A hybrid architecture framework for simulations in a service-oriented environment

Jo Erskine Hannay, Karsten Brathen, Ole Martin Mevassvik

FFI – Norwegian Defence Research Establishment

Abstract

Service orientation, and more recently, the notions of cloud technology that service orientation enable, are designed to accommodate the need for flexible enterprise business processes. Through standardized interfaces, a service-oriented architecture (SOA) should enable one to build and rebuild software systems readily and rapidly in a methodological manner. However, certain domains have specialized architectural standards; an example in point is modelling and simulation (M&S), for which there exist mature architectural standards, that may even have many of the characteristics strived for in SOA. An important issue is, therefore, how to integrate specialized architectures into a wider SOA. Using defence information systems and M&S as a case, we outline a hybrid architecture framework for specialized architectures in an encompassing SOA. Although it may be possible to dissolve a specialized architecture into the encompassing SOA at implementation time, we argue that it is important to be able to model the specialized architecture as an integral intact part. We further advocate a pragmatic notion of reference architecture in terms of appropriate level of abstraction and domain specificity to avoid pit-falls that may render architecture work unusable.

Keywords: architecture framework, overarching architecture, reference architecture, target architecture, systems portfolio, Service-Oriented Architecture (SOA), NATO C3 Taxonomy, Distributed Simulation Engineering and Execution Process (DSEEP), simulation environment, High Level Architecture (HLA)

1. Introduction

The functionality of present information systems must be adaptable in an altogether different manner than before. To meet rapidly changing business processes, functionality must be network enabled, accessible anywhere and put together readily and rapidly to serve various needs. The concepts of service-oriented architecture (SOA); e.g., [Erl, 2007] and capability-based development (CBD); e.g., [Danesh and Yu, 2015] explicitly seek to partition business and IT functionality into loosely coupled, stable pieces that constitute meaningful units at both the enterprise level and the information-technological level.

An example in point is defence capabilities. To enable flexible reconfiguration of defence activities in unpredictable and complex environments and to enable combined multi-nation warfare across heterogeneous defence platforms as is necessary in the North Atlantic Treaty Organization (NATO), planning and development should now be done in terms of capabilities (what to achieve and perform), rather than in terms of specific assets and material. This necessitates the appropriate definition of capabilities at both operational and IT levels. These capabilities must be highly recomposable and support multiple dimensions of quality of service. At the IT level, capabilities must be readily accessible globally from repositories and registries in the form of secure clouds.

For all this to work, portfolios can no longer consist of stand-alone stove-piped systems, and principles of service orientation are pertinent: Functionality must be *loosely coupled* in terms of data, processing, space and time, so that it can be used in various contexts. For composability and interoperability, functionality and communication must be described in standardized formats. Then, decisions have to be made on what capabilities are needed and how functionality should be provided by what services.

There are substantial challenges in achieving service-orientation [Ahmed and Letchmunan, 2015; Li et al., 2010]. Defence information systems portfolios, and portfolios in other complex and heterogeneous operational domains related to crisis management, exacerbate these challenges. The architectural challenges one faces in defence information systems portfolios therefore highlight issues relevant to enterprise systems portfolios in general.

Some sort of architectural coordination mechanism is necessary for handling the enormous complexity of all aspects of such portfolios. So-called overarching and reference architectures are intended as standards and guidances to ensure a common architectural basis across systems in a portfolio to ensure interoperability.

In addition, subsystems in a portfolio might adhere to their own specialized architecture guidelines and stan-

Email addresses: jo.hannay@ffi.no (Jo Erskine Hannay), karsten.brathen@ffi.no (Karsten Brathen), ole-martin.mevassvik@ffi.no (Ole Martin Mevassvik)

dards. For example, the need for loose coupling and interoperability has been recognized for some time in the modelling and simulation (M&S) domain, as reflected in the work of the Simulation Interoperability Standards Organization (SISO). There now exist several standards and protocols for M&S systems; e.g., the High Level Architecture (HLA) [IEEE Standards Association, 2010a], the Distributed Interactive Simulation (DIS) standard [IEEE Standards Association, 2015], and the Test and Training Enabling Architecture (TENA) [Test Resource and Management Center, 2002]. Due to presumed and factual increases in benefit/cost ratios, M&S are mandated to be an integral part of portfolios in many domains. For defence capabilities, they are mandated to be used in education, operations training and rehearsal and long- and short-term planning [NATO Modelling and Simulation Group, 2012]. Recently, frameworks for M&S analyses of critical infrastructure [Grogan and de Weck, 2015; Rome et al., 2014; Tolone et al., 2009] have been developed. In other domains, such as in the oil industry and construction, simulations are indispensable core technologies that give industrial capabilities which would otherwise not be possible.

The question then arises as to the degree of internalizing specialized architectures; e.g., for simulations, into an encompassing SOA. Using HLA and defence portfolios as a case, we will discuss simulations and simulation components in terms of service orientation, and what it means for simulations to be integral in a larger service-oriented federation of systems. From this, we will suggest answers to the following research questions on how specialized architectures might be embedded into an encompassing SOA:

1. To what extent is it desirable and possible to promote specialized architectures as integral parts to the encompassing architecture?
2. To what extent is it desirable and possible to dissolve specialized architectures; hereunder,
   (a) can components in a specialized architecture rather be components in the encompassing architecture?
   (b) can the service management mechanism of the encompassing architecture be used in place of that of the specialized architecture?

As the main contribution, we will then suggest a hybrid architecture framework in line with our findings to the above questions. The framework will have means to be specific in terms of *topology*, in that it will delineate boundaries between sub architectures. Further it will point to a dynamic repository of architecture artefacts that represent domain-specific capability descriptions.

Deliberations into simulation and SOA have been undertaken at the theoretical and experimental level; see e.g., [Wang et al., 2011, 2008; Pan et al., 2007; Dragoicea et al., 2012; Wei and Tsai, 2014] for an overview. Our focus is on setting the stage in terms of architecture principles. The discussion is at the level of capabilities, and we do not discuss important performance issues such as time criticality, connectivity and security. Further, our discussion is at the conceptual level, and we do not provide empirical evidence (quantitative or qualitative results) on effects of using the architecture framework. To do so would require empirical studies over time and is left for future work.

We start by reviewing architectural concepts in Section 2 and then simulation environments with emphasis on HLA in Section 3. We address the research questions in Section 4 by discussing past research. We present the main contribution, a hybrid architecture framework, in Section 5, and give examples of its use in Sections 6 and 7. We conclude in Section 8.

2. Architecture

Agreeing that systems in dynamic portfolios cannot be stand-alone stove-piped systems, but have to be designed to interoperate, a relevant term is *federation of systems*; which, in our discussion, is a system of loosely coupled collaborating information systems. An individual information system may consist of purely human routines; e.g., strategies and processes for information handling. It may be purely technical in terms of information technology. Or it may consist of all levels of information handling from human to technical. Usually, individual systems are designed with the intent that their constituent parts *interoperate*; i.e., work together and communicate with each other to fulfil the system's goals. When assembling systems into a federation of systems it is not obvious that the systems will interoperate well – or at all – without considerable effort. This is because individual systems are often not designed to interoperate with other systems; and in particular, not designed to interoperate with a range of systems that may not be known at the time of design. The idea of service orientation is that systems *can* be designed to interoperate with other, perhaps unknown or future, systems.

An *architecture* of a system or of a federation of systems is, according to [International Organization for Standardization, 2011]: "The fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution." Thus, an architecture provides plans or blueprints for a system.

Architectures can be designed at various levels of abstraction. There is little consensus in general on the various levels of abstraction or on how to name them, but we declare the notions that are relevant for our discussion (Figure 1). Two notions are central: *architectural building block* (ABB) and *architecture pattern* (AP) [The Open Group, 2011a]. ABBs are the elements that constitute an architecture and APs are are high-level suggestions for ways of composing ABBs into architectures.

At the very high level, an *architecture ontology* might declare *types* of ABB and AP. For example, [The Open Group, 2011a] declares ABB types, such as '(business) process', 'service', 'repository', 'service container'; and AP

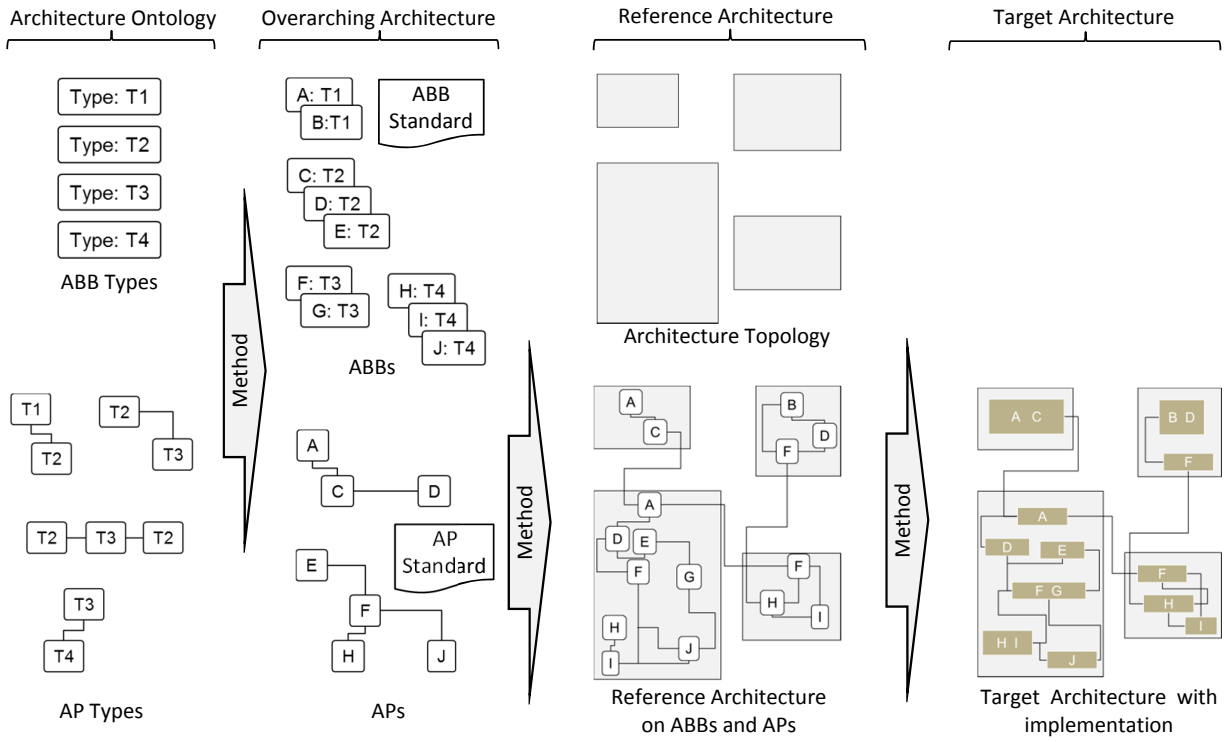

Figure 1: Architecture framework. An architecture ontology provides types of architecture building block (ABB) and architecture pattern (AP). An overarching architecture consists of specific ABBs and APs of various types, with standards for specifying ABBs and APs. Various architecture topologies specifying system and interoperability boundaries aid in designing reference architectures using ABBs and APs. From this, target architectures with implementation-specific systems (olive) can be designed.

types, such as 'consumer pattern', 'service invocation pattern', that are pertinent for any SOA.

Next, actual ABBs and APs of the various ABB types and AP types can be used for declaring a domain-specific *overarching architecture*. The manner in which ABBs and APs are specified might be standardized.

Then, reference architectures are designed by composing ABBs guided by APs from the overarching architecture. In addition, we hold that a *architecture topology* (or several) should be designed at the reference architecture level to delineate intended systems boundaries and the boundaries in which interoperability standards are enforced. From a reference architecture, individual *target architectures* [North Atlantic Treaty Organization, 2016] that specify implementations may be derived. There should be methods for refining architectures at one abstraction level to the next [The Open Group, 2011b].

The spectrum of architecture abstraction levels and such methods are what we here refer to as an *architecture framework* (Figure 1).

Although, the notion of 'architecture framework' is not consistently defined, key points include that such frameworks are ontology based, open and extensible [Hilliard, 2013], and that they provide "conventions, principles and practices for the description of architectures established within a specific domain of application and/or community of stakeholders" [International Organization for Standardization, 2011]. Various frameworks also cover different aspects of architecting [Garnier et al., 2013]; for example, the NATO Architecture Framework (NAF) [North Atlantic Treaty Organization, 2016] is a view-based description framework for architecture and The Open Group Architecture Framework (TOGAF) [The Open Group, 2011b] emphasizes architecture governance and the transformation of one type of architecture into another.

It should also be noted that notions of 'overarching architecture', 'reference architecture' and 'target architecture' differ. For example, The Open Group SOA Reference Architecture [The Open Group, 2011a] is a generic template with ABBs and APs that are pertinent for any SOA, and is, in our terminology, an architecture ontology providing ABB types and AP types, rather than a reference architecture. Moreover, there seems to be a tendency to ignore the full spectrum of architecture abstraction; also observed by [Cloutier et al., 2010]. In the defence domain, the existence of NAF notwithstanding, there have been difficulties both in elaborating what is referred to as reference architectures and commencing development of portfolios according to reference architectures [Hannay et al., 2013; Bloebaum et al., 2013]. Seemingly, problems have arisen due to the following *reference architecture pitfalls*:

- the conception that a reference architecture should be all-encompassing
- the conception that a reference architecture should be fully abstract; e.g., independent of domain, commu-

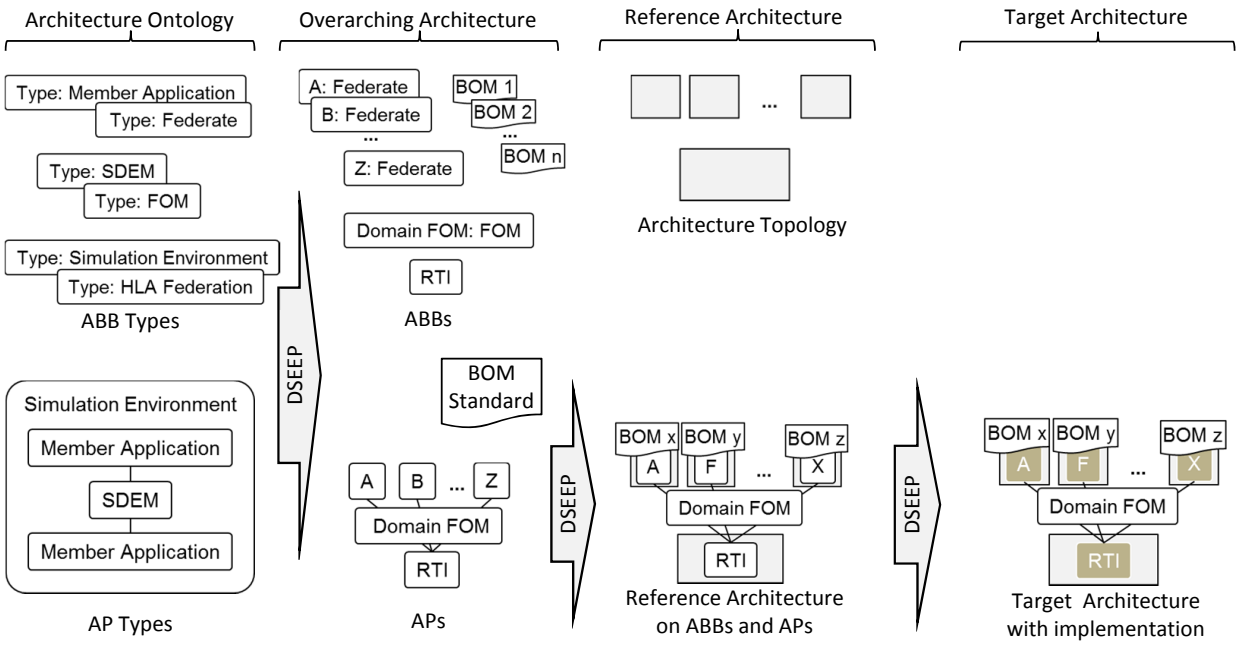

Figure 2: Simulation architecture framework. DSEEP architecture ontology with HLA specialization as sub-types. Within HLA, a FOM gives rise to an overarching architecture consisting of domain-specific ABBs and APs in line with the FOM, which can be described using the BOM standard. Using these ABBs and APs, various reference architectures can be designed, from which target architectures with implementation-specific federations (olive) can be designed. The DSEEP provides a method through these levels of abstraction.

nity of interest and technology.

- the conception that there should be one single reference architecture

Individually, these conceptions of what a reference architecture should be might not be unreasonable. However, we hold that their accumulated effect is to disable sensible action and manifests a severe overloading of the notion of 'reference architecture' with what should be factored out into several levels of abstraction. Observed failures are that reference architectures are hard to devise, since it is unclear what exactly a reference architecture should describe, and that architecture sketches that have been devised are hard to use, because it is unclear what the intention and practical implications of a reference architecture are meant to be. In our discussion, we will therefore use a wide spectrum of architecture abstraction levels, and we will promote a notion of reference architecture that is on a more specific abstraction level than seems to be common.

3. Simulation architecture

We cast simulation architectures in the above architecture framework. The Distributed Simulation Engineering and Execution Process (DSEEP) [IEEE Standards Association, 2010b] is a methodology for developing (and utilizing) distributed simulation systems; see [Tolk, 2012b] for an overview. The DSEEP implies an architecture ontology with three main ABB types; so-called *Member Application*, *Simulation Data Exchange Model* (SDEM) and *Simulation Environment*. Member applications are components serving defined roles in a simulation such as executing simulation assets or utility programs such as data loggers or visualization tools. Member applications exchange information declared in a SDEM. A named set of member applications along with the SDEM and a set of agreements constitute a simulation environment. The DSEEP comes with overlays for HLA, DIS and TENA (Section 1) that specialize the DSEEP. NATO leverages HLA, and we will therefore elaborate HLA for our discussion.

The M&S community has been at the forefront with regards to interoperability due to explicit demands on reuse and composability in complex simulation systems. The High Level Architecture (HLA) [IEEE Standards Association, 2010a] is a design, development, and runtime standard for distributed simulation systems [Petty and Gustavson, 2012; Kuhl et al., 1999]. It enables viable practices for loose coupling and interoperability.

HLA specializes the ABB type 'Member Application' to *Federate*, 'SDEM' to *Federation Object Model* (FOM) and 'Simulation Environment' to *HLA Federation*, see Figure 2. In HLA, information exchange according to a given FOM is coordinated by a *runtime infrastructure* (RTI) which also does advanced time management. HLA prescribes a publish/subscribe protocol: federates publish object attributes and interactions between attributes, and federates may subscribe to updates of published attributes and interactions. The RTI brokers these messages. Federates may also query the RTI on-the-spot for updates.

For the defence domain, several overarching FOMs exist

which set out to declare what is needed in various defence simulation scenarios. Of special interest is the Realtime Platform Reference Federation Object Model (RPR-FOM) and adjoining guide, the Guidance, Rationale & Interoperability Modalities (GRIM) [Simulation Interoperability Standards Organization, 2015b,a]. The RPR-FOM declares a range of defence entities and interactions between them which are relevant for simulations at the weapons platform level. It is the result of data model development under the Distributed Interactive Simulation (DIS) standard [IEEE Standards Association, 2015] (whose architecture is a bus which distributes data between simulator components in a standard format, Protocol Data Units (PDU), without the filtering (e.g., publish/subscribe) and timing control of which HLA is capable [Tolk, 2012c]). Other overarching FOMs exist; e.g., the NATO Education and Training Network (NETN) FOM [NATO Research and Technology Organisation, 2012].

In our framework, a FOM gives rise to an overarching architecture for a given domain. The ABBs of the overarching architecture are a collection of possible federates relating to that FOM and an RTI. The APs describe interactions between such federates. The Base Object Model (BOM) standard [Simulation Interoperability Standards Organization, 2006] for defining abstract object models of entities and interactions in a simulation can be used to specify ABBs and APs.

For designing reference and target architecture, the HLA's Object Model Template (OMT) describes how to declare the various object models in HLA federations: In addition to a FOM for the federation, each individual federate has a Simulation Object Model (SOM) associated to it which declares the federate's contribution to the shared state space, and the RTI is associated with a Management Object Model (MOM). Further, the Federation Engineering Agreements Template (FEAT) [Simulation Interoperability Standards Organization, 2013] describes how to write a Federation Agreement Document (FAD). While a FOM ensures a degree of syntactic interoperability, a FAD ensures a degree of semantic interoperability.

Note that when model-centric development becomes mature in the simulation engineering community, simulation protocol-independent architectures can be developed in a standard manner, and the choice of using HLA, DIS or TENA (or other protocols and standards) can be delayed to the reference architecture or target architecture stage.

4. Simulation environments and service orientation: past research

We will start to address Research Questions 1 and 2 by reviewing past research on loose coupling and interoperability for simulation environments. This pertains to simulation environments as parts in a larger federation as well as member applications as parts in a simulation environment. We will form the discussion around central themes of service-oriented architecture.

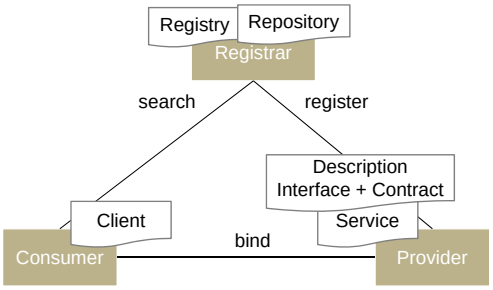

Figure 3: SOA Triangle (adapted from [Erl, 2007]).

4.1. Service-Oriented Architecture

Service-Oriented Architecture (SOA) is an architectural systems design and run-time approach which promotes abstraction, loose coupling, reusability, composability and discovery [Erl, 2007]. In use, SOA has three main roles: the *Service Provider*, the *Service Consumer* and the *Service Registrar*; see Figure 3.

A service is declared by means of an implementation-independent *description* consisting of a standardized *interface* to ensure a degree of syntactic interoperability and a standardized *contract* to ensure a degree of semantic interoperability [The Open Group, 2014]. Descriptions also specify the standardized data exchange formats that are used for consuming a service.

At the overarching and reference architecture levels of our framework, descriptions are technology and implementation independent; whereas in target architecture, specific technology may be used for descriptions; e.g., Web Services Description Language (WSDL) [World Wide Web Consortium, 2007] for Big Web Services (WS*) [World Wide Web Consortium, 2004], or Web Application Description Language (WADL) [World Wide Web Consortium, 2009] for Representational State Transfer (REST) style technology [Fielding and Taylor, 2002]. Although a choice has been made for SOA technology at this point, services as such are still implementation independent (see below). Current practice around popular technologies (WebSocket [Fette and Melnikov, 2011], Advanced Message Queue Protocol (AMQP) [Organization for the Advancement of Structured Information Standards, 2012], JavaScript Object Notation

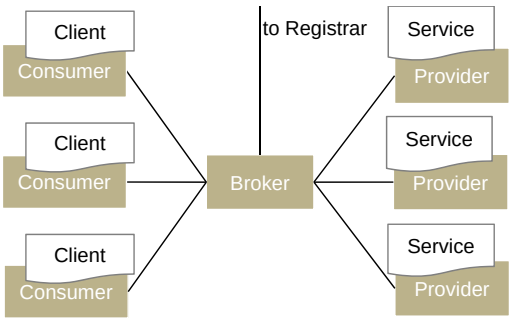

Figure 4: Broker technology.

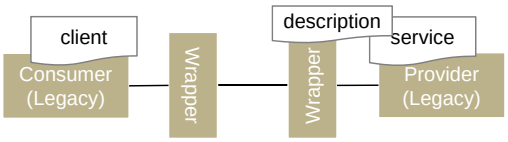

Figure 5: Gateways (wrappers) are used to service-enable software which was not designed service oriented at the outset.

(JSON) [Ecma International, 2013], REST) at the target architecture level usually avoid or omit descriptions. In our framework, it is essential that all services have descriptions at the reference architecture level, at least.

In our framework, services exist on the level of capabilities; i.e., they are abstract in the sense that they persist independently of implementation and, ideally, context. Therefore, a service is different from a service *provider*, since the latter is a specific configuration of resources in a given situation that implements and offers the service. A service provider may provide several services and a service may be provided by several providers [The Open Group, 2014]. Likewise, a *client* is a persistent and implementation-independent user of a service that relates to the service description. A service consumer is an actual configuration of resources in a given situation that uses functionality in the service provider.

A service provider registers a service it wishes to provide to the community with a registrar. The registrar deposits the description in a repository and the concrete information for run-time binding in a registry. A client consults the registrar for descriptions from the repository to prepare for service consumption, and the consumer consults the registrar for binding information from the registry to an appropriate provider.

Types of interaction are mainly request/response (synchronous) and publish/subscribe (asynchronous). *Broker* technology facilitates publish/subscribe, in that a consumer relates to a broker which administers publish/subscribe requests; see Figure 4.

The ever-present need to integrate legacy software haunts SOA. A legacy system can become a service provider by exposing a part of its functionality through a gateway (wrapper) which translates communication from the software system to the standard communication used in the SOA, and likewise for a consumer; see Figure 5.

4.2. Simulation environments as parts in federations

In the defence domain, there has, for some time, been a desire to interoperate operational systems with simulations. In line with strategic decisions, this should be done as part of a service-oriented portfolio [Hannay et al., 2016].

Somewhat independently of concerted SOA initiatives, research in the defence simulation and operational systems communities has strived for an important aspect of service orientation; namely standardized data exchange formats, but without emphasis on standardized descriptions as a whole. Many command and control (C2) systems for planning, commanding and monitoring military operations support XML-based web technologies for communicating their data [Tolk, 2012a,d]. Moreover, data models such as the Joint Consultation, Command and Control Information Exchange Data Model (JC3IEDM) (NATO standard STANAG 5525) and its successor, the Multilateral Interoperability Program Information Model (MIM) [Multilateral Interoperability Programme, 2015], specify the minimum set of data that needs to be exchanged in coalition or multinational operations. Systems that communicate in terms of these common data structures thus gain a level of interoperability.

For communication with simulation environments, the Coalition Battle Management Language (C-BML) [Simulation Interoperability Standards Organization, 2014b] is a formalized language for expressing a commander's plans, orders and reports across C2 systems, simulation systems and autonomous systems. The Military Scenario Definition Language (MSDL) [Simulation Interoperability Standards Organization, 2008] is a standard for describing scenarios used for initializing systems. Both C-BML and MSDL are XML-based and relate to JC3IEDM.

These standards make it possible to achieve interoperability between C2 systems and simulation systems. Efforts toward further interoperability between C2 systems and simulation systems are consolidated in a C2SIM initiative [Heffner et al., 2014]. A number of system demonstrators have been developed to explore the concept.

Figure 6 outlines a demonstration [Pullen et al., 2012], where inter-system collaboration was achieved using standards such as MSDL, C-BML, DIS and JC3IEDM. A Scripted BML (SBML) server coordinated the MSDL initialization and the exchange of C-BML orders and reports. Legacy systems were "standards enabled" using wrappers.

For federations like this to become truly service oriented, however, the involved systems must become providers and consumers of services; that is, C2 and simulation systems must expose their functionality as services in terms of standardized service descriptions (here on C-BML and DIS) that are deployed through a registrar (Section 4.1). At the abstract architecture level, these services should correspond to appropriate ABBs in overarching and reference architectures. Although existing federations may use technology associated with service orientation, such as WS*, REST, etc., this is in itself not sufficient for high SOA maturity [The Open Group, 2009].

It has been argued that "SOA in practice" often does not include the registrar and only involves the lower part of the SOA triangle [Michlmayr et al., 2007, 2010]. This is certainly true for many federations such as the one above. The deployment of training federations is usually static; i.e., trainees, C2 systems and simulation systems are at distributed, but pre-planned, locations. However, it is desirable that trainees are able to complete training locally in their natural environment. This creates the need for composing training federations dynamically.

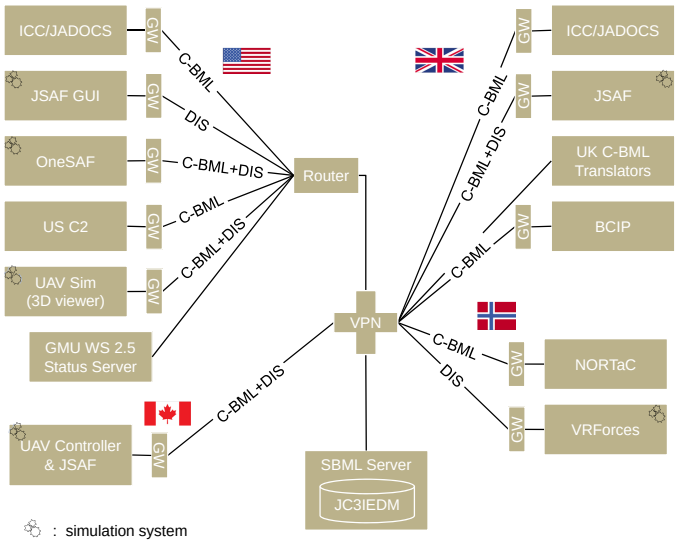

Figure 6: Demonstration of inter-system collaboration by standards between C2 and simulation systems [Pullen et al., 2012]. Systems are standards enabled using wrapper gateways (GW). Target architecture with concrete implementations.

Even more flexibility is needed in the case of simulation in support of operations, where the warfighter uses simulation, e.g., to plan an operation [Bruvoll et al., 2015; Hannay et al., 2015]. In this case, simulation systems would preferably be located on mobile platforms. Then, services must be loosely coupled with respect to location and time; i.e., the physical location of available service providers – which may vary over time – must be transparent to consumers. Further, providers must provide varying levels of quality of service to cater for mobile units that operate in and out of disadvantaged grids [Johnsen et al., 2012]. In other words, when moving the use of simulation closer to the warfighter or to operations, there will be a need for the registrar's discovery mechanisms in order to take the full advantage of service orientation.

The meaning of "dynamic" and "flexible" here is limited by current technology. Many of SOA's ideas are not possible to implement elegantly at present. It is therefore important to be aware of the following distinction: *design-time discovery* and *runtime discovery*. Today, service discovery is mostly done manually at design time [Erl, 2007]; in contrast to the true loose coupling ideal. For example, a systems designer may consult a registrar to determine whether an appropriate service implementation exists or needs to be developed. Runtime discovery is limited to consumers being able to choose among services or switch between service implementations. This limited runtime discovery is still important, since it provides loose coupling w.r.t. platform, location and time and abstracts away from method-level specificity. However, run-time discovery in the sense of a consumer checking at runtime for entirely new functionality would require semantic interoperability and more [Tolk, 2012c]; technologies which are in their infancy. Currently, more static variants of the ideal scenario are possible. For example, a consumer may probe for services, and bind to services at runtime, provided that the consumer has been sufficiently prepared at design-time for consuming those services. Thus, run-time service discovery is more akin to a service availability check, where the actual services are known at design time. See [Lund et al., 2012] for an example, where a user application gives the option to invoke available services on the fly.

In another demonstration [Allen and Schroeder, 2011; Coolahan and Allen, 2012; Allen et al., 2010], a prototype interoperability layer was developed to combine the Army Joint Land Component Constructive Training Capability (JLCCTC) Multi-Resolution Federation (MRF) and the JLCCTC Entity Resolution Federation (ERF) training system. One federate joined each federation; Joint Conflict and Tactical Simulation (JCATS) in MRF and Joint Non-Kinetic Effects Model (JNEM) in ERF.

Both MRF and ERF are HLA federations. Their stated value to the demonstration is that they have widely different FOMs and time management, and therefore challenge interoperability [Allen and Schroeder, 2011]: ERF uses an entity-centric DIS-based data model, while MRF uses an aggregate unit-centric data model based on the Aggregate Level Simulation Protocol (ALSP) [Wilson and Weatherly, 1994]. The MRF is time managed, where all simulations coordinate the advancement of simulation time. The ERF runs in real-time.

To offer coordination across these disparate federations, a service was developed for centralized monitoring to provide the status of connected federations, and a second service was developed for situational awareness providing ground truth for all entities within a geographic bounding box (defined in Universal Core (UCore); a XML format to share intelligence across U.S. government systems). In the demonstration, the situational awareness feed was used to populate a Common Operating Picture (COP) displayed inside the Google Earth environment.

To support these services, a set of common data abstraction services where developed to maintain the minimal set of data required to share object state across entities [Allen and Schroeder, 2011]. Also, development of an enumerations translation service was under way. Enumerations pertain to entity identity and representation, and problems arise when different systems use different enumeration schemes. The service is to provide a set of common methods for data producers and consumers to translate from native representations to common data representations. In this manner, common functionality for gateways may be factored out and offered as services.

This demonstration illustrates inter-simulation system coordination management offered as services; see [Drake and Morse, 2012; Drake et al., 2011] for further elaboration. In this demonstration, the concept of 'service' was more explicitly present than in the former demonstration.

In summary, for simulation environments as parts in federations:

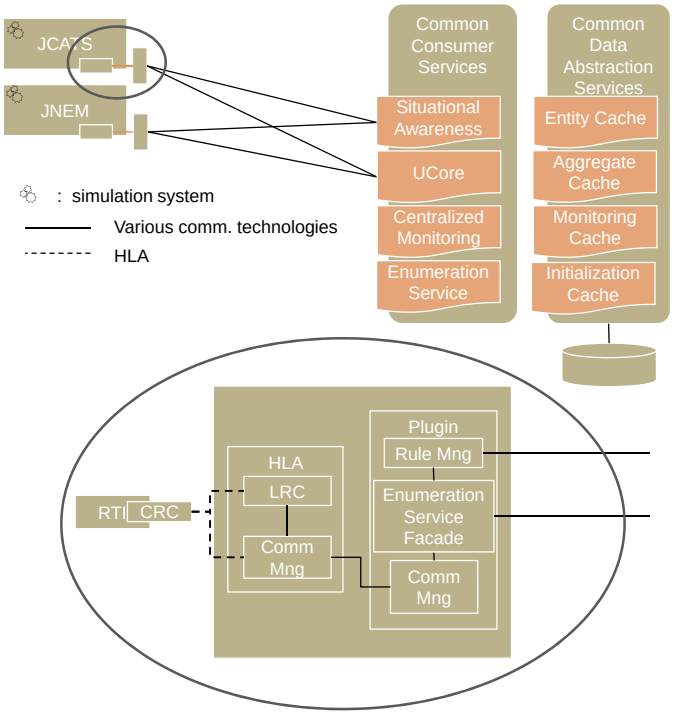

Figure 7: Demonstration of inter-system interoperability involving two HLA federations, common consumer services and common data abstraction services [Allen and Schroeder, 2011]. Gateways (standing rectangles) enable the HLA federations to participate in the overall system. More detailed view of gateway architecture in lower part of figure: The RTI is implemented as a Central Runtime Component (CRC) and Local Runtime Component (LRC), and various gateway components consume the services. Target architecture with concrete implementations (olive), with services represented by descriptions (orange).

- Existing data exchange standards enable simulation environments to participate intact in a wider federation with C2 systems.
- Components are often not defined as service providers or consumers that associate to service descriptions.
- Common coordination and gateway services can facilitate interoperability between simulation systems.
- Discovery has not been used in existing C2-simulation federations, but there are obvious operational needs for it that should trigger the development of registrar-related functionality.

4.3. Member applications as parts in a simulation environment

We now discuss whether member applications in a simulation environment can be seen as service providers. We consider two aspects: that of services within a simulation environment seen as a SOA, and that of services that give functionality across simulation environments.

4.3.1. Services within a given simulation environment

For the first aspect, what type of loose coupling do member applications in a simulation environment enjoy? In HLA, federates communicate with the RTI by means of APIs for Java and C++, or a Web Service API which exposes RTI functionality as web services within the federation [Möller and Löf, 2006; Möller and Dahlin, 2006; Möller et al., 2008]. The emerging WebLVC standard [Granowetter, 2013; Simulation Interoperability Standards Organization, 2014a] enables HLA federates to join a hosted simulation federation in a cloud via web applications.

The RTI is the central point where federates can create, discover and join federations. Thus, in a HLA federation, the RTI middleware plays a role similar to the registrar (Section 4.1). There is therefore loose binding in that federates may join or leave a federation during runtime and in that federates are not aware of other federates in the federation; all they need to do is to publish and subscribe to information. The RTI plays a crucial role in this.

Further, the RTI functions as a state management deferral mechanism, since shared state management has been delegated away from the federates to the RTI [Erl, 2007]. This gives the federates a high level of autonomy with respect to other federates, but a heavy dependency of federates on the RTI. Thus, the RTI can be seen as providing a utility service which intentionally violates the SOA ideal of statelessness so that other services may enjoy their level of state processing deferral [Erl, 2007]. The RTI is relatively active, since it is capable of administering complicated time management schemes and engages a "chatty" mode of communication.

Thus, there are technical mechanisms prescribed in HLA that support service orientation. But for federates to be viewed as service providers within a HLA federation, they must have service descriptions. In a sense, the FAD and the FOM (Section 3) holds the union of descriptions, and individual SOMs hold syntactic descriptions for federates.

So, a simulation environment can be seen as a SOA with certain caveats. Although there is a Web Service API in HLA, this is for exposing RTI functionality as services, not for federates to expose their functionality as services. Further, only publish/subscribe through a broker – the RTI – is possible at present. Although the publish subscribe concept in HLA is an effective way to obtain shared state in the federation, there are cases where a federate may request information that is only relevant for that federate at a certain point of time. One example is information on the visibility of an entity from given points in the terrain that could be given by a line-of-sight service. This information should be given by request/response, rather than published to the entire federation. Ongoing work on the next version of the HLA standard [Simulation Interoperability Standards Organization, 2016] is looking into extending to other types of interaction.

In summary:

- HLA enables loose coupling to a degree that member

applications (HLA federates) may be viewed as service providers relative to a given FOM and FAD.

- HLA's RTI can be viewed as providing service discovery and state deferral.

4.3.2. Services across simulation environments

For the second aspect, [Gustavson et al., 2005] remark that the reliance on a FOM precludes loose binding in the SOA sense. The data types of information shared in a simulation environment must be known at design time, which is a tight coupling that contrasts with SOA. Although federates do not manage the shared state, they nonetheless have to relate to relevant portions of the shared state variables. Thus, a federate cannot simply join a federation operating on a different FOM.

The idea of a HLA federate as a service provider to arbitrary simulation environments is therefore not straightforward, and in light of the above discussion, it does not help that one can use web technology to join a HLA federate to a HLA federation. For a federate to function as a service provider across FOMs, it must be able to digest and adapt to various FOMs [Gustavson et al., 2005] which seems unrealistic; particularly in light of the complexity of many FOMs in use.

However, for implementation-independent architecture, it is possible to write descriptions for services that are independent of data model. At the overarching level of abstraction, these should, for HLA, be FOM independent. In a concrete implementation, these services are provided by member applications that observe a given FOM.

Indeed, one may consider the following approach [Möller et al., 2007; Sisson et al., 2006; Chase and Gustavson, 2005; Chase et al., 2006]: The Base Object Model (BOM) template (Section 3) describes how to model objects and interactions between them. In other words, BOMs are a way to provide interface information – in XML style, and there are guidelines on how to aggregate BOMs and on how to translate BOMs to a FOM, using Extensible Stylesheet Language Transformations (XSLT). One may therefore use BOMs as simulation service descriptions, and when combining services, the aggregated BOMs can be automatically translated to the appropriate FOM necessary for the resulting federation to run. With modular FOMs, one can also extend the federation at runtime. In this manner, the declaration for shared state (the FOM) can be constructed during composition time using abstract simulation service descriptions.

Figure 8 illustrates two ideas along this line of thought. On the left hand side of the figure, is an example from [Möller et al., 2007; Gustavson et al., 2005] where the RPR-FOM is used as the basis for defining BOMs. These BOMs model states and behaviour (interaction patterns) on an abstract level independent of the RPR-FOM, and thus function as abstract service descriptions. BOMs can be automatically translated into modular FOMs using a XSLT service, generating the FOM *du jour* on the fly;

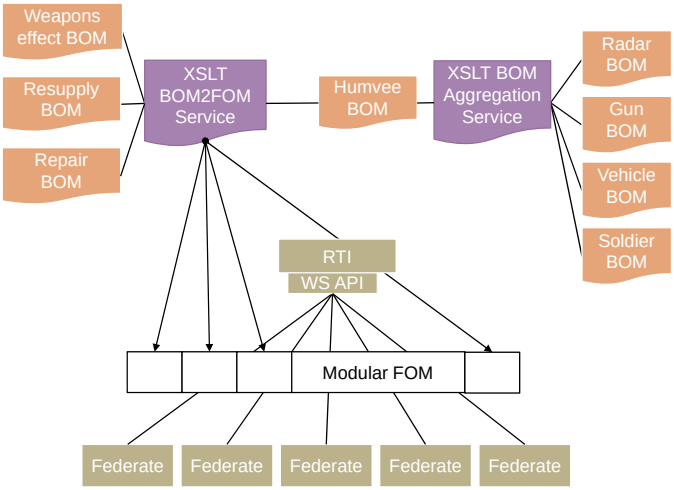

Figure 8: BOMs as service descriptions for simulation services. The RPR-FOM used as the basis for defining BOMs (orange descriptions left-hand side of figure) [Möller et al., 2007; Gustavson et al., 2005] that give rise to FOM modules via a XSLT service (violet description). Entity level BOMs (orange descriptions right-hand side of figure) (de-)aggregated via a XSLT service (violet description) [Sisson et al., 2006].

which then accommodates federates providing the services specified in the BOMs. On the right-hand side of the figure, is an example from [Sisson et al., 2006], where entity-level BOMs are defined and aggregated and de-aggregated using a XSLT service. Entity BOMs are abstract specifications for entities as a service, and composing entity services yields a new service specified by the corresponding BOM aggregation; which can then be automatically translated to a FOM module to join the FOM.

Composing components at this level of granularity faces the challenges of component-based software engineering in full; particularly the verification and validation of the composition based on the descriptions of the components. For modelling and simulation, additional issues apply; see [Tolk, 2012e]. At lower levels of granularity, components act in detail on the same state space which complicates things further [Tolk and Mittal, 2014]. The complication depends on the extent to which the components interact with each other. In the case of the aggregation/de-aggregation example shown in Figure 8, the entities may simply be aggregated so as to represent one entity without components interacting; e.g., for computational economy. When the aggregate has reached its destination, say, the entities may be deaggregated. A BOM Aggregation Framework (BAF) with a BOM Aggregation Support Server (BASS) has been developed which provides this type of aggregation/deaggregation functionality as runtime services [Sisson et al., 2006; Gustavson et al., 2005].

On the other hand, if components are to interact, then demands on interoperability and verification in shared state mount the scene. It is argued that the BOM template is not sufficient for component matching and that it gives weak support for verification and validation of a

composition; e.g, [Mahmood et al., 2011]. To amend these issues, semantic enhancements of BOMs have been developed [Mojtahed et al., 2010, 2008; Moradi et al., 2007], and methods for verifying the functionality of compositions have been researched [Mahmood et al., 2011, 2012].

In summary:

- For implementation-independent architecture, services that offer simulations can be described (using BOMs) independent of FOM.
- For implementation, relevant BOMs can be used to generate a FOM dynamically (and FAD – Section 3), so that, in theory, simulation environments can be composed from loosely coupled services.

Notice that these points imply that descriptions of services that deliver simulations (e.g., BOMs) should be FOM independent. This, in turn, implies that FOMs are shifted toward the more specific parts of the architecture framework, compared to the framework in Figure 2. We will discuss this in the next section.

4.4. Concluding observations

It is evident that principles of service-orientation are pertinent for simulation environments; both for the environment seen from the outside (Section 4.2) and for environments internally (Section 4.3). We answer our research questions as follows:

1. It is desirable and possible to promote specialized architectures as integral parts to an encompassing SOA. For defence simulation, early efforts have demonstrated a need for, and the feasibility of, federating simulation environments with defence systems. There are concerted efforts in the M&S and C2 communities to do this better, since operational personnel must be able to access M&S functionality through their regular systems used in operations. On the one hand, it is beneficial to view a simulation environment as a black box offering simulation as a service to, e.g., C2 systems. On the other hand, it is beneficial to view simulation environments as SOAs operating on highly specialized rules optimized for simulation purposes.

2. Therefore, while designing architecture at the implementation-independent level, it must be possible to model specialized architectures as integral parts of the whole, rather than dissolving the specialized architecture into the encompassing architecture.

2(a). However, the feasibility and usefulness of factoring out common functionality as services for use across simulation systems (e.g., HLA federations) means that the system boundaries of what is included in a "simulation architecture" become less clear. It is necessary to cater for various views on whether these services are part of a specialized simulation architecture or part of the encompassing federation. This becomes even more pertinent when specifying simulation services in a SDEM-independent manner by, e.g., BOMs.

2(b). Certain elements of standard service-oriented technology are usable and useful to enhance the coordination mechanisms (e.g., the RTI) in simulation architectures. For example it is useful and possible to factor out common gateway and coordination functionality for integrating simulation environments running under differing SDEMs and time management schemes. It is also useful and possible to provide entity aggregation and deaggregation as a service beyond the standard capabilities. Features of simulation middleware are definable in standard service-oriented terminology; for example, the RTI can be seen as state-deferral service relieving HLA federates from keeping track of the total state of the HLA federation. However, the RTI is finely tuned to the demands and performance criteria of simulations. Rather than attempting to replace it by standard middleware, its abstract specification – in the form of ABBs and APs – should be available in the overarching architecture. Then, functionality for combining and coordinating simulations can be architected, and then implemented as one finds fit.

5. Hybrid architecture framework

In light of the preceding discussion, we propose a hybrid architecture which caters for integrating specialized subarchitectures in an encompassing SOA. More specifically, we will instantiate the architecture framework sketched in Figure 1 into a *hybrid-architecture framework*.

5.1. Integrating architecture frameworks

Somehow, we want to integrate the simulation architecture framework in Section 3 into a larger architecture framework for SOA. One approach would be to start with the two architecture ontologies given by The Open Group SOA Reference Architecture (Section 2) and the DSEEP and to coalesce the resulting frameworks. However, on the one hand, although we do use the central concepts of ABB and AP from The Open Group SOA Reference Architecture, the entire ontology is extremely general and comprehensive so as to cover everything a SOA needs. On the other hand, the DSEEP ABBs are not service oriented.

Our solution is to use a capability taxonomy as a common modifier for both angles, in the role of both an architecture ontology and overarching architecture. We exemplify with a particular taxonomy, but the ideas are valid for any capability taxonomy that holds ABBs and APs for the domain(s) of interest.

5.2. The C3 Taxonomy

To support the transition to a capability-based and service-oriented portfolio, NATO has developed the Consultation, Command and Control (C3) Taxonomy [NATO Communications and Information Agency, 2016]; see Figure 9 for a high-level view. C3 encompasses military management capabilities, but these capabilities are also relevant for civilian authorities that handle crises, such as

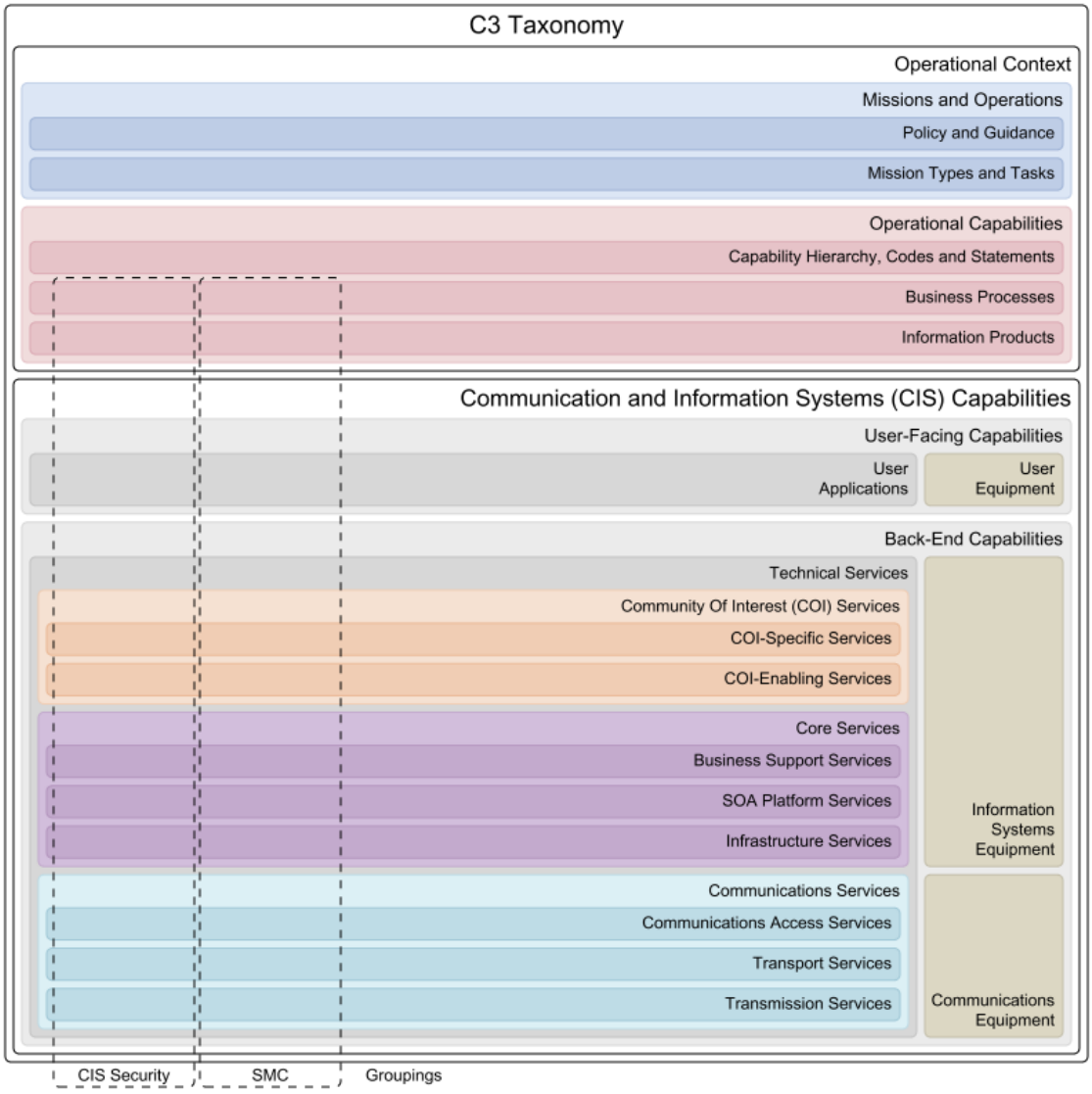

Figure 9: C3 Taxonomy – top-level view [NATO Communications and Information Agency, 2016]

emergency health authorities, police, fire brigades, customs, etc. Many of the management capabilities embody principles which are central in organization and management in any kind of business.

The taxonomy enables the defence community to sort requirements and activities into capabilities. It explicitly includes, in the same picture, the operational context (Operational Context frame) and the IT context (Communication and Information Systems (CIS) Capabilities frame). The Operational Context is layered into Capability Hierarchy, Codes and Statements which is a mapping out of operational capabilities together with high-level requirements describing what the capabilities are, or should be. The operational capabilities are supported by the two layers below; Business Processes and Information Products.

The Communication and Information Systems (CIS) Capabilities present themselves to the end user in the form of User-Facing Capabilities geared toward User Applications for specific defence domains (air, land, maritime, joint, etc.) and communities of interest (M&S, environment, missile defence, etc.). Below this layer are various layers of Back-End Capabilities, which may be used to support the user-facing capabilities. The Back End Capabilities are layered into Community of Interest (COI) Services, subdivided into COI-Specific Services and the more generic COI-Enabling Services. The COI Services are supported by Core Services and Communication Services, both of which provide generic infrastructure services.

Two cross-cutting concerns are defined in the taxonomy. The CIS Security grouping holds functionality for safety and security. The Service Management and Control (SMC) grouping holds functionality for managing and federating a SOA, such as service discovery, mediation and quality of service management.

Figure 10 shows a closer detail of the taxonomy for CIS capabilities; which will be our focus. For example, M&S is explicitly represented at the User Applications, COI-Specific

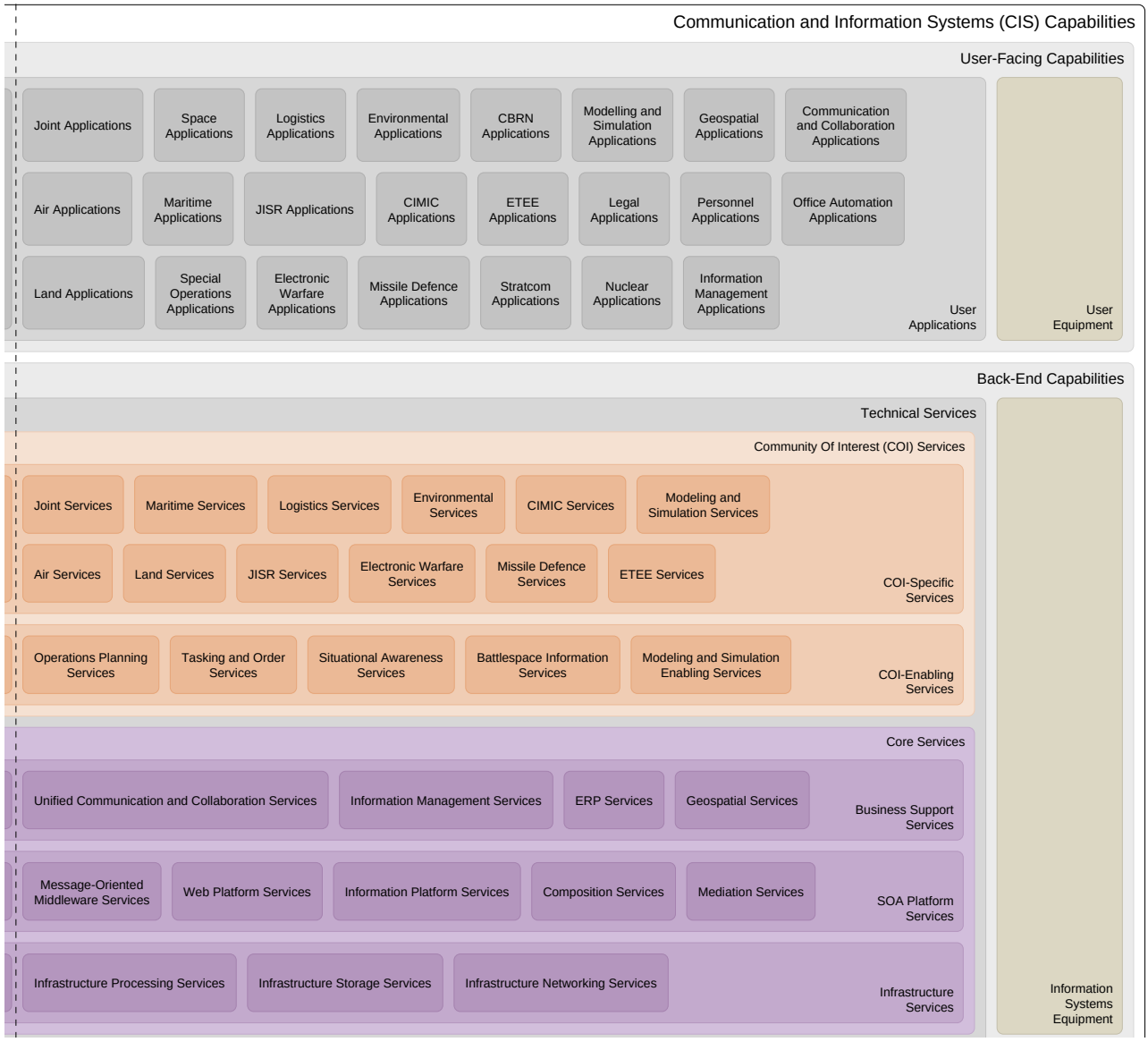

Figure 10: C3 Taxonomy (detail of CIS Capabilities) [NATO Communications and Information Agency, 2016]

Services and COI-Enabling Services levels. Each category (oval box) represents a division into capabilities and is further divided into sub-categories; i.e, sub-capabilities. At the leafs of these capability trees, one finds individual operational processes (under Operational Capabilities), user applications (under User-Facing Capabilities) and services (under Back-End Capabilities). For example, the M&S Enabling Services have sub-capabilities Battlespace Simulation Services, Radio Simulation Services, Ground Truth Battlespace Objects Services and Ground Truth Battlespace Events Services that divide M&S functionality into meaningful groups of services. The taxonomy is interfaced by a semantic wiki which allows one to declare capabilities and to describe them in terms of abstract implementation-independent requirements.

The taxonomy's high-level layers (Operational Capabilities, User-Facing Capabilities, Back-End Capabilities) constitute ABB types and the C3 Taxonomy is thereby an architecture ontology in terms of Figure 1. Further, the capabilities at various levels of detail and refinement down to processes, applications and services constitute C3 ABBs of the various types. In the wiki, one also finds APs that show how the C3 ABBs are intended to interoperate. Thus, the taxonomy also plays the role of an overarching architecture for the C3 domain. More specifically, the taxonomy can be used as repository of ABBs and APs that the defence communities can build and refine over time.

In the following, we will use colour coding in figures according to the C3 Taxonomy. Incidently, we used the colours in Figures 7 and 8 already.

5.3. A service-oriented repository

For the hybrid architecture framework, the C3 Taxonomy (or a similar taxonomy) can be use as a repository for ABBs and APs for both the simulation environment and

the encompassing SOA. This gives a uniform framework for defence portfolio management and development. As argued in [Hannay et al., 2016], this is important for being able to coordinate and prioritize smaller development projects within a portfolio, and in particular, to facilitate the coordinated integration of M&S – and other specialized development efforts – within a larger portfolio.

Using a capability taxonomy as a repository for ABBs and APs also means that the most dynamic part of the architecture framework is factored out. It is the responsibility of the domain community as a whole, rather than just the architects, to keep such as taxonomy updated in line with the evolving understanding of capabilities.

As a service-oriented repository, the C3 Taxonomy contains ABBs that should persist over various implementations, over time and over various contexts of use. Such ABBs only persist in terms of their abstract implementation-independent descriptions; analogously to what we said for services (Section 4.1). As argued in [Hannay et al., 2016], the C3 Taxonomy should be populated with these descriptions – or more extensively, with requirements at the degrees of detail and refinement corresponding to the structure of the taxonomy. Thus, we here speak of ABBs in terms of their descriptions. A C3 Taxonomy ABB of the type User-Application or COI Service, for instance, is an implementation-independent description, not a piece of software that provides that service.

For simulation environments to be service-oriented, this means two things. In general, member applications should be represented by their descriptions, namely BOMs. Taking the step further, since member applications are pieces of software they should be seen as service providers (Section 4.1) and the BOMs of interest are those that describe these services, not the member applications as such. Furthermore, these BOMs can be deposited in the C3 Taxonomy along with other descriptions for the larger SOA.

5.4. The hybrid topology

With a common architecture ontology and a common overarching architecture provided by the C3 Taxonomy, the main enforcer of a hybrid structure becomes the topology element at the reference architecture level. We exemplify with an HLA topology, but other topologies are of course, possible; e.g., a more general simulation environment topology, another simulation standard/protocol topology; e.g., for DIS or TENA, or a topology that integrates multiple simulation architectures as specified in the DSEEP multi-architecture overlay (DMAO) [IEEE Standards Association, 2013]. This hybrid approach is applicable to specialized architectures other than for simulation, and topologies for those approaches can then be provided.

Figure 11 illustrates the hybrid architecture framework with the C3 taxonomy for simulation environments specialized to HLA. The upper-level structure of the C3 Taxonomy (User-Facing Applications, COI-Services, Core Services) is an architecture ontology, while the deeper more detailed and refined levels amount to an overarching architecture and functions as a repository for C3 ABBs and APs. These ABBs are abstract descriptions of user applications and services. Here, NAF (Section 2), or some other description framework, is used as a generic standard for ABBs and APs. For the specialized architecture capabilities, BOMs (Section 4.3) expressed in NAF diagrams is used for ABBs and APs.

Then, hybrid architecture topologies, here for HLA simulation environments, enforce boundaries for specialized architectures in an encompassing SOA. The topology delineates areas of interoperability with special emphasis on the specialized architecture. ABBs and APs are used to design reference hybrid architectures and hybrid target architectures.

Although we exemplify the hybrid architecture framework with a HLA simulation environment, the framework can be used for any specialized federation within an encompassing service-oriented federation. In general, the following three principles apply:

1. *Service-oriented specialized federations*: Federations adhering to a specialized architecture are components in a wider service-oriented federation adhering to a different architecture. Designated components within the specialized federation expose the federation's functionality as services and/or user applications through standardized service and user application descriptions, and consume services from the encompassing federation by the same standards.
2. *Service-oriented specialized components*: Components in specialized federations are seen as service providers internally to the specialized federation if the design principles of the specialized federation have sufficient service-oriented features for declaring service descriptions and for ensuring loose coupling in the SOA sense.
3. *General service orientation*: All components – both within the specialized federation and with respect to the encompassing federation – can consume services *ad lib* in the encompassing federation by means of standards that enable service orientation.

The concept of providing the entire specialized federation as a service is encompassed by the above. Providing large pieces of stateful functionality that persist for longer sections of time goes beyond the "micro services" style of request-response and publish-subscribe as perhaps commonly associated with SOA, but is, all the same, supported by the hybrid reference architecture. The provision of federations as a service (FaaS) in this way enters the realm of cloud computing and gives the associated possibilities for deployment in cloud environments.

6. Simulation Support for Operations Planning

We will now give a defining example of how one might use the hybrid architecture framework sketched in Figure 11.

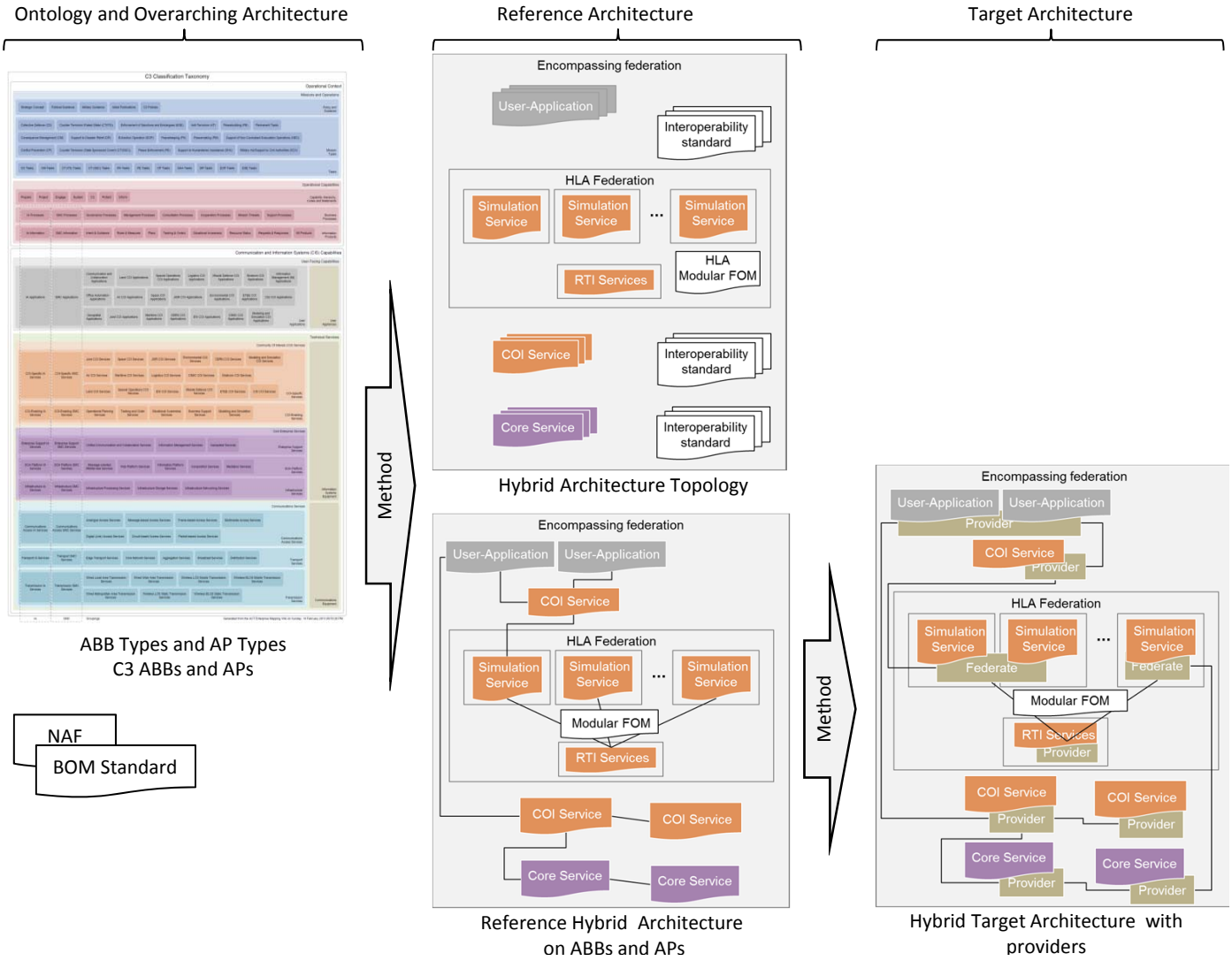

Figure 11: Hybrid-Architecture Framework with C3 Taxonomy for HLA simulation environment. The C3 Taxonomy, as a repository for C3 ABBs and APs, functions as both architecture ontology and overarching architecture. NAF and the BOM standard are chosen for writing ABBs and APs. Hybrid architecture topologies, here for HLA simulation environments, enforce boundaries for specialized architecture. Reference architectures are designed using ABBs (user application and service descriptions) and APs, from which target hybrid architectures with concrete systems (olive) can be designed.

6.1. Topology

For this example, we declare a topology for HLA federations in a C2SIM encompassing environment; see Figure 12. The topology states that the MSDL and C-BML standards (Section 4.2) are to be used for interoperability between C2 systems (which commanders use to monitor and control operations) and simulations, and that HLA and RPR-FOM standards are to be used for simulation systems. Further, the NATO Interoperability Standards and Profiles (NISP) document [NATO Consultation, Command and Control Board, 2014] should be consulted for interoperability standards for COI and Core services.

C2 systems offer user-facing capabilities by their being user applications. Descriptions for these would reside in the appropriate C3 Taxonomy User-Applications categories. Although existing systems implementing these descriptions may be obese and stove piped, the descriptions can be used subsequently to develop thin clients that offer the same, or better, user-facing capabilities consuming services in the layers below, more in line with SOA.

It further states that the HLA federation is exclusively that which observes the HLA protocol. This may include federates, such as gateways, that interoperate with other standards, but does not include components that do not relate to HLA in the encompassing federation. For example, a MSDL/CBML gateway that enables C2 systems to interoperate with a HLA federation is part of the federation, but the C2 system is not; unless is connects directly to HLA. This is somewhat in line with [Heffner et al., 2014]. However, this relates to a general ongoing discussion on what a "simulation environment" as specified in DSEEP and the FEAT should encompass, and there are other

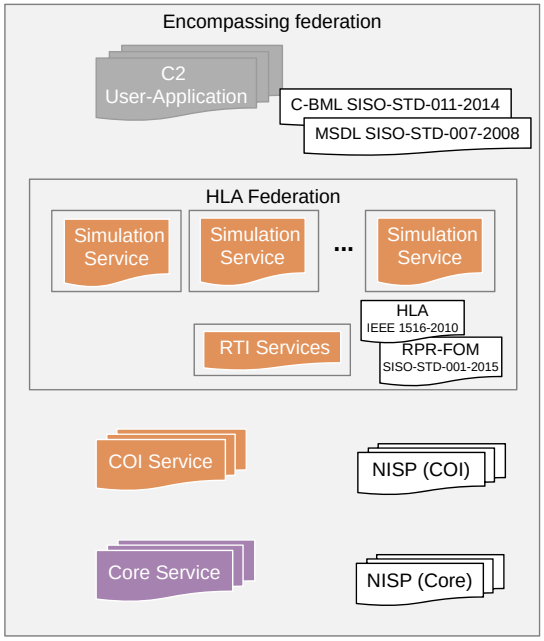

Figure 12: C2SIM hybrid architecture topology. C2 systems communicate over interoperability standards with HLA federations, which adhere to their own specialized interoperability standard. Other interoperability standards may also be at play. Colour coding according to C3 Taxonomy.

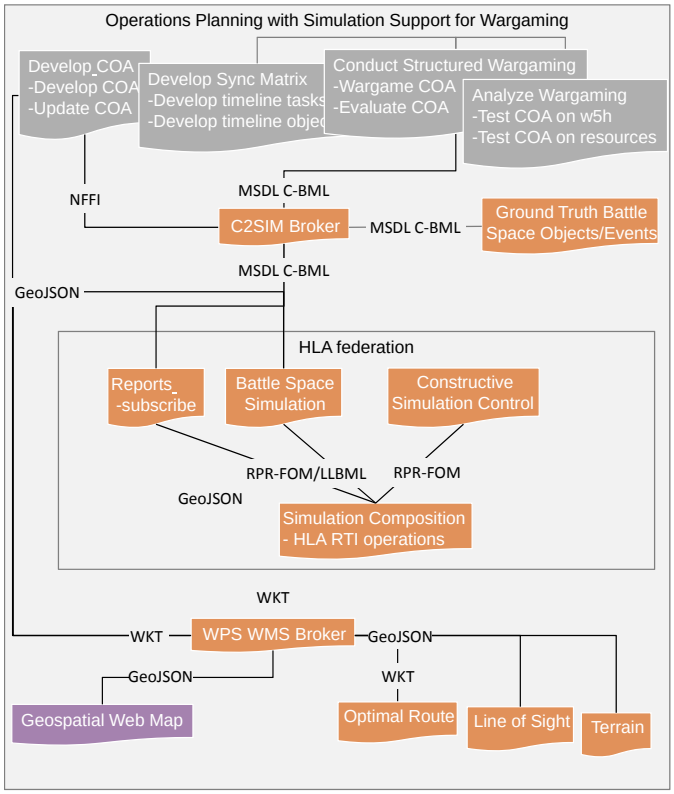

Figure 13: Reference hybrid architecture for "Simulation Support for Operations Planning" with architecture building blocks from C3 Taxonomy.

views than that of our example. For instance, the NATO M&S task group MSG-136 "M&S as a Service (MSaaS)" is considering extending the concept of simulation environment to include components that provide simulation services regardless of protocol, while excluding components that provide real-world services. The salient point here is that those other views can be accommodated, and should be made precise, by corresponding topologies at the reference architecture level.

For our example, a topology with more than one specialized architecture could have been declared. For example, one might want to clearly delineate a cluster of C2 systems that relate to a C2 interoperability standard, in addition to delineating the simulation environment. A reference architecture is a pragmatic tool, and in our example it suffices to delineate the simulation environment.

6.2. Reference architecture

Moving on, we pick ABBs from our repository – the C3 Taxonomy. We want to design a reference architecture for federations of systems that enable operations planners to play out their plan in a simulation. For the full design, one should start at the operational level in our repository with operations planing ABBs situated in the Business Process category (Figure, 9). For brevity, we are focussing on the CIS aspect of architecture. In Figure 13, we have assembled ABBs from the User Applications, COI Services and Core Services layers.

The User Applications ABBs are descriptions of pieces of user-facing functionality in, say a C2 system, that operations planners use for planning. This functionality supports planners in developing alternative courses of action – Develop Course of Action (COA), in developing a synchronization matrix for coordinating movements of forces and equipment – Develop Sync Matrix, in acting out the various courses of action – traditionally done on a large table-top map with toy-sized figures, but now moving to computer simulations – Conduct Structured Wargaming and in analysing wargaming results – Analyze Wargaming. The lines interconnecting the ABBs indicate APs; i.e., patterns of interplay between the pieces of functionality. The C3 Taxonomy is under construction and will always be in flux due to changing requirements. Most ABBs and some APs that we show here reside in the taxonomy, while others may be candidates for future inclusion.

The User Applications ABBs are dependent on a number of services – dependencies which should be specified as APs in the taxonomy. Here, we only show the interaction with the simulation environment, which presents itself as a service.

The simulation, which is a HLA federation, is built up from four ABBs: The Constructive Simulation Control ABB represents the simulation engine. The Battle Space Simulation ABB exposes the simulation environment as a service by specifying service operations to decompose orders implied by a COA into lower-level commands for entities (e.g., vehicles) and entity interactions that enable detailed simu-

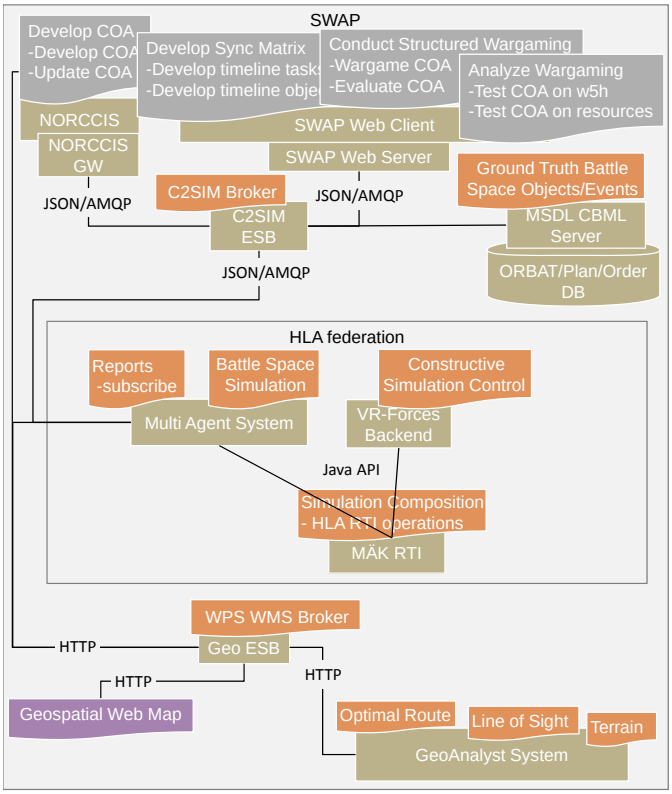

Figure 14: Target hybrid architecture for "Simulation Support for Operations Planning" with concrete systems in olive.

lation of a course of action; for this a FOM module for low-level BML (LLBML) is necessary [Alstad et al., 2013]. The Reports ABB also exposes the simulation environment as a service by specifying service operations for subscribing to reports on events in the simulation. These ABBs would reside in the COI Services layer.

A C2SIM Broker ABB facilitates interoperability between C2 systems and the HLA federation. It also handles storage and retrieval of plans and orders on a Ground Truth Battle Space Objects/Events ABB. The Develop COA ABB uses the NATO Friendly Force Information *de facto* standard [North Atlantic Treaty Organization, 2014] for updating the position of forces.

Distributed systems raise the challenge of verifying and validating the composition of its parts. As mentioned in Section 4.3.2, composing simulations from services (which are very loosely coupled) adds to this challenge; even when service implementations are verified and validated according to their service descriptions. However, the service approach itself is an important means to meet this challenge, as it enables factoring out common functionality that must be equivalent across all systems. In this example, where simulations and C2 systems interoperate, both C2 user application ABBs and simulation ABBs consume common geospatial services: Optimal Route for route planning, Line of Sight for assessing coverage from enemy detection, and Terrain for terrain data. Thus planning personnel develop plans, and the simulation plays the plans, using the same models. This use of common functionality increases consistency across systems and reduces the burden of verification and validation.

These common services, which reside outside the simulation environment, are stateless, and necessary data from the HLA federation is provided as input to the services. The services are brokered by a Web Processing Services (WPS) [Open Geospatial Consortium Inc., 2007] and Web Map Services (WMS) [International Organization for Standardization, 2005] WPS WMS Broker ABB that manages interoperability between geospatial standards such as GeoJSON [Butler et al., 2008] and Well-known Text (WKT) [International Organization for Standardization, 2015].

This assembly of ABBs and APs is implementation independent but at the same time focused on a certain type of use and domain. It is a reference architecture for a concept of "Simulation Support for Operations Planning".

6.3. Target architecture

From this reference architecture, various target architectures can be designed. Figure 14 illustrates one possibility of concrete systems that provide the functionality specified in the ABBs. The User Applications ABBs are implemented by a C2 system (NORCCIS) and a web-based front-end for simulation-supported wargaming (since the particular C2 system does not support C2SIM interoperability and is used for developing COAs only). In the future, all this user-facing functionality could rather be implemented in the form of loosely coupled apps to be downloaded in deployable and mobile devices. In the HLA federation, a Multi Agent System provides both the Reports and Battle Space Simulation services, and VR-Forces (a configurable COTS simulation framework) provides simulation control. A GeoAnalyst System developed at FFI provides the geospatial services. The Web Map Services (violet) shall be consumed from elsewhere so have no implementation here. Service-oriented communication technology choices (Section 4.1) are indicated. An earlier version of this concrete federation is implemented and is described in [Bruvoll et al., 2015] under the name of "Simulation-Supported Wargaming for Analysis Plan" (SWAP).

7. LVC Simulation for Operations Training

For further illustration, we show another reference hybrid architecture based on the C2SIM topology in Figure 12.

So-called *Live* simulation has live personnel using live equipment, but where, e.g., live ammunition is replaced by laser pulses and detectors networked with positional data so as to provide a comprehensive digitized view of events. *Virtual* simulation has live personnel in a virtual environment (e.g., vehicle manoeuvre in gaming), and *Constructive* simulation has totally computer-simulated entities. Combining the three modes in so-called LVC simulation in a distributed system is assumed to increase training

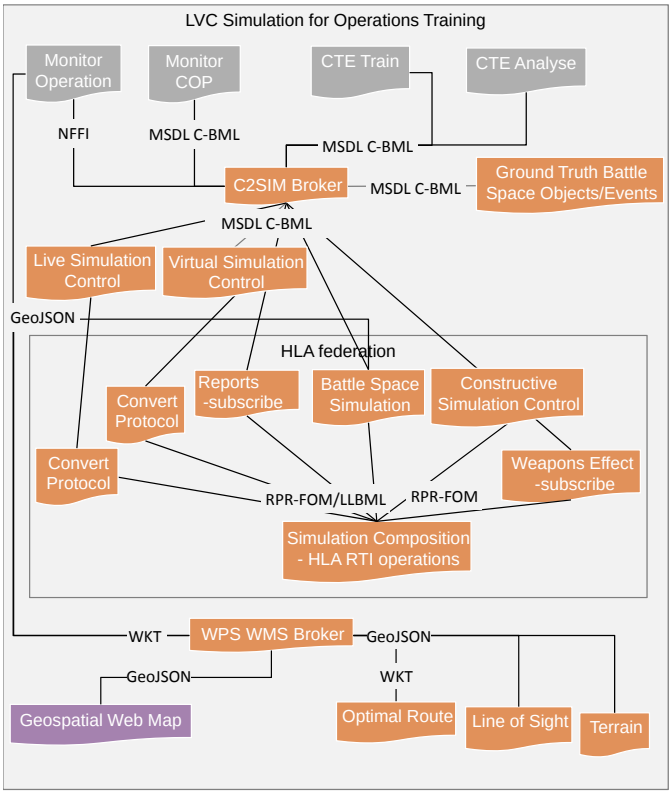

Figure 15: Reference hybrid architecture for "LVC for Operations Training" with architecture building blocks from C3 Taxonomy.

capabilities and training effect. In Figure 15, ABBs representing *Live* and *Virtual* (as black boxes) are federated with the *Constructive* HLA federation via Convert Protocol ABBs.

During training in a simulated environment, simulated entities should be fed to the training forces' C2 systems (which commanders use to monitor and control operations) and Battle Management Systems (BMS) (which war-fighting personnel use to monitor positions of own forces and detected opposing forces). Thus, a Monitor Operation ABB and a Monitor Common Operational Picture ABB interoperates with the LVC simulation systems via a C2SIM Broker ABB. Exercise Control personnel use the Collective Training and Education (CTE) Train and CTE Analyse ABBs to control, monitor and analyse the exercise. The exercise can be initialized by retrieving plans and orders (perhaps generated during "Simulation Support for Operations Planning") from the Ground Truth Battle Space Objects/Events ABB.

Another aspect of the distributed systems challenge when federating multiple simulations, is that simulation engines operate on their own (often proprietary) representations and models. This entails that simulated entities may be treated unequally in the different systems. By bypassing proprietary functionality by using common services, the "fair fight" criterion can be met to the extent that all systems can use the same environmental data, the same models for simulating on that data and the same user-facing capabilities to render the effects. This ensures that players across simulations experience objects (e.g., vegetation for cover from enemy detection) and interactions (e.g., effects of artillery strikes on armoured vehicles) equivalently. In this example, in addition to shared geospatial ABBs, a Weapons Effects ABB offers subscription to centralized damage assessment for use both within and outside the simulation environment.

This assembly of ABBs and APs is a reference architecture for a concept of "LVC Simulation for Operations Training". A federation which implements aspects of this reference architecture, but with little use of service-oriented technology, is described in [Hannay et al., 2014].

8. Conclusion

There are ongoing discussions on the role of simulation environments. For example, since following and implementing HLA can be laborious, more lightweight looser solutions are often sought for. One contemplates whether the simulation environment can be dissolved completely and RTI functionality replaced by some other standard middleware. We do not answer that question on the implementation level, but at the implementation-independent architecture level, we hold that it is necessary to cater for the simulation environment as an integral logical unit relative to any larger context, regardless of choices of implementation. On the one hand, it is beneficial to view a simulation environment as a black box offering simulation as a service to, e.g., C2 systems. On the other hand, it is beneficial to view simulation environments as SOAs operating on highly specialized rules optimized for simulation purposes. In both cases, dissolving the simulation environment into the surrounding context will not enable these two views.

To cater for this, we proposed a hybrid architecture framework – derived from a a general architecture framework – which allows specialized architectures to reside intact in a larger SOA. Service orientation and cloud technology are leveraged strategically, and our hybrid architecture framework supports offering the specialized federation as a service to the encompassing federation and also viewing the specialized federation itself as composed of services. All this can be modelled coherently and cohesively using domain overarching repositories for ABBs and APs such as the C3 Taxonomy. Further, hybrid architecture topologies can be used as a tool to delineate specialized architectures.

This work has put forth concepts for better architectural work. These concepts should be further refined and elaborated together with empirical studies, iteratively and incrementally. In addition, executable architecture models [Wagenhals and Levis, 2009], and more recently, executable models of federations of SOAs [Abusharekh et al., 2011], can be used for studying performance issues; particularly pertinent for simulations in hybrid architectures. We encourage instantiations of the hybrid architecture framework and our research questions in other domains. We

encourage architects and developers to use the reference hybrid architecture outlined here and to provide feedback on their experiences.

Several "reference architectures" at diverse levels of abstraction for modelling and simulation in the context of service orientation have been put forth in the defence domain [NATO Science and Technology Organisation, 2013; Neugebauer et al., 2009]. We think it would be beneficial for further work to consolidate and coordinate these efforts in the hybrid architecture framework and its wide span of architecture abstraction.

References

Abusharekh, A. M., Gloss, L. E., Levis, A. H., 2011. Evaluation of service oriented architecture-based federated architectures. Systems Engineering 14 (1), 56–72.

Ahmed, M. M., Letchmunan, S., 2015. A systematic literature review on challenges in service oriented software engineering. Int'l J. Software Engineering and Its Applications 9 (6), 173–186.

Allen, G. W., Lutz, R., Richbourg, R., 2010. Live, virtual, constructive, architecture roadmap implementation and net-centric environment implications. ITEA Journal 31 (3).

Allen, G. W., Schroeder, L., 2011. Utilization of Service Oriented Architecture (SOA)-based commercial standards to address Live, Virtual, Constructive (LVC) interoperability challenges. In: Proc. Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC) 2011. National Training and Simulation Association.

Alstad, A., Mevassvik, O. M., Nielsen, M. N., øvlid, R. A., Henderson, H. C., Jansen, R. E. J., Reus, N. M. d., 2013. Low-level battle management language. In: Proc. 2013 Spring Simulation Interoperability Workshop (SIW). Simulation Interoperability Standards Organization.

Bloebaum, T. H., Hannay, J. E., Hedenstad, O.-E., Haavik, S., Lillevold, F., 2013. Architecture for the Norwegian Defence Information infrastructure (INI) – remarks on the C3 Taxonomy. FFI-rapport 2013/01729, Forsvarets forskningsinstitutt.

Bruvoll, S., Hannay, J. E., Svendsen, G. K., Asprusten, M. L., Fauske, K. M., Kvernelv, V. B., Løvlid, R. A., Hyndøy, J. I., 2015. Simulation-supported wargaming for analysis of plans. In: Proc. NATO Modelling and Simulation Group Symp. on M&S Support to Operational Tasks Including War Gaming, Logistics, Cyber Defence (STO-MP-MSG-133).

Butler, H., Daly, M., Doyle, A., Gillies, S., Schaub, T., Schmidt, C., 2008. The GeoJSON Format Specification.

Chase, T., Gustavson, P., 2005. RPR-BOM initiative: Providing a set of applicable BOMs to the M&S community. In: Proc. 2005 Spring Simulation Interoperability Workshop (SIW). Simulation Interoperability Standards Organization.

Chase, T., Gustavson, P., Root, L., 2006. From FOMs to BOMs and back again. In: Proc. 2006 Spring Simulation Interoperability Workshop (SIW). Simulation Interoperability Standards Organization.

Cloutier, R., Muller, G., Verma, D., Nilchiani, R., Hole, E., Bone, M., 2010. The concept of reference architectures. Systems Engineering 13 (1), 14–27.

Coolahan, J. E., Allen, G. W., 2012. LVC Architecture Roadmap Implementation—results of the first two years. In: Proc. 2012 Spring Simulation Interoperability Workshop (SIW). Simulation Interoperability Standards Organization.

Danesh, M. H., Yu, E., 2015. Analyzing IT flexibility to enable dynamic capabilities. In: Advanced Information Systems Engineering Workshops. Vol. 215 of Lecture Notes in Business Information Processing. Springer, pp. 53–65.

Dragoicea, M., Bucur, L., Wei-Tek, T., Sarjoughian, H., 2012. Integrating HLA and service-oriented architecture in a simulation framework. In: Proc. 12th IEEE/ACM Int'l Symp. Cluster, Cloud and Grid Computing (CCGrid).

Drake, D. L., Martins, I. X., Roca, R. A., Carr, F., 2011. Live-Virtual-Constructive Service-Oriented Architecture. Service-Oriented Architecture application to Live-Virtual-Constructive simulation: Approach, benefits, and barriers. Tech. Rep. NSAD-R-2011-025, National Security Analysis Department, The Johns Hopkins University, Applied Physics Laboratory.

Drake, D. L., Morse, K. L., 2012. Use of SOA for distributed simulation: A way forward. In: Proc. 2012 Spring Simulation Interoperability Workshop (SIW). Simulation Interoperability Standards Organization.

Ecma International, 2013. ECMA-404 – The JSON Data Interchange Format.

Erl, T., 2007. SOA principles of Service Design. Prentice Hall.

Fette, I., Melnikov, A., 2011. The WebSocket Protocol—Internet Engineering Task Force (IETF) Request for Comments: 6455.

Fielding, R. T., Taylor, R. N., 2002. Principled design of the modern web architecture. ACM Transactions on Internet Technology 2 (2).

Garnier, J.-L., Bischoff, L., André, M., Lavit, B., Peyrichon, M., Blanquart, J., Scuto, N., 2013. Architecture frameworks – a standard to unify terms, concepts, life-cycles and principles. In: Proc. NATO Information Systems Technology Panel Symp. on Architecture Definition and Evaluation (STO-MP-IST-115).

Granowetter, L., 2013. The WebLVC protocol: Design and rationale. In: Proc. Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC) 2013. National Training and Simulation Association.

Grogan, P. T., de Weck, O. L., 2015. Infrastructure system simulation interoperability using the High-Level Architecture. IEEE Systems Journal 99, 1–12.

Gustavson, P., Chase, T., Root, L., Crosson, K., 2005. Moving towards a Service-Oriented Architecture (SOA) for distributed component simulation environments. In: Proc. 2005 Spring Simulation Interoperability Workshop (SIW). Simulation Interoperability Standards Organization.

Hannay, J. E., Brathen, K., Hyndøy, J. I., 2015. On how simulations can support adaptive thinking in operations planning. In: Proc. NATO Modelling and Simulation Group Symp. on M&S Support to Operational Tasks Including War Gaming, Logistics, Cyber Defence (STO-MP-MSG-133).

Hannay, J. E., Bråthen, K., Mevassvik, O. M., 2013. Simulation architecture and service-oriented defence information infrastructures – preliminary findings. FFI-rapport 2013/01674, Forsvarets forskningsinstitutt.

Hannay, J. E., Brathen, K., Mevassvik, O. M., 2016. Agile requirements handling in a service-oriented taxonomy of capabilities. Requirements EngineeringOnline `http://dx.doi.org/10.1007/s00766-016-0244-8`.

Hannay, J. E., Mevassvik, O. M., Skjeltorp, A., Brathen, K., 2014. Live, Virtual, Constructive (LVC) simulation for land operations training: Concept development & experimentation (CD&E). In: Proc. NATO Modelling and Simulation Group Symp. on Integrating Modelling & Simulation in the Defence Acquisition Lifecycle and Military Training Curriculum (STO-MP-MSG-126).

Heffner, K., Mevassvik, O. M., Gautreau, B., De Reus, N., 2014. A proposed process and toolset for developing standardized C2-to-simulation interoperability solutions. In: Proc. NATO Modelling and Simulation Group Symp. on Integrating Modelling & Simulation in the Defence Acquisition Lifecycle and Military Training Curriculum (STO-MP-MSG-126).

Hilliard, R., 2013. The role of architecture frameworks: Lessons learned from ISO/IEC/IEEE 42010. Unpublished.

IEEE Standards Association, 2010a. 1516-2010 – IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA).

IEEE Standards Association, 2010b. 1730-2010 – IEEE Recommended Practice for Distributed Simulation Engineering and Execution Process (DSEEP).

IEEE Standards Association, 2013. 1730.1-2013 – IEEE Recommended Practice for Distributed Simulation Engineering and Execution Process Multi-Architecture Overlay (DMAO).

IEEE Standards Association, 2015. 1278.2–2015 – IEEE Standard for Distributed Interactive Simulation (DIS) – Communication Services and Profiles.

International Organization for Standardization, 2005. ISO 19128:2005 Geographic information – Web Map Server interface.

International Organization for Standardization, 2011. ISO/IEC 42010:2007 Systems and software engineering – Architecture description.

International Organization for Standardization, 2015. ISO 19162:2015 Geographic information – Well-known text representation of coordinate reference systems.

Johnsen, F. T., Bloebaum, T. H., Brannsten, M. R., 2012. Quality aspects of Web services. FFI-rapport 2012/02494, Forsvarets forskningsinstitutt.

Kuhl, F., Weatherly, R., Dahmann, J., 1999. Creating Computer Simulations—An Introduction to the High Level Architecture. Prentice Hall PTR.

Li, Z., Zhang, H., O'Brien, L., Dec 2010. Facing service-oriented system engineering challenges: An organizational perspective. In: 2010 IEEE Int'll Conf. Service-Oriented Computing and Applications (SOCA). pp. 1–4.

Lund, K., Johnsen, F. T., Bloebaum, T. H., Skjervold, E., 2012. SOA pilot 2011—service infrastructure. FFI-rapport 2011/02235, Forsvarets forskningsinstitutt.

Mahmood, I., Ayani, R., Vlassov, V., Moradi, F., 2011. Fairness verification of BOM-based composed models using Petri nets. In: Proc. 11th IEEE Workshop on Principles of Advanced and Distributed Simulation. IEEE Computer Society, pp. 1–8.

Mahmood, I., Ayani, R., Vlassov, V., Moradi, F., 2012. Verifying dynamic semantic composability of BOM-based composed models using colored Petri nets. In: Proc. 2012 ACM/IEEE/SCS 26th Workshop on Principles of Advanced and Distributed Simulation (PADS). IEEE Computer Society, pp. 250–257.

Michlmayr, A., Rosenberg, F., Leitner, P., Dustdar, S., 2010. End-to-end support for QoS-aware service selection, binding, and mediation in VRESCo. IEEE Transactions on Services Computing 3 (3), 193–205.

Michlmayr, A., Rosenberg, F., Platzer, C., Treiber, M., Dustdar, S., 2007. Towards recovering the broken SOA triangle—A software engineering perspective. In: Proc. 2nd Int'l Workshop Service Oriented Software Engineering (IW-SOSWE'07). ACM, pp. 22–28.

Mojtahed, V., Andersson, B., Kabilan, V., Zdravkovic, J., 2008. BOM++, a semantically enriched BOM. In: Proc. 2008 Spring Simulation Interoperability Workshop (SIW). Simulation Interoperability Standards Organization.

Mojtahed, V., Svee, E.-O., Zdravkovic, J., 2010. Semantic enhancements when designing a BOM-based conceptual model repository. In: Proc. 2010 European Simulation Interoperability Workshop (SIW). Simulation Interoperability Standards Organization.

Möller, B., Dahlin, C., 2006. A first look at the HLA Evolved Web Service API. In: Proc. 2006 European Simulation Interoperability Workshop (SIW). Simulation Interoperability Standards Organization.

Möller, B., Gustavson, P., Lutz, R., Löfstrand, B., 2007. Making your BOMs and FOM modules play together. In: Proc. 2007 Fall Simulation Interoperability Workshop (SIW). Simulation Interoperability Standards Organization.

Möller, B., Löf, S., 2006. A management overview of the HLA Evolved Web Service API. In: Proc. 2006 Fall Simulation Interoperability Workshop (SIW). Simulation Interoperability Standards Organization.

Möller, B., Morse, K. L., Lightner, M., Little, R., Lutz, R., 2008. HLA Evolved—a summary of major technical improvements. In: Proc. 2008 Fall Simulation Interoperability Workshop (SIW). Simulation Interoperability Standards Organization.

Moradi, F., Ayani, R., Mokarizadeh, S., Shahmirzadi, G. H. A., Tan, G., 2007. A rule-based approach to syntactic and semantic composition of BOMs. In: Proc. 11th IEEE Int'l Symp. Distributed Simulation and Real-Time Applications (DS-RT 2007). IEEE Computer Society, pp. 145–155.

Multilateral Interoperability Programme, 2015. MIP Website. Accessed June 2015.

NATO Communications and Information Agency, 2016. The C3 Taxonomy. Accessed January 2016.

NATO Consultation, Command and Control Board, 2014. NATO Interoperability Standards and Profiles, v8.0.

NATO Modelling and Simulation Group, 2012. NATO Modelling and Simulation Master Plan (version 2.0). Doc. no. AC/323/NMSG(2012)-015.

NATO Research and Technology Organisation, 2012. NATO Education and Training Network. Technical Report RTO-TR-MSG-068.

NATO Science and Technology Organisation, 2013. Modelling and simulation as a service: New concepts and service-oriented architectures. Technical Report STO-TR-MSG-131 AC/323(MSG-131)TP/608.

Neugebauer, E., Nitsch, D., Henne, O., 2009. Architecture for a distributed integrated test bed. In: Proc. NATO Modelling and Simulation Group Symp. on Current Uses of M&S Covering Support to Operations, Human Behaviour Representation, Irregular Warfare, Defence against Terrorism and Coalition Tactical Force Integration (RTO-MP-MSG-069).

North Atlantic Treaty Organization, 2014. STANAG 5527 (draft) NATO Friendly Force Information.

North Atlantic Treaty Organization, 2016. NATO Architecture Framework v4.0 Documentation (draft).

Open Geospatial Consortium Inc., 2007. OpenGIS Web Processing Service (WPS) Interface Standard.

Organization for the Advancement of Structured Information Standards, 2012. Advanced Message Queuing Protocol (AMQP) Version 1.0.

Pan, K., Turner, S. J., Cai, W., Li, Z., 2007. A service oriented HLA RTI on the grid. In: Proc. IEEE Int'l Conf. Web Services (ICWS 2007).

Petty, M. D., Gustavson, P., 2012. Combat modeling with the High Level Architecture and Base Object Models. In: Tolk, A. (Ed.), Engineering Principles of Combat Modeling and Distributed Simulation. Wiley, Ch. 19, pp. 413–448.

Pullen, J. M., Corner, D., Brook, A., Wittman, R., Mevassvik, O. M., Alstad, A., 2012. MSDL and C-BML working together for NATO MSG-085. In: Proc. 2012 Spring Simulation Interoperability Workshop (SIW). Simulation Interoperability Standards Organization.

Rome, E., Langeslag, P., Usov, A., 2014. Federated modelling and simulation for critical infrastructure protection. In: D'agostino, G., Scala, A. (Eds.), Networks of Networks: The Last Frontier of Complexity. Springer, Ch. 11, pp. 225–253.

Simulation Interoperability Standards Organization, 2006. SISO-STD-003-2006 – Standard for Base Object Model (BOM) Template Specification.

Simulation Interoperability Standards Organization, 2008. SISO-STD-007-2008 – Standard for Military Scenario Definition Language (MSDL).

Simulation Interoperability Standards Organization, 2013. SISO-STD-012-2013 – Standard for Federation Engineering Agreements Template (FEAT).

Simulation Interoperability Standards Organization, 2014a. SISO-PN-012-2014 Product Nomination for WebLVC Protocol, Version 1.0.

Simulation Interoperability Standards Organization, 2014b. SISO-STD-011-2014 – Standard for Coalition Battle Management Language (C-BML) Phase 1, Version 1.0.

Simulation Interoperability Standards Organization, 2015a. SISO-STD-001-2015 – Standard for Guidance, Rationale and Interoperability Modalities for the Real-time Platform Reference Federation Object Model (RPR FOM), Version 2.0.

Simulation Interoperability Standards Organization, 2015b. SISO-STD-001.1-2015 – Standard for Real-time Platform Reference Federation Object Model (RPR FOM), Version 2.0.

Simulation Interoperability Standards Organization, 2016. SISO-PN-016-2016 – Product Nomination for High-Level Architecture Ver-

sion 3.0.
Sisson, B., Gustavson, P., Crosson, K., 2006. Adding aggregate services to the mix: An SOA implementation use case. In: Proc. 2006 Spring Simulation Interoperability Workshop (SIW). Simulation Interoperability Standards Organization.
Test Resource and Management Center, 2002. TENA The Test and Training Enabling Architecture – Architecture Reference Document, Version 2002.
The Open Group, 2009. The Open Group Service Integration Maturity Model (OSIMM) Technical Standard. Doc. no. C092.
The Open Group, 2011a. SOA Reference Architecture Technical Standard. Doc. no. C119.
The Open Group, 2011b. TOGAF Version 9.1 Enterprise Edition. Doc. no. G116.
The Open Group, 2014. SOA Ontology, Version 2.0 Open Group Standard. Doc. no. C144.
Tolk, A., 2012a. Integration of M&S solutions into the operational environment. In: Tolk, A. (Ed.), Engineering Principles of Combat Modeling and Distributed Simulation. Wiley, Ch. 15, pp. 295–327.
Tolk, A., 2012b. Modeling and simulation development and preparation processes. In: Tolk, A. (Ed.), Engineering Principles of Combat Modeling and Distributed Simulation. Wiley, Ch. 13, pp. 243–262.
Tolk, A., 2012c. Standards for distributed simulation. In: Tolk, A. (Ed.), Engineering Principles of Combat Modeling and Distributed Simulation. Wiley, Ch. 12, pp. 209–241.
Tolk, A., 2012d. Terms and application domains. In: Tolk, A. (Ed.), Engineering Principles of Combat Modeling and Distributed Simulation. Wiley, Ch. 4, pp. 55–78.
Tolk, A., 2012e. Verification and validation. In: Tolk, A. (Ed.), Engineering Principles of Combat Modeling and Distributed Simulation. Wiley, Ch. 14, pp. 263–294.
Tolk, A., Mittal, S., 2014. A necessary paradigm change to enable composable cloud-based M&S services. In: Proc. 2014 Winter Simulation Conference.
Tolone, W. J., Johnson, E. W., Lee, S.-W., Xiang, W.-N., Marsh, L., Yeager, C., Blackwell, J., 2009. Enabling system of systems analysis of critical infrastructure behaviors. In: Setola, R., Geretshuber, S. (Eds.), Critical Information Infrastructure Security. Vol. 5508 of Lecture Notes in Computer Science. Springer, pp. 24–35.
Wagenhals, L. W., Levis, A. H., 2009. Service oriented architectures, the DoD Architecture Framework 1.5, and executable architectures. Systems Engineering 12 (4), 312–343.
Wang, W. G., Wang, W. P., Zhu, Li, Q., 2011. Service-oriented simulation framework: An overview and unifying methodology. Simulation: Transactions of the Society for Modeling and Simulation International 87 (3), 221–252.
Wang, W. G., Yu, W. G., Li, Q., Wang, W. P., Liu, X. C., 2008. Service-oriented high level architecture. In: Proc. 2008 European Simulation Interoperability Workshop. Simulation Interoperability Standards Organization.
Wei, X., Tsai, W., 2014. HLA-based SaaS-oriented simulation frameworks. In: Proc. IEEE 8th Int'l Symp. Service Oriented System Engineering (SOSE).
Wilson, A. L., Weatherly, R. M., 1994. The aggregate level simulation protocol: an evolving system. In: Proc. 1994 Winter Simulation Conference. pp. 781–787.
World Wide Web Consortium, 2004. Web Services Architecture – W3C Working Group Note.
World Wide Web Consortium, 2007. Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language.
World Wide Web Consortium, 2009. Web Application Description Language (WADL) Submission.