role. The topic used by the Data Analysis Modules to publish the data is:

Message topic: “analysedresult”.<text>.<image>.<video>.<role>

The fields “text”, “image”, “video” and “role” are the same as in the case of the messages coming from the crowdsourcing solution. Therefore, their usage is described in Table 1: Crowd messages topic definition.

Message contents (example):

The message contents of analysed results are subject to later iterations of the Integrated Platform Demonstrator task.

3.5. Information Flow from the Command and Control Centre

This section is subject to further iteration of Task 1.5. It is mentioned here only for completeness reasons.
4. Conclusions and Future Work

The integration platform demonstrator has been implemented as an event-driven architecture using RabbitMQ because of its hub-and-spoke topology and the AMQP standard protocol. The type of exchange chosen for the RESCUER system is ‘topic’ since this allows the easy definition of message flows using routing key structure.

The current state of the middleware broker only takes into account the information flow from the crowd sourcing solution, since the first iteration focuses on information gathering from the crowd to the command centre. The other information flows will be analysed and the routing keys improved and added, as needed, in later iterations.

In future iterations it will be investigated if RESCUER could benefit from other RabbitMQ features that have not been used, as for example persistent queues to allow messages to be received later, even when the subscriber was offline when the message was sent. Another interesting concept is task queues, which dispatch messages between several subscribers of the same message topics using a round robin approach. This simplifies the design of very scalable distributed systems.
Appendix A: Questionnaire for Information Needs and Offers

Questionnaire: Information needs and offers of RESCUER partners

Information Needs

1. Which information do you need?
2. Can you give a detailed explanation about this information?
3. What will you use this information for?
4. Do you know who provide this information?

<table>
<thead>
<tr>
<th>No.</th>
<th>Information Need</th>
<th>Detailed Description</th>
<th>What do you need it for?</th>
<th>Who offers it?</th>
</tr>
</thead>
<tbody>
<tr>
<td>N1</td>
<td>People Coordinates</td>
<td>The available coordinates from the mobile app running with a latency of 5s.</td>
<td>To analyse it and obtain the crowd density data.</td>
<td>Mobile Application</td>
</tr>
<tr>
<td>N2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Information Offers

5. Which information do you offer?
6. Do you know who need this information?

<table>
<thead>
<tr>
<th>No.</th>
<th>Information Offer</th>
<th>Detailed Description</th>
<th>Who may need it?</th>
</tr>
</thead>
<tbody>
<tr>
<td>O1</td>
<td>Crowd Density Data</td>
<td>JSON data describing the density of people vs coordinates.</td>
<td>Emergency Response Toolkit to show it in the map.</td>
</tr>
<tr>
<td>O2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Name of partner:
Component(s) to develop:
Programming Language(s):
Platforms (Hardware, Operating System):
Appendix B: Information Needs and Offers per Component

Questionnaire: Information needs and offers of RESCUER partners

### Information Needs

1. Which information do you need?
2. Can you give a detailed explanation about this information?
3. What will you use this information for?
4. Do you know who provide this information?

<table>
<thead>
<tr>
<th>No.</th>
<th>Information Need</th>
<th>Detailed Description</th>
<th>What do you need it for?</th>
<th>Who offers it?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Name of partner: Fraunhofer, DFKI, MTM
Component(s) to develop: RESCUER Mobile Solution
Programming Language(s): PhoneGap: JavaScript, HTML5, and CSS3
Platforms (Hardware, Operating System): Android, iOS
### Information Offers

5. Which information do you offer?
6. Do you know who need this information?

<table>
<thead>
<tr>
<th>No.</th>
<th>Information Offer</th>
<th>Detailed Description</th>
<th>Who may need it</th>
</tr>
</thead>
<tbody>
<tr>
<td>O1.1</td>
<td>Crowd Density Data</td>
<td>JSON data describing the density of people vs coordinates.</td>
<td>Emergency Response Toolkit to show it in the map.</td>
</tr>
<tr>
<td>O1.2</td>
<td>Incident local</td>
<td>Where is the incident?</td>
<td>Emergency Response Toolkit to show it in the map.</td>
</tr>
<tr>
<td>O1.3</td>
<td>Incident perimeter</td>
<td>Definition of incident perimeters i.e. critical zone, isolation zone,…</td>
<td>Emergency Response Toolkit to show it in the map.</td>
</tr>
<tr>
<td>O1.4</td>
<td>Incident type</td>
<td>What it happened?</td>
<td>Emergency Response Toolkit to show it in a summary board.</td>
</tr>
<tr>
<td>O1.5</td>
<td>Incident time</td>
<td>When it happened?</td>
<td>Emergency Response Toolkit to show it in a summary board.</td>
</tr>
<tr>
<td>O1.6</td>
<td>Affected people</td>
<td>Who is affected?</td>
<td>Emergency Response Toolkit to show it in a summary board.</td>
</tr>
<tr>
<td>O1.7</td>
<td>User profile</td>
<td>who is using the app and consequently providing the information</td>
<td>Emergency Response Toolkit to calculate reliability.</td>
</tr>
<tr>
<td>O1.8</td>
<td>Potential hazardous materials</td>
<td>Dangerous materials stored in the incident area.</td>
<td>Emergency Response Toolkit to show it in the map.</td>
</tr>
<tr>
<td>O1.9</td>
<td>Integrity of physical infrastructures</td>
<td>Description of damages in buildings, roads... and their identification</td>
<td>Emergency Response Toolkit to show it in a summary board.</td>
</tr>
<tr>
<td>O1.10</td>
<td>Traffic Information</td>
<td>The current usage of the road infrastructure.</td>
<td>Emergency Response Toolkit to show it in the map.</td>
</tr>
<tr>
<td>No.</td>
<td>Information Offer</td>
<td>Detailed Description</td>
<td>Who may need it?</td>
</tr>
<tr>
<td>------</td>
<td>-------------------------------------------</td>
<td>---------------------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>O1.11</td>
<td>People movement dynamic</td>
<td>Movement patterns of people in the incident area and neighbourhood</td>
<td>Emergency Response Toolkit to show it in the map.</td>
</tr>
<tr>
<td>O1.12</td>
<td>Conducted protective actions</td>
<td>Protective action that have been taken</td>
<td>Emergency Response Toolkit to show it in the map and in a summary board (?)</td>
</tr>
<tr>
<td>O1.13</td>
<td>Public safety and security status</td>
<td>Occurrence of violent actions, e.g. robberies, plunder, aggressions</td>
<td>Emergency Response Toolkit to show it in the map and in a summary board (?)</td>
</tr>
<tr>
<td>O1.14</td>
<td>Incoming and understanding of messages</td>
<td>Confirmation that instruction were read and understood, especially in the case of instructions for the operational forces in venue or specialists working on the response against HAZMATs</td>
<td>Emergency Response Toolkit to show it in a summary board.</td>
</tr>
<tr>
<td>O1.15</td>
<td>Human resources</td>
<td>Information about the usage of human resources for the emergency situation.</td>
<td>Emergency Response Toolkit to show it in a summary board.</td>
</tr>
<tr>
<td>O1.16</td>
<td>Equipment</td>
<td>Information about the usage of equipment for the emergency situation.</td>
<td>Emergency Response Toolkit to show it in a summary board.</td>
</tr>
<tr>
<td>O1.17</td>
<td>Images/Videos + Meta Data</td>
<td>Images and videos of the incident. This should support to extract anyway the information need described above.</td>
<td>Image/Video Analysis Component.</td>
</tr>
</tbody>
</table>

We are refining this information need in the data model shown in Figure “Data model for the RESCUER app” of deliverable D2.2.1.
Questionnaire: Information needs and offers of RESCUER partners

Information Needs

1. Which information do you need?
2. Can you give a detailed explanation about this information?
3. What will you use this information for?
4. Do you know who provide this information?

<table>
<thead>
<tr>
<th>No.</th>
<th>Information Need</th>
<th>Detailed Description</th>
<th>What do you need it for?</th>
<th>Who offers it?</th>
</tr>
</thead>
<tbody>
<tr>
<td>N2.1</td>
<td>(Raw) Text</td>
<td>The textual description of the incident sent by people with the mobile application OR textual description sent with images and videos</td>
<td>To analyse it and obtain semantic annotations for the Emergency Response Toolkit</td>
<td>Mobile Crowdsourcing Solution (Dispatcher component)</td>
</tr>
</tbody>
</table>

Information Offers

5. Which information do you offer?
6. Do you know who need this information?

<table>
<thead>
<tr>
<th>No.</th>
<th>Information Offer</th>
<th>Detailed Description</th>
<th>Who may need it?</th>
</tr>
</thead>
<tbody>
<tr>
<td>O2.1</td>
<td>Semantic text annotations about incident</td>
<td>(Probably JSON) data structure containing the original text and the semantic annotations covering at least type of incident (what), additional location descriptions (where), and information about affected people (who)</td>
<td>Emergency Response Toolkit</td>
</tr>
</tbody>
</table>
Questionnaire: Information needs and offers of RESCUER partners

Information Needs

1. Which information do you need?
2. Can you give a detailed explanation about this information?
3. What will you use this information for?
4. Do you know who provide this information?

<table>
<thead>
<tr>
<th>No.</th>
<th>Information Need</th>
<th>Detailed Description</th>
<th>What do you need it for?</th>
<th>Who offers it?</th>
</tr>
</thead>
<tbody>
<tr>
<td>N3.1</td>
<td>Image and video with structured data (e.g. geographic location, time, image and video resolution) that presents fire, smoke and/or explosion.</td>
<td>We will use only the images/videos that present minimum digital quality so that we can process it analytically (Filtering Step) and return the data with annotation of crisis situations and extracted features generated in the analysis module.</td>
<td>We will classify, cluster, match and perform similarity querying with the data to identify crises, and to spot regions of interest that indicate fire, smoke, and/or explosion.</td>
<td>These data must come from T3.1 leaded by UPM.</td>
</tr>
</tbody>
</table>

Information Offers

5. Which information do you offer?
6. Do you know who need this information?

<table>
<thead>
<tr>
<th>No.</th>
<th>Information Offer</th>
<th>Detailed Description</th>
<th>Who may need it?</th>
</tr>
</thead>
<tbody>
<tr>
<td>O3.1</td>
<td>We offer sets of images/videos geo refereed and time contextualized that have the potential of assisting the Command Centre in monitoring the crises.</td>
<td>The images/videos shall be tagged (what, who, where) indicating crisis situations along with details about the fire, smoke, and/or explosion that started the crisis.</td>
<td>Emergency Response Toolkit</td>
</tr>
</tbody>
</table>

Name of partner: USP (University of Sao Paulo)
Component(s) to develop: T3.2, modules that form the data analysis system
Programming Language(s): C++/Java
Platforms (Hardware, Operating System): X86 PCs and X86 high-end servers; Windows
Questionnaire: Information needs and offers of RESCUER partners

Information Needs

1. Which information do you need?
2. Can you give a detailed explanation about this information?
3. What will you use this information for?
4. Do you know who provide this information?

<table>
<thead>
<tr>
<th>No.</th>
<th>Information Need</th>
<th>Detailed Description</th>
<th>What do you need it for?</th>
<th>Who offers it?</th>
</tr>
</thead>
<tbody>
<tr>
<td>N4.1</td>
<td>Location</td>
<td>All data received should be geo-located.</td>
<td>To cluster data based on its location and then filter them.</td>
<td>Mobile Application</td>
</tr>
<tr>
<td>N4.2</td>
<td>Video sequences</td>
<td>Video sequences recorded by end-users, whose duration is more than 15 seconds.</td>
<td>To analyse them and extract as much information about the crowd as possible in the Video Analysis Component.</td>
<td>Mobile Application</td>
</tr>
<tr>
<td>N4.3</td>
<td>Frame stamps</td>
<td>Timestamps of each frame that compose the video sequence. Alternatively, we could work with the timestamp of the beginning of the recording and the frame rate to obtain a temporal reference for each frame.</td>
<td>To analyse the quantity of motion in the scene for the Video Filtering Component.</td>
<td>Mobile Application</td>
</tr>
<tr>
<td>N4.4</td>
<td>Orientation measurements</td>
<td>Orientation measurements during the video recording with the objective to associate the orientation of the mobile to each frame of the video.</td>
<td>To analyse the quantity of motion in the scene for the Video Filtering Component.</td>
<td>Mobile Application</td>
</tr>
</tbody>
</table>

Name of partner: UPM
Component(s) to develop: Video Fusion & Filtering Components, Video Analysis Component
Programming Language(s): C++
Platforms (Hardware, Operating System):
- Two computers for parallelize processes at least.
  - Minimum requirements: Intel Core i7, 2.8 GHz, 4GB RAM (using now).
  - OS: Linux.
<table>
<thead>
<tr>
<th>No.</th>
<th>Information Need</th>
<th>Detailed Description</th>
<th>What do you need it for</th>
<th>Who offers it?</th>
</tr>
</thead>
<tbody>
<tr>
<td>N4.5</td>
<td>Orientation timestamps</td>
<td>Timestamps of each orientation sample.</td>
<td>To synchronize the temporal references of the frames with the orientation samples in the Video Filtering Component.</td>
<td>Mobile Application</td>
</tr>
<tr>
<td>N4.6</td>
<td>Acceleration measurements</td>
<td>Acceleration measurements during the video recording with the objective to associate the acceleration of the mobile to each frame of the video.</td>
<td>To analyse the quantity of motion in the scene for the Video Filtering Component.</td>
<td>Mobile Application</td>
</tr>
<tr>
<td>N4.7</td>
<td>Acceleration timestamps</td>
<td>Timestamps of each acceleration sample.</td>
<td>To synchronize the temporal references of the frames with the acceleration samples in the Video Filtering Component.</td>
<td>Mobile Application</td>
</tr>
</tbody>
</table>

**Information Offers**

5. Which information do you offer?
6. Do you know who need this information?

<p>| No. | Information Offer | Detailed Description                                                                 | Who may need it?                                                                                     |
|-----|-------------------|-------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------|-----------------------|
| O4.1| Location          | All data provided by the Video Analysis Component will be geo-located.              | Emergency Response Toolkit to show it in the map.                                                    |
| O4.2| Kind of event     | Kind of event that is contained in the video-sequence:                              | Emergency Response Toolkit to show it in the map.                                                    |
|     |                   | • Smoke                                                                              |                                                                                                     |
|     |                   | • Fire                                                                               |                                                                                                     |
|     |                   | • Crowd                                                                              |                                                                                                     |
|     |                   | • Combination.                                                                      |                                                                                                     |
| O4.3| People by m²      | Estimation of the people/m² presents in the video sequence.                          | Emergency Response Toolkit to show it in the map.                                                    |</p>
<table>
<thead>
<tr>
<th>No.</th>
<th>Information Offer</th>
<th>Detailed Description</th>
<th>Who may need it?</th>
</tr>
</thead>
<tbody>
<tr>
<td>O4.4</td>
<td>Risk of congestion</td>
<td>Risk of congestion alarm based on the density temporal analysis that the Video Analysis Component realises.</td>
<td>Emergency Response Toolkit to show it in the map.</td>
</tr>
<tr>
<td>O4.5</td>
<td>Expanse velocity</td>
<td>Crowd/Fire/Smoke expanse velocity based on the density temporal analysis that the Video Analysis Component realises.</td>
<td>Emergency Response Toolkit to show it in the map.</td>
</tr>
</tbody>
</table>
# Questionnaire: Information needs and offers of RESCUER partners

## Information Needs

1. Which information do you need?
2. Can you give a detailed explanation about this information?
3. What will you use this information for?
4. Do you know who provide this information?

<table>
<thead>
<tr>
<th>No.</th>
<th>Information Need</th>
<th>Detailed Description</th>
<th>What do you need it for?</th>
<th>Who offers it?</th>
</tr>
</thead>
<tbody>
<tr>
<td>N5.1</td>
<td>People Coordinates</td>
<td>The available coordinates from the mobile app uploaded once a minute.</td>
<td>To analyse it and obtain the crowd density data.</td>
<td>Sensing Library (built into the mobile application)</td>
</tr>
<tr>
<td>N5.2</td>
<td>User ID</td>
<td>The unique ID of each app user.</td>
<td>To be able to assign uploaded data to an anonymous entity and to filter crowd data.</td>
<td>Sensing Library (built into the mobile application)</td>
</tr>
<tr>
<td>N5.3</td>
<td>Messages (*)</td>
<td>Messages to be received by the mobile app.</td>
<td>To present the mobile app users with messages from the command and control centre.</td>
<td>Later on: the message queue (which will be fed by the Emergency Response Toolkit) Right now: DFKI message server (fed by a DFKI Web UI)</td>
</tr>
<tr>
<td>N5.4</td>
<td>Outgoing Text Data (***)</td>
<td>Everything text based that the app is sending to the command and control centre.</td>
<td>Since the DFKI Library is responsible for enabling the Ad-hoc mode, it will need to route all outgoing text data to make sure that it will be sent out via Ad-hoc during scenarios where the network is down.</td>
<td>Mobile application (after the mobile application created all messages for all analysis components, the message should be handed to the DFKI library, which sends it out).</td>
</tr>
</tbody>
</table>

Name of partner: DFKI  
Component(s) to develop: DFKI Library / Crowd Density Server  
Programming Language(s): DFKI Library: Objective-C/Java, Server: Python  
Platforms (Hardware, Operating System): Server Cluster running Linux (Debian Wheezy AMD 64)
*: receiving messages is being handled through the DFKI Library since we’re also responsible for switching between normal mode and AdHoc mode which directly affects the way messages are received.

**: due to size limitations, the AdHoc mode won’t be able to send out binary data. Hence, the DFKI library will only route text data.

Information Offers

5. Which information do you offer?
6. Do you know who need this information?

<table>
<thead>
<tr>
<th>No.</th>
<th>Information Offer</th>
<th>Detailed Description</th>
<th>Who may need it?</th>
</tr>
</thead>
<tbody>
<tr>
<td>O5.1</td>
<td>Crowd Density Data</td>
<td>JSON data describing the density of people vs coordinates.</td>
<td>Emergency Response Toolkit to show it in the map.</td>
</tr>
<tr>
<td>O5.2</td>
<td>Textual Data</td>
<td>Messages and other text-based information (like reports) from the mobile app.</td>
<td>Emergency Response Toolkit</td>
</tr>
</tbody>
</table>
Appendix C: Data Format for Crowd Density

{
  "max": 7,
  "stats": {
    "active_users": 1788,
    "ts": 1373047440,
    "max_timestamp": 1373241360.0
  },
  "data": [
    {
      "lat": 47.37340013035328,
      "count": 1,
      "lng": 8.542822217576711
    },
    {
      "lat": 47.36695102421234,
      "count": 1,
      "lng": 8.543362421557969
    },
    {
      "lat": 47.36456543677241,
      "count": 1,
      "lng": 8.53699210100671
    },
    {
      "lat": 47.36904095397086,
      "count": 1,
      "lng": 8.543561752893536
    }
  ]
}
Appendix D: Data Format for Incident Report

```json
{
    "reportTimestamp":"2014-07-04T12:48:06.4147832+02:00",
    "reportSender":{
        "personRole":1, /* 1=visitor, 2=employee, 3=workforce */
        "personPosition":{
            "latitude":47.3742753222995,
            "longitude":8.5428106828621733,
            "sequenceNumber":0,
            "altitude":0.0
        },
        "personExperience":false,
        "movementProfile":[
            {
                "position":{
                    "latitude":47.377275322299504,
                    "longitude":8.5498106828621729,
                    "sequenceNumber":0,
                    "altitude":0.0
                },
                "date":"2014-07-04T12:48:06.4147832+02:00"
            },
            {
                "position":{
                    "latitude":47.3712753222995,
                    "longitude":8.5468106828621728,
                    "sequenceNumber":0,
                    "altitude":0.0
                },
                "date":"2014-07-04T12:48:06.4147832+02:00"
            }
        ],
        "text":"Injured people Explosion",
        "mediaPhoto":[
            {
                "mediaPath":"http://exampleURL.com/images/fire3.jpg", /* URL of blob storage server, see section 3.3 */
                "mediaIdentifier":"1132451",
                "size":null,
                "quality":null,
                "type":"image/png",
                "mediaTimestamp":"2014-07-04T09:43:00.4147832-03:00",
                "location":{
                    "latitude":47.3742753222995,
                    "longitude":8.5428106828621733,
                    "sequenceNumber":0,
                    "altitude":0.0
                },
                "accelerometer":0.0,
                "gyroscope":0.0
            },
            {
                "mediaPath":"http://exampleURL.com/images/incident.png", /* URL of blob storage server, see section 3.3 */
```
"mediaIdentifier":"1132452",
"size":null,
"quality":null,
"type":"image/png",
"mediaTimestamp":"2014-07-04T09:43:00.4147832-03:00",
"location":{
  "latitude":47.3742753222995,
  "longitude":8.5428106828621733,
  "sequenceNumber":0,
  "altitude":0.0
},
"accelerometer":0.0,
"gyroscope":0.0
},
"mediaVideo":[
  {
    "mediaPath":"http://exampleURL.com/data/videos/massa.mp4", /* URL of blob storage server, see section 3.3 */
    "mediaIdentifier":"1132453",
    "size":null,
    "quality":null,
    "type":"video/mp4",
    "mediaTimestamp":"2014-07-04T09:43:00.4147832-03:00",
    "location":{
      "latitude":47.3742753222995,
      "longitude":8.5428106828621733,
      "sequenceNumber":0,
      "altitude":0.0
    },
    "accelerometer":0.0,
    "gyroscope":0.0
  }
],
"keyword":"Fire",
"reportPosition":{
  "latitude":47.3742753222995,
  "longitude":8.5428106828621733,
  "sequenceNumber":0,
  "altitude":0.0
},
"injuredPeople":true,
"hazardMaterial":{
  "materialDivision":{
    "DivisionID":12,
    "DivisionName":"divisionName",
    "Description":"description",
    "Media":{
      "mediaPath":"http://exampleURL.com/images/mass3.jpg", /* URL of blob storage server, see section 3.3 */
      "size":null,
      "quality":null,
      "type":"image/png",
      "mediaTimestamp": "2014-07-04T09:43:00.4147832-03:00",
      "location":{
        "latitude":47.3692753222995,
        "longitude":8.5448106828621722,
        "sequenceNumber":0,
        "altitude":0.0
      }
    }
  }
}
"altitude":0.0,
"accelerometer":0.0,
"gyroscope":0.0
},
"ClassID":0
},
"materialClass":{
"ClassID":0,
"ClassName":"Physical Hazard",
"Description":"explosive water reaction, detonation or explosive decomposition, polymerization, or self-reaction at normal temperature and pressure",
"Media":{
"mediaPath":"http://exampleURL.com/images/mass3.jpg",
"size":null,
"quality":null,
"type":"image/png",
"mediaTimestamp":"2014-07-04T09:43:00.4147832-03:00",
"location":{
"latitude":47.3692753222995,
"longitude":8.5448106828621722,
"sequenceNumber":0,
"altitude":0.0
}
}
},
"materialPosition":{
"latitude":47.3692753222995,
"longitude":8.5448106828621722,
"sequenceNumber":0,
"altitude":0.0
},
"timestamp":"2014-07-04T12:48:06.4147832+02:00"
},
"quantityInjuriedPeople":"22",
"reportIdentifier":"113245"}