beAWARE

Enhancing decision support and management services in extreme weather climate events

700475

D7.1 Technological Roadmap

Dissemination level:	Public
Contractual date of delivery:	Month 6, 30 June 2017
Actual date of delivery:	Month 6, 30 June 2017
Workpackage:	WP7 System development, integration and evaluation
Task:	T7.1 System Requirements and Architecture
Type:	Report
Approval Status:	Final Draft
Version:	1.0
Number of pages:	52
Filename:	d7.1_beaware_technological_roadmap_2017-6-30_v1.0.docx

Abstract

The objective of this document is to define the outline of the time plan for the development of the beAWARE platform. In this context, the deliverable provides a high level view of the beAWARE platform architecture and a specification of the infrastructure layout. In addition the deliverable includes an initial functional description of each of the technical modules, describing the supporting technologies, the increasing functionalities over time and the development timeline. Finally, the deliverable includes a summary of the use cases and a global project timeline, describing the platform's scheduled iterations and the levels of functionality at the project milestones.

Co-funded by the European Union

History

Version	Date	Reason	Revised by
0.1	05.06.2017	Deliverable structure	All partners
0.2	22.06.2017	Contributions by all partners	All partners
0.3	23.06.2017	Integrated document	CERTH
0.4	24.06.2017	Internal review of the document	B. Mandler (IBM)
1.0	30.06.2017	Final version	CERTH

Author list

Organisation	Name	Contact Information
CERTH	Yiannis Kompatsiaris	ikom@iti.gr
CERTH	Anastasios Karakostas	akarakos@iti.gr
CERTH	Konstantinos Avgerinakis	koafgeri@iti.gr
CERTH	Efstratios Kontopoulos	skontopo@iti.gr
IBM	Benjamin Mandler	MANDLER@il.ibm.com
IBM	Evgeniy Hvostenko	EVGENIYH@il.ibm.com
MSIL	Israel Altar	Israel.altar@MotorolaSolutions.com
MSIL	Yaniv Mordecai	Yaniv.Mordecai@MotorolaSolutions.com
IOSB	Hylke van der Schaaf	hylke.vanderschaaf@iosb.fraunhofer.de
UPF	Stamatia Dasiopoulou	stamatia.dasiopoulou@upf.edu
UPF	Jens Grivolla	jens.grivolla@upf.edu
UPF	Gerard Casamayor	gerard.casamayor@upf.edu
UPF	Simon Mille	simon.mille@upf.edu
UPF	Leo Wanner	leo.wanner@upf.edu
AAWA	Daniele Norbiato	daniele.norbiato@adbve.it
AAWA	Michelle Ferri	michele.ferri@adbve.it
PLV	Carmen Castro	proyectosplv@valencia.es
HRT	Iosif Vourvachis	projects@hrt.org.gr
FBBR	Amalie Janniche	amjan@fbbr.dk

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 700475

Executive Summary

This deliverable presents the technological roadmap towards the implementation of the beAWARE platform.

First, this roadmap presents a high-level view of the envisioned architecture for the beAWARE platform, illustrating the conceptual architecture and a draft of the complete technical architecture. Then, the deliverable describes the modules that are developed in each Work Package and comprise the beAWARE platform. For each module, a functional description is provided including a summary of its scientific and technical objectives and supporting technologies, input/output specifications, the increasing functionality over time, the development timeline, as well as the detailing schedule and resource allocation. In addition, D7.1 provides a summarised vision of the proposed use cases and their implementation strategy. Finally, a global project timeline is presented, describing the platform's scheduled iterations and the levels of functionality at the project milestones.

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 700475

Abbreviations and Acronyms

ASR	Automatic speech recognition
CCTV	Close Circuit TV
DTr	Dynamic texture recognition
GIS	Geographical Information System
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
KB	Knowledge Base
ObjD	Object detection
OWL	Web Ontology Language
PSAP	Public-safety answering point
RDF	Resource Description Framework
REST	Representational State Transfer
STL	Spatio-temporal localization
TLc	Traffic level classification
UC	Use Case
WP	Work Package
XML	Extensible Mark-up Language

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 700475

Table of Contents

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 700475

List of Figures

List of Tables

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 700475

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 700475

1 Introduction

This document presents a roadmap of how the Consortium plans to develop the beAWARE platform. The objective of the deliverable is to specify in detail: (i) the progressively increasing functionality (in particular at the time points of the milestones) of the individual modules as well as of the platform as a whole, (ii) the temporal and functional synchronisation of the individual modules to ensure this functionality; (iii) the resources that will be needed to achieve this functionality; (iv) the technical infrastructure specifications.

To this end, and based on a model of iterative development, the roadmap includes:

1. A high-level view of the envisioned Platform architecture, illustrating the conceptual architecture and a draft of the complete technical architecture. A specification of the infrastructure layout and topology is also provided.
2. A functional description of each of the technical modules, describing:
    a. A summary of its scientific and technical objectives, and supporting technologies;
    b. Its progressively increasing functionality over time;
    c. Its timeline, detailing schedule and resource allocation;
3. A summarised vision of the proposed use cases and their implementation strategy;
4. A global project timeline, describing the platform's scheduled iterations and the levels of functionality at the project milestones;

Therefore, this deliverable is structured as follows. Section 2 presents the architecture. Section 3 provides the functional description of the modules involved in alignment with the project Work Packages (WPs). Section 4 presents the development cycle and briefly discusses the use cases. Finally, Section 5 provides the overall project timeline and the milestones, while Section 6 concludes the deliverable.

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 700475

2 Architecture

The forming architecture is formulated with the ultimate goal of providing accuracy and functionality for decision support systems to be able to respond well to weather related disaster scenarios.

The architecture is roughly made up of the following layers:

1. **Ingestion** layer, containing mechanisms and channels through which data is brought into the platform;
2. **Internal services**, generic data repositories and communication services being used by the different components;
3. **Business** layer, containing the components that perform the actual platform-specific capabilities;
4. **External facing** layer, including the end-users' applications and PSAP (Public-safety answering point) modules, interacting with people and entities outside the platform (end-users of the platform).

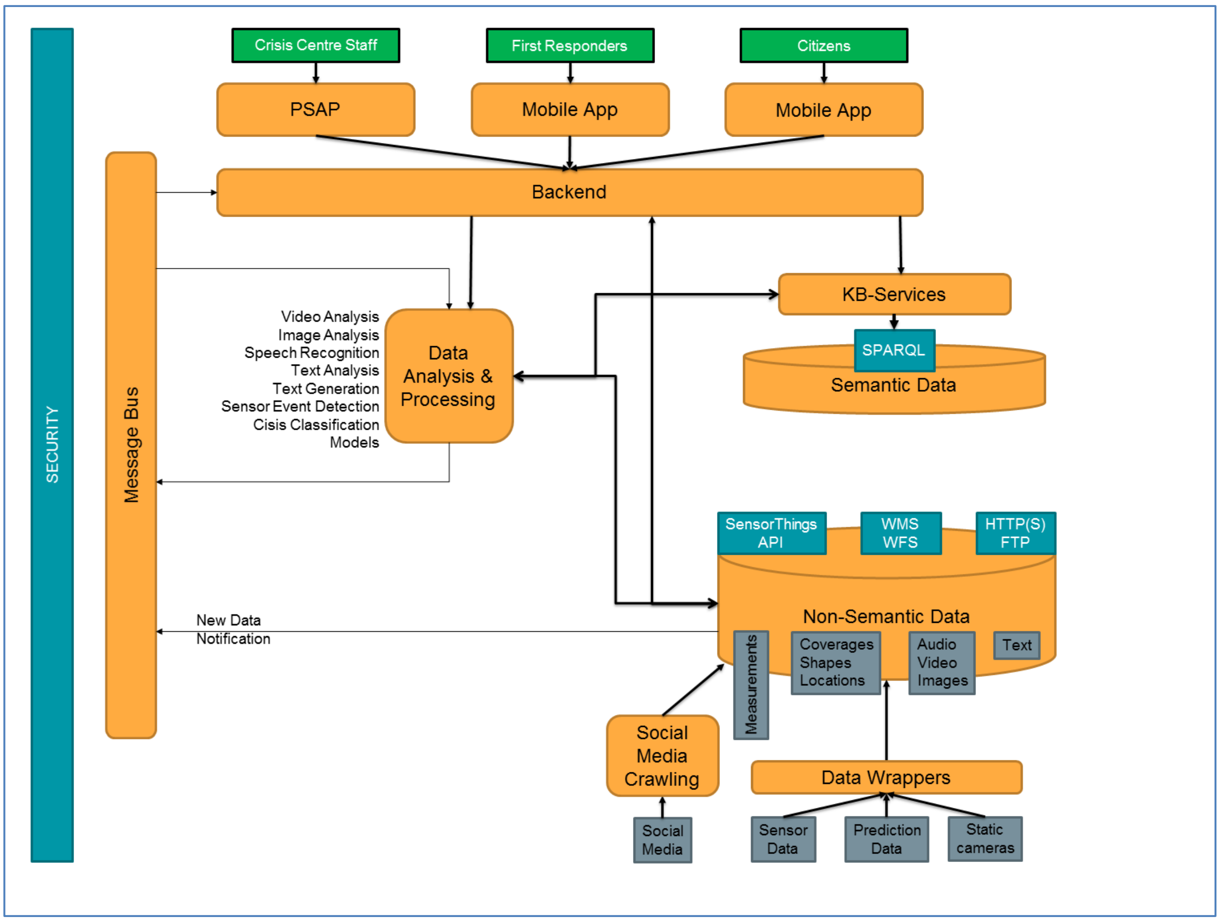


Figure 1: Architectural high-level view

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 700475

2.1 Description

As can be derived from Figure 1, the first step in a generic flow of the platform consists of new data being pushed into the platform. The new data can originate from specific beAWARE applications used by civilians (end-users) or first responders. Note that these applications may be bi-directional as they can serve to communicate back with the users upon need. Moreover, incoming data to the platform can originate from other sources as well, such as IoT devices, social media pots grabbed by crawlers, and weather data, to name a few.

Once a new piece of data is successfully ingested into the system, the data is stored in a temporary raw data store and the availability of the new piece of data is broadcasted to all interested components using a specific topic of the messaging service. There shall be a separate topic for each kind of information flowing into the system, such that only components which are interested in that specific kind of data will be made aware of the arrival of a new relevant piece of data.

All interested parties will receive the information, which includes a pointer enabling to access the data, and shall perform their specific analysis on the new data. Note, that the result of such an analysis may in turn create a new piece of data which is of interest to other components, that once again will be made aware and access the data in the same manner explained above, providing services for crisis classification and early warning.

The final results of the analysis can be added to the knowledge base, and if required will notify PSAP related components of the identification of a new state.

The PSAP in turn may use beAWARE components, including apps to manage and alert the different stakeholders (citizens, first responders, and authorities).

2.1.1 Platform components

Ingestion layer – Serves as the input mechanism into the platform. Different kinds of information can serve as input to the system. For example, IoT devices and additional input mechanisms, such as dedicated applications, can produce different kinds of data such as measurements (time series), pictures, videos, audio, and more. An additional source of incoming information is weather related data. Typically, once a new piece of data has been ingested into the platform, it is stored temporarily in a raw storage system, and a proper notification is sent via the service bus to the interested parties.

Internal services – These services are used internally for the proper functioning of the capabilities provided by the various components. These services are used mostly for data storage and communication. Some generic data analytics and processing services may be used as well. Examples of this layer include a central (raw) data repository, a central message bus, and a generic knowledge base. These services will be accessible to all platform components.

Business layer – This layer encompasses the components that perform the actual platform specific capabilities. The bulk of the platform is concentrated at this layer, which interacts in turn with all additional layers.

A particular group of components in this layer tackles the analysis of different kinds of data flowing into the platform. A main aspect of the components in this category is extraction of

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 700475

semantic data flowing into the platform from various different sources. Among this group we can find the following components:

- Social Media Analysis Services
- Image analysis
- Video analysis
- Multimedia analysis
- Automatic speech recognition
- Sensor analysis

These components help to determine the crisis classification and drive the detection of events which leads in turn to meaningful decision support.

A second group of components deal more specifically with the analysis of weather-related data. Among these components are:

- Climate Emergency Modeling and Prediction Services
- Weather Forecast Services
- Flood Prediction Services

Both sub-categories listed above contribute to the proper warning generation capabilities of the platform, including measures of crisis classification.

A third group of related components deal mostly with text, both at the input and output aspects. Namely, incoming text messages, possibly in a variety of languages, are processed and analyzed, and outgoing text messages that need to be delivered by the platform are prepared:

- Multilingual Report Generator
- Multilingual Text Analyzer

External layer – This layer handles the interaction of the platform with external entities, both as input providers and output recipients. There are two main groups of components making up this layer, namely:

- Mobile applications
    - Civilians Mobile Application
    - First Responder Mobile Application
- Control centers
    - Public Safety Answering Point (PSAP)
    - Valencia Local Police Control Center

The **REST architectural pattern** will be used to model the services used by the different services. REST proposes a very lightweight, HTTP-based method of stateless operations between resources in the Web.

2.2 Technology stack

In order to keep the architecture coherent and manageable, and simplify hosting and maintenance, a controlled set of technologies will be used to implement the shared components of the platform.

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 700475

As a general principle, open source technologies will be used to implement the platform ("open source" defined as software licensed under an OSI[1]-approved license that is included in the list of EUPL-compatible licenses[2]). Exceptions will be proprietary or non EUPL-compatible licenses for software belonging to a partner in the beAWARE consortium, or specific products, where no EUPL-compatible alternatives exist.

2.2.1 Programming languages

A preliminary list of the selected programming languages for module implementation is provided in Table 1.

Language/framework	Versions allowed	Notes
Python	2.7	For component development
Java	JSE7, JSE8, JEE7	For component development
C/C++	ANSI-C 99 / ISO C++	For component development
R	3.2	For component development
JavaScript	Any	Node.js 0.10.x

Table 1: Programming languages

2.2.2 Databases/Repositories

A preliminary list of selected database/storage solutions is provided in Table 2.

Data store	Versions allowed	Notes
GraphDB	8.1	RDF repository – access via SPARQL and HTTP
MongoDB	3.4.2	Social media and multimedia storage
PostgreSQL	9.x	Relational DB (Time-series sensor data and metadata)
ObjectStorage	-	IBM® Object Storage for Bluemix (raw data store)

Table 2: Database and repository packages

2.2.3 Other technologies

A preliminary list of miscellaneous technological packages (application servers, special purpose stacks, etc.) is provided in Table 3.

Product	Versions allowed	Notes
Kafka	0.8.x	Messaging service

[1] Open Source Initiative, see http://opensource.org/.

[2] See https://joinup.ec.europa.eu/software/page/eupl/eupl-compatible-open-source-licences for complete list of EUPL-compatible open source licences.

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 700475

(MessageHub)		
Jetty	9.x	J2EE application server (preferred)
Tomcat	7.x	J2EE application server
Node.js	0.10+	JavaScript application server
Apache Jena	3.x	Semantic Web framework for Java

Table 3: Other technologies

2.2.4 Exchange formats

The preferred exchange format for structured data will be **JSON**[3]. **UTF-8** encoding will be used in order to ensure correct and consistent handling of multilingual content.

For semantic data representation and serialisation, the **OWL2/XML** format[4] will be used. **JSON-LD**[5] will be used where appropriate to represent Linked Data.

Finally, for annotations of multimedia content, the **MPEG-7** standard[6] will be used.

[3] http://json.org/

[4] https://www.w3.org/TR/owl2-xml-serialization/

[5] http://json-ld.org/

[6] http://mpeg.chiariglione.org/standards/mpeg-7

This project has received funding from the European Union’s Horizon 2020 research and innovation program under grant agreement No 700475

3 Functional Description of Modules

3.1 Early warning generation (WP3)

WP3 addresses the provision of the necessary technological solutions that will allow the beAWARE framework to provide early warning and decision support to the PSAP. This is accomplished through: i) the crisis classification module, ii) the module that deals with concept and conceptual relation extraction from textual information, iii) the multimedia concept/event detection module, and iv) the automatic speech recognition (ASR) module.

3.1.1 Crisis classification module

The beAWARE crisis classification module deals with the identification and classification of crisis events, based on 1) weather forecasting data (provided by FMI), 2) data from heterogeneous sources (e.g. photos, written text, social media, etc.) and 3) rules that will be defined with the help and guidance of user experts. The module serves as a two-layered crisis classification system (first layer: early warning of upcoming crisis events, second layer: real-time monitoring of an identified crisis event's severity level).

Input	• Weather forecasting data (early warning layer) • Outputs from WP3-WP5 modules (monitoring layer)
Output	• Identification of upcoming crisis event (early warning layer) • Level of severity for identified crisis event (monitoring layer)
Programming languages	Java, R
Dependencies	• WP2 weather forecasting data • WP2 user requirements • Output from WP3-WP5 modules
Critical Factors	• Difficulties in utilizing the outputs of some WP3-WP5 modules in the monitoring layer of the module • Insufficient integration of the user partners' domain expertise into the monitoring layer

Table 4: Crisis classification module summary

3.1.2 Concept and conceptual relation extraction from textual information module

The concept and conceptual relation extraction components implement the multilingual text analysis functionalities of beAWARE, enabling to process textual inputs in the targeted languages and abstract them into structured representations that faithfully capture their meaning, and can be subsequently reasoned upon for intelligent decision making in WP4.

Concept extraction component

The concept extraction component tackles the recognition of named and nominal entities (objects, locations, etc.) and events from the pertinent to beAWARE textual inputs (i.e. social media posts, SMS messages, transcribed language, etc.), their disambiguation, and their mapping to existing structured lexical and knowledge resources such as BabelNet[7] and

[7] http://babelnet.org/

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 700475

DBpedia[8]. It serves as the first step of the beAWARE semantic, multilingual text analysis. It involves several steps, including the normalisation of the textual inputs.

Input	Raw text, plus pertinent metadata (e.g. geolocation)
Output	RDF representations, fed to the KB via UPDATE queries
Programming languages/tools	Java, UIMA/DKPro software architecture, Jena API
Dependencies	• Components affording (via the BUS) textual inputs, namely: social media monitoring, automatic speech recognition, and mobile application backend • Semantic representation and reasoning: the extracted RDF representations are used to populate the KB
Critical Factors	Poor context due to brevity and nature of input texts

Table 5: Concept extraction component summary

Relation extraction component

The relation extraction component deals with the identification of semantic frames, i.e. relational contexts that denote events and situations between entities (frame participants) and the semantic roles (frame roles) of the participating entities. It serves as the second step of the beAWARE semantic text analysis.

Input	Raw text plus annotations extracted by the concept extraction module
Output	RDF representations, fed to the KB via UPDATE queries
Programming languages/tools	Java, UIMA/DKPro software architecture, Mate tools, Jena API
Dependencies	• Concept extraction performance • Semantic representation and reasoning: the extracted RDF representations are used to populate the KB
Critical Factors	• Poor context due to brevity and nature of input texts • Incomplete and/or partially ungrammatical inputs

Table 6: Relation extraction component summary

3.1.3 Concept and event detection from multimedia module

This module consists of several components that deal with several aspects of concept and event detection. More specifically, the concept and event detection from multimedia module comprises the following components: (a) The dynamic texture recognition (DTr), (b) the object detection (ObjD) in video frames and images, (c) the traffic level classification (TLc) and (d) the Spatio-Temporal localization (STL) of fire, smoke and flood incidents. More information for each component is provided in the following subsections.

[8] http://wiki.dbpedia.org/

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 700475

Dynamic texture recognition (DTr) component

The dynamic texture recognition component is responsible for analysing video samples and determining whether they contain a fire, smoke, flood or traffic incident. It is used as a prerequisite before running object detection and traffic classification components.

Input	Video file (.avi, .mp4) from static or mobile camera.
Output	**.xml** with the prediction score of the classification task
Programming languages/tools	C/C++, Python, openCV, vlfeat
Dependencies	• WP2 user requirements • WP4 modules that will provide the input videos from tweets • WP4 modules (KB service) that will retrieve information from the DTr
Critical Factors	It is essential that the input files have been acquired from video samples that do not contain a lot of camera motion, and good lighting condition.

Table 7: Dynamic texture recognition component summary

Object detection (ObjD) component

Object detection will take place on video samples that have been indicated by the dynamic texture component to include a fire, smoke, flood or traffic incidents. Object detection will also take place on images that the text retrieval module which analyses tweeter posts has indicated that they contain fire, smoke of flood incident.

Input	Video (.avi, .mp4) from static of mobile camera or image file (.png, .jpg) from mobile camera
Output	**.xml** with the prediction scores for each object that exists inside the image frame and its bounding box (i.e. where this object is located inside the image)
Programming languages/tools	C/C++, python, openCV, vlfeat, caffe
Dependencies	• WP2 user requirements • DTr component • WP4 modules (KB service) that will provide the input images and videos from tweets • WP4 modules that will retrieve information from the ObjD
Critical Factors	The objects that we expect to identify should not be occluded more than 50% by water, smoke or flood regions.

Table 8: Object detection in video frames and images component summary

Traffic level classification (TLc) component

The traffic level classification component will be responsible for classifying the traffic incidents that have been detected by the dynamic texture component. Traffic level incidents could be classified as light, medium or high density ones. The algorithm could work twofold so that it can classify both images and video input files. On the one hand, the component will count the number of cars by using the object detection component to classify image files,

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 700475

while on the other hand it will use dynamic texture representation features to clarify the motion pattern that the cars exhibit and distinguish the traffic level in videos.

Input	Video (.avi, .mp4) or image file (.png, .jpg) from static surveillance camera or Close Circuit TV (CCTV)
Output	**.xml** with the traffic level of an image or video file
Programming languages/tools	C/C++, python, openCV, vlfeat, caffe
Dependencies	• WP2 user requirements • DTr and ObjD components • WP4 modules (KB service) that will retrieve information from the TLc
Critical Factors	We would need long duration videos to classify appropriately the traffic level classification model.

Table 9: Traffic level classification component summary

Spatio-temporal localization (STL) of fire, smoke and flood incidents component

The spatio-temporal localization component will analyse video files so as to identify the start and end boundary detection frames where a critical event occurs (dynamic texture temporal detection). Afterwards, spatial localization will take place to identify this incident specifically in each frame (dynamic texture frame-based spatial localization). This component will produce a more detailed analysis of where a critical event occurs inside a long hour video sample.

Input	Video (.avi, .mp4) from static or mobile camera
Output	**.xml** with the boundary timestamps of a critical event (start and end frame) and corresponding bounding boxes for each video frame inside
Programming languages/tools	C/C++, python, openCV, vlfeat, caffe
Dependencies	• WP2 user requirements • WP3 component: DTr • WP4 modules (KB service) that will retrieve information from the STL
Critical Factors	This module will work only for long duration videos with critical events

Table 10: Spatio-temporal localization of fire, smoke and flood incidents component summary

3.1.4 Automatic Speech Recognition (ASR) module

The automatic speech recognition module provides a channel for analysis of spoken language in audio and video files. The module considers both audio extraction and audio transcription functionalities. The former deals with the extraction of audio, in the case that the input to the module is a video file. The latter deals with the conversion of the audio input into written texts. The module is able to accept audio and video input in various

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 700475

formats and encodings. Within beAWARE, ASR is performed for three languages regarding the pilot use cases, namely Italian, Spanish and Greek. ASR is also performed for English (for presentation/demonstration purposes).

Input	Audio or video file, language parameter (English, Italian, Spanish or Greek)
Output	The output is retrieved asynchronously: • In the first step a file-ID • In the second step, either the recognised text as a plain UTF-8 text or the CTM file with time steps for each recognised word
Programming languages	C++, Python
Dependencies	• In order to be able to adapt the ASR for the use case domains, there will be a need to collect (monolingual) corpora for the defined domains. Thus, the ASR quality will depend on the quality and the use-case closeness of the data which will be used for the training.
Critical Factors	• The recognition quality might be low for the selected domains, if the training data differs from the testing data.

Table 11: Automatic speech recognition module summary

3.1.5 Activity table and workload

Activity	**Subactivity**	**Description**	**Dependencies**
Crisis classification	Early warning	Identification and early warning of upcoming crisis events	WP2 weather forecasting data
	Severity level monitoring	Real-time monitoring of an identified crisis event's severity level	• WP2 user requirements • Output from WP3-WP5 modules • User partners' domain expertise
Concept and conceptual relation extraction from textual information	Concept extraction	Textual inputs normalization; identification and disambiguation of named (locations, temporal, etc.) and nominal entities (events, objects)	• Feed of textual inputs from social media monitoring, mobile application & ASR components • KB modeling and population requirements
	Relation extraction	Distill events and situations along with -centric	• Concept extraction • KB modeling

		representations that capture the participating entities and their relations	and population requirements
Concept and event detection from multimedia	Dynamic texture recognition	Classify video samples as critical or not	• WP2 user requirements • WP4 modules • KB service
	Object detection in video frames and images	Detect persons and cars within video and image files	• WP2 user requirements • Dynamic texture recognition component • WP4 modules • KB service
	Traffic level classification	Determine the traffic level of a video or image file	• WP2 user requirements • Dynamic texture recognition and object detection components • WP4 modules
	Spatio-temporal localization of fire, smoke and flood incidents	Localize the existence of critical event in a spatio-temporal manner inside video files	• WP2 user requirements • Dynamic texture recognition component • KB service
Automatic speech recognition	Baseline system	ASR trained on general-domain data	N/A
	Domain-adapted system	ASR adapted for the use-case domains and tuned by in-domain data	Collected corpora for the defined use-case domains

Table 12: WP3 activity table

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 700475

3.1.6 Timeline and dependency from other modules

ACTIVITY	Y1												Y2												Y3											
WP3	🟦	🟦	🟦	🟦	🟦	🟦	🟦	D3.2	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	D3.3	D3.1	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	D3.4	🟦	🟦
Crisis classification			🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧				
Early warning			🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩																		
Severity level monitoring									🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩				
Multilingual text analysis			🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧				
Concept and conceptual relation extraction v1			🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩																			
Concept and conceptual relation extraction v2																		🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩				
Concept and event detection from multimedia			🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧		
Dynamic texture recognition v1			🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩																			
Dynamic texture recognition v2																		🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩								
Object detection v1				🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩																			
Object detection v2																		🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩					
Traffic level classification v1									🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩																			
Traffic level classification v2																		🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩		
Spatio-temporal localization v1									🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩																		
Spatio-temporal localization v2																			🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩		
Automatic speech recognition			🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧		
Baseline system			🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩																			
Domain-adapted system																🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩		

Table 13: WP3 timeline

3.2 Aggregation and semantic integration of emergency information for decision support and early warnings generation (WP4)

WP4 addresses the collection, aggregation and semantic integration of content relevant to emergency information in order to facilitate decision support and early warnings generation. This is accomplished through: i) the social media monitoring module, ii) the module responsible for monitoring machine sourcing information from IoT and M2M platforms, and iii) the semantic representation and reasoning module.

3.2.1 Social Media Monitoring module

The aim of the social media monitoring module is to collect posts from Twitter that appear to be relevant to the three main pilots, i.e. floods, fire, and heatwave. The crawling process needs to be real-time and efficient, able to handle large streams of data, especially when keywords such as “fire” have multiple meanings. The module collects tweets in English, Greek, Italian, Spanish, and Danish, that are published by any citizen, civil protection organisation or online news websites.

In order to gain access to Twitter’s global stream of data, we have exploited the Streaming APIs[9], a streaming client that receives tweets the moment that they occur. Compared to Twitter’s REST APIs[10] , this option offers a real-time stream of tweets instead of constantly making requests and thus overrides any rate limiting, i.e. maximum number of requests. The only limitation when using the Streaming APIs is that each account is allowed to create only one standing connection.

There are various streaming endpoints that can be divided into the following categories: Public streams, User streams, and Site streams. In our case, the “POST statuses/filter” endpoint of public streams is the most suitable, since it focuses on public data flowing through Twitter that matches one or more filter predicates. Specifically, the “track” field can be used to define up to 400 search keywords, combined with an OR operator, so that the API will return tweets matching any of these keywords.

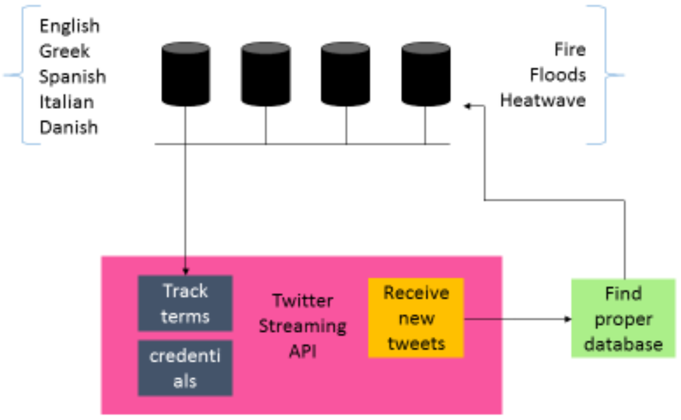


Figure 2: Allocation of Twitter posts to its proper data collection

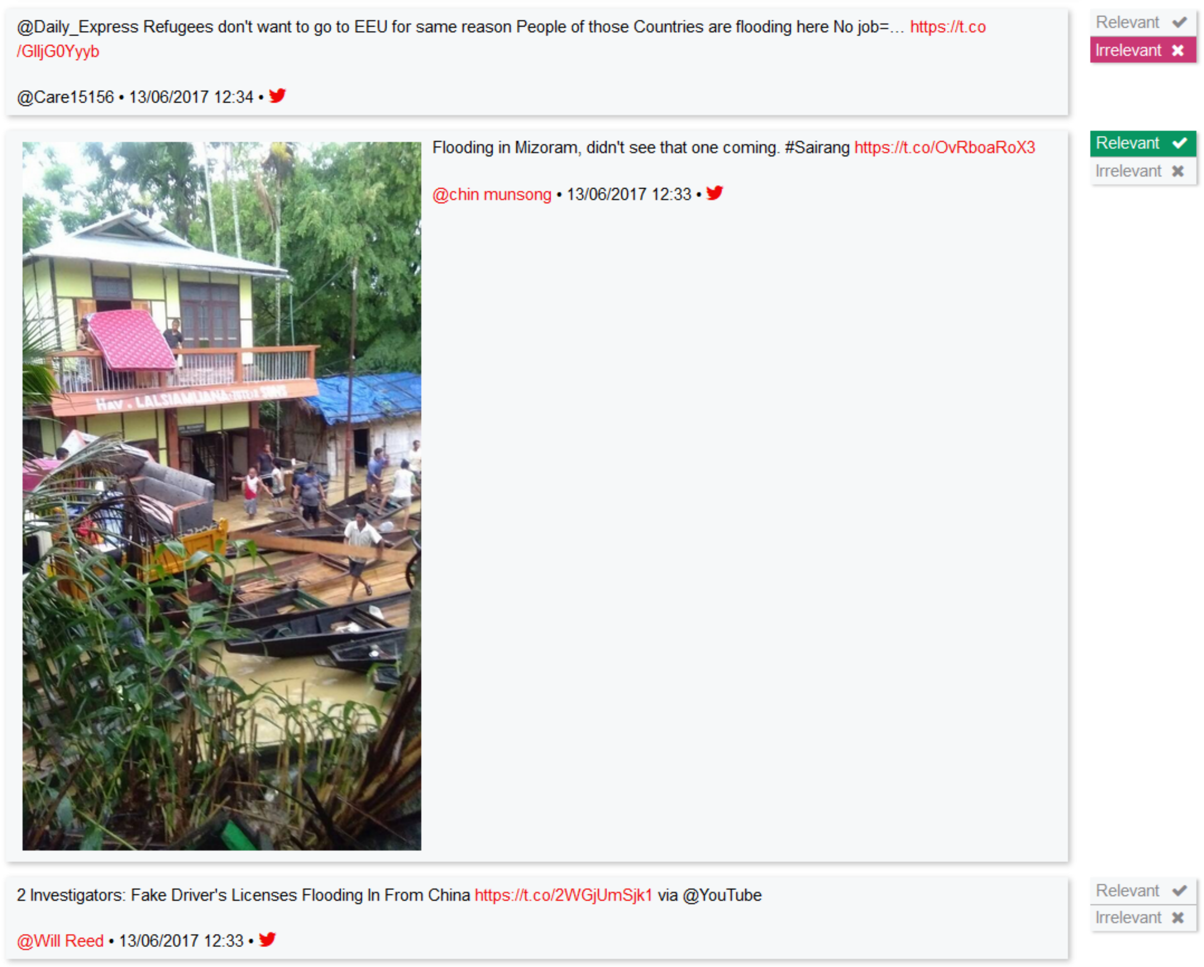
[9] https://dev.twitter.com/streaming/overview

[10] https://dev.twitter.com/rest/public

This project has received funding from the European Union’s Horizon 2020 research and innovation program under grant agreement No 700475

To consume Twitter's Streaming API, we chose to adopt the Hosebird Client (hbc)[11], an open-source and easy-to-use Java HTTP client. A required parameter is the user account's credentials, while an optional parameter is the track keywords. For each pilot and each language we sustain separate collections in a MongoDB database, not only to store the crawled tweets, but also to keep a collection of relevant keywords. By communicating with these collections we are able to define a set of words and use them as track terms. After connection with the Twitter API is established, every time a new tweet is retrieved, we examine which of the keywords exists in the tweet's text. In this way we can match the tweet to the corresponding language and pilot (e.g. "inundación" matches to Spanish and floods), and then insert it to the respective database, maintaining the structure provided by the API, since the JSON format fits well for MongoDB databases structures.

A service is created to classify all posts as relevant or irrelevant, so only the relevant ones proceed to the beAWARE analysis component, aiming mainly to extract locations and to detect key events. The training of the classification model is based on user feedback, as collected from our dedicated interface that demonstrates all social media collections, offering the annotation option as relevant/irrelevant, which is stored as a binary attribute in MongoDB.


@Daily_Express Refugees don't want to go to EEU for same reason People of those Countries are flooding here No job=... https://t.co/GlljG0Yyyb

@Care15156 • 13/06/2017 12:34 •

Flooding in Mizoram, didn't see that one coming. #Sairang https://t.co/OvRboaRoX3

@chin munsong • 13/06/2017 12:33 •

2 Investigators: Fake Driver's Licenses Flooding In From China https://t.co/2WGjUmSjk1 via @YouTube

@Will Reed • 13/06/2017 12:33 •

Figure 3: Interface for classifying posts as relevant/irrelevant

[11] https://github.com/twitter/hbc

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 700475

The classification service collects the stored annotated posts every midnight (00:00 GMT+2) to update the training set. The updated training set is used to assign an estimated relevance score to send only the relevant Twitter posts for further analysis. The classification is based on open-source code, which is also available as a library in R (RTextTools[12]), in combination with the tm[13] library.

Once a new Twitter post is collected and identified as relevant, a message is created and published to the communication bus and all subscribers can access the marked Twitter posts directly from the MongoDB database.

The overall input, output, dependencies and critical factors are listed in Table 14.

Input	Twitter API, Streaming API, relevant keywords for each language and use-case scenario
Output	Relevant Twitter posts
Programming languages	Java, R
Dependencies	Concept extraction module and mainly Task 3.2.
Critical Factors	The percentage of relevant to irrelevant Twitter posts; the total amount of collected posts

Table 14: Social media monitoring module summary

3.2.2 Monitoring machine sourcing information from IoT and M2M platforms module

Sensor data is crucial for tracking the onset and progress of a crisis. There is a large variation in sensors and an almost equally large variation in interfaces to access the data from those sensors. The aim of this module is to collect time-series sensor data from various on-line sensors and make this sensor data, and the sensor metadata, available through a standardised interface.

A modern, RESTful interface for accessing time-series sensor data and sensor metadata is The OGC SensorThings API[14]. It specifies a data model, a JSON data format for transferring the data, and a RESTful interface for accessing the data. The data model is based on the ISO/OGC Observation and Measurement (O&M) model. The data format and RESTful interface is based on the OASIS OData Version 4.0 specification. Several implementations of the SensorThings API exist, both open- and closed-source.

The sensor data will be fed into the system through data wrappers that mediate between the (often proprietary) interface of the sensor and the SensorThings API. The module will contain an extension for setting thresholds and for sending notifications through the communication bus (WP7) when thresholds are passed.

[12] https://cran.r-project.org/web/packages/RTextTools/index.html

[13] https://cran.r-project.org/web/packages/tm/index.html

[14] http://docs.opengeospatial.org/is/15-078r6/15-078r6.html

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 700475

Input	Time-series sensor data in various formats
Output	Time-series sensor data and metadata in JSON format, conforming to the OGC SensorThings API specifications
Programming languages	Java, PLSQL
Dependencies	• Modules supplying sensor data (WP2, WP7) • Modules accessing sensor data (WP3, WP6, WP7) • Communication bus (WP7)
Critical Factors	Access to on-line sensor data

Table 15: Monitoring machine sourcing information from IoT and M2M platforms module summary

3.2.3 Semantic representation and reasoning module

The semantic representation and reasoning module consists of the following components: (a) the **beAWARE Knowledge Base** (KB), also referred to as "**ontology**", (b) the **KB Service**, (c) the **semantic enrichment and reasoning framework**. More information on these components is given in the following subsections.

Semantic KB component

The semantic KB, or ontology, constitutes the core means for semantically representing the pertinent knowledge and for supporting decision-making. Based on a well-defined formalism (**OWL – Web Ontology Language**), the beAWARE ontology will encompass, amongst others, information regarding domain knowledge (types of crises, risks and impacts), multimodal input (image, video, text and speech) and contextual information, climate and environmental conditions, geolocations, aspects relevant to events and time.

Typical constructs stored in an ontology contain the classes representing the main entities in the domain, the instantiations of these classes (i.e. objects) and the relations between the entities. Specifically, for the beAWARE ontology, the stored notions and interrelationships will depend on the output of the requirements analysis activities within WP2, and most notably on D2.1, delivered in M5. The content of the ontology will be updated in later stages in the project lifetime, according to more up-to-date needs arising from the pilot deployments.

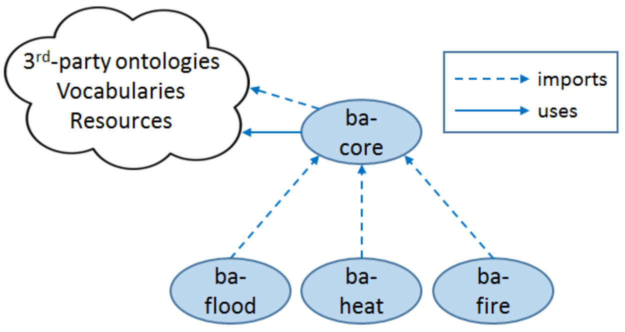


Figure 4: beAWARE ontology network

The design and development of the ontology will follow a well-known ontology engineering methodology, called "**NeOn methodology**", which is extremely suitable for collaborative ontology design and authoring (i.e. two beAWARE partners are involved in developing the ontology, CERTH and IOSB). The actual deployment of the ontology will rely on open-source established software tools: **Protégé** (v.5+) ontology editor for authoring the ontology, **GraphDB** or **WebGenesis** triple store serving as the ontology repository.

In order to conform to a more modular approach, the beAWARE ontology will consist of a set of interconnected sub-ontologies, one per class of crises (flood, fire, heatwave).

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 700475

Additionally, existing standards and ontologies will be adapted or extended according to beAWARE needs.

After the basic schema of the ontology is finalised, the input to the ontology will come (in the form of instances) from various other beAWARE components, like e.g. the component extracting concepts and conceptual relations from text (T3.2), or the component retrieving concepts and events from multimedia (T3.3). Since we are expecting massive volumes of instances to be added into the ontology in the case of a crisis event, the choice of a scalable ontology repository is of utmost importance.

Input	The KB simply stores information – input and output are handled by the KB service (see next subsection).
Output	
Ontology language	OWL 2
Dependencies	Ontology has dependencies with: • WP2 user requirements; • Knowledge extracted from WP3 and WP4 modules; • WP5 modules, which will retrieve information from the KB.
Critical Factors	• Inefficient communication with use case partners on the most relevant notions to be stored in the KB. • Scalability of the ontology repository.

Table 16: Semantic KB component summary

KB Service component

The KB Service is responsible for accessing and updating the ontology content, by providing a RESTFul API service. Calls to the API from end-users will be translated to appropriate SPARQL 1.1 queries, which will then be submitted to the GraphDB triple store maintaining the ontology. Thus, this component is responsible for the ontology's **semantic integration**, i.e. semantically integrating information that is the output from other modules into the ontology.

Other aspects that will be investigated (and potentially added to the KB Service) include: ontology population and ontology enrichment. The former process involves the addition of objects/instances into the ontology, while semantic enrichment refers to the process of augmenting the semantics of an ontology by establishing links to external sources and by appending new knowledge retrieved from those sources. It is imperative that the external third-party knowledge sources serve data in an appropriate format (i.e. as RDF Linked Data), thus we will consider knowledge sources such as DBpedia, Geonames and Freebase.

Input	Select/Update queries from other components.
Output	Response containing the result set of the 'Select' query. In the case of 'Update' queries, the response will be the number of triples added to/removed from the ontology.
Programming languages	Any (RESTful APIs are language-independent).
Dependencies	• Communication Bus (WP7); • Modules accessing the KB Service (through the Bus): WP3, WP4, WP5 modules.

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 700475

Critical Factors	• Concurrency of transactions to triple store. • Scalability of the ontology repository.

Table 17: KB service component summary

Semantic Reasoning component

This component is responsible for performing semantic reasoning on the ontology, which refers to the process of inferring implicit knowledge from the explicitly asserted facts residing in the ontology. The reasoning mechanisms will support sophisticated interpretation tasks relevant to the management of multimodal incoming information and to the retrieval of information pertinent to the users' needs.

A critical issue for beAWARE involves handling uncertain information. For this point, we will rely on fuzzy and/or non-monotonic reasoning approaches, in order to cope with incomplete and inconsistent information and to resolve conflicts and prioritize the proposed actions. Existing reasoning engines will be the baseline for the semantic reasoning activity and will be effectively extended, in order to mitigate their current limitations, like e.g. scalability issues, non-seamless integration with ontologies (in the case of uncertainty reasoning engines) and more implementation-specific limitations.

Input	A set of facts (in RDF) relevant to the query from the user.
Output	A graph of query-relevant facts extracted from the triple store. Relevant information will be encoded in the graph.
Programming languages	Java and/or Python
Dependencies	• Ontology and KB Service from WP4; • Use case needs and requirements from WP2 with respect to data types and reasoning.
Critical Factors	• The accuracy of this service is crucial to the quality of user supporting information. Flaws/errors in the reasoning and inference procedures might produce sub-optimal results. • The size of the knowledge base might be too large for the reasoner to handle efficiently. • The lack of specificity can lead to poor results with this component. • Uncertainty handling should be meaningful and not trivial.

Table 18: Semantic reasoning component summary

3.2.4 Activity table and workload

Activity	Subactivity	Description	Dependencies
Social Media	Social media crawling	Collect social media information from social media (real-time)	• Twitter Streaming API
	Classification of social media posts	Classify the collected social media posts as relevant or irrelevant	• WP2 user requirements for annotating a training set of Twitter posts
	Event detection and localisation	Detect locations in raw text from social media and cluster posts by location	• Task 3.2 (concept detection from textual information)
Sensor data	Sensor data integration	Mediating between sensor	• WP2 User requirements

		interface and the SensorThings API	and Sensor data sources
	Threshold detection	Implementing threshold detection and notification	• WP7 Communication bus
Semantic KB	Ontology specification & conceptualization	Design the ontology, using requirements extraction and conceptualization techniques (e.g. diagrams)	• WP2 user requirements • WP5 modules, which will retrieve information from the KB
	Ontology formalization & implementation	Formalize the ontology using an appropriate ontology language	• WP2 user requirements • WP5 modules, which will retrieve information from the KB
	Ontology mapping and alignment	Establish links to other vocabularies/ontologies	• Knowledge extracted from WP3 and WP4 modules
KB Service	Investigate semantic integration techniques	Examine approaches for semantic integration	N/A
	KB Service v1	Integrate in v1 operational prototype	• Communication Bus (WP7) • Modules accessing the KB Service: WP3, WP4, WP5 modules
	KB Service v2	Integrate in v2 operational prototype	
Semantic Reasoning	Investigate uncertainty & scalable reasoning	Examine relevant approaches	N/A
	Implement uncertainty reasoning v1	Integrate in v1 operational prototype	• Ontology and KB Service from WP4 • Use case needs and requirements from WP2
	Implement uncertainty reasoning v2	Integrate in v2 operational prototype	
	Integrate scalable reasoning v1	Integrate in v1 operational prototype	• Ontology and KB Service from WP4 • Use case needs and requirements from WP2
	Integrate scalable reasoning v2	Integrate in v2 operational prototype	

Table 19: WP4 activity table

This project has received funding from the European Union’s Horizon 2020 research and innovation program under grant agreement No 700475

3.2.5 Timeline and dependency from other modules

ACTIVITY	Y1												Y2												Y3											
WP4	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	D4.1	🟦	🟦	🟦	🟦	🟦	D4.2	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	D4.3	🟦	🟦
Social media monitoring	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧						
Social media crawling	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩																		
Classification of social media posts				🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩																		
Event detection and localisation									🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩						
Sensor data integration	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧				
Semantic KB (ontology)	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧																		
Ontology specification & conceptualization	🟩	🟩	🟩	🟩																																
Ontology formalization & implementation				🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩																		
Ontology mapping and alignment with other vocabularies													🟩	🟩	🟩	🟩	🟩	🟩																		
KB Service			🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧		
Investigate semantic integration techniques for multimodal input			🟩	🟩	🟩																															
Implement KB Service v1					🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩																		
Implement KB Service v2																			🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩		
Semantic Reasoning						🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧		
Investigate uncertainty & scalable reasoning						🟩	🟩	🟩	🟩																											
Implement uncertainty reasoning framework v1								🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩																		
Implement uncertainty reasoning framework v2																			🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩		
Integrate scalable reasoning techniques v1								🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩																		
Integrate scalable reasoning techniques v2																			🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩		

Table 20: WP4 timeline

3.3 Multilingual report generation (WP5)

WP5 addresses the generation of the emergency reports and messages that need to (or should) be communicated to the different types of stakeholders that beAWARE considers (authorities, first responders, citizens, etc.) during an emergency in order to provide them with tailored support. It accomplishes this through the following two modules.

3.3.1 Text planning module

The text planning module deals with the selection of content from the semantic repository (KB) and the creation of a discourse plan for the reports and messages that are to be generated. It serves as the first step in the beAWARE emergency report and message generation pipeline, and it is responsible for tailoring contents selection and their structuring to the information needs pertinent to the specific target user group and context.

Input	• RDF triples obtained via SELECT queries from the KB • Targeted user group (e.g. authorities, first responders, citizens, etc.), with the possibility for further potentially relevant information (e.g. filters) from the triggering modules (semantic reasoning and PSAP)
Output	Partially ordered sequence of the selected RDF triples
Programming languages/tools	Java, BabelNet synset annotated Wikipedia dump
Dependencies	• Modules triggering report generation, namely semantic reasoning and PSAP • Domain knowledge modelling, as it affects the selection and discourse structuring criteria • Specification of the information needs of the different user groups addressed
Critical Factors	• Scalability of content selection algorithm • Update effectively with respect to what has been communicated previously • The richer the semantic domain model, the more changes for coherent content selection and discourse planning

Table 21: Text planning module summary

3.3.2 Multilingual linguistic generation module

The multilingual linguistic generation module deals with the verbalization of emergency reports in the language of end user groups targeted in the beAWARE pilots. It takes as input the abstract (conceptual) representations produced by the text planning module and successively produces all the layers foreseen by the Meaning-Text Theory through a pipeline of graph transducers and classifiers.

Input	RDF triples as selected by the text planning module
Output	Natural language reports and messages to be shown to the involved stakeholders (authorities, first responders, citizens, etc.)

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 700475

Programming languages/tools	Java, Jena API, Mate tools
Dependencies	• Text planning module performance • User partner's specifications about the desired/appropriate language style for the different types of generated reports/messages in the considered pilot contexts
Critical Factors	Flexible projection of RDF constructs to predicate-argument structures that are suitable for linguistic generation

Table 22: Multilingual linguistic generation module summary

3.3.3 Activity table and workload

Activity	Subactivity	Description	Dependencies
Text planning	Content selection	Select relevant contents form the KB, upon pertinent system/user request	• Semantic reasoning/PSAP triggers for content selection • Expressivity of reference domain ontology • User partner delineated criteria for assessing relevance/importance of the contents to be included
	Discourse structure	Arrange selected contents in a coherent way	• Content selection performance • Expressivity of reference domain ontology
Multilingual linguistic generation	Mapping from conceptual representations to linguistic semantic structures	Projection of ontological representations to linguistic semantic structures	• Text planning completeness • Underlying semantics of reference domain ontology
	Mapping from linguistic semantic representations to surface-syntactic ones	Projection of linguistic semantic structures to syntactic ones	N/A
	Linearisation & morphological realisation	Projection of linguistic syntactic structures to natural language text	N/A

Table 23: WP5 activity table

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 700475

3.3.4 Timeline and dependency from other modules

ACTIVITY	Y1												Y2												Y3											
WP5	🟦	🟦	🟦	🟦	🟦	🟦	🟦	D5.1	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	D5.2	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	D5.3	🟦	🟦
Emergency report generation specifications and requirements	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧																												
Text planning				🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧		
Content selection & discourse planning v1				🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩																		
Content selection & discourse planning v2																			🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩		
Multilingual linguistic generation					🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧		
Projection of conceptual reprensetations to NL text v1					🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩																		
Projection of conceptual representation to NL text v2																			🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩		

Table 24: WP5 timeline

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 700475

3.4 Main Public Safety Answering Point for emergency multimedia enriched calls (WP6)

The objectives of WP6 are to develop a unified platform deployed in public safety answering points (PSAP) for efficient emergency incidents management through the provision of a common operational picture and improved situational awareness, based on 4 main pillars:

1. Collecting and correlating multimodal events coming from disparate data sources;
2. Video management;
3. Geographical Information System (GIS) analysis;
4. Incident and resource management.

One of the main objectives of this WP is the design and development of an advanced presentation layer based on cutting edge GIS, analytic and visualization techniques. This will include research on novel interaction techniques and investigation of new presentation concepts in data analysis.

PSAP has a three-tier technological architecture as shown in Figure 5:

- **Front End** – Desktop and Web Clients, endpoint security.
- **Back End** – Desktop and Web Services, Inbound/Outbound Communication Services, Server Security (including Client Authentication Services).
- **Infrastructure** – ArcGIS Geographical Information Services, IBM messaging services.

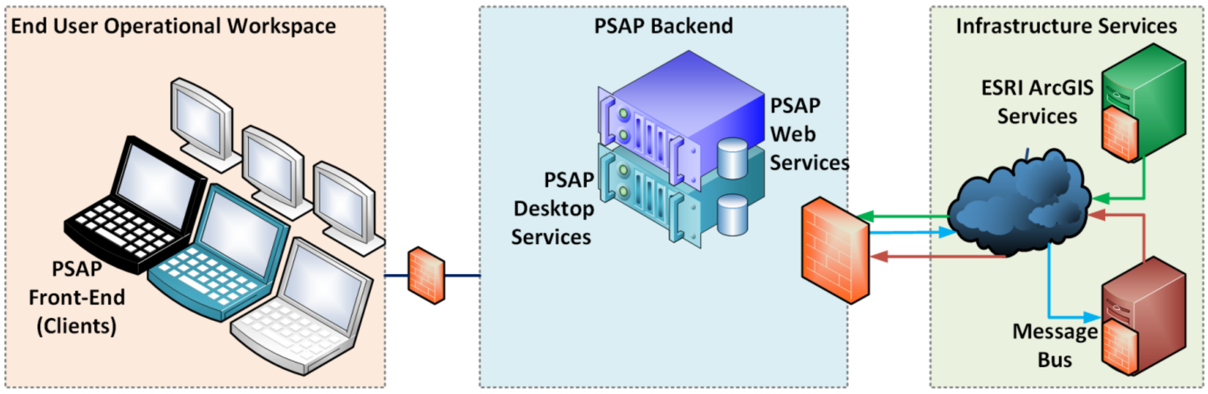


Figure 5: PSAP Technological Architecture

3.4.1 PSAP capabilities

The following capabilities are planned as part of the PSAP functionality:

Pre-Emergency Decision-Supporting Risk Visualization

PSAP shall display pre-emergency risk assessment and crisis classification to the decision makers.

Input	Risk assessment, Crisis Classification, Alerts, Position-based metrics, Analyzed video and images, analyzed sensor reading reports	Sources: Flood Risk Analysis Services, Crisis Classification Services, Video Analytics, Social Media Analytics, Sensor Analytics

Output	Risk display, decision supporting information display, geo-based information layer display Video playback, Image display Risk indicator and metric charts	Recipients: Decision Makers, Emergency Managers, Emergency Analyst
Programming languages/tools	N/A	
Components	DSS, Front-End, Back-End Services, GIS	
Dependencies	Flood Risk Analysis Services, Crisis Classification Services, Video Analytics, Social Media Analytics, Sensor Analytics, Message Bus	
Critical Factors	Flood Risk Analysis Services, Crisis Classification Services, Outcomes of D6.1	

Table 25: Pre-Emergency Decision-Supporting Risk Visualization summary

Emergency Incident Collection and Display

PSAP shall collect and display emergency incident information to the emergency control center operators.

Input	Incident	Sources: Video Analytics, Social Media Analytics, General public, First responders, Local Police / Calltaking service
Output	Geo-based incident display Incident record display	Recipients: Emergency Managers, Control Center Operators
Programming languages/tools	N/A	
Components	Front-End, Back-End, GIS	
Dependencies	Video Analytics, Social Media Analytics, First Responder (FR) Mobile App, Public Mobile App, Calltaking service, External Incident Data Provider, Message Bus	
Critical Factors	N/A	

Table 26: Emergency Incident Collection and Display summary

First Responder Position Collection and Display

PSAP shall collect and display first responder team positions and status information to the emergency control center operators.

Input	First Responder Position	Sources: FR Mobile App
Output	Geo-based workforce position display Workforce information display	Recipients: Emergency Managers, Control Center Operators
Programming languages/tools	N/A	
Components	Front-End, Back-End, GIS	

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 700475

Dependencies	FR Mobile App, Message Bus
Critical Factors	FR Mobile App position sending

Table 27: First Responder Position Collection and Display summary

Incident Assignment to First Responder

PSAP shall allow assigning incidents to first responder teams and incident information distribution to the first responders by the emergency control center operators.

Input	Incident assignment to workforce	Sources: Control Center Operator
Output	Incident information (position and details)	Recipients: First responders (with FR Mobile App)
Programming languages/tools	N/A	
Components	Front-End, Back-End	
Dependencies	FR Mobile App, Message Bus	
Critical Factors	FR Mobile App Task Reception Interface	

Table 28: Incident Assignment to First Responder summary

Public Alert

PSAP shall allow sending public alerts regarding incidents and general risk assessments to citizens who have the Public App on their mobile device.

Input	Public Alert, incident cue	Sources: Control Center Operator
Output	Public Alert Message	Recipients: Citizen (with Public Mobile App)
Programming languages/tools	N/A	
Components	Front-End, Back-End	
Dependencies	Public Mobile App, Message Bus, Alert Transcripts	
Critical Factors	N/A	

Table 29: Public Alert summary

Incident Management

PSAP shall receive and display incident information, status, and footage from first responders in the field or bystanders (citizens) who will provide the information and footage from their mobile device and FR Mobile app.

Input	Incident information, Incident Footage (video, stills)	Sources: FR Mobile App, Public Mobile App

Output	Incident information display Incident lifecycle management On-map incident information display Video playback, Image display	Recipients: Control Center Operator
Programming languages/tools	N/A	
Components	Front-End, Back-End, GIS	
Dependencies	Public Mobile App, FR Mobile App, Message Bus	
Critical Factors	N/A	

Table 30: Incident Management summary

3.4.2 Activity table and workload

Activity	Subactivity	Description	Dependencies
Main Public Safety Answering Point for emergency multimedia enriched calls	Pre-Emergency Decision-Supporting Risk Visualization	Display pre-emergency risk assessment and crisis classification to the decision makers	Flood Risk Analysis Services, Crisis Classification Services, Video Analytics, Social Media Analytics, Sensor Analytics, Message Bus
	Emergency Incident Collection and Display	Collect and display emergency incident information to the emergency control center operators	Video Analytics, Social Media Analytics, FR Mobile App, Public Mobile App, Calltaking service, External Incident Data Provider, Message Bus
	First Responder Position Collection and Display	Collect and display first responder team positions and status information to the emergency control center operators	FR Mobile App, Message Bus
	Incident Assignment to First Responder	Assign incidents to first responder teams and incident information distribution to the first responders by the emergency control center operators	FR Mobile App, Message Bus
	Public Alert	Send public alerts regarding incidents and general risk assessments to citizens who have the Public Mobile App on their mobile device	Public Mobile App, Message Bus, Alert Transcripts
	Incident Management	Receive and display incident information, status and footage from first responders in the field or bystanders (citizen) who will provide the information and footage from their mobile device	Public Mobile App, FR Mobile App, Message Bus

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 700475

		and FR Mobile app	

Table 31: WP6 activity table

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 700475

3.4.3 Timeline and dependency from other modules

ACTIVITY	Y1												Y2												Y3											
WP6	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	D6.1	🟦	🟦	D6.2	🟦	🟦	D6.3	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	D6.4	🟦	🟦
Pre-emergency decision-supporting risk visualization	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧		
Research and specification	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩																								
Design and development													🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩						
Deployment, integration, demonstration																			🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩		
Emergency incident collection and display	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧		
Design and development	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩			
Deployment, integration, demonstration																🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩		
First responder position collection and display	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧		
Design and development	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩			
Deployment, integration, demonstration																🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩		
Incident Assignment to First Responder														🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧		
Design and development														🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩				
Deployment, integration, demonstration																					🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩		
Public Alert							🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧		
Design and development							🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩			
Deployment, integration, demonstration																			🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩		
Incident Management													🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧		
Design and development													🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩			
Deployment, integration, demonstration																					🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩		

Table 32: WP6 timeline

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 700475

3.5 System development, integration and evaluation (WP7)

Due to the distributed approach of the design and implementation of the beAWARE platform, in which different partners maintain ownership over different components of the system, combined with the complexity of the platform, which is comprised of many different components we opted to follow a micro-services approach to make the entire process of design, implementation, and integration more manageable, in a distributed manner

Using this approach, we maintain the independence of the different components, and the integration focus is on the exposed capabilities of each component. The integration between components is being realized via two main mechanisms, first, inter-communication via the platform communication service bus, and second through exposed REST interfaces of various components, potentially aided by a discovery service.

Micro-services is an architecture form and methodology for breaking up a large and complex system into small units, each fulfilling a unique well-defined task, in an independent manner. Each such micro-service is deployed separately and can interact with its peer micro-services using standard communication protocols. Such an approach eases the task of long term maintenance and evolution of a large system, compared to a monolithic system. In addition, internal changes within a micro-service do not influence any other system component as long as the external contract of the micro-service is adhered to. Moreover, each component can be implemented in a different programming language, using different middleware.

As mentioned earlier, there are two main mechanisms for micro-services to interact, namely, the communication service bus and a REST interface (possibly with the help of a discovery service). For communicating through the communication bus the components need only to agree on the topic via which they will be interacting and the format of the messages published on that topic. For communicating via a REST interface, there needs to be a way for one micro-service to find another micro-service it wishes to invoke. For that purpose, a service discovery component may be deployed. Such a component serves as a central registry of available micro-services, responding to queries about the location of a certain micro-service. In actual deployments, this capability can be provided by packages such as Netflix Eureka, amalgam8, or Consul.

Additional helping capabilities that may be employed in our environment consists of a gateway, health monitoring, and logging (ELK stack).

3.5.1 Twelve-Factor Applications[15]

The platform will be composed of multiple micro-services and aspires to seamlessly deliver services to its users, thus we are encouraged to use the following guidelines:

1. Codebase

Maintain a one-to-one relationship between codebase and the application (one codebase multiple deploys).

[15] https://12factor.net/

This project has received funding from the European Union’s Horizon 2020 research and innovation program under grant agreement No 700475

2. Dependencies

They should not rely on system-wide packages or functions. Dependencies must be declared inside the scope of the application

3. Configuration

Separation between configuration and code, as configuration varies across deployments.

4. Back-end Services

A micro-service should make no distinction between local and third party services.

5. Build, release, run

Separate between build, release, and run stages.

6. Processes

Stateless. Any persistent data is stored using external services.

7. Port binding

Application should be standalone, rather than relying on a running instance of an application server. Services are provided to externals via a certain port bound to the application.

8. Concurrency

Should easily be scaled horizontally and having the workload distributed amongst multiple instances of the same micro-service.

9. Disposability

Support failures by easily being started and stopped.

10. Development / production parity

Designed for continuous deployment. Therefore, the difference between development and production environments should always be minimal.

11. Logs

Provide a continuous stream of logging information, which is retrieved by a separately configured service.

12. Admin processes

Run admin/management tasks as one-off processes, on an environment similar to production.

3.5.2 System architecture

This will involve the definition of the global architecture for the beAWARE platform, from its high-level component design to the server layout definition. A roadmap will also be defined, to drive the process of implementation.

Dependencies	User requirements, specification of pilot use cases and validation plan: the Use Cases will drive the processes and ultimately the architecture will be built to service those use cases, so a clear definition is essential.

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 700475

Critical Factors	• Incomplete Use Case definition: UCs must be clearly defined and well understood to ensure the appropriate systems, processes and capabilities are put in place in the architecture. UCs must include comprehensive scenarios and validation plans to ensure the architecture is built correctly to specifications. • Incomplete service definition: each of the technical development WP is responsible for defining the service API for components related to its research. Failure to provide the API specification in a timely and thorough manner can lead to delays or errors in definition of the architecture.

Table 33: Architecture definition summary

3.5.3 Technical infrastructure

This implies the provisioning and operation of the infrastructure, on which the beAWARE system will run. This includes the internal services, business services, associated repositories and external entities (mobile applications, control centers).

It easily follows that some subsystems of the beAWARE platform cannot be deployed to the platform, due to licensing reasons. Others may be dependent on secondary systems that cannot be easily replicated in a different environment, or may require concentrated computing power that is provided by a highly specialized expert system in a partner's data centre. In such cases, proxy services will be built by partners and exposed to the rest of the platform. Proxy services will delegate work to the actual services and do any extra work as required in a transparent way.

Dependencies	System architecture: technical infrastructure will be provisioned to implement the system architecture.
Critical Factors	• Distributed infrastructure: parts of the infrastructure are foreseen to be distributed and include systems which are run in-house by some partners. Issues with firewalls, authentication, etc. are to be expected, and acted upon as needed. • Performance: the scope and complexity of the beAWARE processes and the size of the data to be handled is not yet known and is likely to evolve as research progresses. Performance problems may arise, and resizing and/or provisioning of extra capacity may be required.

Table 34: Infrastructure definition summary

3.5.4 System development

This is the task for all development activities related to the technical implementation of the Use Cases and any other work required for the completion of the objectives.

The system development will be iterative, from an initial dummy prototype to the final version, with the following cycle planned:

Operational Prototype

Dummy end-to-end architecture; all service dummies in place, operating on test/mock data.

First Prototype

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 700475

Using real services from WP3-WP6; First Prototype milestone (MS3).

Second Prototype

Using real services from WP3-WP6; Second Prototype milestone (MS4).

Final System

Complete beAWARE platform; Final System milestone (MS5).

Dependencies	• All technical WP services • WP2 Use Cases and Evaluation
Critical Factors	• Changing requirements: Use Cases should remain stable during the project. While minor changes are always to be expected, major breaking changes may cause throwing away work that has already been done and impact on delivery capabilities.

Table 35: System development summary

3.5.5 Communication bus

The communication bus serves as a central point of communication between different system components. Its main mode of operation is that of publish / subscribe, which supports different parts of a composite application to be unaware of each other but still manage to communicate upon need. The only items that need to be agreed upon between a publisher and a subscriber is the name of the topic (channel) through which they will be communicating and the format of the messages flowing through that topic.

The proposed bus is in charge of notifying interested and registered components when new items which are of interest to them have been received or calculated by another component. In addition, the bus may perform some light transformations on incoming information, upon need.

The communication bus will be configured, upon deployment, with the necessary set of topics as agreed upon between the different components. In addition, the message structure of each message in each topic will be agreed upon and documented by the cooperating components. The communication bus needs to support the number of different topics required for a beAWARE installation, along with the associated aggregated throughput in all topics.

The communication bus is realized by using an instance of a MessageHub service, deployed in IBM's BlueMix cloud. The back-end is based on a Kafka cluster, and the interaction with the service is realized using standard Kafka clients.

Dependencies	• Kafka – a popular open source messaging platform • MessageHub – a cloud deployment based on Kafka
Critical Factors	• Needs to support the number of topics required along with the aggregated throughput. Requires cooperating components to agree upon messages format and abide by them.

Table 36: Communication bus summary

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 700475

3.5.6 End-users applications

There are two end-user applications for mobile devices (smartphones or tablets): one used by the first responders and one used by the public. These mobile applications will communicate with a backend that will connect to the rest of the beAWARE system.

With these applications, the first responders can communicate with the PSAP, send information about the emergency event, receive tasks and report on those assigned tasks. The public will be able to receive information about emergency events and send information about the event to the PSAP using the communication bus. Both first responders and the public will be able to send text, image, video and vocal information.

Dependencies	• Communication bus • Mobile device hardware (camera, voice, gps, gsm)
Critical Factors	• Bad reception on the mobile network • Bad GPS reception

Table 37: End-users applications summary

3.5.7 Activity table and workload

Activity	Subactivity	Description	Dependencies
System architecture	Technological roadmap	Current document	N/A
	System requirements and architecture	Description of technical requirements derived from Use Cases and detailed design of platform architecture	WP2 user requirements
Technical infrastructure	N/A	Provisioning and operation of the infrastructure, on which the beAWARE system will run	System architecture
System development	Operational Prototype	Initial dummy prototype	WPs 3-6
	First Prototype	First iteration of beAWARE platform	WPs 3-6
	Second Prototype	Second iteration of beAWARE platform	WPs 3-6
	Final system	Final version of beAWARE platform	WPs 3-6
Communication bus	N/A	Central point of communication between different system	• Kafka • MessageHub

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 700475

		components	
End-users applications	N/A	Mobile application used by first responders and the general public	• WP2 user requirements • Integrated system backend (T7.2) • Communication infrastructure (T7.4)

Table 38: WP7 activity table

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 700475

3.5.8 Timeline and dependencies from other modules

ACTIVITY	Y1												Y2												Y3											
WP7	🟦	🟦	🟦	🟦	🟦	D7.1	🟦	🟦	🟦	D7.2	🟦	D7.3	🟦	🟦	🟦	🟦	🟦	D7.4	🟦	🟦	🟦	🟦	🟦	D7.5	🟦	D7.6	🟦	🟦	🟦	🟦	🟦	🟦	D7.7	D7.8	🟦	D7.9
System architecture	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧																								
Technological roadmap	🟩	🟩	🟩	🟩	🟩	🟩																														
System requirements and architecture					🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩																								
Technical infrastructure	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧
System development					🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧
Operational Prototype					🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩																								
First Prototype													🟩	🟩	🟩	🟩	🟩	🟩																		
Second Prototype																			🟩	🟩	🟩	🟩	🟩	🟩												
Final system																									🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
Communication bus	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧																
End-users applications						🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧			

Table 39: WP7 timeline

This project has received funding from the European Union’s Horizon 2020 research and innovation program under grant agreement No 700475

4 Development Cycle and Use Cases

The overall strategy for the development of the beAWARE platform is to start with simple scenarios and proceed stepwise towards the final system. The detailed work plans of the individual modules presented in Section 3 will follow the general cycle: after the completion of each prototype version (timing defined by Milestones), the functioning of the prototype is assessed, the needs for further development are identified and agreed upon, a detailed work plan for the next prototype version is finalized and the development cycle towards the next prototype is initiated. One important predefined concept for the different prototypes is the definition of the practical use cases each prototype is serving.

4.1 Scenario 1: Flood

Target group:

In each development cycle different user groups (target groups) will be involved in pilot-testing in a real-world environment and with real-world data:

- AAWA personnel
- Decision makers involved during the flood emergencies: in the C.O.C (the Municipal operative centre):
  - The Mayor
  - The head of the Municipal group of Vicenza Civil Protection
  - The head of the National Association of Carabinieri
  - The head of the National Alpine Trooper Association
  - The coordinator of Voluntary Associations of Civil Protection, Province of Vicenza
  - The head of the Local Italian Red Cross
  - The head of the Vicenza Fire Brigades
  - The head of the Vicenza Local Police
- Citizens
- First responders
- Research community
- SMEs
- Flood Risk Management authority at national and international scale

Objectives: beAWARE should:

- Support the collection and visualization of spatially distributed and multi-sourced information related to a flood emergency (flood reports, weather forecasts, flood early warning system results, elements at risk, extension of flooded areas, crisis classification based on all available data).

- Enable the automatic detection of flood hazardous situations, such as river level overtopping/breaking.
- Provide authorities with the ability to manage first responder assignments.
- Facilitate the communication between authorities and citizens, also in different languages. More specifically, beAWARE should:
    - Provide authorities with the ability to warn people with messages, once they are approaching a flooded area.
    - Allow citizens to send flood reports with text, images, audio and video.

4.2 Scenario 2: Fires

Target group:

The target groups listed below are those who will use and interact with the beAWARE platform in the phase of preparedness and response during a forest fire emergency:

Users Involved in Fire Use cases

- Local entities
    - Civil Protection- Valencia City Council: Volunteers
    - Local Police- Valencia City Council
    - Local Firefighters: Professional body
    - Devesa- Albufera Service
- Local authorities
    - Mayor
    - City Councilors

Provincial End-User Involved in Fire Use cases

- Provincial entities: Valencia Provincial Fire Consortium
- Provincial authorities: President of the Provincial Council

Regional Users Involved in Fire Use cases

- Emergencies Rural Brigades
- Airborne and Heliborne Brigades
- Forest Fire Prevention Service of the Department of Agriculture, Environment, Climate change and Rural development
- Natural Park Guards

Health services

- Center of Coordination for Emergency Information and Coordination (CICU)
- Urgent Medical Assistance Service Units (SAMU)

- Basic Life Support Units (SVB)
- Not Assisted Transport (TNA)

<u>Regional Authority in Fire Use cases</u>

- Regional Authority
- Regional Director for Prevention, Fire Extinction and Emergencies
- Valencian Safety and Emergency Response Agency (112- regional PSAP)

<u>National Users Involved in Fire Use cases</u>

- Guardia Civil Traffic Group
- Guardia Civil Nature Protection Service (SEPRONA)
- National Police
- UME (military emergency unit)

<u>National First responders Involved in Fire Use cases</u>

- Aircrafts of the Ministry of Environment

Objectives: beAWARE should:

- Send information regarding people in danger – where are they, how many, how will they be affected – in order to optimize the response and possible evacuation.
- Provide visualization on incoming information (mobile apps, videos, calls, text messages, sensors, social media).
- Give the ability to manage the tasks assigned to the first responders and track these tasks, as well as personnel and material to allow appropriate mobilization and coordination of resources (water supply etc.).
- Provide information about the fire and weather conditions (development, flame height, wind direction, drought index etc.).
- Provide information to the public (safe locations, safe evacuation directions/ways, warnings, forbidden activities/behaviors).

4.3 Scenario 3: Heatwave

Target group:

The target groups listed below are those, who will use and interact with the beAWARE platform in the phase of preparedness and response during a heatwave emergency. These groups are categorized based on their geographical level of involvement.

<u>Users involved in heatwave cases in local level</u>

- Municipal Civil Protection
- Municipal Social Services
- Mayor

- Local Firefighters: Professional body
- National Emergency Aid Centre (EKAV)
- Hospitals
- Volunteer organizations: First responders
- Local traffic police (in case of a traffic jam)

<u>Users involved in heatwave cases in regional level</u>

- Regional Directorate of Fire Department
- Regional Civil Protection
- Regional Health Directorate
- Regional Governor
- Regional PSAP (112, 166, 199)

<u>Users involved in heatwave cases in national level</u>

- National Health Operations Centre
- General Secretariat for Civil Protection
- Ministry of Health
- Ministry of for Citizen Protection
- Ministry of National Defense

Objectives: beAWARE should provide the following support to first responders and decisions makers:

- Issue an alert on an imminent heatwave in order to inform authorities to take proactive measures.
- Provide an assessment on fire risk, the period during and after a heatwave.
- Send information of where people are in danger in order to send rescue teams or ordering evacuations/confinement, especially in the cases where elder or sick people are confined in houses with no A/C or people are trapped in an elevator.
- Send information and create an estimate based on past events, on how many people are in danger and how many are/could be affected from the heatwave.
- Visualization on relevant information provided via mobile apps, calls, text messages, as well as social media streams (social media monitoring).
- Assign tasks to first responders and evaluate their development in real time.
- Monitor heatwave development and the weather conditions.
- Keeping track of personnel, volunteers and equipment/vehicles.
- Allowing the appropriate mobilization of resources.
- Monitor traffic conditions in order to mobilize first responders more effectively.
- Monitor the level of occupancy in places of relief that are offered to general public during a heatwave.

beAWARE should also provide the following support to citizens affected by the heatwave:

- Send information on places for relief, e.g. where they are and what is the level of their occupancy in real time, as much as possible.
- Send warnings to citizens in order to avoid interferences with first responders.
- Inform citizens of best practices/behaviours during a heatwave.
- Warn citizens of an imminent heatwave.

5 Project Timeline and Milestones

The platform will be built incrementally and iteratively, starting with a system made up of dummy services for formal end-to-end validation and delivering increasingly enriched functionality and further capabilities on each milestone.

The figure below illustrates the timeline for the technical milestones in the evolution of the system.

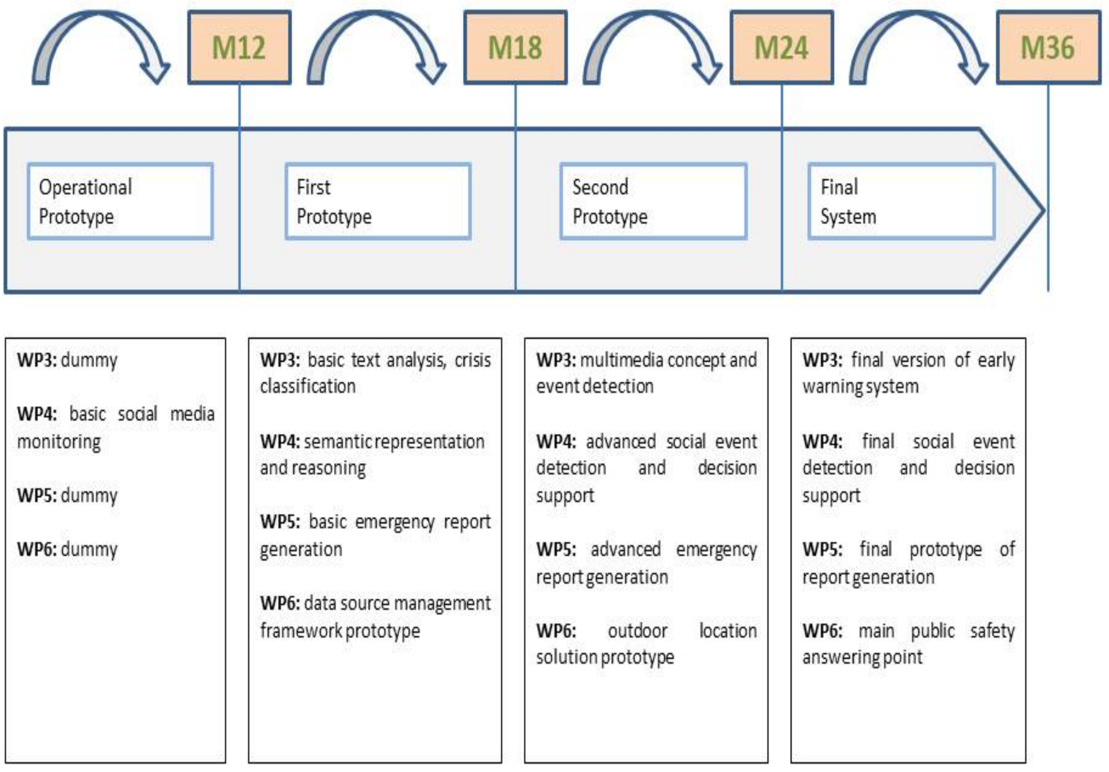

M12
M18
M24
M36
Operational Prototype
First Prototype
Second Prototype
Final System
WP3: dummy
WP4: basic social media monitoring
WP5: dummy
WP6: dummy
WP3: basic text analysis, crisis classification
WP4: semantic representation and reasoning
WP5: basic emergency report generation
WP6: data source management framework prototype
WP3: multimedia concept and event detection
WP4: advanced social event detection and decision support
WP5: advanced emergency report generation
WP6: outdoor location solution prototype
WP3: final version of early warning system
WP4: final social event detection and decision support
WP5: final prototype of report generation
WP6: main public safety answering point

Figure 6: beAWARE walking skeleton

6 Summary

In this deliverable, the technical roadmap of the beAWARE platform was presented. This document will guide the development of the project with a view to attaining the scientific and the technical objectives envisaged.

However, since the project is following an iterative development procedure (use cases-requirements-development-evaluation), it is expected that small adaptations and deviations from the initial technical specifications and the module functionalities foreseen by this document will be needed to satisfy the final user requirements. Such changes/adaptations might be required at component or subcomponent level. The platform architecture together with the detailed specification of the modules (including any updates required) will be reported in D7.2 (System requirements and architecture).