EuroSIM Constructive Training System for CBRN Incident Response Training

Dr. James England
Riskaware Ltd.
UNITED KINGDOM

james.england@riskaware.co.uk

Dr. Martyn Bull
Riskaware Ltd.
UNITED KINGDOM

martyn.bull@riskaware.co.uk

ABSTRACT

The EuroSIM Constructive Training System prototype was developed under a Defence and Security Accelerator (DASA) funded project for the Defence Science and Technology Laboratory (Dstl) with the aim of developing a training system for coordinating a response to a Chemical, Biological or Radiological (CBR) event. The system provides a web application for commander and trainer use, and a GPS-enabled Android mobile application providing simulated chemical sensor data and health status information for military or civilian first response trainees operating in the field. A distributed modelling capability provides a shared simulated ground-truth and casualty modelling based upon trainees' GPS locations, providing Modelling and Simulation as a Service (MSaaS).

In this paper, we show how a training scenario can be set up in a user-specific location to respond to a synthetic CBR event. We demonstrate how a Commander can track trainees' positions, exposure to contaminant and health status as they move through the simulated plume, how they create virtual cordons and direct trainees as appropriate. We also show how trainees can see their simulated status through their mobile app and apply a Personal Protective Equipment (PPE) status as directed. We further demonstrate how scripted entities can be added to the system to consider CBR effects on people moving through the vicinity.

1.0 INTRODUCTION

The aim of the EuroSIM CTS project was to provide a proof of concept prototype of a novel training system for CBR responders, that could be developed further to provide a full CBR training capability, either as part of a larger Information Management system (such as the UrbanAware system currently being developed at Riskaware), or as a standalone system. The system is designed to be cloud deployable and run on existing client devices, making the set up simple and cost-effective. The prototype could be extended to develop a complete training system for the use of military, civil emergency response, aid agency and civil users. The modularity of the future system allows for user-specific tailoring to match any agency or body's needs.

The system is designed to be used by three types of user:

- Trainer – the trainer is responsible for setting up a CBR training scenario (creating and adding CBR incidents, setting meteorological observations, starting the real-time dispersion simulation, placing cordons, etc.), adding live trainees to the scenario, and monitoring the real-time scenario – he/she is able to see the simulated plume in the map view and is able to change the scenario as it unfolds. At the end of the scenario, the trainer can interrogate the events that took place during the scenario, analyse the response, and report back to the trainees. The trainer may also add scripted entities (simulated people who move around the scenario), who are also affected by the simulated dispersion.

- Incident Commander – the incident commander is being trained in managing a CBR event. They can view trainee locations on a map (for both live trainees and scripted entities) and see live sensor readings. They can also see the simulated current health status and PPE status of all people in the field based upon the simulated contaminant dispersion. Whilst he/she may see the incident location on the Web Application map, he cannot see the simulated dispersion plume. The commander can place cordons on the map, which can be seen by the trainer, and should direct trainees as appropriate to manage the CBR incident.
- Trainee – the trainee is in the field where a simulated CBR release is being modelled. They have a mobile sensor application that reports back to them a simulated CBR dispersion reading. The trainee should react as if this was a real-life sensor reading. The application also retrieves their health state given the calculated simulated dose that the user has been exposed to. The trainee is able to change their PPE state via the app to show what they would do in a real-world scenario had they been using an actual CBR sensor. This would reduce the effect of the incident release but may hamper movement. The trainee should be in contact with the incident commander and follow any command given, including the setting up of cordons.

2.0 SYSTEM/SERVICE ARCHITECTURE

The EuroSIM CTS system is designed to be easily deployable on local or cloud-based services using Docker containers. The system consists of multiple web service components, each with a REST (Relational State Transfer) API and two applications. Each service is responsible for a specific element of the system:

- **Modelling Service** – responsible for the real-time dispersion simulation and publishing of concentration and dosage output. The modelling service is also responsible for the calculation of dose accumulation and health effects of trainees and scripted entities depending on their location within the simulated plume and their PPE dress state.
- **Incident Controller Service** – responsible for the creation and storage of CBR incidents. A library of incidents can be created and used in any training scenario.
- **Entity Controller Service** – responsible for monitoring the position, exposure, and health status of all entities (trainees or scripted entities). Also responsible for creation and movement of scripted entities.
- **CTS Web Application** – the web service provides the web app front-end serving as a system interface for both commander and trainer. It also manages the scenario definition and allows scenarios to be saved. Multiple users can join a scenario.
- **Android Mobile Sensor Application** – provides a mock CBR sensor that also reports back a simulated health status according to simulated exposed dose. Also allows the trainee to indicate PPE dress state.

The services running in Docker containers and the applications are shown in Figure 1 below.

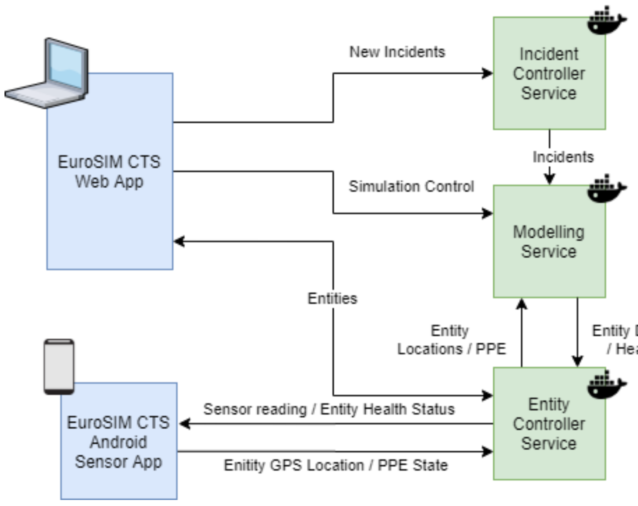

EuroSIM CTS Web App
New Incidents
Incident Controller Service
Incidents
Simulation Control
Modelling Service
Entities
Entity Locations / PPE
Entity Dose / Health
EuroSIM CTS Android Sensor App
Sensor reading / Entity Health Status
Entity Controller Service
Enitity GPS Location / PPE State

Figure 1: EuroSIM CTS Apps and Service Design

2.1 High Level Design

A more detailed high-level design of the system is shown in 2 below, showing the use of the HEART modelling framework within the Modelling Service and the data storage components. Elements of the design are shown in the following sections.

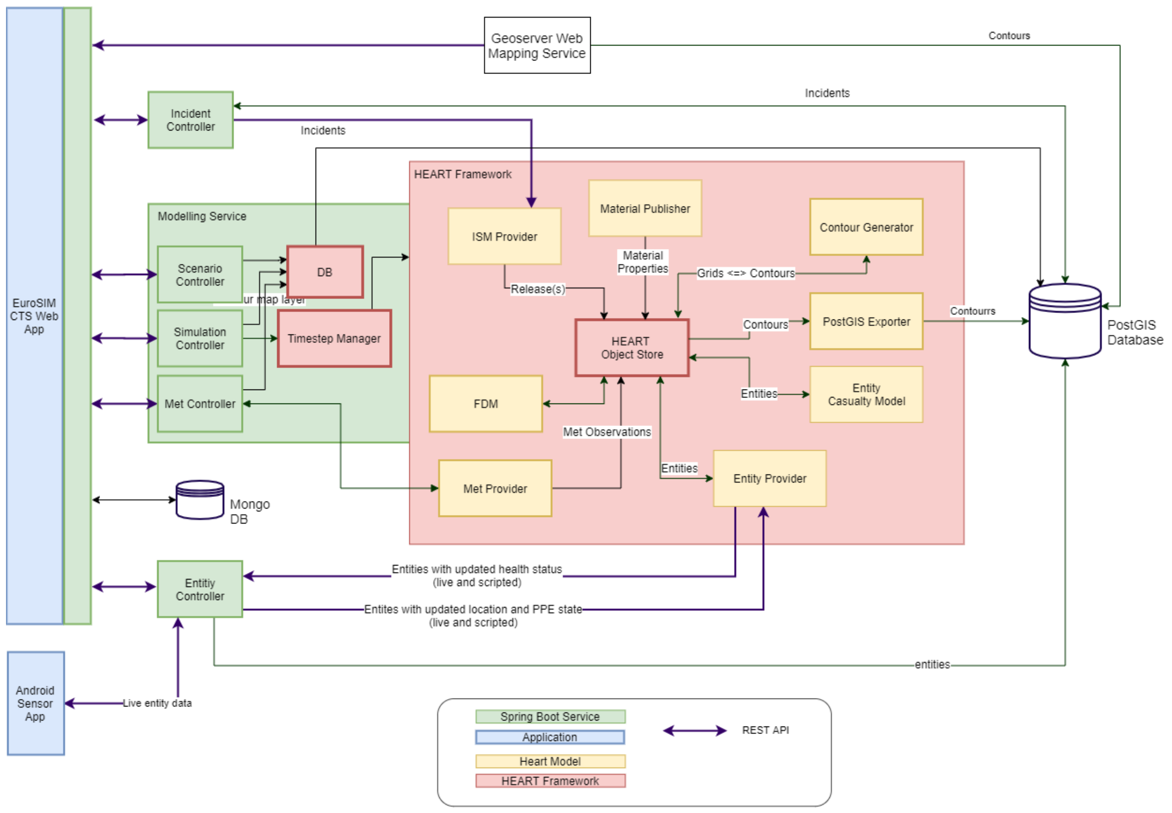


Figure 2: EuroSIM CTS High-Level Design

2.2 Modelling Service

The Modelling Service is a Java Spring Boot Service. It was created to model a simulated dispersion of a CBR incident and provide casualty estimation for entities that are present in the plume location. The core of the service is based upon the Riskware HEART framework, which allows multiple models to be run together in a time-stepped simulation mode.

The HEART framework utilises a time stepped architecture whereby data is exchanged between models via a shared object data store. The architecture means that it is straightforward to integrate additional models, and non-simulation components.

The Modelling Service currently has three REST APIs:

- Scenario API – the Scenario API is used to create and retrieve scenario definitions. Incidents and Meteorology timelines can be added to the scenario. Note, incidents are defined in the Incident Controller; see Section 2.3.
- Simulation API – the Simulation API is used to start and stop a simulation of a specified scenario. Multiple instances of a scenario can be simulated at a time, although for the EuroSIM CTS, only a single simulation is performed at a time.
- Meteorology API – the Meteorology API allows the user to specify a Meteorology Timeline and add observations to the timeline. The Meteorology Timelines may then be added to the scenario via the Scenario API. At present, only simple observations with wind speed and direction at a reference height are available. More complex meteorological modelling may be implemented in future, with the Meteorology API would be moved into a separate service.

2.2.1 Timestep Manager

The HEART framework steps through the time steps of a simulation for the specified simulation time lime – a simulation runs as fast as it can, which varies dependent on the processor speed. For a training simulation providing mock dispersion results, it is required to retrieve the concentration at the current time. Therefore, the dispersion needs to run in real-time, providing external concentration output and health effects at the current time. A Time Step Manager was created to advance the HEART framework in real-time. This requires the dispersion model and results retrieval to be able run at least as fast as real-time.

2.2.2 Dispersion Modelling

Dispersion modelling is currently provided by the Riskaware Flexible Dispersion Model (FDM). In future, this could be provided by Dstl's Urban Dispersion Model [5] with urban topography supplied by the Geographic and Environmental Database Information System (GEDIS).

The configuration of a HEART scenario defines all model inputs in the scenario properties files that are read at the start of the simulation. A desired capability of the system was that the modelling would be able to be modified as the modelling progresses – e.g. adding another release or changing the wind direction. This functionality was added by creation of two extra models and the modification of the FDM model:

- A Meteorology Provider - this HEART model stores the Meteorology Timeline (the list of met observations at time that they were observed) and post it to the object store. The dispersion model retrieves the Meteorology Timeline from the object store and adds it to the dispersion model if they are within the current time step.
- A Release Provider – this HEART model requests incidents from the Incident Controller (see Section 2.3) for the current scenario. Releases for the current time step are then posted to the Object Store. The dispersion model listens for releases being added to the store and adds them to the

dispersion model when notified.

These modifications allow the dispersion model to be dynamically changed whilst running providing a dynamic real-time simulation.

2.2.3 Casualty Modelling

To provide casualty modelling, assessing the dose and health status of all entities (both live trainees and scripted entities), a simple casualty model was developed and integrated into the HEART framework. Additionally, an Entity Provider model was created – this model retrieves all entities from the Entity Controller via its REST API (see next section), and places them on the Object Store. The Casualty Model retrieves the entities from the store, updates their dose and health status and posts the updated entities back to the store. The Entity Provider then posts the updated entities back to the Entity Controller.

The Casualty Model calculates the probability of effects using simple probit calculations for eye effects, incapacitation, and death. A realisation is generated using a lucky number that is generated when the entity is first created. More complicated hazard effects modelling may be modelled in future, including different effects for chemical and biological materials and percutaneous effects.

2.3 Incident Controller

The Incident Controller service is a Spring Boot service that controls incident definitions, storing a library of generated incidents that are available. Users can add incidents and retrieve incidents via the REST API. These incidents can be added to any scenario, once defined, and are stored in a PostgreSQL database.

At present there are two incident types supported:

- A Simple Incident – this incident is a simple continuous release of material. The release has a release diameter, release mass rate and release duration.
- A Container Incident – this incident models a release from a container

2.4 Entity Controller

The Entity Controller service is also a Spring Boot service. It handles the registration and database management of live and scripted entities. Live entities are registered and location updates from these users are sent to the Entity Controller via the CTS Mobile App. Scripted entities created in the Web App are managed and run entirely within the Entity Controller. The updates to entity health from the Modelling Service are also managed by the Entity Controller. All updates to entities in active scenarios on the Web App are automatically forwarded to its own local database.

2.4.1 Scripted Entities

In addition to real-world live ground-based entities, EuroSIM CTS also models the toxic response of scripted entities – these are simulated persons (both red and blue entities) that follow pre-defined routes, and are subject to the same simulated toxic dose as a real entity would, had they followed the same path.

Several options for the scenario definition were investigated to see if they were suitable for scripted entity definition: C-BML (Coalition Battlefield Management Language) [1], MSDL (Modelling and Simulation Definition Language) [2], PLDL (Pattern of Life Definition Language) [3] and C2SIM (Command and Control Systems - Simulation Systems Interoperation) [4]. It was decided that aiming to use PLDL or C2SIM in the long term would provide the required functionality and could provide realistic scripted behaviours. However, for the prototype, a simple bespoke format for scripted entities was implemented in

JSON format.

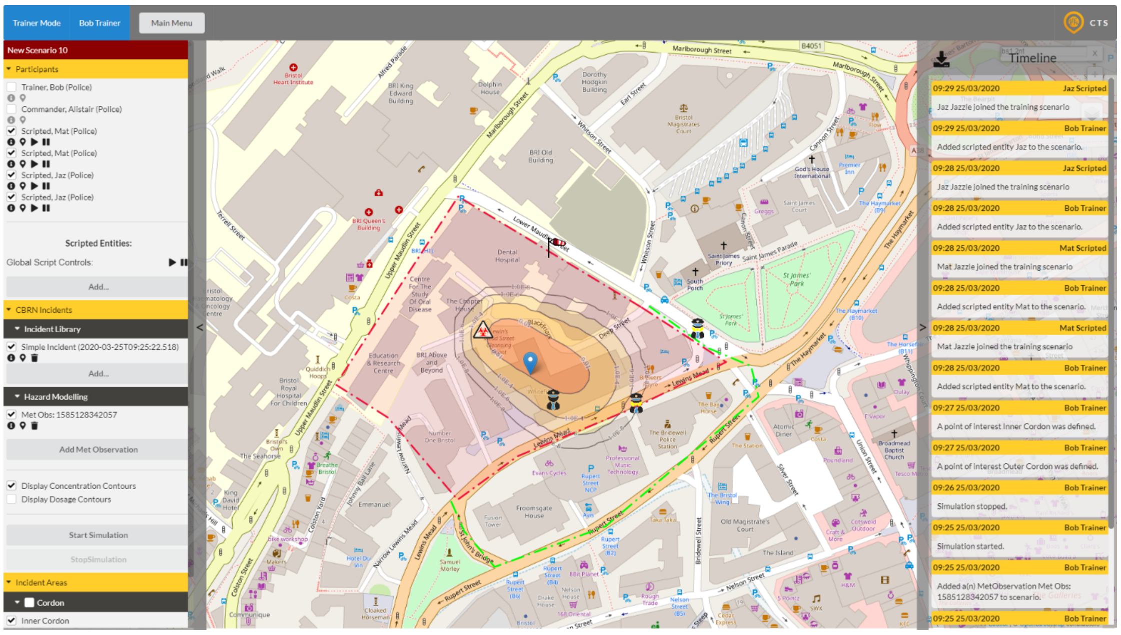
2.5 GeoServer

GeoServer is a WMS (web mapping server) and takes dispersion data from the Modelling Service saved to the PostGIS database, converts this to layers for a map, and serves them up to the Web App. It is a service separate to the Web App in the system, however it is solely used by the Web App, and provides a core part of the Web App's capability to enable visualisation of a scenario.

3.0 USER APPLICATIONS

3.1 CTS Web Application

The Web Application front end is written in React JavaScript using Redux, which gives the system its modern application-style appearance and allows for easy modification to suit user needs. It interacts with the Java back end via a REST API. Redux allows for centralised state management, making the application data consistent and simplifying data management.



New Scenario 10

Participants
- [ ] Trainer, Bob (Police)
- [ ] Commander, Alistair (Police)
- [✓] Scripted, Mat (Police)
- [✓] Scripted, Mat (Police)
- [✓] Scripted, Jaz (Police)
- [✓] Scripted, Jaz (Police)

Scripted Entities:
Global Script Controls:

CBRN Incidents
Incident Library
- [✓] Simple Incident (2020-03-25T09:25:22.518)

Hazard Modelling
- [✓] Met Obs: 1585128342057
- [✓] Display Concentration Contours
- [ ] Display Dosage Contours

Incident Areas
Cordon
- [✓] Inner Cordon

```
09:29 25/03/2020                 Jaz Scripted
Jaz Jazzle joined the training scenario
09:29 25/03/2020                 Bob Trainer
Added scripted entity Jaz to the scenario.
09:28 25/03/2020                 Jaz Scripted
Jaz Jazzle joined the training scenario
09:28 25/03/2020                 Bob Trainer
Added scripted entity Jaz to the scenario.
09:28 25/03/2020                 Mat Scripted
Mat Jazzle joined the training scenario
09:28 25/03/2020                 Bob Trainer
Added scripted entity Mat to the scenario.
09:28 25/03/2020                 Mat Scripted
Mat Jazzle joined the training scenario
09:28 25/03/2020                 Bob Trainer
Added scripted entity Mat to the scenario.
09:27 25/03/2020                 Bob Trainer
A point of interest Inner Cordon was defined.
09:27 25/03/2020                 Bob Trainer
A point of interest Outer Cordon was defined.
09:26 25/03/2020                 Bob Trainer
Simulation stopped.
09:25 25/03/2020                 Bob Trainer
Simulation started.
09:25 25/03/2020                 Bob Trainer
Added a(n) MetObservation Met Obs:
1585128342057 to scenario.
09:25 25/03/2020                 Bob Trainer
```

Figure 3: EuroSIM CTS Web Applications Interface

The Web App back end is a Spring Boot Java service, it has a MongoDB database, which it uses to manage the Scenario. The database is updated by the Web App with information provided by services at runtime following the observer pattern; at start up the Web App posts its address to other services it requires updates from, and this address is used to send updates and so does not require the other services to be preconfigured to the web server's address.

The Web App is used by both the trainer and incident commander to view the running scenario; the trainer has full control of all modelling capability, while the commander is limited to real-world capability, as if viewing a real incident unfolding. The trainer's view can be seen in Figure 3. Note, the concentration contours are not available to the commander.

The Web Application menu contains all the controls used in-scenario. Expandable lists contain participants, incidents, hazard modelling and cordons.

In Trainer Mode (see Figure 4 (a)), the participants list allows users to be invited to the scenario and displayed on the map with the checkbox, access to an information submenu for each participant, creation of scripted entities, and individual or global start and stop of scripted entities. The incident list allows creation and deletion of simple incidents, along with viewing their location, adding them to the scenario, and inspecting their information. The hazard modelling menu allows the same for simple met observations, along with viewing contours and starting and stopping the simulation. Finally, the incident areas list allows addition and deletion of cordons, along with displaying them on the map.

In Commander Mode (see Figure 4 (b)) the participant menu is reduced, allowing only for adding and removing users from scenario, and viewing their locations and information.mThe incident list only allows information and location viewing, and the hazard modelling menu is unavailable. The incident areas list remains the same, since a commander would define these themselves.

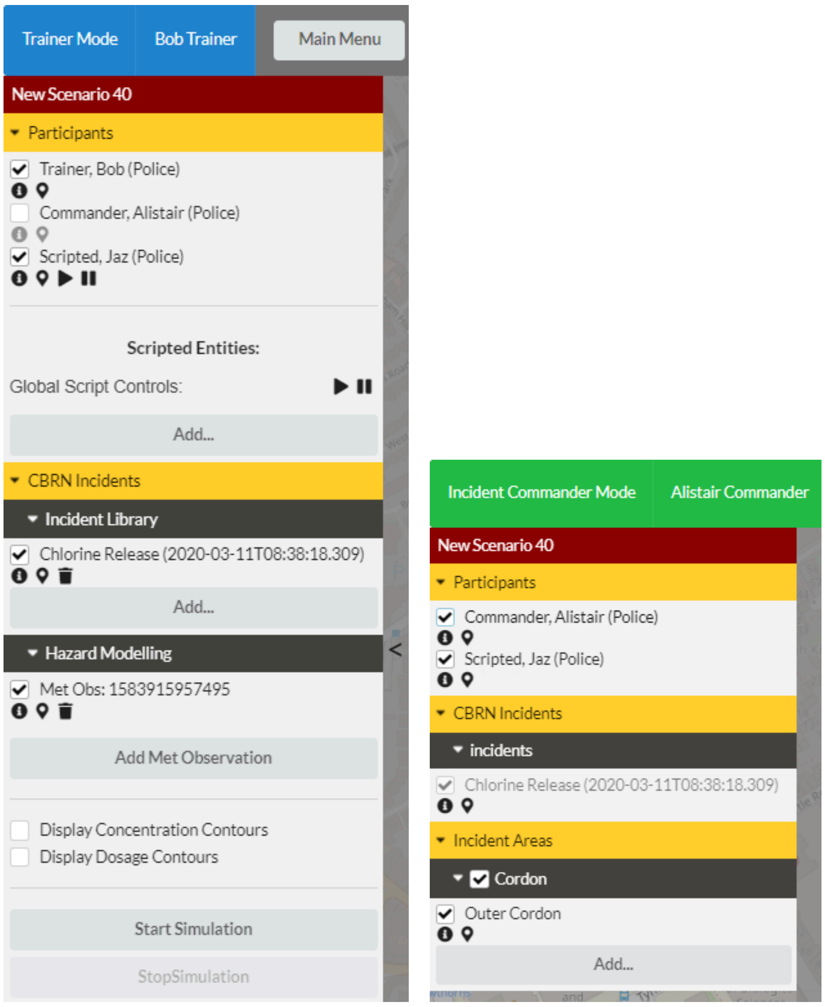
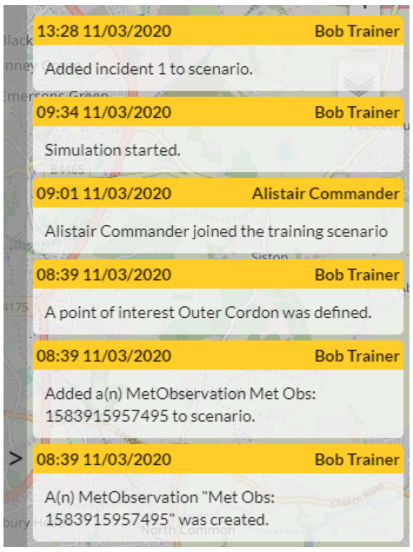


Figure 4: EuroSIM CTS Web App (a) trainer and (b) commander menus

The timeline (Figure 5) displays a log of actions undertaken by users enabling post-scenario review of user performance. The download button saves the timeline locally in CSV format.


13:28 11/03/2020 Bob Trainer
Added incident 1 to scenario.

09:34 11/03/2020 Bob Trainer
Simulation started.

09:01 11/03/2020 Alistair Commander
Alistair Commander joined the training scenario

08:39 11/03/2020 Bob Trainer
A point of interest Outer Cordon was defined.

08:39 11/03/2020 Bob Trainer
Added a(n) MetObservation Met Obs: 1583915957495 to scenario.

08:39 11/03/2020 Bob Trainer
A(n) MetObservation "Met Obs: 1583915957495" was created.

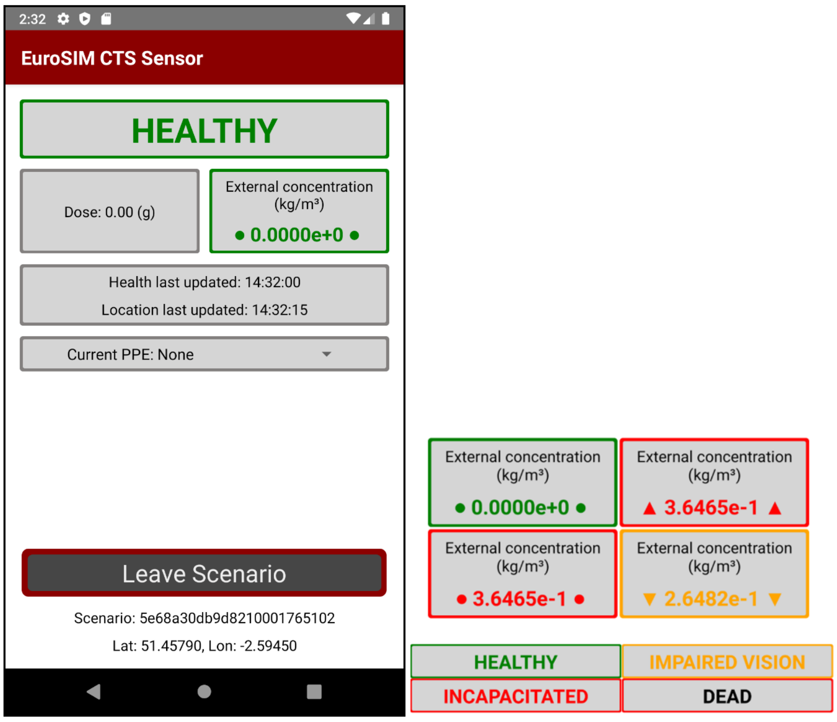
Figure 5: EuroSIM CTS Web App Scenario Timeline

3.2 CTS Mobile Sensor Application

The Mobile Sensor App is a simulated sensor for first responders, it allows registration of a user, or login of an existing user, and joining a scenario. It is written in React Native JavaScript with Redux and can be deployed on both Android and iOS devices (although presently it has only been built and tested on Android). Redux allows centralised state management, meaning all data, whether updated locally by mobile location services or externally in the Entity Controller and retrieved by fetch API calls, can be saved once to the Redux store and immediately passed to update all components within the app.

Data is transferred between the Mobile App and Entity Controller via a REST API on the Entity Controller and fetch API calls on the Mobile App. As such, the Entity Controller has no reliance on mobile devices, and only sends information when requested by the app. This means that all the setup required is configuring the host address and port of the Entity Controller on the Mobile App, simplifying deployment significantly.

After the user has logged into the app, it begins to poll the Entity Controller for scenario invites, and once an invite is received, the user can join the scenario. Once in a scenario, the sensor interface is displayed, which can be seen in Figure 6. The sensor interface displays the simulated user health and dose, along with the external concentration at the user's location.



2:32

EuroSIM CTS Sensor

HEALTHY

Dose: 0.00 (g)

External concentration (kg/m³) ● 0.0000e+0 ●

Health last updated: 14:32:00

Location last updated: 14:32:15

Current PPE: None

Leave Scenario

Scenario: 5e68a30db9d8210001765102

Lat: 51.45790, Lon: -2.59450

External concentration (kg/m³) ● 0.0000e+0 ●

External concentration (kg/m³) ▲ 3.6465e-1 ▲

External concentration (kg/m³) ● 3.6465e-1 ●

External concentration (kg/m³) ▼ 2.6482e-1 ▼

HEALTHY

IMPAIRED VISION

INCAPACITATED

DEAD

Figure 6: EuroSIM CTS Mobile App Simulated Sensor Interface

When in a training scenario, the user’s location is updated periodically and forwarded to the Entity Controller. After the location is successfully updated, the most recent simulated health and environment data are requested. The interface will change colour depending on the simulated danger to the user, health status data display can be seen in Figure 6.

Personal Protective Equipment (PPE) can be set in the dropdown menu. The current dress states were decided after discussion with Dstl. There are four dress states, each with and without a respirator. These will update the entity in and modify the exposure to contaminant in the casualty modelling.

4.0 TECHNOLOGIES

4.1 Spring Boot

The Modelling Service, Entity controller service, Incident Controller Service and Web UI back-end all use Spring Boot to provide a web service application. This runs an embedded Tomcat web server, and each service has a REST API, where URL calls on the service are mapped to controller Java methods.

4.2 React and Redux

The Web UI uses the React JavaScript framework for a modern web front end. This enables rapid development of pre-defined and custom controls. State is managed using a Redux store

4.3 Leaflet

The Web App uses the leading open-source JavaScript map library Leaflet for creating the application map. This can easily link to different source data such as OpenStreetMap and Google Satellite imagery (subject to licensing conditions). It also provides ability to easily add layers to the map for incidents, dispersion plumes, entities, and cordons. The dispersion plume contours can are retrieved from the GeoServer Web Mapping Service.

4.4 PostgreSQL/PostGIS

The Entity Controller, Incident Controller and Modelling Service store data in a PostGIS PostgreSQL database, which contains tables unique to each controller. PostgreSQL is an open-source database management system well suited to being accessed by multiple services at once, as in our application. PostGIS itself extends PostgreSQL by adding GIS (Geographic Information System) features to store geographic data; this is necessary in the case of dispersion modelling, in order to manage concentration and dose values across the affected geographical area.

4.5 MongoDB

The Web App has its own local MongoDB database, which is used to store the training scenarios within the Web UI. MongoDB is a document-oriented database and allows the Web App to simply store whole Java and JSON objects with minimal extra formatting required. The basis for using a separate database was that the EuroSIM CBR prototype was already implemented using this database; the final version would ideally rely entirely on the PostGIS database for all data storage.

5.0 DEPLOYMENT

The system uses Docker for containerised deployment on a server or in the cloud. This simple deployment is a key feature of EuroSIM CTS, vastly reducing the time required to set up a training scenario. The prototype allows for configuration via a Docker-compose YAML file; once configured in this file and uploaded to the server, it can be deployed with a single command.

6.0 USE CASE

As an example of the applicability of EuroSIM CTS in a real-world training situation, the following use case was imagined:

1) The trainer wishes to set up a new training scenario - they log in to the Web App and click to create a new scenario. They specify the training scenario title and location.

2) The trainer adds a simple incident and met observations.

3) The trainer loads scripted entities from local files, they are automatically invited to the training scenario. The scripted entities are stationary until their movement is started by the trainer.

4) The trainer then invites live users – these users will appear in the participants list if the user has logged into their mobile sensor app.

5) Once the training scenario is fully set up, the trainer starts the modelling – all scripted entity movement is initialised, and the dispersion model is started. The system is online, so it is available from any devices with a web browser and internet access to the system host.

6) When the scenario is setup, the incident commander can sign into the Web App and join the created scenario,

7) The commander can add or remove participants if required. They can see all live and scripted entities on the Web App map, along with their external concentration, dose, health status and PPE status.

8) The commander can add cordons to match the real-world scenario. These can be defined in the Web App and communicated with first responders via an external communications method.

9) The commander can also communicate orders to change PPE status via external communication methods.

10) Mobile App users will register at any point, simply connecting to the Entity Controller service via mobile or local network and registering their name and username. Once registered they can await invitation to a scenario from that moment.

11) Once in a scenario, Mobile App users will see their health and dose displayed on their mobile device, along with the simulated external concentration from the "sensor". They can respond to this data accordingly as if it represented a real threat. Users further have the ability to set their PPE in the app, which in turn will affect how their dose and health transform with time.

12) During scenario runtime, the trainer can add or remove users, define new incidents or remove existing ones, and add and remove met observations. This dynamic functionality can be used to test the commander and first responders' responses to these threats and changes.

Any change to the scenario or users is logged, and these logs can be downloaded directly from the Web App as a CSV file, for review later. The modelling can also be restarted by the trainer at any point, allowing for the scenario to be restarted with no extra setup.

7.0 CONCLUSIONS

The EuroSIM CTS prototype demonstrates the basis for a CBR response training system using up-to-date and easy to deploy technologies. It has scope to be developed further with the option of integration into full Information Management systems to provide training capability.

Possible next steps for the project have been identified below.

- Enhance the modelling – integrate the Urban Dispersion Model (UDM) [5] within the modelling service and extend it to model different incident types.
- Enhance Mobile App – Add a map display to allow users to see their location, and add commands pushed from the commander to instruct the user where to go and what actions to take (including a destination on the map).
- Web Application – Add the ability for grater control over simulation output, allowing the trainer to rewind and replay the simulation. Allow the commander to push commands to the mobile app users. The Web app storage database should also be changed to the PostGIS database used for the modelling data.
- Scripted entities – Implement standards for scripted entity behaviour – either PLDL or C2SIM – for more realistic and stochastic behaviour.
- Integration with Augmented Reality headset – Allow users to see simulated sensor reading in real-time.

8.0 REFERENCES

[1] Simulation Interoperability Standards Organization (SISO), "Standard for Military Scenario Definition Language, SISO-STD-007-2008," May 11, 2015.

[2] Simulation Interoperability Standards Organization (SISO), "Standard for Coalition Battle Management Language (C-BML) Phase 1, SISO-STD-011-2014," April 14, 2014.

[3] Simulation Interoperability Standards Organization (SISO), "Standard for Command and Control Systems - Simulation Systems Interoperation, SISO-STD-019-2020," 25 April 2020.

[4] A. Easton and R. Ranawana, "Toward Megacity Simulation: A Proposed Pattern-of-Life Definition Standard," I/ITSEC 2018.

[5] D. J. Hall, A. M. Spanton, M. Hargrave, S. Walker, I. H. Griffiths and D. R. Brook, "The Urban Dispersion Model (UDM): Version 2.9," Defence Science and Technology Laboratory (Dstl), 3 August 2004.