EUROPEAN COMMISSION

pulse

Platform for European Medical Support During Major Emergencies

D4.3 Logistics Tool

skytek · CESS Centre for European Security Strategies · ONEST SOLUTIONS · TRILATERAL RESEARCH · UNIVERSITÀ CATTOLICA del Sacro Cuore · Selex ES A Finmeccanica Company · IAEMO

PULSE

Platform for European Medical Support during major emergencies

WP4 - Tools

Deliverable D4.3 - Logistic tool

30/11/2015

Contractual Delivery Date:	Actual Delivery Date:	Deliverable Type*-Security**:
30/11/2015	30/11/2015	R - PU
607799	PULSE	Platform for European Medical Support during major emergencies

*Type: P: Prototype; R: Report; D: Demonstrator; O: Other.

**Security Class: PU: Public; PP: Restricted to other programme participants (including the Commission); RE: Restricted to a group defined by the consortium (including the Commission); CO: Confidential, only for members of the consortium (including the Commission).

Responsible:	Organisation:	Contributing WP:
Silvano Mignanti	Selex ES	WP4

Authors (organisation)
Silvano Mignanti (SES), Francesco Malmignati (SES)

Abstract:
The purpose of this document is to provide a report on the Logistic tool component. This is an important module of the PULSE platform since it is used to store all the information available regarding the crisis resources' status and allows a real-time retrieval of these data. In addition to this, the Logistic tool is able to calculate the optimal dispatch of the casualties to the hospitals. Therefore, this deliverable also provides a detailed description of the algorithm that have been defined for the generation of this optimal dispatch.

Keywords:
Logistic, optimal, heuristic, resources, crisis, overview

Revisions:

Revision	Date	Description	Author (Organisation)
0.1	21/09/15	First inputs to the document	Silvano Mignanti (SES)
0.2	20/10/15	Finalization of chapters 4 and 5	Silvano Mignanti (SES)
0.3	03/11/15	Chapter 6	Francesco Malmignati (SES)
0.4	06/11/15	Editing of the document	Francesco Malmignati (SES)
1.0	09/11/15	Final version of the document	Francesco Malmignati (SES)
1.1	26/11/15	Internal review	Paul Kiernan (SKY)
1.2	27/11/15	Addressing final review comments	Francesco Malmignati (SES)

Table of contents

List of figures

List of Tables

1 List of acronyms

Acronym	Definition
API	Application Programming Interface
(S)CRUD	(Search), Create, Read, Update, Delete
DB	DataBase
DBMS	DB Management System
DSVT	Decision Support Validation tool
GPS	Global Positioning System
GUI	Graphical User Interface
JDBC	Java DataBase Connectivity
JPA	Java Persistence API
JSON	JavaScript Object Notation
MPORG	MultiPlayer Online Role Playing Game
REST	REpresentational State Transfer
SCGT	Surge Capacity Generation Tool
SQL	Structured Query Language

2 Executive Summary

The aim of the deliverable D4.3 is to describe the PULSE Logistic Tool, with its sub-components, the provided functionalities and the relations with the use cases defined in WP2.

The aim of the Logistic Tool component is to provide both functionalities to manage data regarding the events, in particular with respect to the crisis management, and an optimization mechanism, able to provide an almost optimal solution that, by using the Health care facility model defined in WP3, assesses the required stockpiles of any necessary equipment, medications and vaccinations present in the different hospitals and is able to assign all the wounded to the proper hospitals using as many hospital resources as possible and sending them in the minimum amount of time.

The document provides also a description of the various components of the Logistic Tool and, in particular, a formalization of the optimization problem being resolved by the tool is presented.

3 Introduction

3.1 Scope of the Document

This document is a covering document to support the software delivery of the PULSE *Logistic tool.*

It summaries the software component delivery and provides high-level details on the architecture, technologies and underlying libraries on which the software component has been developed.

3.2 Structure of the Document

This document is structured into the following sections.

- Description of the *Logistic tool* objective.
- Main functionalities of the Logistic component including a detailed description of the optimization algorithm.
- Description of the *Logistic tool* internal architecture and relation of this component with other elements of the PULSE platform architecture;
- List of technologies adopted for the implementation of *the Logistic tool*;
- List of underlying 3rd party libraries used for *Logistic tool* implementation and summary of the corresponding licenses

3.3 Relation with other Deliverables

The work presented in this deliverable is related to the following WP2 and WP4 deliverables:

- D2.2 - Use case specification [5] – This document describes the use cases related to two PULSE reference scenarios: SARS-like epidemics and Stadium crush.
- D4.1 – Decision support Validation tool [1] – It is the tool that invokes the API provided by the Logistic tool and provides a graphic front-end for the management of the crisis resources.
- D4.5 – Training tools [3] – The MPORG described in D4.5 [3] invokes the Logistic tool API to obtain the status of the crisis resources.

4 Tool description

4.1 Objective

The **Logistic tool** is a fundamental module of the PULSE platform. It is in fact the component in charge of managing all the data regarding events, such as the Ambulances, First responders, Wounded, Hospitals in an incident-like scenario and the probable and confirmed cases and weak signals in a SARS-like scenario.

In addition to this, the **Logistic tool** is able to calculate the optimal dispatch of the casualties to the available hospitals. It is in fact able to provide an almost optimal solution to send all the wounded to the proper hospitals using as many hospital resources as possible and sending them in the minimum amount of time. In order to achieve this, it considers the actual resources present in the different hospitals and it exploits the models defined in WP3, by invoking the *Surge capacity generation tool* (SGCT) [2], for assessing the required stockpiles of any necessary equipment, medications and vaccinations. These required stockpiles are crucial information for the optimization algorithm (as described in 4.2.2.1) considering that it would be impossible to efficiently predict the best allocation of people (that need resources) to the hospitals (that provide resources) without considering such kind of information.

4.2 Functionalities

All the functionalities are made available to other components of the PULSE architecture (e.g., DSVT) through a standard Web Service which can be invoked by a RESTful interface. The messages exchanged with the Logistic tool by means of this interface are HTTP-based requests and responses. They include a message body whose information content is represented in JavaScript Object Notation (JSON). Subsections 4.2.1 and 4.2.2 provide more details about these functionalities.

4.2.1 Real-time crisis information management

The main objective of the PULSE platform is to develop an operational framework that allows the platform's stakeholders to have access to timely key data, planning and decisions that efficiently help them to manage a major healthcare crisis. For this reason, it is crucial to have inside the PULSE platform a component that would be able to store all the information available regarding the crisis resources' status and to allow a real-time retrieval of these data. The component in question is the Logistic tool that is in charge of the management of the crisis information.

The resources handled by the tool are the following:

- Ambulance:
  - Status (e.g. Free, with a patient),
  - Category
  - Destination (e.g. Hospital, incident location),
  - Medical equipment on board,
  - Time needed to reach the destination,
  - GPS coordinates,
  - Person currently assisted
- Person:
  - Triage code,

    - Symptoms,
    - Health condition,
    - Required resources to be cured,
    - GPS coordinates,
    - Full name (if available)
- Hospital:
    - GPS coordinates,
    - Set of medical resources at disposal (e.g. Doctors, beds, etc.),
    - Set of persons currently assisted
- Resource (e.g. equipment, medications and vaccinations):
    - Category,
    - Quantity,
    - Unit of measure
- Rescuers:
    - Category,
    - GPS coordinates,
    - Full name,
    - Qualification,
    - Set of medical resources at disposal,
    - Tasks
- Task:
    - Description,
    - Completion percentage

The tool provides a proper RESTful interface to (S)CRUD (Create – Read – Delete – Update – Search) all the described data. This functionality can be triggered with any RESTful client by calling the URL: http://hostname:port/crisismanagement/ with the following HTTP methods:

- **GET**: to retrieve (Search) information on instances on the database
- **POST**: to persist (Create) new instances on the database
- **PUT**: to persist updates (Update) to instances already existing on the database
- **DELETE**: to remove (Delete) existing instances from the database.

and by adding at the end of the URL the name of the resource that we want to manage. For example, if we want to retrieve the list of Ambulances we can send a **GET** request to http://hostname:port/crisismanagement/ambulances. This method returns the JSON file depicted in the following Figure 1.

```
[
  {
    "id": 1,
    "status": "free",
    "category": "A",
    "timeToDestination": 50,
    "coords": {
      "lat": 41.9,
      "lng": 12.4833333
    },
    "destination": {
      "lat": 41.84,
      "lng": 12.38321
    },
    "resources": [
      {
        "category": "blood 0-",
        "quantity": 5,
        "unitOfMeasure": "unit",
        "ambulanceId": 1,
        "id": 1
      },
      {
        "category": "defibrillator",
        "quantity": 1,
        "unitOfMeasure": "unit",
        "ambulanceId": 1,
        "id": 2
      }
    ]
  }
]
```


Figure 1: Example of JSON file

4.2.2 Optimized dispatch

Currently, when an incident happens, the people in charge usually decide for the dispatch of the wounded to the hospitals by following the personal expertise and experience. The Logistic tool tries to complement this by providing an almost optimal solution that considers the actual resources present in the different hospitals and is able to assign all the wounded to the proper hospitals using as many hospital resources as possible and sending them in the minimum amount of time.

The optimisation tool calculates the optimal dispatchment of the various casualties to the available hospitals, trying to maximise some indicators (e.g. the number of survivors). It could be accessed through a RESTfull interface passing to the system the entire status of the critical event, it returns the optimal dispatchment of the provided set of casualties. A detailed description of the optimization problem is provided in 4.2.2.1.

4.2.2.1 Details on the optimization problem

Core to the Logistic Tool, is the optimization problem, i.e. the problem the tool aims to solve. Here follows a formal definition of the whole problem.

Let's define a set of patients as follows:

$$P(t) = \{p_1(t), ..., p_{a(t)}(t)\}$$

where each patient p_i is $p_i(t) = p_i(S_i(t), N_i(t))$, with $i \in I = \{1, ..., a(t)\}$ and where:

- $S_i(t) = \{V_i(t), \Delta V_i(t)\}$ is the status of the patient p_i at time t;
- $V_i(t) = \{v_1^i(t), ..., v_b^i(t)\}$ is the set of values of the various (vital) parameters of the patient p_i at time t;
- $\Delta V_i(t) = \{\Delta v_1^i(t), ..., \Delta v_b^i(t)\}$ is the variation function of the parameters of the patient p_i during the time, calculated at time t;
- $N_i(t) = \{n_1^i(t), ..., n_c^i(t)\}$ is the set of the necessities, i.e. resources the patient p_i requires to be cured, at time t;

Note:

$a(t)$ = number of patients at time t;

b = number of vital parameters (at ANY time) of the patients

c = number of possible types of resources (at ANY time) for patients to be required (and also being possibly available at the hospitals)

Let's also define a set of hospitals:

$H(t) = \{h_1(t), ..., h_{d(t)}(t)\}$, where, each hospital h_j is $h_j(t) = h_j(R_j(t), ?\Delta R_j(t)?, T_j(t))$, with $j \in J = \{1, ..., d(t)\}$, where:

- $R_j(t) = \{r_1^j(t), ..., r_c^j(t)\}$, available resources at the hospital h_j at time t, $j \in J$;
- $?\Delta R_j(t) = \{\Delta r_1^j(t), ..., \Delta r_c^j(t)\}?$, is the variation function of the resources of the hospital h_j during the time, calculated at time t;
- $T_j(t)$ is the time needed to reach the hospital h_j at time t.

Note:

c = number of possible types of resources being possibly available at the hospitals (at ANY time)

$d(t)$ = number of hospitals at time t;

Let's finally introduce a set of ambulances as follows:

$A(t) = \{A_1, ..., A_{e(t)}\}$

Where

$e(t)$ = number of available ambulances at time t.

The overall idea is to dispatch the various patients (injured/wounded) to the available hospitals in order to:

a) Maximize the number of patients surviving the critical event
b) Minimize the overall time required to cure all the patients (including time to reach the hospitals)
c) Evenly distribute the patients among the hospitals

The overall idea of the first objective is to maximize the number of persons surviving to the critical event. At the event location, a triage will be made to catalogue people among red, yellow and green ones, in descending order of gravity of their situation. A possible fourth colour is "black", reserved for people so unlucky to be in a situation

there will be no possibility they survive even if cured. In such a dramatic case, usually going to be adopted only in very exceptional cases, considering it would be “useless” to send such a person to a hospital, the system will decide to reserve the cures to other people having chances to survive.

The second objective is not important as the first one (actually, it is infinitely less important than the first one), but, introducing it, it would be possible to obtain that:

- Patients are going to be cured as fast as possible
- Patients are going to be cured in the minimum possible overall time. This is very important: typically, persons are cured in a strict “red, yellow, green” order, which is not always the best choice: in fact, if a person of higher priority requires resources not available at that moment, there could be space to cure a person with lower priority, if this one requires less resources and its cures end “in time” (which means, the patient with lower priority releases resources sufficiently fast to not cause delays in the cure of patients with higher priorities)

The third objective is (again, infinitely) less important than the first two, but “evenly” distributing patients among hospital will avoid to overload specific hospitals which, in chain, means to reduce the overall time of cures. An important note about “evenly”: this not implies that patients are sent in an equal number to each hospital, but that patients are sent to the hospitals based on the actual capacity of the hospital to cure them. For example, if an hospital is “as big as twice” another hospital, meaning it has twice the resources, the doctors… of the other one, an even distribution is to send to the first hospital two times the patients sent to the first one.

Let’s continue the formalization defining a decision function:

$$d_{i,j}(t) = \begin{cases} 1, \text{ if the patient } p_i \text{ is sent to the hospital } h_j \text{ at time } t \\ 0, \text{ otherwise} \end{cases}$$

Let’s also define a decision matrix

$$D(t) = \begin{bmatrix} d_{1,1}(t) & d_{1,2}(t) & \dots & d_{1,d(t)}(t) \\ d_{2,1}(t) & d_{2,2}(t) & \dots & d_{2,d(t)}(t) \\ \dots & \dots & \dots & \dots \\ d_{a(t),1}(t) & d_{a(t),2}(t) & \dots & d_{a(t),d(t)}(t) \end{bmatrix}$$

containing all the decision taken for all the patients at time t (i.e., the matrix indicates, at time t, in which hospital each patient is sent). Note that sending a certain patient p_i at time t to the hospital h_i does not mean that patient will be cured at time t; the patient will be cured at a time $t+\tau_i$, where τ_i is potentially different for each patient. For this reason, we will define a set $T(t) = \{\tau_1, \tau_2, \dots, \tau_{a(t)}\}$ containing all the time instants all the patients (from 1 to $a(t)$) will be cured in the hospitals they are sent at time t.

While pursuing the objectives introduced above, several constrains hold.

At any instant t, if a certain patient p_i is decided to be sent to the hospital h_j at time t, for being cured at time $t+\tau$, it should be (for that τ): $R_j(t+\tau) \geq N_i(t+\tau)$. The previous inequality among vectors is intended as meaning that each element of the

two vectors is respecting the inequality (i.e. $r_k^j(t+\tau) \geq n_k^i(t+\tau),\ \forall k \in \{1,...,c\}$). In behalf of the previous one, we will consider the following inequality: $r_k^j(t+\tau) \geq n_k^i(t+\tau) \cdot d_{i,j}(t),\ \forall k \in \{1,...,c\}$, which also considers whether or not a patient p_i is decided to be sent to the hospital h_j at time t.

This means that, at time $t+\tau$, all the required resources for the patient are available at that hospital: if at that time the constrain is not respected, the hypothesis is that patient could not be cured at all!

Another fact we must assure is that each patient has to be assigned (at time t) only to a single hospital:

$$\sum_{j=1}^{d(t)} d_{i,j}(t) = 1, \forall i \in I$$

The previous is not sufficient: we also have to assure each patient is assigned once to (one among) the hospitals, so that we need to guarantee that:

$$\sum_{t=1}^{a(t)} \sum_{j=1}^{d(t)} d_{i,j}(t) = 1, \forall i \in I$$

Finally we also need to guarantee that the sum of the needs for each resource of all the patients sent to a certain hospital to be cured at a certain time t is lower than the amount of the available resources (for each type) in that hospital at that time. To this aim, we need to define a presence function (a "cure" function):

$$c_{i,j}(t) = \begin{cases} 1, \text{ if the patient } p_i \text{ is cured at hospital } h_j \text{ at time } t \\ 0, \text{ otherwise} \end{cases}$$

Let's also define a presence matrix:

$$C(t) = \begin{bmatrix} c_{1,1}(t) & c_{1,2}(t) & ... & c_{1,d(t)}(t) \\ c_{2,1}(t) & c_{2,2}(t) & ... & c_{2,d(t)}(t) \\ ... & ... & ... & ... \\ c_{a(t),1}(t) & c_{a(t),2}(t) & ... & c_{a(t),d(t)}(t) \end{bmatrix}.$$

We can now introduce the constraints:

$$\sum_{i=1}^{a(t)} n_k^i(t) \cdot c_{i,j}(t) \leq r_k^j(t)\ \forall j \in J, \forall t \in T(t)$$

.

Summarizing the constraints are:

1) $r_k^j(t+\tau) \geq n_k^i(t+\tau) \cdot d_{i,j}(t),\ \forall i \in I, \forall j \in J, \forall k \in \{1,...,c\}, \tau \in T(t)$

2) $\sum_{t=1}^{a(t)} \sum_{j=1}^{d(t)} d_{i,j}(t) = 1, \forall i \in I$

3) $\sum_{i=1}^{a(t)} n_k^i(t) \cdot c_{i,j}(t) \leq r_k^j(t), \forall j \in J, \forall k \in \{1,...,c\}, \forall t \in T(t)$

Where:

1) guarantees any patients is sent to an hospital having at least all the resources (e.g. equipment, medications and vaccinations) he needs to be cured,
2) guarantees each patient is sent just to single hospital, and
3) guarantees that the sum of the resources of a certain type k requested by the patients sent to a certain hospital j and cured in that hospital at a certain time instant t, is at maximum equal to the available resources of that type k in that hospital j in that time instant t.

Now let's introduce the optimizations to be performed.

The simplest is the second (b), i.e. to minimize the overall time required to cure all the patients; in formulas, it could be expressed as follows:

$$\min \sum_{i=1}^{a(t)} \tau_i$$

Another optimization objective quite simple to be introduce is the "fairness". First of all we have to define what we consider as "fair". The possible choices are mainly two:

1) define fair an assignment if it assigns the patients evenly distributing them among the hospitals (considering all the constrains satisfied and any other optimization objective having the same value – in few words, not considering neither the constrains nor any other optimization objective).
2) define fair an assignment if it assigns the patients distributing them proportionally to the "capacity" of the various hospitals.

The formula in order to obtain the first objective could be introduced as follows.

First of all let's introduce an error function, defined as follows:

$e_j(t) = N_j(t) - \sum_{i=1}^{a(t)} d_{i,j}(t)$, where

$e_j(t)$ is exactly the error function of the hospital h_j at time t;

$N_j(t)$ is the "nominal capacity" of the hospital h_j at time t: such a capacity is a measure about how many patients a certain hospital could cure, at the same time, at a certain time instant. Generally speaking, we can consider $N_j(t) = f(R_j(t), \Delta R_j(t))$, i.e. the capacity is a given function $f(\cdot)$ of the resources (and of their variation) of the hospital h_j at time t. The rationale to introduce the variation is that a certain hospital, for example with a certain number of doctors active at the same time, has a certain number of doctors waiting for their turn to go to work in that hospital. For this reason, a hospital having in media 10 doctors active at the same time, it is expected to have more or less the same number of doctors active during all the day. Furthermore, in cases of crisis, it is possible to expect the number of doctors even to increase; in few words, considering 3 turns of 8 hours, in such a hospital, it could be expected the total number of doctors working in that hospital to be more or less 25-30. Consequently, for a hospital with 100 active doctors it could be expected a pool of about 250-300 doctors.

Defined in this way, the error is the difference among the nominal capacity of the hospital h_j at time t and the patients requested (going) to be cured in that hospital in the same time instant.

With these definitions, we can introduce the following optimization objective:

4) $\min_{D(t)} \sum_{j=1}^{d(t)} e_j(t)^2$ which exactly leads to a fair assignment as indicated in point (1) (we remember that $D(t)$ is the decision matrix).

In our opinion, for the reasons indicated above for the concept of "nominal capacity", (4) is not the fairest assignment: consider the following example. Suppose to have two hospitals, the first with capacity 5, the second with capacity 50, and to have to assign 1000 patients. The various constrains could assign, in different order, up to the first 55 patients among the two hospitals, depending on their actual needs. Let's suppose the optimal assignment being 5 patients to the first hospital, 50 to the second, with both the hospitals having all their resources assigned for 4 hours. The 56th patient will be assigned, if all the remaining parameters are equals between the two hospitals, to the first one, because this minimizes the (4). As well, patients from 57 to 100 will be assigned to that hospital. From that point, the patients will be assigned one each to the two hospitals, leading to 500 patients sent to each hospital. The point is that the nominal capacity of the first hospital is just 5, while the second has 50, so that one could expect the second hospital to be able to take care of much more patients of the first one: not only 45 more, but something near to ten times the first one! For this reason it is better to define a different error function:

5) $e_j(t) = \dfrac{\sum_{i=1}^{a(t)} d_{i,j}(t)}{D_j(t)}$

This error function is (inversely) proportional to the nominal capacity of the hospitals, and leads to much better results. Summarizing, we will consider the optimal function defined in (4) where the error function is defined as in (5).

Finally we can introduce the first optimization objective. In order to do so, we need to introduce some further notation.

First of all, we should define a function which measures the probability of survival of a certain patient. It is important to notice that, in every hospital, or a the location of the critical event, the patients incur in a triage phase, in which a triage code is assigned to each of them. Such a code indicates the severity of the status of a patient, ranging from white/green (minor disease, not critical), to red (patient at risk of life). Based on this, and based on the vital parameters of each patient, we will assign a probability of survival equals to 1 to each patient with triage colour different from red. Red patients will have probability 0 to survive if not all the resources they require are available for them, while a probability between 0 and 1 if the required resources are available. We will also consider a further "triage" code, black, which indicate patients with no chances to survive: in this group we will include patients requiring specific resources which are not available and patients with no probability of survival even if cured. In formulas, we will introduce:

$$Tc_{i,j}(t) = g(p_i(t), h_j(t+\tau_i)) \in \{white, green, yellow, red, black\}$$

Where $Tc_{i,j}(t)$ is the triage code assigned to the patient p_i going to be cured at hospital h_j at time $t+\tau_i$ and $g(\cdot)$ is a function that consider the status of the patient p_i and of the hospital h_j at times t and $t+\tau_i$ respectively. As described in [10], the possible values are:

- white: (dismiss) are given to those with minor injuries for whom a doctor's care is not required; 100% chance of survival
- green: (wait) are reserved for the "walking wounded" who will need medical care at some point, after more critical injuries have been treated; 100% chance of survival.
- yellow: (observation) for those who require observation (and possible later re-triage). Their condition is stable for the moment and, they are not in immediate danger of death. These victims will still need hospital care and would be treated immediately under normal circumstances; 100% chance of survival
- red: (immediate) are used to label those who cannot survive without immediate treatment but who have a chance of survival; [0-100%] chance of survival
- black: (expectant) are used for the deceased and for those whose injuries are so extensive that they will not be able to survive given the care that is available; 0% chance of survival.

Note that the triage code is assigned to a patient at a certain time, but it could change during the time: typically, a person with a yellow code which is not cured for a long time could become a red code.

We will also define a function indicating the chance of survival, at time t, of a certain patient p_i expected (assigned) to be cured at hospital h_j at time $t+\tau_i$:

$$S_{i,j}(t)=\begin{cases} 1, if\ Tc_{i,j}(t)\in\{white, green, yellow\} \\ s(p_i(t),h_j(t+\tau_i))\in(0,1], if\ Tc_{i,j}(t)=red \\ 0, if\ Tc_{i,j}(t)=black \end{cases}$$

Where $S_{i,j}(t)$ is the survival function and $s(p_i(t),h_j(t+\tau_i))$ is a function of the status of the patient p_i, at time t, going to be cured at hospital h_j at time $t+\tau_i$. With these definitions, we can finally introduce the most important optimization objective:

6) $\max_{J(t)} \sum_{i=1}^{a(t)} S_{i,j}(t)\cdot d_{i,j}(t)$,

i.e to maximize the chances of survival of all patients, optimizing their distribution among hospitals.

4.3 Relation with WP2 Use Cases

Part of the work carried out in WP2, was focused on the definition of a set of use cases for PULSE that have been detailed in the deliverable D2.2 [5]. In the following paragraph, we will try to associate the functionalities described in section 4.2 with the use cases where the DSVT is actually involved.

The Logistic tool, for its own nature, has to be considered as a “core” functionality of the PULSE platform. Nevertheless, it is anyhow possible to determine some specific use cases more strictly related to the tool itself:

Table 1 - Logistic tool relations with WP2 use cases

Use cases	Step	Role
UC-SARS LIKE – 02 UC-SARS LIKE – 03 UC-SARS LIKE – 04	Step: 9.vii	The Logistic Tool, in particular the Optimization Tool, will provide the optimal dispatchment information about the various casualties
UC–STADIUM CRUSH – 02 UC–STADIUM CRUSH – 04 UC–STADIUM CRUSH – 08	N/A	The Logistic Tool, in particular the Logistic Manager, will act as the background DB for the MPORG (UC–SC–2), for the Surge Capacity and Bed Management (UC–SC–4) and for the researches of the Casualty Bureau Operation (UC–SC–8).
UC – STADIUM CRUSH – 07	N/A	The Logistic Tool, in particular the Logistic Manager, will act as the source of information for the post crisis evaluation.

5 Tool architecture

5.1 Software Component Architecture

The Logistic Tool architecture is relatively simple, and it is depicted in the following figure.

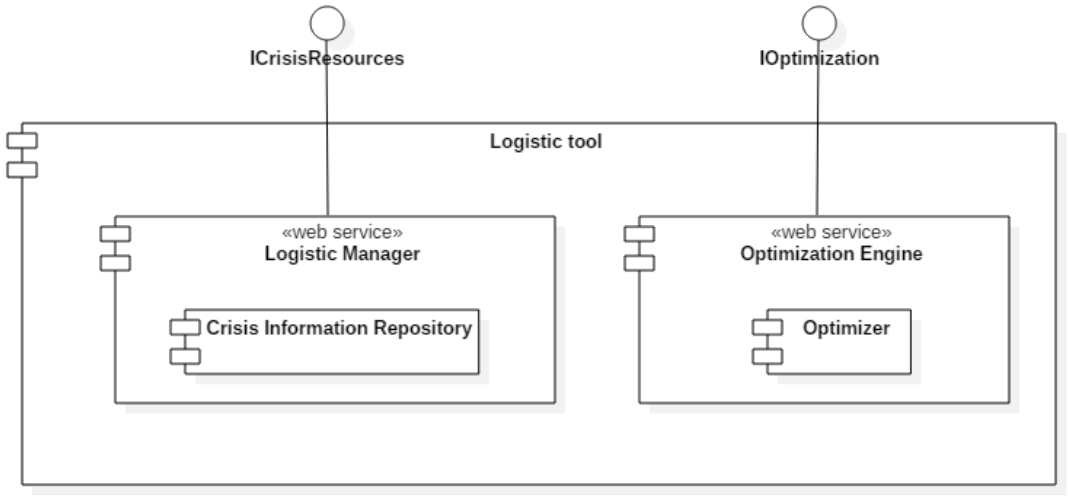

Figure 2: Logistic tool component architecture

The Logistic tool is composed of two sub-components: the *Logistic Manager* (see 5.1.1), which provides a web service interface for the management of the crisis resources and the *Optimization Engine* (see 5.1.2), which provides a web service interface for the generation of optimal dispatchment of the various casualties to the available hospitals.

5.1.1 Logistic Manager

The aim of the *Logistic Manager*, within the *Logistic Tool*, is to provide means to store and manage (SCRUD) data regarding events, among which, for example, the location of the event, the hospitals, the ambulances, the casualties, etc.

The *Logistic Manager* is constituted by one sub-component called *Crisis Information Repository*, which is the internal Database used by the Logistic Manager to store the crisis information described in 4.2.1.

The Logistic Manager provides a RESTful interface called *ICrisisResources*, using JSON syntax and the “standard” RESTful approach for the supported methods:

- GET: to retrieve information on instances on the database
- POST: to persist new instances on the database
- PUT: to persist updates to instances already existing on the database
- DELETE: to remove existing instances from the database.

The described operation can be triggered with any RESTful client by calling the HTTP request method POST at the following URL:

- http://hostname:port/crisismanagement/storeSARSSnapshot.

5.1.2 Optimization Engine

The optimization tool calculates the optimal dispatchment of the various casualties to the available hospitals, trying to maximise some indicators (e.g. the number of survivors), as described in 4.2.2. The Optimization Engine contains internally a module called *Optimizer* that is in charge of the execution of the Optimization algorithm. It is accessed through a RESTfull interface called *IOptimization* that accept as input the entire status of the critical event. This status must contain:

- A set of wounded people (containing information regarding the number of resources needed to cure each single injured person)
- A set of hospitals (containing information regarding the number of medical resources available)

After the execution of the optimization algorithm, the Optimizer returns then the optimal dispatchment of the provided set of casualties.

5.2 Component relation to overall PULSE architecture

Figure 3 shows a component diagram of the PULSE Platform Architecture where the relations between the *Logistic Tool* and other tools are highlighted. As said in 5.1, the Logistic Tool provides two different interfaces: the *ICrisisResources* and the *IOptimization*. These two interfaces are invoked, in different occasions, by other tools of the PULSE platform.

In particular the *ICrisisResources* interface is invoked by:

- the DSVT when the operational overview of the crisis must be depicted on the GUI front-end. In this case the DSVT retrieves or updates the status of the crisis resources (as described in 4.2.1) by invoking the Logistic tool interface.
- the MPORG (described in D4.5 [3]) when the tool wants to retrieve the status of the crisis resources to simulate the scenario starting from the actual scenario conditions.

The *IOptimization* interface is invoke instead by:

- the DSVT when a user asks for an immediate optimized dispatch of the wounded to the hospitals surrounding the incident location.
- The DSVT during the simulation of the possible crisis evolution.
- The MPORG to compare the results obtained by the user during the “game” with the “optimal” solution generated by the Logistic Tool.

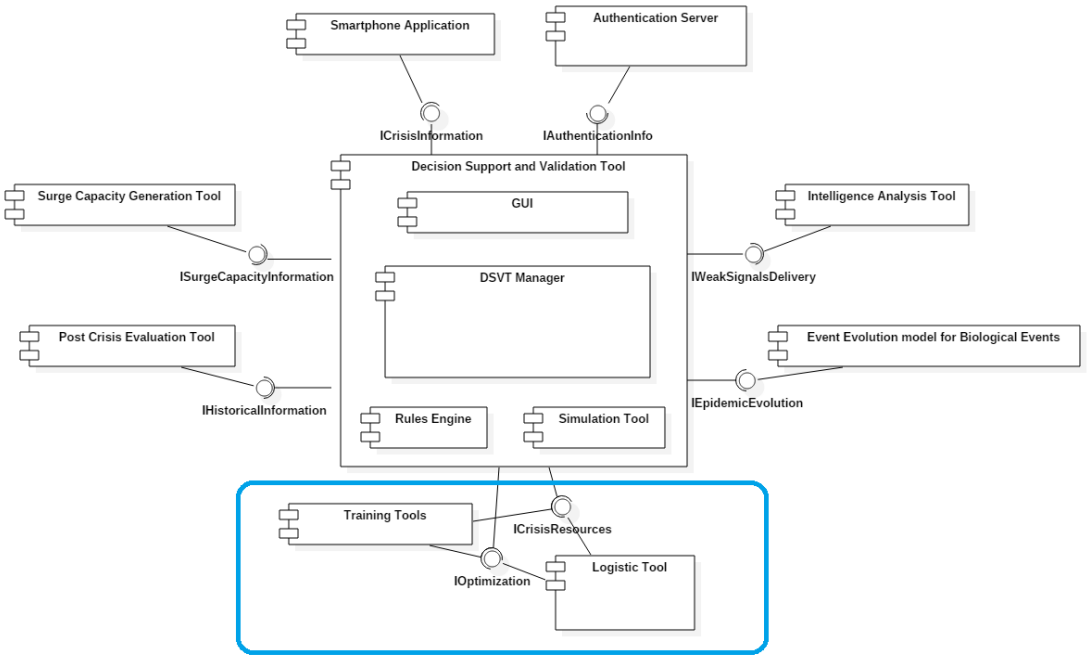

Figure 3: PULSE Architecture

6 Component Technologies

6.1 List of core technologies

6.2 3rd Party libraries and licenses

As said in 5.1, the Logistic Tool is composed of 2 sub-modules the Logistic Manager and the Optimization Engine.

The core technologies selected for the implementation of these components are:

- **Logistic Manager**
  - *Jetty* – It is a Java-based Web Server used to deploy the web application exposing the *ICrisisResources* interface. It implements an HTTP server and servlet container capable of serving static and dynamic content either from a standalone or embedded instantiations. This technology is developed as a free and open source project as part of the Eclipse Foundation [6].
  - *Jersey* – It is an open source framework [7] that provides an API extending the JAX-RS [8] toolkit with additional functionalities, with the purpose of simplifying the implementation of RESTful services. This technology is used to build the Web Service RESTful interface offered by the *Logistic Manager*.
  - *H2* [12] – it has been selected as the DBMS constituting the Database of the Logistic Manager, considering it is lightweight and could be easily integrated in a "all-in-one" package, not requiring a mandatory preliminary installation. Nonetheless, any kind of relational DB could be used as DBMS, considering it is the code of the DB Manager itself that automatically creates the structures of the tables of the database: this was obtained by using of the Java Persistence API (JPA) that provides the proper decoupling.
- **Optimization Engine**
  - *Jetty* – It is the Web Server used to deploy the web application exposing the *IOptimization* interface.
  - *Jersey* – This technology is used to build the Web Service RESTful interface offered by the *Optimization Engine.*
  - *OptaPlanner* – it is a lightweight, embeddable planning engine [11]. It allows solving optimization problems efficiently. OptaPlanner combines sophisticated optimization heuristics and metaheuristics (such as Tabu Search [13], Simulated Annealing [14] and Late Acceptance [15]) with very efficient score calculation. It is the core of the Optimizer sub-component and it is exploited to calculate the optimization algorithm defined in 4.2.2.1.

Below is a list of third party libraries/frameworks used and the licenses under which they are distributed.

Table 2 - DSVT 3rd party libraries and licenses

Product	Version	Vendor	License
Jersey	9.2.10	Eclipse	Apache 2.0
Jetty	2.17	Oracle Corporation	Dual License: - CDDL 1.1 - GPL 2
Hibernate ORM	5.0.2	RedHat	LGPL V2.1
H2	1.3.176	Thomas Mueller	Mozilla Public License Version 2.0
OptaPlanner	6.1.0	RedHat	Apache 2.0

References

[1] PULSE Project Deliverable – D4.1 Decision Support Validation tool

[2] PULSE Project Deliverable – D4.4 Surge Capacity Generation tool

[3] PULSE Project Deliverable – D4.5 Training tools

[4] PULSE Project Deliverable – D4.8 Smartphone application

[5] PULSE Project Deliverable – D2.2 Use case specification

[6] Jetty, http://www.eclipse.org/jetty/

[7] Jersey, https://jersey.java.net/

[8] JSR 311 JAX-RS: Java™ API for RESTful Web Services, version 1.1, https://jsr311.java.net/nonav/releases/1.1/spec/spec.html

[9] Hibernate ORM, http://hibernate.org/orm/

[10] Medical Triage: Code Tags and Triage Terminology, http://www.medicinenet.com/script/main/art.asp?articlekey=79529

[11] OptaPlanner, http://www.optaplanner.org/

[12] H2, http://www.h2database.com/html/main.html

[13] Tabu search, Fred Glover (1989). "Tabu Search - Part 1". ORSA Journal on Computing 1 (2): 190–206. doi:10.1287/ijoc.1.3.190

[14] Kirkpatrick, S.; Gelatt Jr, C. D.; Vecchi, M. P. (1983). "Optimization by Simulated Annealing". Science 220 (4598): 671–680. Bibcode:1983Sci...220..671K. doi:10.1126/science.220.4598.671

[15] E. K. Burke and Y. Bykov. The late acceptance hill-climbing heuristic. Technical Report CSM-192, University of Stirling, 2012