Easy-to-Use, Web-Based Graphical User Interface for Controlling Entities in Constructive Simulations

Per-Idar Evensen, Kristian Selvaag, Dan Helge Bentsen,
Helene Rødal Holhjem, Håvard Stien
Norwegian Defence Research Establishment (FFI)
Instituttveien 20
NO-2027, Kjeller
NORWAY

per-idar.evensen@ffi.no, kristian.selvaag@ffi.no, dan-helge.bentsen@ffi.no,
helene-rodal.holhjem@ffi.no, havard.stien@ffi.no

ABSTRACT

At the Norwegian Defence Research Establishment (FFI) we investigate how to increase combat effectiveness in land force operations. As part of this work we need to conduct detailed, entity-level simulations of battalion to brigade level operations, to assess and compare the performance of different land force structures and operational concepts.

For our use, traditional constructive simulation systems often do not have the required level of resolution, are too complex and cumbersome to use, or are not flexible enough with respect to representation of new technologies. We are therefore developing webSAF, an easy-to-use, web-based graphical user interface (GUI) for controlling semi-automated entities in constructive simulations. So far we have developed functionality for controlling indirect fire and manoeuvre entities simulated in Virtual Battlespace (VBS) and air defence entities simulated in VR-Forces. Since we conduct simulations for experimentation and analysis purposes (and not procedure training), the system has been designed to only require a minimum amount of input from the operators. Furthermore, our simulations need to be conducted with a minimum number of operators on each side.

In this paper we describe the overall design and implementation of the GUI system, as well as the experiences from the initial experiments with the system

1.0 INTRODUCTION

At the Norwegian Defence Research Establishment (FFI) we investigate how to increase combat effectiveness in land force operations. As part of this work we need to conduct detailed, entity-level simulations of battalion to brigade level operations, to assess and compare the performance of different land force structures and operational concepts.

For our use, traditional constructive simulation systems often do not have the required level of resolution, are too complex and cumbersome to use, or are not flexible enough with respect to representation of new technologies (e.g. new sensor systems, weapon systems, and protection systems). We are therefore developing webSAF, an easy-to-use, web-based graphical user interface (GUI) system for controlling semi-automated entities in constructive simulations [1]. So far we have developed functionality for controlling indirect fire and manoeuvre entities simulated in Virtual Battlespace (VBS) and air defence entities simulated in VR-Forces. In the future we plan to extend the system with functionality for controlling combat service support entities simulated in VBS, and air entities simulated in VR-Forces.

Since we conduct simulations for experimentation and analysis purposes, and not command and staff

training, the system has been designed to only require a minimum amount of input from the operators. It is a goal that military officers should be able to control the entities with minimal instruction. In addition, the simulations should be conducted with a minimum number of operators on each side.

First, in this paper, we give a brief description of the background for this work. Secondly, we describe the intended use of the simulation system, including a list of requirements. Thirdly, we give an overall description of the simulation system we are composing and its components. After this, we describe the design and implementation of the web-based GUI system in more detail. Finally, we summarize the initial experiences with the system, and outline our plans for further work.

2.0 BACKGROUND

The first time constructive simulation with semi-automated forces (SAF) was used to support analysis of land force structures at FFI, was in 2010. In project "Future Land Forces" the performance of five fundamentally different land force structures were evaluated through a series of simulation experiments [2][3]. The goal was to rank these structures based on their relative performance. In addition, the experiments revealed several strengths and weaknesses of the evaluated structures. The lightweight simulation tool mōsbē from BreakAway was used in the experiments. The main reasons for this choice were that mōsbē supports simulation of brigade level operations and has a user interface that makes it easy to control large groups of entities. The experiments were conducted as simulation-supported wargames, where military officers planned and controlled the operations, and the simulation tool kept track of the movement of units and calculated the results of duels and indirect fire attacks. The development of mōsbē was discontinued in 2008, but the tool was in use at FFI until 2014.

After this, GESI (GEfechts-SImulation System) from CAE (which is the command and staff training system at the Norwegian Army Land Warfare Centre) has been used a few times in a similar manner. Most recently, simulations in GESI have been used to support the special review of the Norwegian land forces, which has been taking place in 2017.

Both mōsbē and GESI have several significant weaknesses which can produce questionable simulation results. For example, they do not support representation of micro-terrain features, and this systematically favours long-range, direct fire weapon systems. In addition, the human behaviour models are very simple, and this entails that the operators have to spend a lot of time micromanaging the units. Furthermore, they do not have an application programming interface (API) for developing additional functionality or plug-ins. When used for experimentation and analysis purposes, it is also a disadvantage that GESI requires a large number of operators (since it is a system for training command and staff procedures). This, of course, limits the convenience and accessibility of GESI simulations for experimentation purposes.

Consequently, there is a need to establish a new capability for conducting more detailed constructive simulations of battalion level operations at FFI. Based on our experience with simulations in mōsbē and GESI, we have identified two main factors that have the potential to improve the fidelity of our constructive simulations: (1) *increased terrain resolution*, and (2) *better tactical artificial intelligence* (AI) that can exploit this terrain [4]. We expect that these two factors will result in more realistic detection and engagement distances.

In the recent years we have seen an increasing use of web technologies for modelling and simulation (M&S). The new version of the HyperText Markup Language (HTML), HTML5 (which was finalized in October 2014), especially provides new abilities for creating interactive web-interfaces for simulations and games [4]. The biggest advantage of using web technology for M&S is *accessibility*. Following this trend, we decided to develop our own easy-to-use, web-based GUI system for controlling constructive entities simulated in VBS and VR-Forces (or any other simulation tool with a suitable API). This way we are able to tailor the

GUI system to our specific needs. The system has been named webSAF.

The next generation distributed simulation environments are envisaged to rely heavily on open standards and service-based architectures [5]. M&S as a service (MSaaS) is an architectural and organizational approach that promotes abstraction, loose coupling, reusability, composability and discovery of M&S services [6][7]. In this paradigm, the vision is that instead of composing a simulation environment as a federation of different individual simulation systems, the users will be able to compose a simulation environment based on services. Web-based GUI systems which are independent from the simulation systems will of course fit perfectly into the MSaaS paradigm.

3.0 SIMULATION OF LAND FORCE OPERATIONS FOR EXPERIMENTATION AND ANALYSIS

We conduct simulations of land force operations for experimentation and analysis purposes, and one of the main research questions we are investigating is *how to increase combat effectiveness*. As part of this work we need to assess and compare the performance of different land force structures, which will vary with regard to: *composition of material and equipment*, *tactical organization*, and *operational concept*.

Our focus is on simulation of the actual combat phases with engagements and skirmishes. However, these phases of combat are also the most complex and therefore the most challenging phases to simulate.

We have not found a single simulation tool that is satisfactory for our use, and it is not possible for us to develop our own simulation system from scratch. Our overall solution is therefore to tailor available commercial off-the-shelf (COTS) simulation tools to suit our needs.

3.1 Simulation-Supported Wargame

Our simulation experiments are conducted as what can be described as *simulation-supported*, *two-sided* (blue and red) *wargames*. Military officers participate as players/operators on both sides. A typical experiment consists of a *planning* phase, a *wargaming* session, and an *after action review* (AAR) session. We normally use tactical vignettes derived from national defence scenarios. After the experiments there is an analysis phase.

3.2 Entity-Level versus Aggregate-Level Models

With today's computing capabilities it should be feasible to simulate operations up to brigade level size using entity-level models. Entity-level models have higher resolution and thus the potential to achieve higher fidelity than aggregate-level models. It is also easier to see what is going on in an entity-level simulation, and this makes them more accessible for face validation [4]. Nevertheless, it is also a known issue that current entity-level models tend to produce attrition levels that are higher than those observed historically [8][9]. “Possible phenomena present in actual combat and accounted for in [the parameters of aggregate-level attrition models (such as the Lanchester models)] but not [in the] entity-level combat models that could explain this include target duplication, shooter non-participation, suppression effects, self-preservation, and suboptimal use of weapons and targeting systems” [8]. In other words, current constructive entity-level combat models lack good representations of the human aspects of combat and combat friction, resulting in that the simulated operations tend to run smoother than they would in the real world. For our use, however, it would have been difficult to calibrate aggregate-level models to represent new combat systems and new concepts due to the lack of data from real operations. Furthermore, our need for higher resolution, and also the possibility of combining virtual and constructive simulations (e.g. having virtual air entities operating together with constructive ground entities), supports that we need a simulation system based on entity-level

models.

3.3 Modelling Human Behaviour

Modelling realistic human behaviour and cognition, including decision-making and creativity, is the hardest and most complex challenge in combat simulation [10]. Human behaviour modelling is challenging because "[h]uman behaviour is not generally yet thought to obey observable laws" [11]. Consequently, the current status for human behaviour simulation is that it can be used "to understand, [but] not necessarily predict, the aggregate behaviour of an inherently complex system for which we have no better model" [12]. When using human behaviour models "it is often possible to perform sensitivity analysis and identify broad trends as opposed to exact predictions" [12]. For example, a constructive simulation may show that increasing the number of main battle tanks (MBTs) has a positive effect on the outcome of a scenario, but it cannot be used to pinpoint the exact number of MBTs required to win a battle with a certain probability [12].

3.4 Summary of Simulation Capability Requirements

The most important requirements for our new simulation capability can be summarized as follows:

- It must support entity-level simulations of battalion level land force operations, and use simulation models with high resolution. Typically, we need to simulate operations that include between one and four maneuver battalions (each with 50–60 combat vehicles and about 200 soldiers), one or two artillery battalions, and one air defence battalion on each side.
- It must represent entities from the following capabilities: maneuver (MBTs, infantry fighting vehicles (IFVs), unmanned ground vehicles (UGVs), infantry, etc.), indirect fire (artillery, missile launchers, close air support, etc.), air defence (missile launchers, radars, etc.), aviation (fixed wing aircrafts, helicopters, unmanned aerial vehicles (UAVs), etc.), combat engineering, and intelligence, surveillance, target acquisition, and reconnaissance (ISTAR) (sensors and facilities).
- It must support high terrain resolution (10 meters between the elevation points, or better) and representation of micro-terrain features.
- It must have a GUI that is easy to use (military officers should be able to control the entities with minimal instruction).
- It must have an API for developing additional functionality (e.g. customized simulation models).
- It should have a tactical AI where the operators are able to issue high-level orders at the company level and more detailed orders at the platoon level for vehicles and squad level for infantry.
- It should have a tactical AI where the entities are able to intelligently take advantage of the terrain.

4.0 SIMULATION SYSTEM

Based on the requirements above, we are composing a simulation system where the ground-to-ground combat entities are simulated in VBS and the air and air defence entities are simulated in VR-Forces. The VBS entities will use behaviour models developed in VBS Control. All the constructive entities are controlled from the web-based GUI system. Figure 1 shows an overview of the components in the simulation system. The simulation tools are briefly described below, and webSAF described in more detail in the next section.

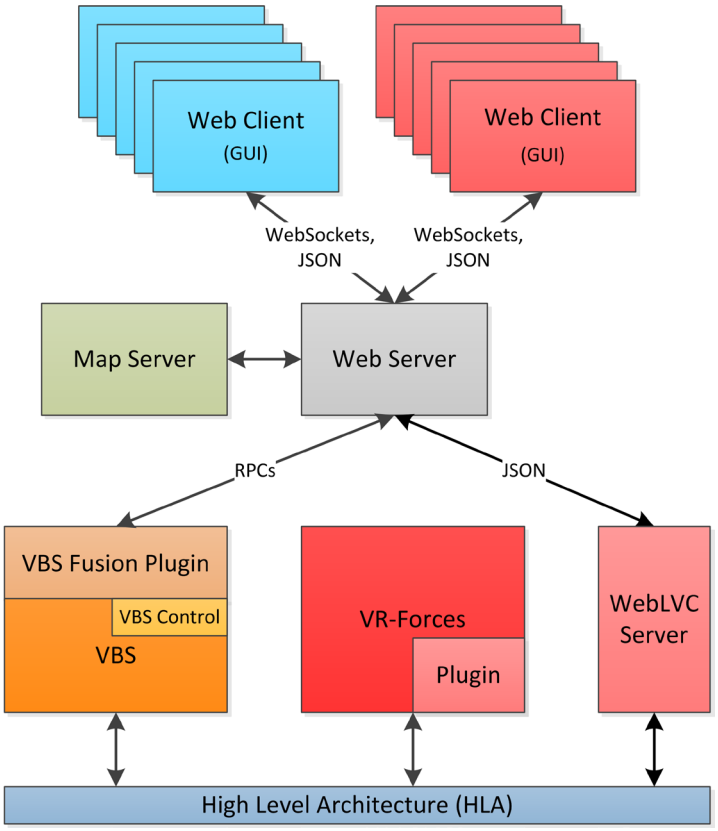

Figure 1: Overview of the components of the simulation system.

4.1 Simulation Tools

4.1.1 Virtual Battlespace (VBS)

VBS is an interactive, three-dimensional synthetic environment, for use in military training and experimentation. It is developed by BISim and is based on game technology from the Armed Assault (ARMA) series. VBS is delivered with a comprehensive content library funded by different nations over the years, and in addition it has its own scripting language for creating new functionality. VBS is used by many military organizations worldwide (including the Norwegian Armed Forces), and has become an industry standard in game-based military simulation.

VBS Fusion is a C++-based API for VBS, developed by SimCentric Technologies. It provides a comprehensive object-oriented C++ library for developing plug-ins for VBS. The plug-ins are compiled as dynamic link libraries (DLLs), which can be loaded by the VBS engine.

4.1.2 VBS Control

VBS Control is a new framework for AI in VBS. The behaviour models used by VBS Control are based on *behaviour trees* (BTs), and users can create customized behaviour models through the VBS Control Editor. The BTs can be visually debugged in real time.

BTs are a relatively new and increasingly popular approach for developing behaviour models [13]. The approach has become especially popular for creating behaviours for non-player characters (NPCs) in computer games, robots, and autonomous vehicles. The first high-profile computer game which used BTs was Halo 2 from Bungie Software [14], which was released in 2004.

BTs are represented as directed trees with a hierarchy of control flow nodes and task nodes that control the behaviour of an agent. The control flow nodes contain some decision logic and have at least one child node. The task nodes are leaf nodes (nodes without children) and contain conditional tasks which test some property in the simulated environment (or the real world in case of robots and autonomous vehicles) or action tasks which alter the state of the simulation (or the real world) in some way [4].

What makes BTs so powerful is their *composability* and *modularity*. Task nodes and control flow nodes are composed into sub-trees which represent more complex actions, and these actions can be composed into higher-level behaviours [15]. Task nodes and action sub-trees can be reused, and different sub-trees can be developed independent of each other. BT editors enable users (e.g. military officers) to create modular behaviour models without needing programming skills. For our system we will typically need to develop BTs for combat drills corresponding to each manoeuvre order that can be issued from webSAF.

4.1.3 VR-Forces

VR-Forces is a framework for computer-generated forces (CGF) developed by VT MAK. It includes simulation models for hundreds of battlefield units and systems in all domains (land, naval, air, and space), and can be used in both aggregate- and entity-level mode. The VR-Forces framework is customizable, and provides several C++-based APIs for different development tasks.

5.0 WEB-BASED GUI SYSTEM (webSAF)

webSAF consists of a *web server*, a number of *web clients*, and a *map server*. The web clients connect to the server through WebSockets, and send and receive data in the form of JavaScript Object Notation (JSON) packets. Map tiles used by the GUI system are streamed from the map server. The ground-to-ground combat entities are simulated in VBS, and the web server communicates with VBS through remote procedure calls (RPCs) using the Apache Thrift framework [16]. The air defence sensors and weapons are simulated in VR-Forces, and the web server communicates with VR-Forces using JSON packets. The communication with VR-Forces goes through a WebLVC Server [17], which wraps the JSON packets inside High Level Architecture (HLA) interactions. The HLA interactions are unwrapped by plug-ins in VR-Forces. The typical data transferred between the web server and the simulation systems are entity status data and orders.

5.1 Components

5.1.1 Web Server

The web server is the management hub of the system. It is the connection link between the web clients and the simulation, and validates all user interactions.

5.1.2 Web Clients

Each operator connects to the web server from a web browser and selects side (blue or red) and role (currently joint fires, manoeuvre, and air defence).

5.1.3 Map Server

The map server is a simple HyperText Transfer Protocol (HTTP) file server, serving map tile images in Portable Network Graphics (PNG) format. Tiles are organized in a file structure according to z-, x-, and y-indexes, where the z-index indicates zoom level, and x- and y-indexes indicate longitude and latitude, respectively.

5.2 Operator Roles

So far the GUI system supports three operator roles: *manoeuvre*, *joint fires*, and *air defence*. Functionality for more roles (e.g. aviation, naval, and ISTAR) will be implemented in the future.

5.2.1 Manoeuvre

The manoeuvre operator controls combat vehicles and infantry. Orders are currently issued at the platoon level for vehicles and at the squad level for infantry. (In the future we plan to implement functionality for issuing higher-level orders at the company level.) In addition, the manoeuvre operator can request indirect fire support from the joint fires operator. The goal is that each manoeuvre operator should be able to control an entire manoeuvre battalion, including organic indirect fire support entities like mortars.

5.2.2 Joint Fires

The joint fires operator receives fire requests from the manoeuvre operators, and prioritizes and forwards fire missions (possibly with adjustments) to the indirect fire entities. Currently, we only have support for land-based indirect fire entities like tube artillery, rocket artillery, and missile launchers, but in the future we plan to also implement functionality for fire support from air and sea entities.

5.2.3 Air Defence

The air defence operator receives detections from the air defence sensors in VR-Forces. VR-Forces also communicates the tracks in a prioritized list, which is presented to the air defence operator. For each track, VR-Forces also sends a list of prioritized launchers which can engage on that track. The air defence operator selects which track to engage, which launcher to use, and when to fire.

5.3 GUI Functionality

The main component of the GUI is the *map area*. In addition, the GUI shows the *order of battle* (OOB) (to the left) and *notifications* (to the right). The OOB and notifications can be easily hidden/shown using the buttons in the upper left and right corners respectively (or using the underlined hotkeys). The top bar shows the operator's role and the current simulation time. Figure 2 shows an example of the web-based GUI for a blue joint fires operator. The GUI functionality has been developed in close collaboration with military subject matter experts (SMEs).

5.3.1 Map Area

Map navigation is similar to most so called "sliding maps" often found online, like OpenStreetMap [18] and Google Maps [19]. The map can be panned by dragging the map with the mouse. When the mouse button is released, the map continues to slide before slowing to a halt. The sliding action enables the user to pan great distances with minimal hand movement. The map view can be zoomed in or out by scrolling the mouse wheel. The bottom bar shows a map scale bar for the current zoom level and the mouse cursor coordinates in Military Grid Reference System (MGRS).

Overlaid on the map are units, spot reports, tracks, and other visual information. Map icons for friendly units are aggregated based on the OOB and zoom level. At the most zoomed-in level, each icon represents a platoon for vehicles and a squad for infantry. At the most zoomed-out level, each icon represents a company (or equivalent).

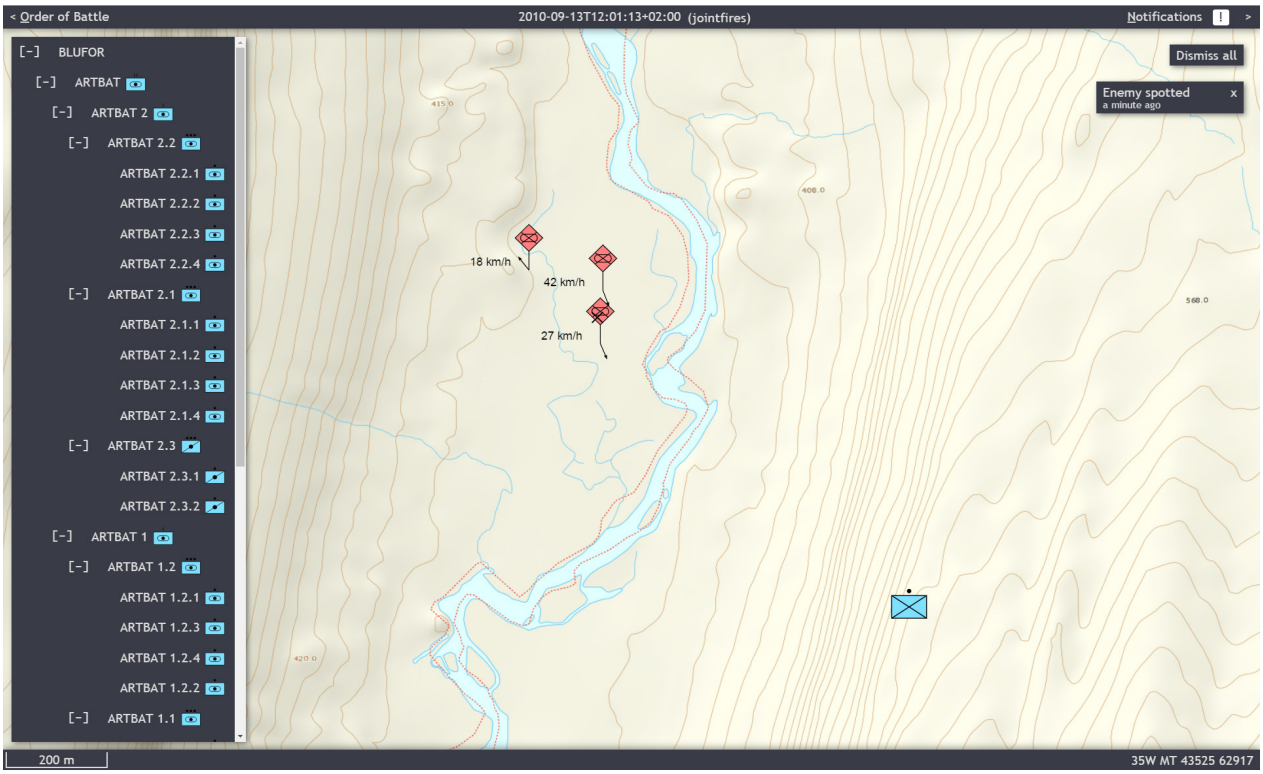

Figure 2: GUI for a blue joint fires operator.

5.3.2 Order of Battle (OOB)

The OOB has a collapsible tree view, similar to the file explorer in many operating systems. The tree represents the command hierarchy of the operator's side (blue or red). To the right of the unit name is a MIL-STD-2525C symbol, indicating the type and size of the unit. Double clicking the unit name zooms and pans the map to bring that unit to the centre. Hovering over the unit name with the cursor highlights that unit on the map. Conversely, hovering over a unit icon in the map highlights that unit in the OOB.

5.3.3 Notifications

Notifications are received when an event of interest occurs, for example, when enemy entities are spotted, or friendly entities are lost. When a notification is received, it appears on the top of the list. The list is kept chronologically sorted, with the newest notification on top. The exclamation mark in the notifications button has a badge that indicates the total number of active notifications. Each notification has a label, for example "Enemy spotted" or "Fire order underway". Clicking the notification label does different things depending on the type of notification. For example, clicking an "Enemy spotted"-notification centers the map on where the enemy was spotted. The time since the notification was received is indicated below the label.

5.3.4 Fog of War

An operator cannot see entities on the opposing side by default. Rather, when a unit under the control of the current operator spots an enemy entity, it is reported in the GUI as MIL-STD-2525C symbols to indicate type. In addition, the entity's approximate speed and heading is shown as a labelled arrow protruding from the map icon's base. If multiple enemies are spotted close to each other, the map icons will stack, as illustrated in Figure 3. The number beside each stacked icon indicates the count of that type. The first time that an enemy entity is spotted in an area, a notification is given. Clicking this notification automatically pans

the map to the location where the enemy entity was spotted.

As long as an enemy entity can be seen by friendly troops, the spot information (type, position, speed, and heading) is continuously updated in the map view. Receiving updates with too high frequency though, might give the operator a situational awareness well above what is realistic. In addition, if a lot of enemy entities can be seen, too frequent updates might cause network congestion. Therefore, the value of the spot update frequency cannot be set until after running through several test scenarios.

Figure 3: Stacked spot report icons.

5.3.5 Issuing Orders

Units can be selected for the purpose of issuing orders. Once they are selected, their icons become highlighted in the map and orders can be issued by bringing up the context menu (right-clicking the map). Different options appear on the context menu depending on the role of the operator and whether any units have been selected. A joint fires operator will for example have the option of moving the selected artillery unit to the clicked location, or designate a new remotely delivered mine field.

5.4 GUI Design Philosophy

The GUI design philosophy is based on balancing the following aspects, roughly prioritized in descending order:

1. *Time efficient user interaction*. The user should be able to do a lot, fast.
2. *Familiar user interaction*. The system should be easy to learn.
3. *Ease of implementation*. The project needs to be completed within time and budget constraints.
4. *Ease of maintenance and extension*. The project is anticipated to expand in the future. The software architecture therefore needs to be amenable to changes and extensions.
5. *Aesthetics*. Visual style should not hinder adoption by users.

One of the main drivers behind the project was to save operator time during constructive simulation, in order to reduce the number of operators required. The GUI was therefore designed to help the operator reach his or her goals with the least amount of manual effort. Operator efficiency takes precedence over other design aspects.

Familiarity was the driving aspect for other design decisions. The "sliding maps" way of panning and zooming in and out in the map view, for example, is largely unchanged compared with widely used online map services. The right-click context menu is also a tried and true way of minimizing UI clutter, while still keeping useful interactions just a click away. The context menu is complemented by a few static menus, mainly the OOB and the notification menus. The user has the option of hiding them, and when not hidden, they are kept to the sides as to not clutter the user's main focus: the map. Apart from the thin bars at the top and bottom of the screen, the map fills the entire GUI area. Menus and available interactions are hidden in optional overlays instead of being ever present surrounding the map. The disadvantage is that, for a new user, it is not immediately obvious what interactions are available. On the other hand, it can be overwhelming for a new user to see all the available options simultaneously. A user who has become familiar with the GUI, however, will prefer the situational awareness given by a larger map.

Another aspect that might discourage new users is poor aesthetics. Though aesthetics is rarely prioritized in internal software projects, slow adoption of new software and routines is a persistent problem across organizations [20]. Thus some consideration was given to the visual design of the GUI, like the medium-dark colour scheme, button highlighting, and flat menus with sharp corners and faint shadows to indicate depth ordering. Perceived performance is closely linked to aesthetics; users get fatigued by slow user interfaces. Performance tuning has not taken significant development time so far, but it was considered during software architecture design and in selecting external software libraries.

5.5 GUI Implementation

The GUI was implemented in the TypeScript programming language [21]. TypeScript provides code type safety which helps catch errors early in development. It compiles to JavaScript that runs in modern web browsers. To ease the development of GUI components, the web framework React was employed. React is maintained by Facebook, and provides an easy way to create scalable and responsive single page applications [22]. React also has an open source ecosystem of ready-made GUI components with permissive licenses. Some of these components were used, and others were developed in-house. The component that contains the map is rendered using OpenLayers [23], an open source web mapping library that provides similar functionality to other online maps. To manage communication, a simple event system sends updates (in JSON format) between GUI components and the web server via a WebSockets client. Figure 4 shows the architecture of the web client.

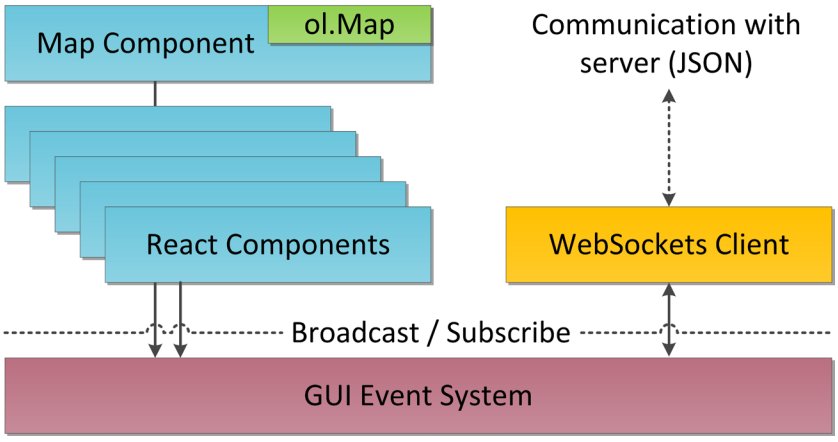

Figure 4: Software architecture of the web client.

6.0 INITIAL EXPERIENCES WITH THE SYSTEM

We have so far tested webSAF in small experiments with up to three operators on each side, and are able to simulate operations with a battalion of manoeuvre entities, a battalion of indirect fire entities, and a battalion of air defence entities on each side. However, we have not yet developed behaviour models (for VBS Control) for the majority of the combat drills for manoeuvre platoons. The initial tests with the web-based GUI system have been successful, and we have so far not observed any performance issues with the web server or GUI. The system has not yet been stress tested, but for simulations with multiple manoeuvre battalions we envisage that we need about eight to ten operators on each side.

6.1 Overall Feedback from Officers

Since the GUI system is being developed in close collaboration with military SMEs and officers, suggested improvements are implemented continuously. However, the overall feedback we have received from military officers (typical users of command and staff training systems and C2 systems) is that webSAF appears to be better than other GUI systems they have seen before, especially with regards to accessibility, ease of use, and visual feedback. Senior military officers have also expressed that they want their command and staff training system, and also their C2 systems, to have GUIs with similar functionality.

6.2 Advantages to Developing a Web-Based GUI System

There are several advantages to developing our own web-based GUI system for controlling entities in constructive simulations:

- It requires minimal hardware for the operator clients.
- No simulation software (and thus no licenses) needs to be installed on the operator clients (only a web browser is needed).
- It can be tailored to a specific use (e.g. two-sided wargaming or command and staff training).
- It is in principle independent of the simulation tools in use (a plugin is needed for each tool), and can be used to control entities in a federation of different simulation tools.
- There are a lot of tools and libraries available (many of them are open source) for developing web-based GUIs and applications.
- In principle, the web-based GUI system allows military participants to join a simulation experiment from any location, even on a classified network.

We expect to see more web-based GUIs for simulation systems in the near future, and we also expect more web-based GUIs for C2 systems. For example, a demonstrator for a system for simulation support during the planning of military operations with a web-based GUI has been developed FFI [24]. In principle, the same web-based GUI system could be used for wargaming, command and staff training, and C2, for example by defining different sets of operator roles.

7.0 FURTHER WORK

Much of the work ahead will be focused on developing behaviour models of combat drills for manoeuvre platoons [13]. It is worth mentioning that the composability and modularity of BT-based behaviour models open up opportunities for collaboration on development, and sharing of behaviour models, between nations using VBS. In addition, we need to include additional capabilities like combat service support and ISTAR in the simulation system. The GUI system must also be further developed with new roles and orders to control entities from the additional capabilities.

Before we can start doing useful experiments, the simulation system requires calibration, validation and extensive testing. We expect to be able to conduct simulation experiments with multiple manoeuvre battalions on each side by the mid-2019. Moreover, it will also be interesting to compare the results from these simulations with the results from our previous simulations, and investigate if the new, more detailed simulations are more useful.

So far the development of webSAF has been focused on creating a working solution. In the future we will look at the possibility of making the GUI system more interoperable by using the Coalition Battle Management Language (C-BML) [25] standard for sending orders to the simulated units. We will also look at the possibility of employing the WebLVC protocol [17] for transferring entity status data from the whole simulation system (and not just VR-Forces).

8.0 SUMMARY AND CONCLUSION

This paper has presented the design and implementation of webSAF, a web-based GUI system for controlling entities in constructive simulations (semi-automated entities in VBS and VR-Forces). The GUI functionality is being developed in close collaboration with military SMEs and officers, and is designed to be easy to use, in order to reduce the number of operators required. The GUI system will be used to operate simulation-supported, two-sided wargames, and so far it has been tested in small experiments with an operator controlling a manoeuvre battalion, an operator controlling a battalion of indirect fire entities, and one operator controlling a battalion of air defence entities on each side.

Developing a web-based GUI system for controlling entities in constructive simulations has several advantages. It requires only minimal hardware, and no simulation software, for the operator clients. In addition, it can be tailored to a specific use (e.g. two-sided wargaming or command and staff training), and is in principle simulation system agonistic. Military officers have expressed that webSAF appears to be better than other GUI systems they have seen before, and that they want their command and staff training system, and their C2 systems, to have GUIs with similar functionality. We believe the web-based GUI approach is the way ahead to make constructive simulations and C2 systems more accessible and easier to use.

9.0 REFERENCES

[1] P-I. Evensen, K. Selvaag, D.H. Bentsen & H. Stien, Web-Based GUI System for Controlling Entities in Constructive Simulations, *Proceedings of the Interservice/Industry Training, Simulation and Education Conference (I/ITSEC) 2017*, Paper No. 17043, 2017.

[2] E.Ø. Hoff, P-I. Evensen, H.R. Holhjem, I.B. Øyan & H.K. Nygård, Simulation in Support of Army Structure Analysis, *Proceedings of the Interservice/Industry Training, Simulation and Education Conference (I/ITSEC) 2012*, Paper No. 12088, 2012.

[3] E.Ø. Hoff, P-I. Evensen, H.R. Holhjem, I.B. Øyan & H.K. Nygård, Interactive Simulation to Support the Transition of Forces, *Proceedings of the NATO Modelling and Simulation Group (NMSG) Annual Symposium 2013 (STO-MP-MSG-111)*, Paper No. 6, 2013

[4] P-I. Evensen & D.H. Bentsen, *Simulation of land force operations – a survey of methods and tools*, FFI-report 2015/01579, Kjeller: Norwegian Defence Research Establishment (FFI), 2016.

[5] R. Siegfried, M. Bertschik, M. Hahn, G. Herrmann, J. Lüthi & M. Rother M, Effective and Efficient Training Capabilities through Next Generation Distributed Simulation Environments, *Proceedings of the NATO Modelling and Simulation Group (NMSG) Annual Symposium 2013 (STO-MP-MSG-111)*, Paper No. 7, 2013.

[6] R. Siegfried, T. Van den Berg, A. Cramp & W. Huiskamp, M&S as a Service: Expectations and challenges, *Proceedings of the 2014 Fall Simulation Interoperability Workshop (SIW)*, Paper No. 040, 2014.

[7] North Atlantic Treaty Organization (NATO) Science and Technology Organization (STO), *Modelling and Simulation as a Service: New Concepts and Service-Oriented Architectures*, STO Technical Report, STO-TR-MSG-131, 2015.

[8] M.D. Petty & J. Panagos, A Unit-level Combat Resolution Algorithm Based on Entity-level Data, *Proceedings of the Interservice/Industry Training, Simulation and Education Conference (I/ITSEC) 2008*, Paper No. 8359, 2008.

[9] M.D. Petty, R.W. Franceschini & J. Panagos, Multi-Resolution Combat Modeling, in *Engineering Principles of Combat Modeling and Distributed Simulation*, Tolk, A. (edited by), John Wiley & Sons, 2012.

[10] J. Thorpe, Trends in Modeling, Simulation, & Gaming: Personal Observations About the Past Thirty Years and Speculation About the Next Ten, *Proceedings of the Interservice/Industry Training, Simulation and Education Conference (I/ITSEC) 2010*, Paper No. IF1001, 2010.

[11] J. Storr, *The Human Face of War*, Continuum, 2009.

[12] Y. Papelis & P. Madhavan, Modeling Human Behavior, in *Modeling and Simulation Fundamentals: Theoretical Underpinnings and Practical Domains*, Sokolowski, J.A. & Banks, C.M. (edited by), John Wiley & Sons, 2010.

[13] P-I. Evensen, H. Stien & D.H. Bentsen, Modeling Battle Drills for Computer-Generated Forces using Behavior Trees, *Proceedings of the Interservice/Industry Training, Simulation and Education Conference (I/ITSEC) 2018*, Paper No. 18081, 2018.

[14] D. Isla, Handling Complexity in the Halo 2 AI, *Proceedings of the Game Developers Conference (GDC) 2005*. 2005.

[15] I. Millington & J. Funge, *Artificial Intelligence for Games*, 2. Edition, Morgan Kaufmann, 2009.

[16] Apache Software Foundation, *Apache Thrift*, Retrieved August 22, 2018, from https://thrift.apache.org, 2016.

[17] L. Granowetter, The WebLVC Protocol: Design and Rationale, *Proceedings of the Interservice/Industry Training, Simulation and Education Conference (I/ITSEC) 2013*, Paper No. 13163, 2013.

[18] OpenStreetMap, *OpenStreetMap*, Retrieved August 22, 2018, from https://www.openstreetmap.org.

[19] Google, *Google Maps*, Retrieved August 22, 2018, from https://www.google.com/maps.

[20] B.H. Hall & B. Khan, *Adoption of New Technology*, National Bureau of Economic Research,

Cambridge, Massachusetts, 2003.

[21] Microsoft, *TypeScript*, Retrieved August 22, 2018, from https://www.typescriptlang.org.

[22] Facebook, *React - A JavaScript Library for Building User Interfaces*, Retrieved August 22, 2018, from https://reactjs.org.

[23] OpenLayers, *OpenLayers*, Retrieved August 22, 2018, from https://openlayers.org.

[24] S. Bruvoll, J.E. Hannay, G.K. Svendsen, M.L. Asprusten, K.M. Fauske, V.B. Kvernelv, R.A. Løvlid & J.I. Hyndøy, Simulation-Supported Wargaming for Analysis of Plans, *Proceedings of the NATO Modelling and Simulation Group (NMSG) Annual Symposium 2015 (STO-MP-MSG-133)*, Paper No. 12, 2012.

[25] Simulation Interoperability Standards Organization (SISO), *Standard for Coalition Battle Management Language (C-BML) Phase 1*, SISO-STD-011-2014, 2013.