PRELIMINARY VERSION 22-Feb 2011

Development of a Distributed Simulation Environment for Crisis Management Training

Erik Vullings - TNO, The Netherlands
Simone De Kleermaeker – Deltares, The Netherlands
Cor-Jan Vermeulen – HKV, The Netherlands

Erik.Vullings@tno.nl, Simone.DeKleermaeker@deltares.nl, CorJan.Vermeulen@hkv.nl

Keywords:
Team training, exercise management, simulation engine, scenario editor, distributed environment, scenario editor, public safety and security, flood control

ABSTRACT: *Disasters happen; you simply cannot prevent them. So you need to be adequately trained for managing a crisis when it occurs. Most crisis management professionals, however, do not have sufficient time to train regularly, as they often have demanding regular jobs too. In addition, most current-day trainings are inflexible and only employ linear scenario lines, i.e. you have to follow the stipulated path as presented by the trainer, and there is little room for divergence. Every time you do not follow the "yellow brick road", you are corrected and put on the right path again. Furthermore, these scenarios take a lot of time and manpower to develop, and the tooling is not great. To make matters worse, you often need many people to simulate the external environment, feeding simulated email and telephone messages to the trainees.*

In order to train efficiently and effectively, we present a solution to train crisis management professionals in a distributed environment. The solution was developed as part of the Flood Control 2015 program in a consortium of four parties: IBM, HKV, Deltares and TNO. It uses Windows Workflow Foundation for creating training scenarios in a user-friendly environment, in combination with an HLA-like Service Bus for sending control and data messages to distributed software modules. The environment was used to train crisis management professionals in safety regions and for water management. This paper will describe the developed environment, the training setup and the results of the training.

1 Introduction

Training a group of persons to act as a team during a crisis situation is a big challenge. On the one hand, you always need to bring the whole team together, and on the other hand, you need to simulate the environment as well. Compare this, for example, to training a soccer team. They can train every day, and have matches regularly. And if they want to have a match, they just need to divide the team up. Also, the environment is fixed, and needs no further preparation.
Training a crisis team might require many trainers: for example, during a recent training in a small city, about 40 persons of the city council were trained by 12 trainers. 2 to observe the two teams, and 10 to act as external stimuli. Dedicated exercise management tools exist, such as JEMM, developed by NATO's NC3A, or the commercial EXONAUT. Typically, they still require several trainers, but the scenario can be better tuned to the learning goals: during the scenario, at predefined time intervals, they instruct the trainers about what to do or how to act. An advantage of this kind of training is the high level of realism that can be attained, since we are dealing with actual people. However, the costs are high too, and this often means that people are not trained very often.
The approach we present here strives to achieve a sufficient level of realism at a low cost, with only one trainer who acts as an observer. In Section 2, we present the training framework. In particular, we will focus on the interoperability requirements for the different components, and comparison to standard frameworks like HLA and DIS. In Section 3, we describe a training session with three teams, and the results achieved. Section 4 will present some preliminary conclusions and future work.

2 Framework for Urban Safety & Security (FUSS)

Our Framework for Urban Safety & Security (FUSS) is based on the distributed High Level Architecture (HLA, see ref. [1]). The main reason why we did not use HLA was that it had proven difficult to connect non-C++ clients written in programming languages like C# or Delphi. And although current HLA implementations may have resolved these limitations, our current framework excels in simplicity and speed, and any competent programmer can apply it in less than an hour.

2.1 Overview of the technical environment

Figure 1 depicts the exercise setup: at the top are the three teams: LCO a.k.a. *Landelijke Coördi-natiecommissie Overstromingsdreiging* (National

Commission Flooding), and two operational teams, WOT, a.k.a. *Waterschappen Operational Team* (Water Board Coordination Commission) and ROT, a.k.a. *Regionaal Operationeel Team* (Regional Coordination Commission), each with several professionals. These teams all have email, and optionally a GIS module and an evacuation model at their disposal. At the left, you see two shared tools, i.e. the simulation time and a virtual news channel. At the bottom, you see the Facilitator, who has access to the log, including all sent and received emails, the evaluation module and the Scenario Editor and Exercise Manager (SEEM).

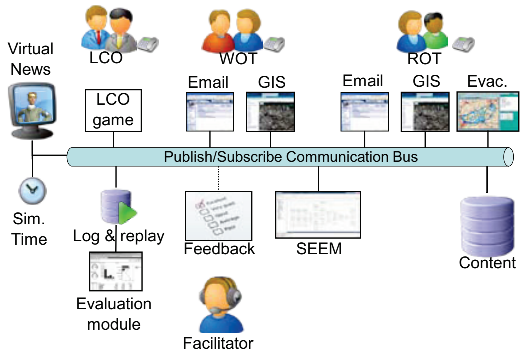
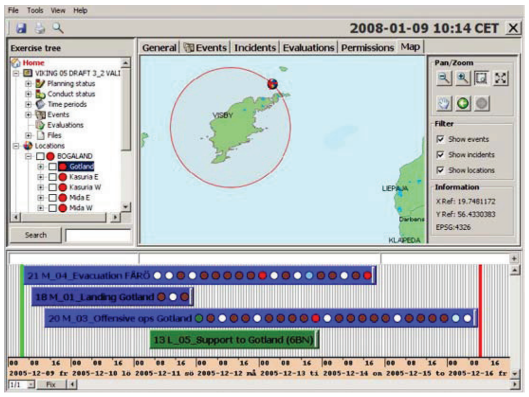


Figure 1. Training setup, with the three teams at the top and the facilitator at the bottom.

We briefly discuss two alternative exercise managers: the commercial Exonaut (see below, www.exonaut.com**),** which has an online version as well, and the freely available Joint Exercise Management Module (JEMM, free for NATO countries). Both are mainly aimed at briefing the role players during an exercise, and not the players, so from a training perspective, the training will be very labor intensive and therefore expensive. The presented solution, however, could also facilitate this, but as this is precisely the aspect we want to tackle (efficiency)

Exonaut
Exonaut Exercise Manager is a Java-based exercise management software that allows directing staff to conduct distributed and simultaneous joint planning, irrespective of geographical location. Through a comprehensive timeline and map function, the software visualizes the exercise scenario in detail, allowing the directing staff to obtain a comprehensive overview of the exercise as it is being planned, executed and evaluated.
Exonaut stores exercise scenarios in a database, thereby allowing recycling and continuous improvement of exercise scenarios while saving hours in exercise planning. Comprehensive evaluation data allows exercise planners to evaluate and reuse successful aspects of an exercise scenario, while the training audience can receive instant feedback on its performance. Exonaut also allows for dynamic scripting, thereby allowing directing staff to increase the realism and efficiency of the exercise in the execution phase.
Exonaut has several filtering functions, thus ensuring that different functions within the directing staff can focus on information relevant to their tasks. As a result, coordination of complex scenarios involving a number of functions and training objectives is achieved.

Exonaut's open interface makes it easily integrated with other systems in the Exonaut suite; Calendar, Intelligence, Training Progression Matrix as well as external systems, thereby allowing Exonaut to be used in a range of areas such as Live Exercise Management, Virtual Exercise Management and Constructive Simulation Exercise Management. Exonaut has been used in a number of complex computer assisted exercises involving a large directing staff and integration with constructive simulators, most recently the Viking 2011 exercise at the Swedish Command & Control regiment and distributed sites.



2.2 The communication bus

The communication service bus is a command line executable (see also [2]), which connects publishers and subscribers in a distributed environment. This means that a client that wishes to publish information, the publisher, needs to connect to the service bus, join a federation, and create a channel. From then on, it can send messages across. Although we chose to use XML-messages, arbitrary byte messages can be used too.

```
imbClient = new IMBClient();
imbClient.InitializeConnection();
imbClient.Publish("TestChannel",
   new XmlSerializer(typeof(TestMessage)));
```

A client that wishes to receive messages, the subscriber, follows a similar procedure, but instead of publishing a channel, it subscribes to one, and listens to incoming events.

```
imbClient = new IMBClient();
imbClient.InitializeConnection();
imbClient.IncomingEventObject += IncomingEventObject;
imbClient.Subscribe("TestChannel",
   new XmlSerializer(typeof(TestMessage)));
…
private static void IncomingEventObject(string eventName,
  object pObject) {
  if (pObject is TestMessage)
     DisplayMessage((TestMessage)pObject);
}
```

For interoperability purposes, although not required by the service bus, we chose to send XML messages across and publisher and subscriber only need to format and parse those messages, respectively.

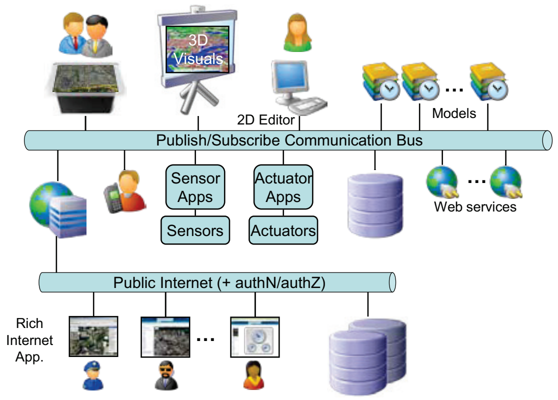


Figure 2. Overall system architecture showing different connection options.

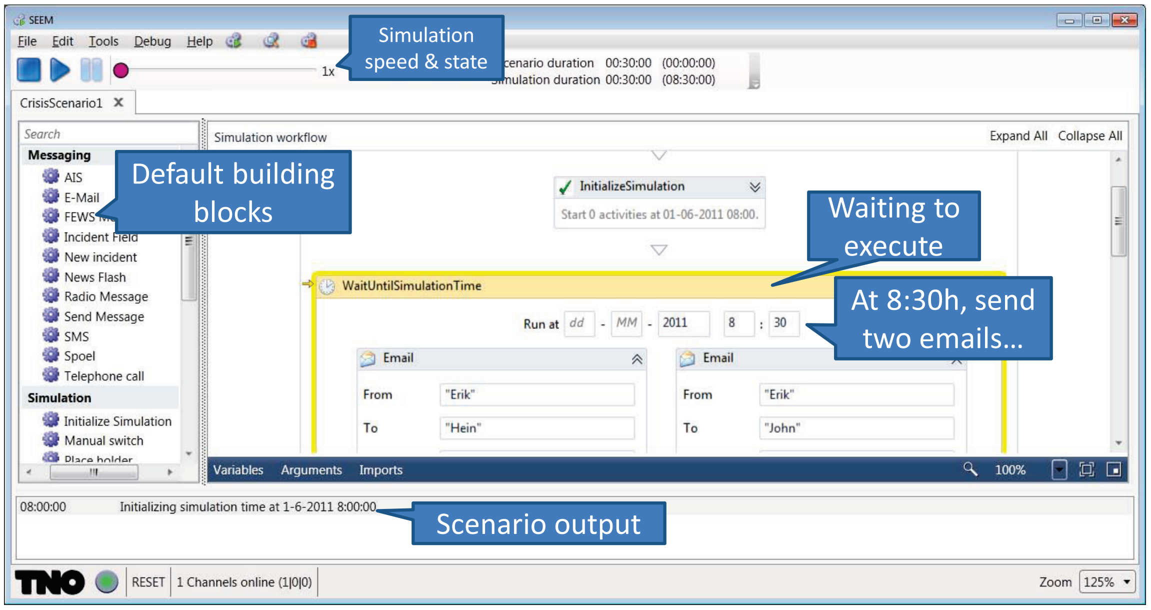
2.3 The Scenario Editor & Exercise Engine (SEEM)

At the heart of our framework is the Scenario Editor and Exercise Engine (SEEM). It allows end users (typically the trainer) to create team-training scenarios by dragging-and-dropping basic building blocks into the scenario. Examples of the building blocks are email messages, SMS messages or news bulletins, which are presented by a virtual character, but also blocks for managing the timeline and flow (branching). With these building blocks, the end user (trainer) can create a training scenario from scratch, sending different messages at different times or under specific conditions.
SEEM is built on top of Windows Workflow Foundation 4 (WF4, see [3]), which provides the drag-and-drop user experience, and has default building blocks to implement simple logic as you would typically define in a flowchart or if/then/else branching.

Using the WF4 bookmarking system, we could easily extend it and make it aware of the simulation time, so it not only allows us to edit a scenario, but also to execute it. Consequently, the simulation time is wrapped inside a message, and communicated across the bus to all clients, so they are aware of the simulation time and state (running, pause, stop) too.

2.4 Other components

The training environment is further enriched with several other components, which we will describe briefly.



Scenario duration 00:30:00 (00:00:00)
Simulation duration 00:30:00 (08:30:00)

Start 0 activities at 01-06-2011 08:00.

Run at dd - MM - 2011 8 : 30

From	"Erik"
To	"Hein"

From	"Erik"
To	"John"

```
08:00:00        Initializing simulation time at 1-6-2011 8:00:00
```

1 Channels online (1|0|0)

Figure 3. Detailed view of SEEM, where a running scenario is depicted. At the left, you see a partial list of building blocks. At the top, you see the exercise control (start, stop, pause, and speed), and at the bottom, the scenario output.

Email
To offer email to trainees, we combined a freeware email server (SmarterMail, www.smartertools.com) with integrated web clients with a small SMTP/IMAP service that is connected to the bus.
The email server is configured with a training domain and accounts of the trainees. In addition, the email server can be configured to add a BCC (Blind Carbon Copy) to each email, so the trainer/facilitator can inspect the messages during the training. In addition, we use this feature for logging, as explained below.
The SMTP service listens to email messages on the bus. Upon receiving an email message, it uses the SMTP protocol to send it via the email server to the trainees, who can read it in their web client.
The IMAP listens to a specific logging account, filled with BCC messages (as explained under bullet 1). Every message that is received is published on the bus for logging purposes.

Logging
The logging is done for evaluation and analysis purposes. All messages and actions are logged for later analyses. In addition, it filters out certain types of messages for automatic evaluation (see evaluation module). The logging is built on top of the GRACE database (see also [4])

Virtual news reader
Often when performing training, a video with the news is shown. Since the news needs to be a good representation of the scenario, these newscasts are done with real anchor women. Unfortunately, this also makes it hard to change the scenario and re-use them, and instead, we opted to have a virtual character read the news aloud, prompted by the scenario.

Evacuation model
Based on the current (simulation) time and situation, the status of the road density was shown. Using the map, trainees could close roads and thereby influence the (potential) evacuation operation.

External game
Since one of the teams in the training originated from a newly erected organisation, they used another serious game as the main interface to play the scenario.

GIS (Geographical Information System)
Map interface to show incidents and discover new information.

Simulation clock
Shows the relative time to the trainees, since we played faster than real-time (one hour of actual time represented one day in the scenario).

Evaluation module
Using a commercial Business Intelligence product, it evaluates and scores the logged data along the OODA (Observe, Orient, Decide, and Act) methodology. The scoring is based on predefined criteria formulated in the workflow procedures of the trainees.

NTP service
The Network Time Protocol service listens to the simulation time, and updates the PC's system time, and acts as an NTP service for NTP clients.

3 Early Experiment

Our first experiment was conducted mid November 2010 as part of the project Flood Control 2015 – Serious Gaming (see www.floodcontrol2015.com). There were three teams, from three different organisations, each team with approximately 4 persons. The training took 3 hours (excluding briefing, breaks, and debriefing) and the scenario comprised three days of a crisis (flooding scenario). Each team was provided with their own interface, typically an email web client Joint Exercise Management Module, and specific tools, such as the GIS map or evacuation model. Since they were co-located, each team could watch the news and simulation clock on a large screen. Directly after the experiment the performance report for each team was available.

4 Early Conclusions and Road Ahead

One of our main goals with the developed environment was to improve training efficiency and effectiveness. From the early evaluation, we conclude that this training environment is much more efficient. All you need is one facilitator to start and manage the exercise, and optionally, observers to observe each team. Effectiveness, on the other hand, is much harder to prove. From the feedback that we've received from the participants, we know that our setup was similar to the one they had experienced before, but which required many more people to run. So presumably, since the experience was quite similar, the potential for learning is similar too. New addition was the instant

generated performance report for direct feedback and for the after action review.
We noticed that the familiarity with the provided tooling was an issue and in the future, we need to reserve more time for introducing them properly. For example, one team experienced some difficulty in using a regular web client for email. During the preparation, when we mentioned that there would be a GIS/map tool available, they immediately felt the need to include someone else in their team, one who would be familiar with using a map.
Due to development time constraints, we did not develop a telephone service, so the players could not use the phone. This was something that they really missed, and we are currently developing such a service.
It's also worth mentioning that the news presence, although appreciated, was not of primary importance. There were even players that were slightly annoyed by its presence, as it disrupted the meeting.
Finally, it's important to mention that most time was spent in setting up the room, including all equipment. We are currently investigating whether it is economically feasible to host all services, so there is no more the need to bring in equipment.

References

[1] IEEE: "IEEE 1516, High Level Architecture (HLA)", www.ieee.org, March 2001.

[2] The Reality Check: Evacuation Planning done by Mixed Reality and Simulation, D. Keus et.al., SIW 2011, 11E-SIW-009

[3] A Developer's Introduction to Windows Workflow Foundation (WF) in .NET 4, M. Milner, November 2009, http://msdn.microsoft.com/en-us/library/ee342461.aspx

[4] Generic Reconstruction and Analysis for Simulations or Live Exercises, R. Witberg, E. van Veldhoven, D. Keus, SIW 2011

Author Biographies

ERIK VULLINGS received his Masters in Mechanical Engineering and his PhD in Electrical Engineering from Delft University of Technology. In 1999, he joined Philips as a Systems Architect, and finally became Program Manager of an Integrated Project in the area of Mechatronics (FP6). In 2004, he went to Australia as the Program Manager of an IT project in the field of Identity and Access Management, thereby laying the foundation for the Australian Access Federation. In 2007, he returned to The Netherlands, and currently works as Senior Project Manager at the Dutch Research Organisation TNO. He is the author of multiple articles and papers, and holds several patents.

SIMONE DE KLEERMAEKER has studied Applied Mathematics at the University of Groningen and received her masters in 2001. She then joined WL|Delft Hydraulics and currently works as senior advisor for Marine and Coastal Systems at Deltares. She has extended knowledge of hydrodynamic modelling of both Marine and Coastal as well as Industrial systems. She complements this knowledge with the application of such models in projects such as Serious Gaming, FEWS installations and calibration using tools such as OpenDA. She has extended experience as teacher and trainer.

COR-JAN VERMEULEN graduated in 1987 in Applied Mathematics (systems and control theory) from the University of Twente. After graduating he joined WL|Delft Hydraulics as a project engineer mathematical modelling. He participated in long and short term assignments as mathematical modelling specialist, researcher and lecturer/instructor. Since 2005 he specialized in disaster management with special attention to water safety, i.e. prepare for water related incidents like floods, and draughts. He specialised in operational water management and disaster management and has been a consultant with over 20 years of experience.