# An interoperability Framework for Trials and Exercises

Martijn Hendriks[2], Erik Vullings[1], Steven van Campen[2], Pieter Hameete[1]
[1]TNO, The Hague, The Netherlands
[2]XVR Simulation, Delft, The Netherlands

**Abstract** —In the Crisis Management (CM) domain, there is a need for quickly setting up a trial or exercise to test or train new or existing CM solutions and procedures. The EU-funded DRIVER+ project developed an open source cloud-based simulation framework, which provides a quick and easy way to connect simulators and (C2-like) solutions in such a way that they can efficiently exchange information between each other, and between simulators and solutions. It has already been used successfully in three CM trials, and in several Military battle labs proof-of-concepts.

## 1 Introduction and Background

DRIVER+ [1] is an EU-funded project that develops a pan-European test-bed for crisis management (CM), offering CM experts a Concept Development & Experimentation (CD&E) environment for testing new CM solutions and/or processes. Its main outcomes are a practical methodology to trial solutions, the so-called Trial Guidance Methodology, a portfolio of solutions (PoS), and an open source interoperability framework, which will be the focus of this paper.

## 2 Technical Approach and Methods

A trial typically starts with a CM organisation having a problem: with an incident such as earthquakes, flooding, or forest fires, inadequate situational awareness, or procedural aspects like resource management. To address the problem, the DRIVER+ Portfolio of Solutions [2], basically a smart catalogue of C2-like solutions and processes, can be searched, and solutions claiming to solve his problem identified. Still, they need to select one or more solutions that fits best within their organisation/region, so a trial is devised using the Trial Guidance Methodology [3].

An important part of any trial is to setup the technical infrastructure: on the one hand, existing C2 systems and new solutions need to be connected in such a way that they can exchange information via the so-called Common Information Space (CIS), see Fig. ***1***. On the other hand, simulators are needed to create a fictitious incident, so the solutions can be tested and experienced in a virtual, albeit sufficiently realistic, background. As one simulator is often not enough to cover all aspects of the fictitious incident, there also needs to be a space where simulators can exchange information with each other: The Common Simulation Space (CSS). In the CSS, a flooding simulator can focus on simulating a flooding, and a traffic simulator makes sure that the vehicles do not drive through the flooded area. Finally, there is the need for a gateway connecting the CIS and CSS, so information about the simulated fictitious incident can be offered to the CIS, and optionally filtered. For example, a flooding simulator can simulate a flooding, but depending on the level of situational awareness, the C2 system may only 'see' a part of the actual flooding. The gateway also offers solutions to request changes to the simulation, e.g. a C2 system dispatching a request for a simulated ambulance to move to a location.

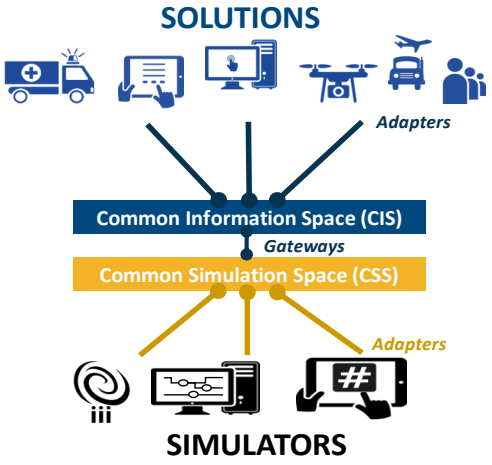


**Fig. 1.** The CIS connects solutions, the CSS connects simulators, and information is exchanged via gateways.

## 3 Technical Requirements

In order to fully comply to the demands and wishes of such a technical infrastructure to be deployed in the CM domain, several technical requirements were defined to steer the implementation of this framework (more detailed in [4] and [5]):

- The primary focus of the framework and its tools should be the CM operations, protocols and software applications. Some data exchange standards do exist, such as the Common Alerting Protocol (CAP) [6] and the Emergency Data Exchange Language (EDXL) [7], but they do not offer a complete end-to-end solution, and are sporadically implemented at best.
- The framework must be able to connect (CM) simulators to create realistic fictitious incidents, and should provide a gateway to exchange data to and from solutions and simulators.
- For both solution and simulation providers, the framework must be understandable and provide easy connectivity for their applications.
- The framework should contain tools focused on the support for running trials and exercises. The need to monitor and control the trial during execution and the need to save all exchanged data for after-action review and analysis needs to be addressed.
- For sustainability reasons, all code of the framework and its tools should be freely available and open-source (MIT license).

## 4 Technical Design

The Common Simulation Space could be implemented using one of the already available standards like High-Level Architecture (HLA) [8], which is well-equipped to solve the problem of information exchange in a simulation. However, within DRIVER+, another approach was taken, since:

- HLA tooling is expensive for CM organisations, there is no real open source alternative, and the community of practitioners is small.
- HLA tooling typically only offers code generators for Java, C++ and C#, which excludes many simulators.
- HLA requires a steep learning curve, and although some work has been done in creating a CM Federation Object Model (FOM), no mature standards are available.
- The CM domain has a slightly different set of requirements and communication standards than the domain HLA is currently predominantly used in.
- Few, if any, CM simulators currently support HLA.

The CSS as well as the CIS, are based on the open source streaming platform, Apache Kafka [9], which provides the following benefits:

- It is open-source, widely used, and supported by the Apache organisation, and can run easily in the cloud using Docker.
- Adapters to connect to Kafka are available in many programming languages, and in DRIVER+, existing connectors in Java, C#, JavaScript, Python and REST were adapted to provide additional Modelling & Simulation functionality.
- Exchange of messages is based on AVRO schemas [10], providing precise information on the syntax of each message.
- Message throughput is high [11] and, using clustering, can be made even higher.

For combining CM simulators and HLA RPR-FOM based military simulators, a two-way gateway in Java was created for the real-time exchange of RPR-FOM entity state messages (name, marking and position of entities) with the CSS as AVRO messages, and vice versa.


**Fig. 2.** The Common Simulation Space can partially be implemented using Apache Kafka, partially using HLA RPR-FOM.

The chosen implementation also has some limitations that in particular developers need to be aware of:

- Time synchronization between simulators and solutions is based on a simple time-driven implementation. Time messages are sent across a Kafka topic, and there is no concept of event handling or waiting for a lagging system. This is acceptable for typical CM simulations and exercises which are running not much faster than real-time, but not suitable if you intend to use monte-carlo simulations. Similarly, our framework is not suitable for high-accuracy simulations, e.g. as needed in the domain of air-defence simulation.
- Ownership: There is no concept of ownership, i.e. one simulator taking over the ownership of an entity created by another simulator.

## 5 Additional Tooling

Exchanging information is only part of the framework's functionality: DRIVER+ also offers tools for observers, after-action review, scenario management, a time service, data services, such as cloud storage or a twitter gateway, and debug tools, such as the message injector and replay service. All these tools are available as Docker containers and can easily be combined in a Docker environment, and hosted in the cloud via container orchestration tools. (see Figure 3).

DRIVER+ also created a Moodle-based e-learning module to educate CM professionals as well as software developers and system administrators on what it takes to setup a trial and to connect their solution or simulator to the interoperability framework.

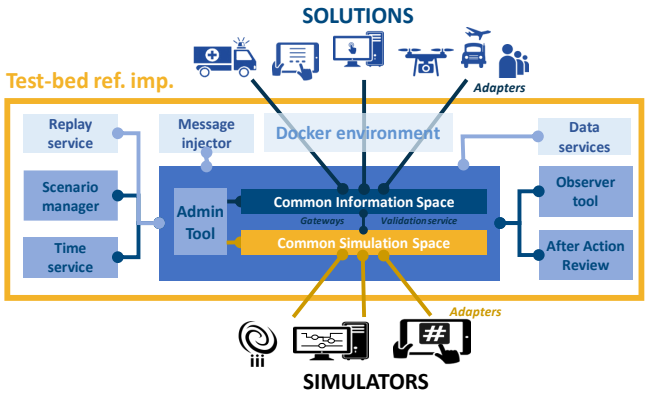


**Fig. 3.** Complete overview of the DRIVER+ interoperability framework. An animation is available at [12].

## 6 Lessons Learned & Conclusions

Based on experience gained so far in several trials:

- The framework enriches the user experience: for example, a 3D first-person simulator can now be connected to a C2 platform or an ambulance dispatcher.
- It allows to connect previously disconnected operational systems, e.g. the Dutch Defence C2 platform ELIAS was connected to the Dutch C2 CM platform, LCMS
- Creating an adapter is straightforward: using one of the existing open source Kafka connectors as the basis, a limited set of functionalities needs to be added, such as a heartbeat signal, processing time messages, and security.
- It is easy to connect solutions: it involves converting AVRO-based messages to and from internal messages in your solution.

- For simulators, messages need to be converted to AVRO, but in their case, time management is much more of an issue. Also, they often need to be more reactive to external requests.

## Acknowledgements

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement n° 607798. The information and views set out in this publication are those of the author(s) and do not necessarily reflect the official opinion of the European Union.

## References

[1] Project DRIVER+, https://www.driver-project.eu.
[2] DRIVER+ Portfolio of Solutions, http://pos.driver-project.eu/
[3] DRIVER+ Trial Guidance Methodology, https://www.driver-project.eu/trial-guidance-methodology/
[4] DRIVER+ Test-bed specification, https://driver-eu.gitbook.io/test-bed-specification
[5] DRIVER+ Test-bed design, https://driver-eu.github.io/test-bed-design
[6] Common Alerting Protocol (CAP), http://docs.oasis-open.org/emergency/cap/v1.2/CAP-v1.2-os.html
[7] Emergency Data Exchange Language (EDXL), http://docs.oasis-open.org/emergency/edxl-de/v2.0/edxl-de-v2.0.html
[8] HLA (High-Level Architecture) simulation standard, https://en.wikipedia.org/wiki/High-level_architecture
[9] Apache Kafka, https://kafka.apache.org.
[10] AVRO standard, http://avro.apache.org.
[11] J. Kreps - "Benchmarking Apache Kafka", 2014, https://engineering.linkedin.com/kafka/benchmarking-apache-kafka-2-million-writes-second-three-cheap-machines
[12] DRIVER+ Test-bed - how can the Test-bed help Crisis Management practitioners?, https://www.youtube.com/watch?time_continue=1&v=si0YEQKNCkM

## Biographies

**Steven van Campen** is a senior product designer and medical incident command instructor at XVR Simulation. He holds an MSc. in Aerospace Engineering and Industrial Design Engineering from Delft University of Technology (NL). He has interests in education and crisis management, and involved in creation of simulation centres and innovations in incident command training.
**Pieter Hameete** is a Junior Scientist Innovator at TNO's Modelling, Simulation & Gaming department. He holds an MSc. In Computer Science from Delft University of Technology (NL). His work involves distributed (simulation) systems, and data-intensive applications.
**Martijn Hendriks** is a senior developer at XVR Simulation. He holds an MSc. in Media & Knowledge Engineering from Delft University of Technology (NL). His main focus lies in integrating external applications within the XVR platform architecture.
**Erik Vullings** is a Senior System Integrator at TNO's Modelling, Simulation & Gaming department. He previously worked as an R&D programme manager in Australia and as systems engineer for Philips. He holds a PhD. in Electrical Eng. and an MSc. in Mechanical Eng. from Delft University of Technology (NL).