FFI-rapport 2013/01674

Simulation architectures and service-oriented defence information infrastructures – preliminary findings

Jo Hannay, Karsten Bråthen and Ole Martin Mevassvik

FFI Forsvarets forskningsinstitutt

FFI-rapport 2013/01674

Simulation architectures and service-oriented defence information infrastructures – preliminary findings

Jo Hannay, Karsten Bråthen and Ole Martin Mevassvik

Norwegian Defence Research Establishment (FFI)

30 August 2013

FFI-rapport 2013/01674

1233

P: ISBN 978-82-464-2270-1
E: ISBN 978-82-464-2271-8

Keywords

Informasjonsinfrastruktur

Tjenesteorientert arkitektur, Service-Oriented Architecture (SOA)

High-Level Architecture (HLA)

Interoperabilitet og løs kopling

NATO Architecture Framework (NAF) og The Open Group Architecture Framework (TOGAF)

Approved by

Karsten Bråthen	Project Manager
Anders Eggen	Director

English summary

In line with strategic decisions, the Norwegian Armed Forces' systems portfolio should be developed in terms of *service orientation*: Software should be organized in parts so as to be presented as *services* which can be used readily and rapidly in a range of contexts. This demands that services are *interoperable*, in the sense that they can function as services for each other; which in turn, demands a common method of communicating data and specifying what the service provides. This also demands *loose coupling* in the sense that services are sufficiently generic to be useful over a range of service consumers, rather than being designed for one consumer only. A service-oriented manner of organizing software (an architecture) will enable one to build and rebuild software systems readily and rapidly—by adding and replacing services. Service-oriented architecture is geared explicitly to handle rapidly changing operational needs. The Norwegian Armed Forces information infrastructure (INI) is a part of, and should underlie, this service-oriented systems portfolio.

Modelling and simulation software must be embedded in this service-oriented portfolio. Well-defined architectural standards already exist for modelling and simulation software. They share many of the characteristics strived for in service-oriented architectures, while differing on other characteristics. An important question therefore pertains to how to integrate modelling and simulation software into the service-oriented portfolio.

An essential aspect of this question is what a service should be; what it should offer, how much it should offer, etc. NATO's C3 Classification Taxonomy is a partitioning of consultation, command and control (C3) functionality (at the enterprise level and at the IT systems level) in a service-oriented spirit. The taxonomy may be used as a tool for elaborating upon a adequate partitioning of functionality into services, and may be used as a starting point for development, such that it defines the building blocks and relationships of a service-oriented systems portfolio and a service-oriented information infrastructure. We outline a method for using the C3 Taxonomy for this purpose.

We apply parts of the method on simulation systems. This then gives a placement of modelling and simulation software according to the C3 Taxonomy. Although the systems portfolio extends beyond C3, this gives a starting point for integrating simulation systems with the Norwegian Armed Forces' systems portfolio and information infrastructure (INI).

The use of the method and the use cases we suggest in this report, are to be considered illustrations. Our user stories should be replaced by ones elicited more systematically, and our analyses should be supplemented by empirical studies, e.g., experiences gathered through the use of demonstrators, and literature studies. Nevertheless, we have mapped out and laid the grounds for further work. It is not self-evident what services should be and where the boundaries are for technical feasibility and cost-benefit. It is necessary to conduct systematic studies at all levels—from operational needs down to technical solutions—in order to gain headway in defining modelling and simulation services. Locally, we suggest that this can be done by conducting small manageable pilots in the various military domains, with tight collaboration between military practitioners and researchers, and where the entire spectrum from operational needs to technical feasibility is considered.

Sammendrag

I tråd med strategiske føringer, skal Forsvarets systemportefølje være *tjenesteorientert*: Programvare skal organiseres i deler slik at de fremstår som *tjenester* som kan tas i bruk enkelt og raskt i ulike anvendelser. Dette krever at tjenestene er *interoperable*; det vil si at de kan fungere som hverandres tjenester, som igjen fordrer en felles måte å kommunisere på og en tydelig bruksanvisning (tjenestekontrakt). Dette krever også at tjenestene er *løst koplet*, i den forstand at de er nyttige for mange tjenestebrukere, snarere enn at de er skreddersydd mot en bestemt tjenestebruker. En tjenesteorientert organisering av programvare, en *arkitektur*, skal gjøre det mulig å bygge opp og endre IT-systemer raskt og enkelt ved å bytte og legge til tjenester. Tjenesteorientert arkitektur er eksplisitt motivert ut fra at de operative behovene endrer seg stadig. Forsvarets *informasjonsinfrastruktur* (INI) er en del av, og skal understøtte denne tjenesteorienterte systemporteføljen.

Modellerings- og simuleringsprogramvare må inngå i en slik tjenesteorientert portefølje. I modellering og simulering har man allerede veldefinerte programvarearkitekturer som til dels har noen av egenskapene man søker etter i tjenesteorienterte arkitekturer, samtidig som det er tydelige ulikheter. Spørsmålet er da hvordan modellering- og simuleringsprogramvare skal innlemmes i porteføljen.

En vesentlig del av dette spørsmålet er hva en tjeneste skal være; hva den skal levere, hvor omfangsrik den skal være, osv. NATOs C3-taksonomi er en inndeling av konsultasjon, kommando og kontroll-funksjonalitet (både på virksomhetsnivå og på IT-system-nivå) etter tjenesteorienterte prinsipper. Taksonomien kan brukes som et verktøy til å finne hensiktsmessige oppdelinger av funksjonalitet og kan brukes som utgangspunkt for utvikling slik at den gir byggesteinene og sammenhengene i en tjenesteorientert systemportefølje og en tjenesteorientert informasjonsinfrastruktur. Vi skisserer en metode for å bruke C3-taksonomien som et slikt utgangspunkt for utvikling.

Vi bruker deler av metoden med henblikk på simuleringssystemer. Dette gir samtidig en plassering av modellerings- og simuleringsprogramvare i henhold til C3-taksonomien. Selv om Forsvarets systemportefølje ommfatter mer enn konsultasjon, kommando og kontroll, gir dette et utgangspunkt for å integrere simuleringssystemer med porteføljen og INI.

Bruken av metoden og brukstilfellene vi foreslår i denne rapporten er å anse som illustrasjoner. Brukstilfellene våre bør erstattes med brukstilfeller som er uthentet systematisk, og analysene vi gjør bør suppleres med mer empiri (for eksempel erfaringer gjennom demonstratorer) og litteraturstudier. Vi har likevel kartlagt grunnen og lagt opp løypa for videre arbeid. Det er ikke innlysende hva tjenester skal være og det er heller ikke innlysende hvor grensene går for hva som er teknisk mulig og hva som er kost-nyttig. Det er nødvendig med ytterligere systematisk arbeid på alle nivåer – fra operative behov og ned til tekniske løsninger – for å komme videre med å definere modellerings- og simuleringstjenester. Vi foreslår at slikt arbeid gjøres i form av mindre overkommelige piloter i forskjellige domener, i tett samarbeid mellom operativt personell og forskere, og der spekteret fra operative behov til teknisk mulighet ivaretas.

Contents

1 Introduction

Modern defence activities are characterized by joint operations at all levels. Such combined operations may involve different units, defence services, national allies, as well as a host of civilian actors at all levels. Computerized information systems, which support the complex flow of information and which structure the presentation of information, have become a key enabler for joint operations. The requirements for such information systems only increase in number and complexity. Further, multiple information systems must be combined and interoperate in so-called federations of systems supported by an information infrastructure. Therefore, a number of architectures have been put forward which set out to catalogue and structure the information necessary for collaboration, as well as giving standards for how to exchange that information between actors in an operation. These architectures give guidelines, and in some cases, strict requirements of compliance, for how defence information infrastructures should represent and exchange information in terms of data. Information infrastructures and their architectures evolve according to perceived short-comings and new information-handling technology. For example, an information infrastructure might provide a common platform to represent and exchange data in a uniform manner, but may be too proprietary and excluding so that useful "third-party" systems cannot be integrated. Such an infrastructure must therefore evolve to new needs or be replaced by a different infrastructure.

Current Norwegian and international developments in defence information infrastructures are service oriented. Service orientation entails that information systems, and also parts thereof, are viewed as integral components that provide well-defined information-handling services to both current and future service consumers, which may be end users or other services. The provision of a service to an unspecified range of service consumers entails that the service cannot be designed with a specific consumer in mind. Services are required to be loosely coupled with each other; they may interact and be combined with any other component as long as there is an agreement on how to specify information format and exchange. In theory, this ensures that information systems can be assembled and combined readily and rapidly according to the particular information handling demands at hand; for example, in a joint operation. Service-oriented architecture (SOA) [26] is an architectural approach intended to enable information systems, and the information infrastructures they both constitute and rely on, to evolve readily and rapidly according to upcoming and unforeseen needs.

Service orientation also permeates the concept of cloud computing, which defines IT services on a broader arena (software-as-a-service, data-as-a-service, platform-as-a-service, security-as-a-service, network-as-a-service, infrastructure-as-a-service, and even everything-as-a-service). Our focus is on software, and we will therefore relate to SOA in our discussion.

For the Norwegian Defence Information Infrastructure (INI), the Norwegian Ministry of Defence states [85] (translated from Norwegian):

> "The information infrastructure of the future must support network-based modes of operation by enabling the organization of all Armed Forces resources in a collaborating network. The Armed Forces must therefore have capabilities which satisfy National and NATO demands to interoperability and functionality, and which make it possible to meet digital threats while developing the Armed Forces toward a higher level of maturity with regards to network-based defence."

The current status of the INI is that a rudimentary architecture has been defined [82], with the intention of SOA explicitly stated. However, the architecture has not been realized into a working information infrastructure, in the sense that services may be accessed or provided or that applications may be routinely built by using modules or components in the infrastructure. At present, the INI is coarse-grained, and it is fair to say that substantial efforts must be made to populate the infrastructure to make it operational. There are general recommendations on how to do so [124, 83, 84].

Prompted by both national strategic relevance – note the reference to NATO in the above quote – and, more technically, to the explicit interoperability requirements of diverse cross-national collaboration [124], we will align our discussion to NATO's framework. NATO has come further in its efforts and its more elaborate architecture makes it easier to discuss concrete methods and points of advancement. The foundational structure of Norway's and NATO's information infrastructure architectures are similar, if not overlapping. Therefore, our discussions and proposed methods will apply directly to the INI.

NATO's information infrastructure goes under the name of the Networking and Information Infrastructure (NII). It is developed under the NATO Network-Enabled Capability (NNEC) program. On NNEC's web pages [78] it is stated that:

> "The networking and information infrastructure (NII) is the supporting structure that enables collaboration and information sharing amongst users and reduces the decision-cycle time. This infrastructure enables the connection of existing networks in an agile and seamless manner.
>
> This leads to Information Superiority, which is the ability to get the right information to the right people at the right time. NATO defines information superiority as the operational advantage derived from the ability to collect, process, and disseminate an uninterrupted flow of information while exploiting or denying an adversary's ability to do the same. [...]
>
> NNEC is about people first, then processes, and finally technology."

The last sentence above is notable. We will return to that point during our discussion.

For the technological part of NNEC, it is the NATO Communications and Information Agency (NCIA) which coordinates activities. A NNEC feasibility study was conducted in 2005. According to recommendations from the NATO Consultation, Command and Control Board (NC3B), who oversees the activities of the NCIA, "... the NNEC FS [feasibility study] postulated that the concept

of SOA is key to meeting the NNEC operational requirements and is an essential part of the overall strategy" [75].

NCIA has developed the Consultation, Command and Control (C3) Classification Taxonomy [73]. If not an architecture in the full sense of the word, the taxonomy may at least be viewed as an architecture sketch of a a C3 systems portfolio and an information infrastructure to support that portfolio. The taxonomy encompasses both the operational context and the technological context. At the technological level, the taxonomy maps out concepts on all layers; from physical layers of wired and wireless communication services, through middle-level services which software developers can use to develop applications in a service-oriented way, and up to user-level service-oriented applications and systems. In other words, it is envisioned that service orientation and service-oriented design and development permeate all levels of both systems portfolio and information infrastructure, in order to meet the demands of NNEC at the operational level. We intend to shed light on how modelling and simulation might fit into this picture. The total portfolio that an information infrastructure must support includes several other activities beyond C3. However, we relate to the C3 Taxonomy in this discussion because of its relatively high level of maturity and aptness for illustrating the points made in our discussion.

Modelling and simulation (M&S) systems are a part of today's computerized information systems capability for defence operations. Among other things, they are used in operations training and long- and short-term planning. However, the integration of simulation into defence information infrastructures is, at present, in its infancy. The current focus on service orientation promises new possibilities for integrating simulation into defence information infrastructures, and there are several initiatives focusing on these possibilities. However, while architectures for service-oriented defence information infrastructures currently exist at the conceptual level, architecture standards and development standards for M&S are well-defined and are realized into working systems, often using off-the-shelf software. In M&S it is desirable to be able to develop new simulation components (e.g., a model of a new combat aircraft), or to upgrade an existing simulation component, and add it to an existing simulation system. The need for loose coupling and flexible extension, modification and integration has therefore been recognized for some time in the M&S domain as well, as reflected e.g., in the NATO Modelling and Simulation Masterplan [76, 77]. Currently, the M&S architecture that has come farthest in terms of standardization and modern technology, is the High Level Architecture (HLA) [38]. This report will therefore focus on how M&S according to HLA might be embedded into an information infrastructure following NATO's framework.

To guide these efforts, it is important that not only technological aspects are considered, but also operational capabilities. It is an essential aspect of service orientation that services should be defined, or at least targeted, ultimately at the enterprise level; in other words at the needs and (often tacit) requirements of end users. To this end, we propose use cases for using and utilizing a defence information infrastructure that incorporates modelling and simulation. We will relate to the C3 Taxonomy when formulating use cases, and we will propose use cases at several layers of the taxonomy. At more technical layers, we will discuss the use cases in terms of characteristics of HLA and tech-

nical aspects of SOA: Both the HLA and SOA promote interoperability and very loose coupling. At first sight, HLA and SOA seem to be motivated by similar aims, to provide similar types of infrastructure, and to provide similar types of extendibility (dynamic and static). We will attempt to clarify relevant differences and similarities.

We hypothesize that a main disabler for developing both the INI and the NII lies in a lack of a practicable method for detailing the architectures and a lack of a practicable method for developing the architecture into a working information infrastructure. We look into the possibility of using the structure of the C3 Taxonomy to discipline a production strategy which could detail the architecture and develop the implied information infrastructure.

In summary, we will seek answers to the following research questions:

1. What are the requirements of users of a service-oriented defence information infrastructure that incorporates M&S software systems?
2. How can these requirements be used to build the information infrastructure?
3. To what extent is it desirable and then possible to package application-size M&S software to loosely coupled and interoperable units in the context of a service-oriented defence information infrastructure?
4. To what extent is it desirable and then possible to use service-orientation internally in M&S software; hereunder,
   (a) can SOA be used as a simulation management framework beyond HLA?
   (b) can component/module-size M&S software be packaged as loosely coupled and interoperable units?

We follow a flow of argumentation as follows: In Section 2, we define our understanding of what an information infrastructure and its architecture are. We describe the architecture sketches for the Norwegian Armed Forces information infrastructure (INI) and NATO's networking and information infrastructure (NII). In Section 3, we suggest how the C3 Taxonomy can be used in systematizing and disciplining a method for eliciting requirements and for refining these into working parts of the information infrastructure according to the desired architecture. In Section 4, we give suggestions as to how such requirements can be formulated and what they might be. We then introduce Service-Oriented Architecture and the High Level Architecture in Sections 5 and 6. We give examples of more technical requirements and give examples of where working software may be placed in the C3 Taxonomy in Sections 7 and 8. Sections 9 and 10 discuss and conclude.

2 Defence information infrastructures and their architectures

There is a bundle of terms denoting complex information systems and their descriptions; e.g., "information infrastructure", "information system framework", "architecture", "reference architecture", "reference model", "architecture framework", "reference information infrastructure", "systems of systems", "federation of systems", etc.; *ad nauseam*. We shall limit our vocabulary to

"information infrastructure", which is a concrete and working information-technological support structure which enables the successful construction and running of a "federation of systems" which is a large and complex, but concrete and working collection of loosely coupled information handling systems. Further, we will use the term "architecture", which is a structural description at some suitable level of abstraction of a (planned) concrete working system; here of an information infrastructure and/or a federation of systems. We will also use the term "architecture framework" to denote a framework for developing architectures.

A *federation of systems* is, in our discussion, a collection of loosely coupled collaborating information systems; see Figure 2.1. An individual information system may consist of purely human routines; e.g., strategies and plans for information handling. It may be purely technical in terms of software or hardware. Or it may consist of all levels of information handling from human through software to hardware. 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 future systems. The purpose of an *information infrastructure* is, in our context, to facilitate this service orientation. This can be done by providing common services for constructing new service-oriented systems, by providing services to service-enable or refactor existing, possibly stove-piped systems, and by providing services to couple systems together into a federation of systems.

An *architecture* of an information system (or of a federation of systems, or of a information infrastructure) is, according to ISO/IEC 42010:20071: "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 the system, as we indicated above. But there should also be a method for constructing an architecture and for maintaining it. Such a method is what we here refer to as an *architecture framework*. In addition, we hold that there should exist a method for constructing a working information system from an architecture; see Figure 2.2. An architecture framework is a specification of what an architecture should be, preferably with a method for constructing a concrete architecture according to the architecture framework. In turn, an architecture should preferably come with a method for constructing a working system (e.g., an information infrastructure or a federation of systems) according to the architecture.

2.1 Architecture for the Norwegian Armed Forces Information Infrastructure

Figure 2.3 shows a sketch of an architecture for the Norwegian Armed Forces Information Infrastructure (INI) [82]. It is layered into decision support services (yellow) and core services (light blue) with a communication infrastructure in the form of communication services (grey). The de-

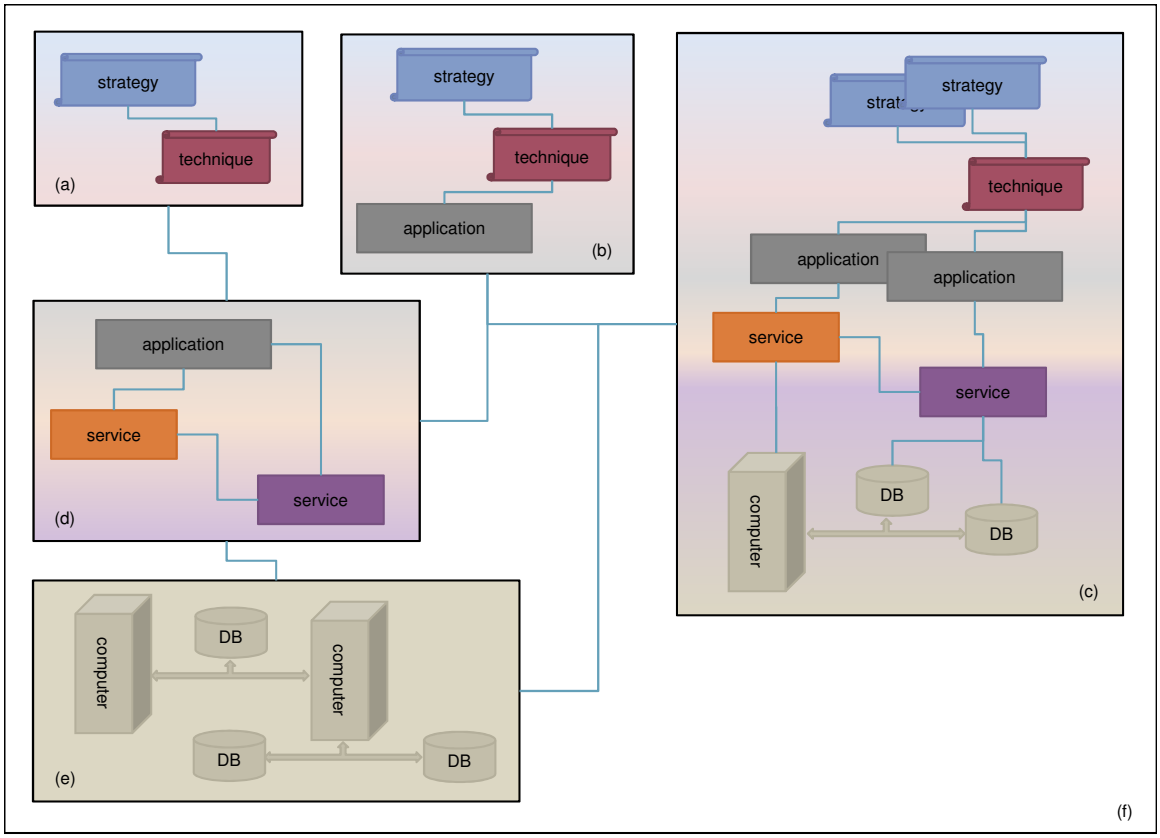

Figure 2.1 *Federation of systems exemplified by systems consisting of human processes (a), of a combination of human processes and software processes (b), of a combination of human, software processes and hardware processes (c), of software processes only (d), of hardware processes only (e). Together, the systems form a federation of systems (f).*

cision support services are divided into Specific functional services and Generic functional services. Information security (red) and service management (green) flank the figure as cross-cutting concerns.

Figure 2.4 shows an alternative sketch of the architecture. Due to input from the Norwegian Defence Research Establishment (FFI) in 2007 regarding M&S, further categories were added and M&S elements tentatively placed: M&S in general in "K2 og ledelse" (C2), simulation-based training and exercise in "Utdanning, trening og øving" (Education, training and exercise), the Runtime Infrastructure of HLA (Section 6) in "Informasjonsutveksling" (Information exchange) and the Coalition Battle Management Language (C-BML) and Military Scenario Definition Language (MSDL) (Section 7) in "Mediering" (Mediation). In general, FFI has made several initiatives toward defining the architecture of INI.

The present architecture sketches are very high level and simply indicate what types of applications and services should be included in the information infrastructure and at which level of end-user versus technical orientation they should be considered. To populate the categories (boxes) with applications and services is a step toward developing an information infrastructure from the archi-

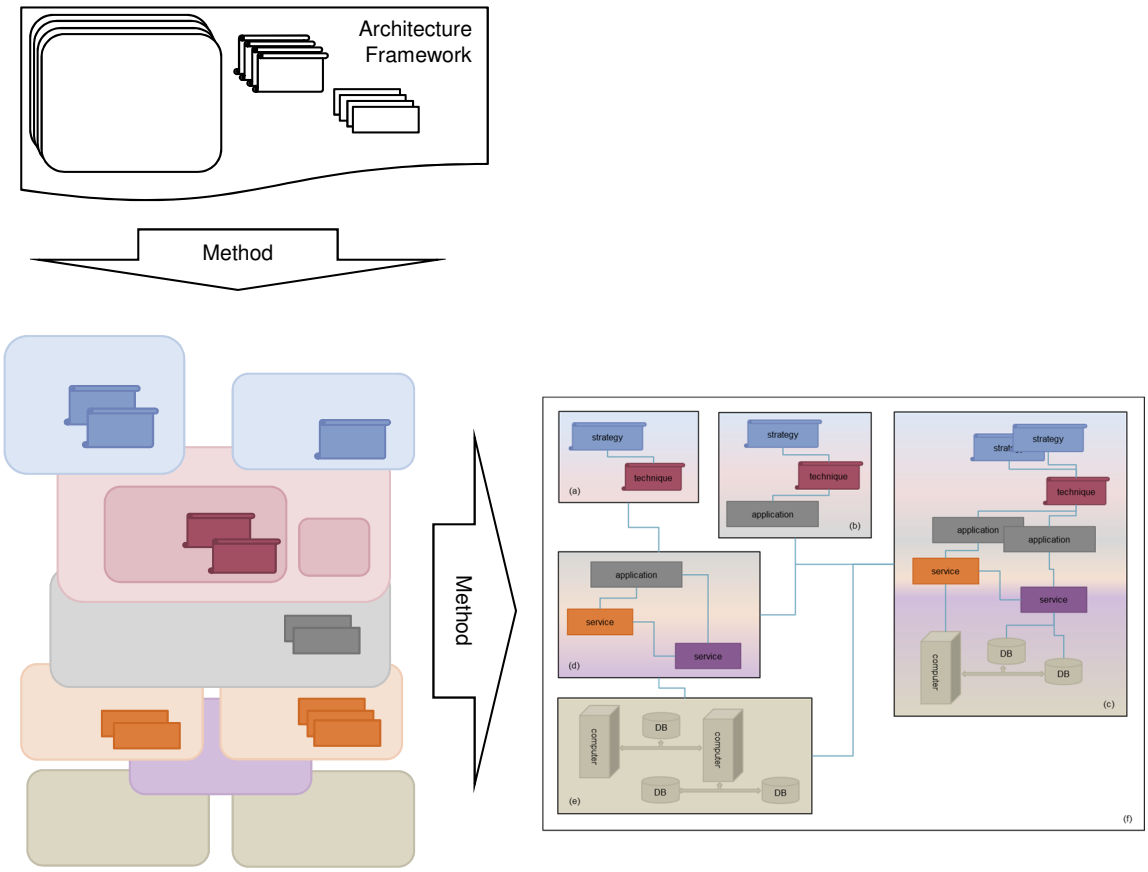

Figure 2.2 *Architecture framework, architecture of system, and system, with methods for transforming one to the other.*

tecture. The architecture lacks explicit and official population of working software, although the category names indicate where several such pieces of software belong. There has not been substantial advancements in detailing the INI architecture beyond what we have reported here.

In the latest IT strategy document from the Norwegian Chief of Defence [86], it is decided that:

- Architecture descriptions follow the NATO Architecture Framework (NAF) [74]
- Architecture development follow The Open Group Architecture Framework (TOGAF) [45]
- The INI is to be developed in pace with NATO's NNEC and is to be harmonized with NATO's NII and the C3 Classification Taxonomy

Together, NAF and TOGAF provide guidelines for implementing a method for constructing architectures. One has to provide the concrete methods for requirements elicitation, entity modelling, etc, but with that in place one has an architecture framework; i.e., a method for constructing an architecture of the INI, in line with our remarks above. The third point entails an explicit focus on NATO's information infrastructure architecture, to which we will now turn.

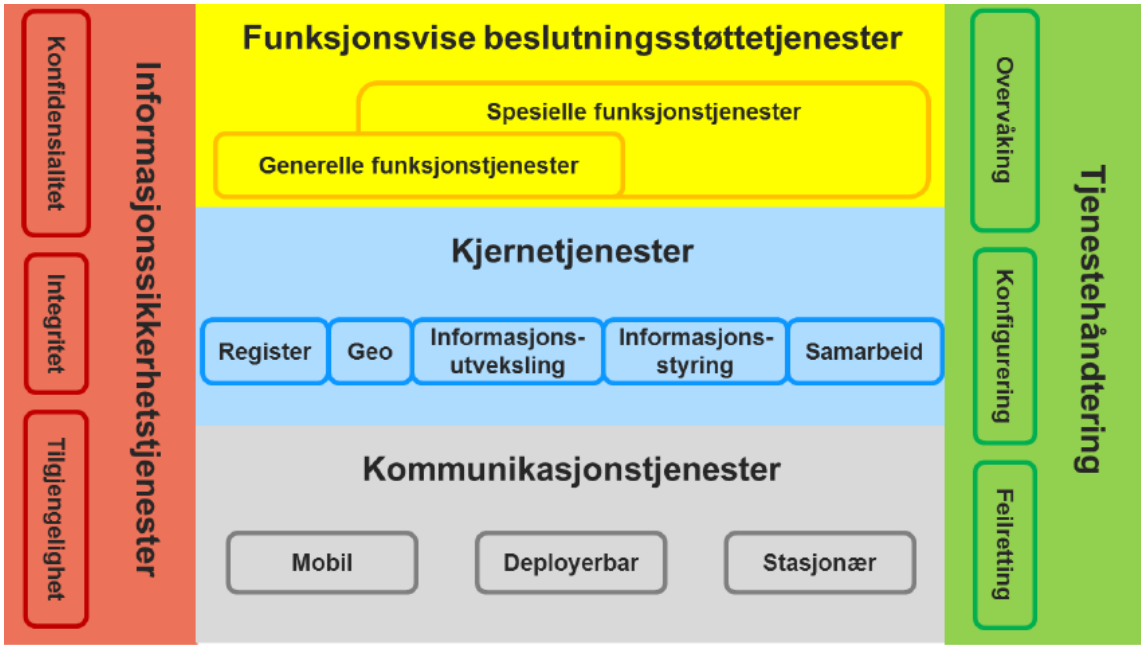

Figure 2.3 Architectural sketch for the Norwegian Armed Forces Information Infrastructure (INI) [124].

2.2 The C3 Taxonomy

The C3 Classification Taxonomy (C3 Taxonomy for short) is a sorting of functionality. More precisely, it is a sorting of capability concepts relevant to producing enterprise computer system support for Consultation, Command and Control (C3) in NATO; see Figure 2.5 and Figure 2.6 (more detailed view). It explicitly includes, in the same picture, the operational context (Operational Context frame in Figure 2.5) and the computing context (Communication and Information Systems (CIS) Capabilities frame in Figure 2.5). The following summarizes the motivation for the C3 Taxonomy:

> "The C3 Classification Taxonomy provides a tool to synchronize all capability activities for Consultation, Command and Control (C3) in the NATO Alliance by connecting the Strategic Concept and Political Guidance through the NATO Defence Planning Process (NDPP) to traditional Communications and Information Systems (CIS) architecture and design constructs [...] Throughout the years, many communities have developed and contributed components to the overall CIS capability of the Alliance but sadly, most groups did their work in splendid isolation. Today we are confronted with a patch-quilt of systems, applications, vocabularies and taxonomies and simple English words such as service or capability have become highly ambiguous. As a result of extreme stove-piping, NATO now faces a very complex fabric of capabilities that are not interoperable and attempts to solve these problems are often hampered by lack of mutual understanding caused by confusing vocabularies" [11].

The guiding of technology by an explicit focus on operational context is very much in line with the motivation behind SOA. The focus on enterprise-driven service definition shifts the definition

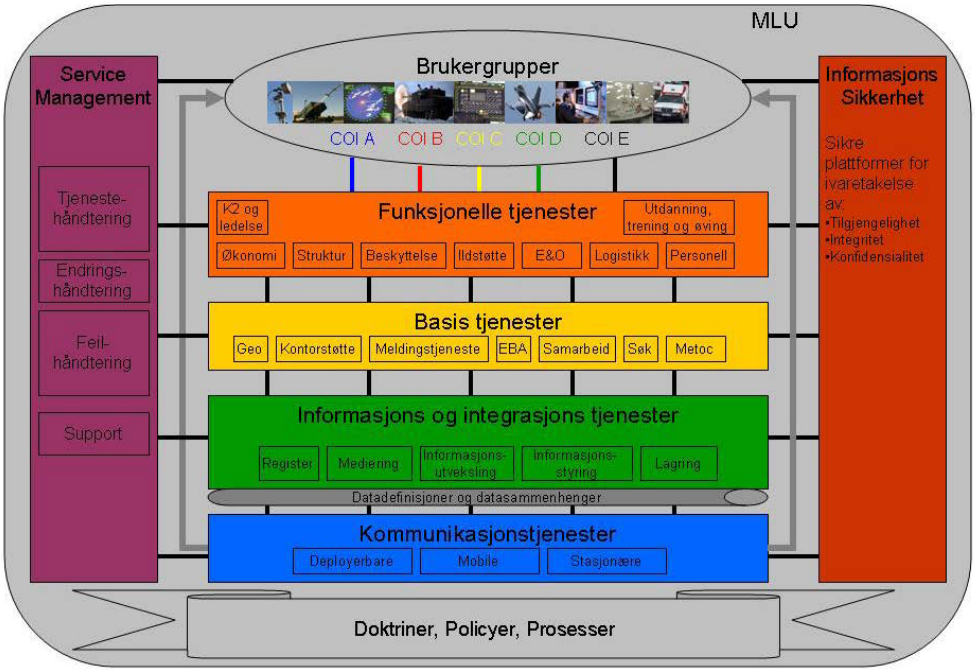

Figure 2.4 Alternative architectural sketch for the INI.

of an organization's enterprise processes back to where they belong, in the organization's strategy domain, and away from the IT-department: Although user requirements are in focus in modern software development practices, it is still technologists who define the architecture of the system and who develop the system. The resulting software system will possibly fulfil needs at the point of deployment, but will probably become an obstacle when the organization's strategy and processes inevitably change. SOA promises to produce software systems which are agile to an organization's strategy and process – here defence strategy and tactics – also after deployment.

As can be seen from Figure 2.5, the taxonomy has an Operational Context at the top, layered into Missions and Operations which are supported by Operational Capabilities; the latter being foundational, generic operational modules on which missions and operations may be built. Below the operational context are the Communication and Information Systems (CIS) Capabilities, which are the information technology support for the operational context. The CIS capabilities presents themselves to the end user in the form of User Applications geared toward specific domains (air, land, maritime, joint, etc.) and communities of interest (modelling and simulation, environment, missile defence, etc.); see Figure 2.6. Below this layer are various layers of Technical Services, which may be used to develop and implement the user-facing capabilities. The Technical Services are layered in an analogous manner to the traditional Open Systems Interconnection (OSI) and Transmission Control Protocol/Internet Protocol (TCP/IP) models, both of which have more domain-dependent IT functionality at the top and increasingly generic and technical functionality, including hardware, lower down. Thus, the upper layers of the C3 Taxonomy's Technical Services consist of the Community of Interest (COI) Services geared toward realizing User Applications.

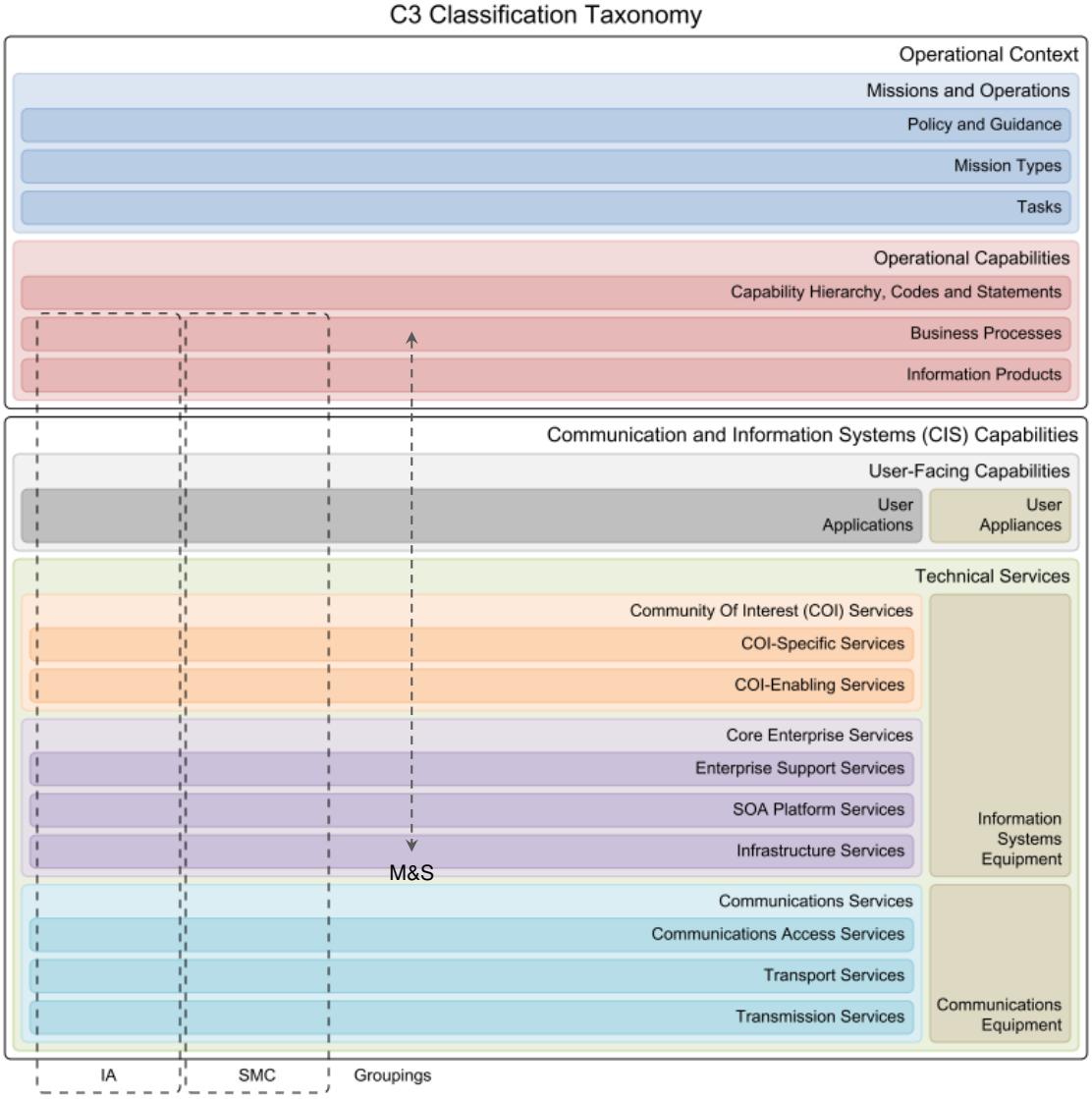

Figure 2.5 *C3 Classification Taxonomy [73] – Main categories; with NATO Exploratory Team 034 (M&S as a Service) M&S-relevant layers indicated [97].*

At the User-Facing Capabilities layer, software is denoted *applications*. Software and hardware at the Technical Services layers are denoted *services*. However, services and applications should both fulfil requirements of interoperability and loose coupling. We will return to this below.

Two cross-cutting concerns are defined in the taxonomy (IA and SMC Groupings in Figure 2.5). The Information Assurance (IA) grouping "provides a collection of measures to protect information processed, stored or transmitted in communication, information or other electronic systems in respect to confidentiality, integrity, availability, non-repudiation and authentication" [11]. The Service Management and Control (SMC) grouping "provides a collection of capabilities to coherently manage components in a federated service-enabled information technology infrastructure. SMC tools enable service providers to provide the desired quality of service as specified by the customer" [11].

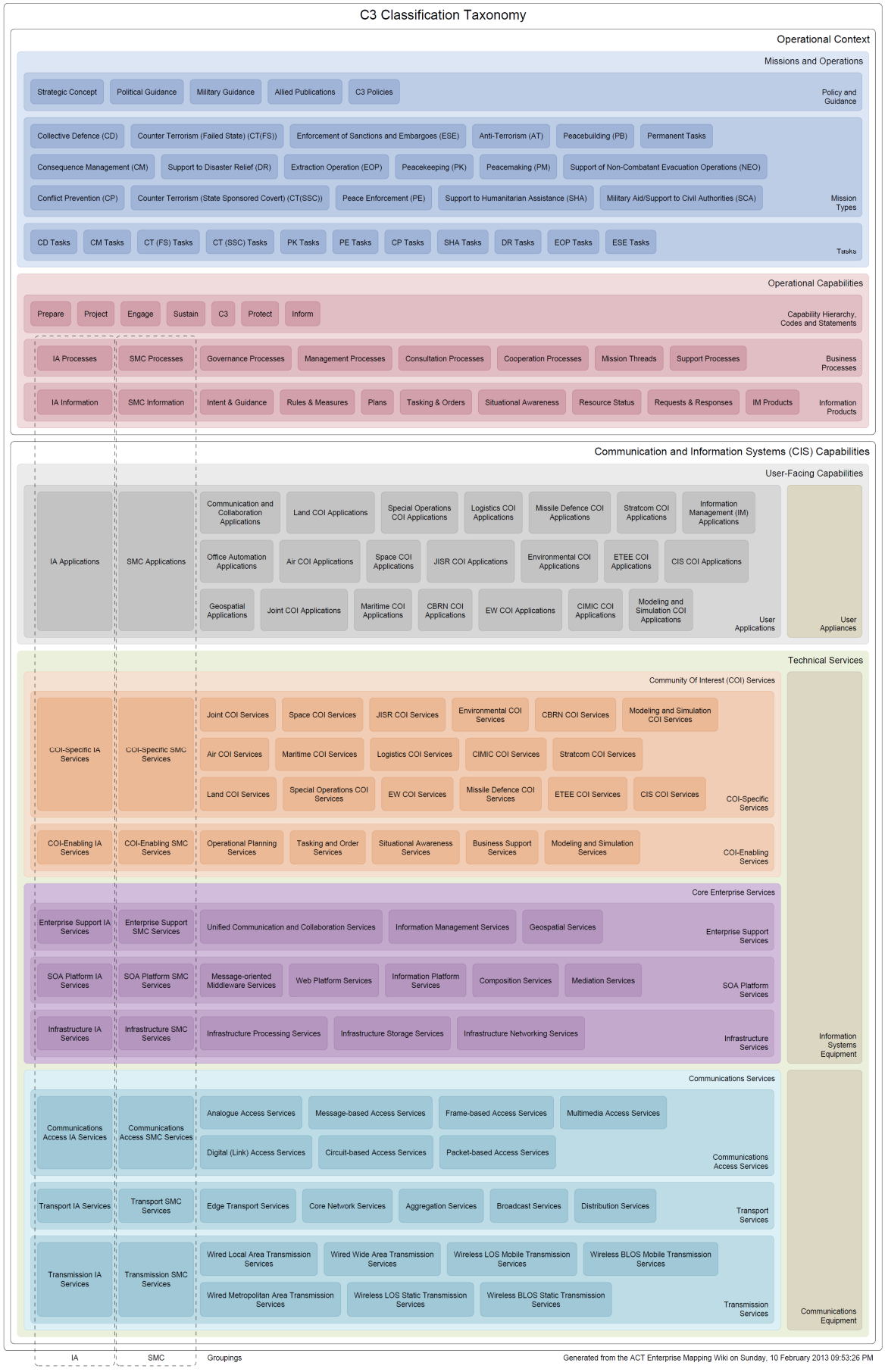

Figure 2.6 C3 Classification Taxonomy [73]

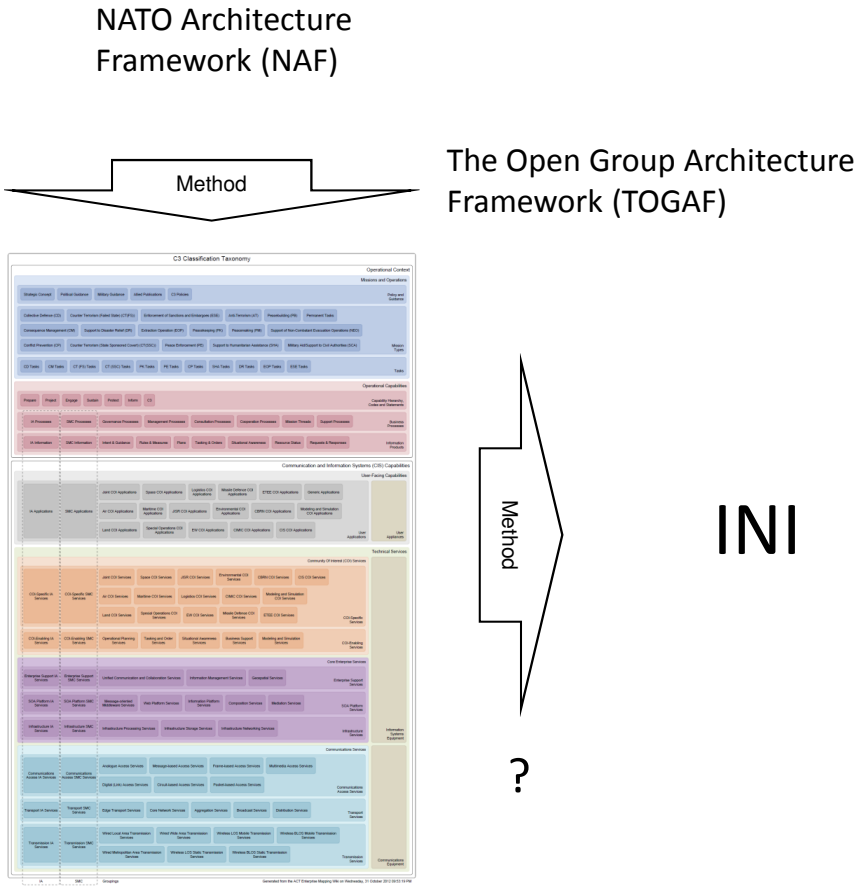

Figure 2.7 *NAF as architecture framework, TOGAF as development method and C3 Taxonomy as architecture for the INI, but no explicit method for developing the INI.*

Both sketch versions of INI's architecture in Figures 2.3 and 2.4 can be mapped onto the C3 Taxonomy. The overall structure of the latest version includes cross-cutting concerns corresponding to Information Assurance (IA) (red) and Service Management and Control (SMC) (violet). This correspondence is why it is meaningful to relate to the C3 Taxonomy when developing the INI. Together with NAF and TOGAF, one has a method for developing the architecture of the INI via the C3 Taxonomy, but not an explicit method for developing the INI itself; see Figure 2.7.

With its service-oriented structure, the C3 Taxonomy may be regarded as an architecture sketch for both human and computerized systems for C3. The information infrastructure (the INI or the NII) lies somewhere as a part of this. There are at present unclarities as to the boundaries of such an infrastructure; should an infrastructure be limited to "pipes" and roads" or should it include maintenance services, "public transport" and other "utility vehicles"? For example, the INI architecture as sketched in the previous section only considers the Technical Services portion of the C3 Taxonomy, and there are arguments for leaving User Applications outside of the INI; in particular regarding stove-piped systems [124]. There are a host of challenges in defining the system boundaries for an infrastructure, but the principles of common applicability and stability should be essential

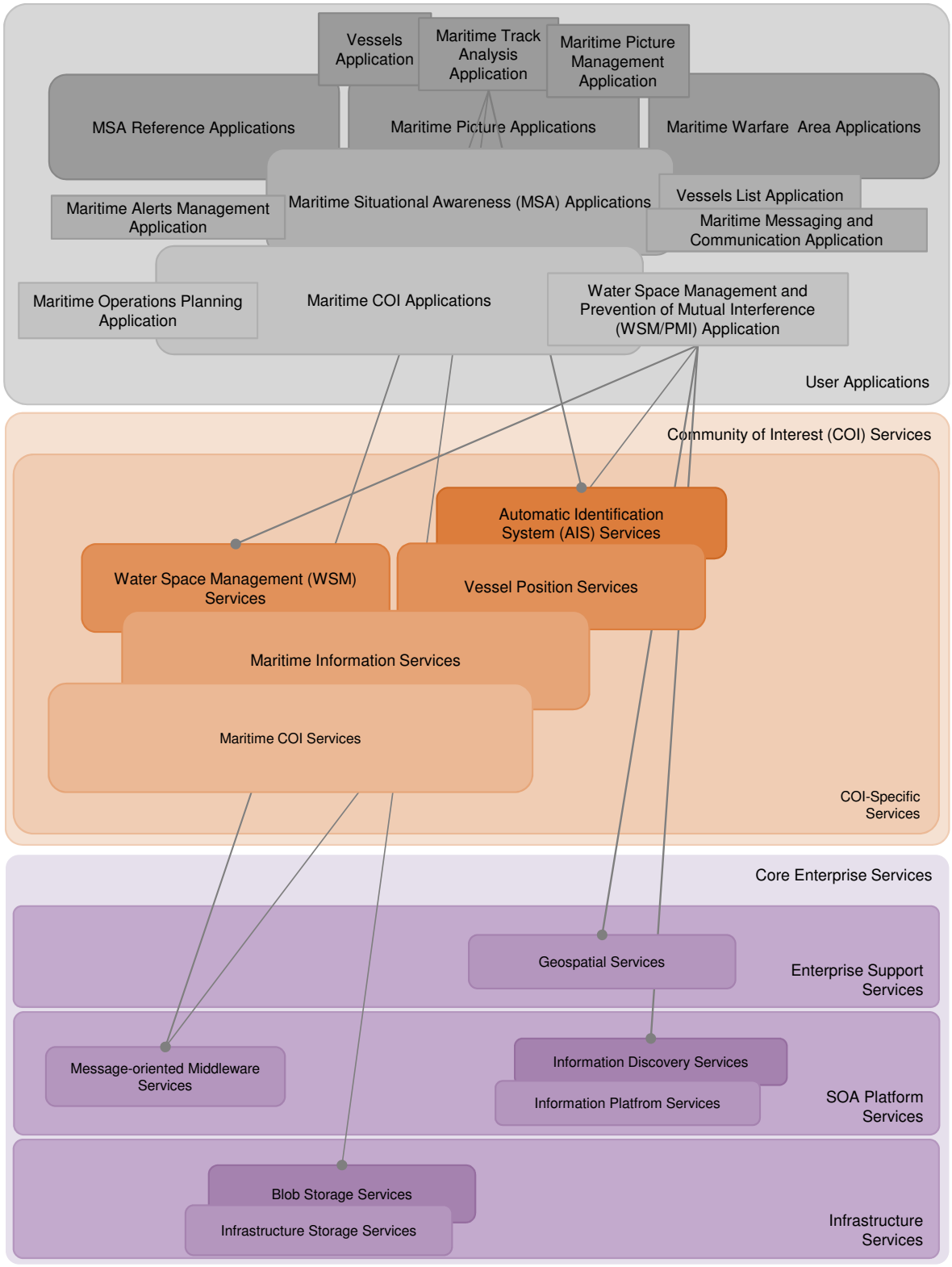

Figure 2.8 Current status of population of the C3 Taxonomy for one of the User Applications categories. Concrete applications in square boxes. Lines indicate required services.

inclusion criteria for whatever lies within the infrastructure.

In our discussion, we do not need to take a particular standpoint regarding the system boundaries of the envisioned information infrastructure. Modelling and simulation systems typically belong in the User Applications, Community of Interest Services and Core Enterprise Services categories, and we will relate to the relevant categories in the taxonomy regardless of the exact boundaries of the infrastructure. Our approach will assume that stove-piped systems are eventually converted to service-oriented systems front-ended by a thin client residing in the User Applications layer, using services at lower layers.

The C3 Taxonomy is maintained via the NATO Allied Command Transformation (ACT) Technology for Information, Decision and Execution superiority (TIDE) Enterprise Mapping (EM) portal on Tidepedia (a NATO Wikipedia analogue). Through EM, the taxonomy functions as a semantic Wiki on which authorized persons may summarize current knowledge, develop a common understanding of concepts and ultimately populate the taxonomy with working systems (human, software and hardware). There is no corresponding interactive gateway for collaboration toward the Norwegian Defence Systems portfolio and INI.

The C3 Taxonomy has been populated with a substantial number of artefacts in some of the categories directly related to specific defence branches. For example, the Maritime COI Applications category has been divided into subcategories – one of them being Maritime Situational Awareness (MSA) Applications, which has further been divided into MSA Reference Applications, Maritime Picture Applications and Maritime Warfare Applications. Several of the categories have been populated with concrete applications; see Figure 2.8 for an overview and the EM wiki for full details. Also at the Core Enterprise Services and the Communications Services levels has there been substantial activity toward populating the taxonomy. When it comes to M&S, however, the current status is that there is somewhat less information present in the taxonomy. A NATO Exploratory Team ET-034 M&S as a Service (MSaaS) has recently conducted introductory studies on how to integrate M&S in the C3 Taxonomy. The extent of M&S relevant layers as perceived by the team is indicated in Figure 2.5. Our current work is in line with this initiative, and we will outline a light-weight method for developing both the architecture and the C3 systems portfolio. We shall also propose tentative placements in the taxonomy of M&S-related artefacts.

3 A method for developing the C3 systems portfolio

The lack of progress on the Norwegian INI is the likely result of many factors. We hold that one important obstacle is the lack of a leveraged practitioner-oriented method for engaging in development work on the INI. The decision to use NAF, TOGAF and the more detailed C3 Taxonomy with its wiki interface are improvements in this respect. However, to reach a stage of actual development, one needs to make the method more concrete.

We outline a use case-driven development method which utilizes the structure of the C3 Taxonomy. While this does not provide a concrete method as such, it provides concepts which enable concret-

ization. Conceptual simplicity and small steps are crucial for a deployed method to be employed, and we suggest using practices from agile management and development [111, 100, 24, 115].

The C3 Taxonomy includes Operational Context and the Communication and Information Systems (CIS) Capabilities. From the point of view of M&S, our main interest lies in populating the CIS Capabilities part of the infrastructure. We therefore start by outlining the method at the CIS Capabilities level.

3.1 Use cases as requirements to CIS capabilities

The development of the C3 systems portfolio, including an information infrastructure such as INI or NII, at the CIS Capabilities level is a very large IT-systems development project. Large IT development projects increasingly adopt agile management and development methods, with the intention to manage the inevitable high levels of complexity and uncertainty in such projects. Appropriate requirements elicitation and handling is of crucial importance when using such methods. For requirements elicitation, nominees from each stakeholder group should engage in systematic methods for determining, formalizing, and content analysing system requirements; see e.g., [56]. These methods should be used both individually and in groups. For requirements handling, there are three key points from agile methodology which we bring forward here:

- Requirements are elaborated and refined over time according to the project's rising level of knowledge and understanding of needs.
- Requirements are partitioned and formulated so that they represent meaningful parts of functionality for those stakeholders who are relevant at a given point of elaboration and refinement.
- Requirements are partitioned so that they represent viable production elements.

In agile development, requirements are formulated in *use cases*, which are specifically formulated to capture how a stakeholder intends to use the system under development. In line with the first point above, at early stages of the development project, use cases are high-level place-holders, often called *epics*. During the course of the project epics get elaborated and refined into *user stories* and ultimately end up as concrete development tasks (*sprint tasks* in Scrum) which produce shippable code – here parts of applications and services; see Figure 3.1. In line with the second point above, epics are usually formulated in terms of the organization's business or operational requirements to the system under development and should be rooted in the business case for the development project. Hence the partitioning of functionality represented by the epics reflects the system as seen from the business or operational context. Moreover, in line with the third point above, an epic represents a piece of the system which, from the vantage point of business or operational context, makes sense to produce as a subsystem. Epics should be explicitly prioritized according to estimated benefit/cost [18, 113].

As an epic becomes elaborated and refined, more technical details become relevant and therefore other stakeholders become involved. User stories and sprint tasks therefore represent a partitioning of the system which is meaningful for more technical concerns. A user story or a sprint task is viable for production from a technical point of view. Note that the operational context present in

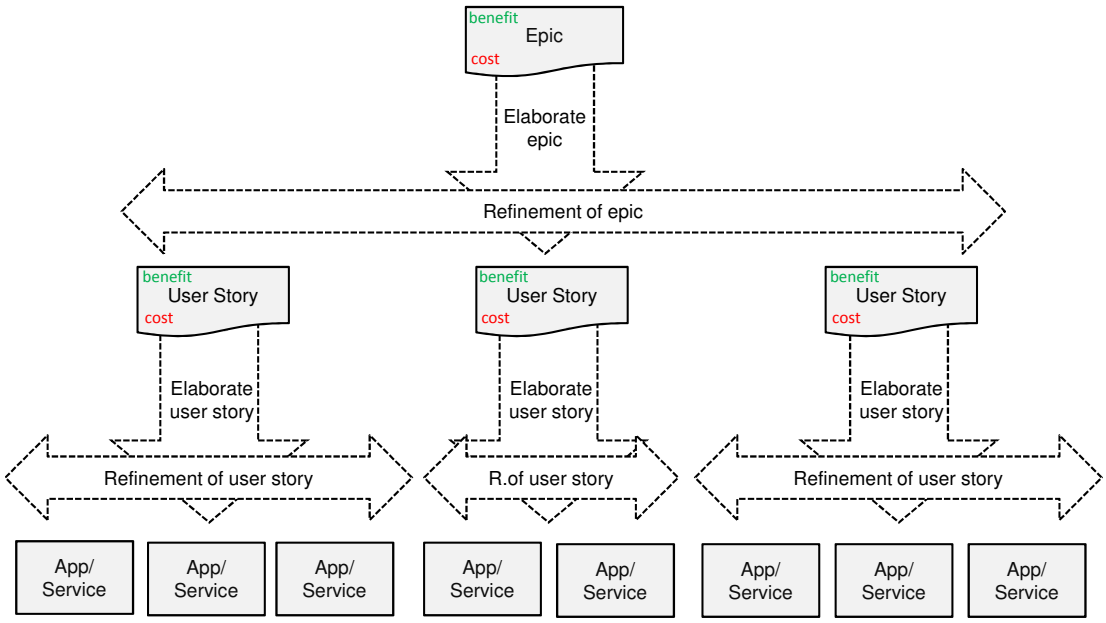

Figure 3.1 *Stepwise epic elaborate-refine tree for applications and services. Epics are elaborated into, possibly, several user stories, which by separating concerns between them, represent a refinement of the epic. Likewise, user stories are elaborated and refined (developed) into applications or services. Note that there may be several layers of both user stories and epics.*

an epic is preserved down through the hierarchical elaboration and refinement into user stories and sprint tasks. The epic's prioritization according to estimated benefit/cost should also be preserved.

Although Figure 3.1 shows a single layer of epics and user stories, it is possible to detail epics into more detailed epics, and likewise to have several levels of user stories. When to call use cases epics and when to call use cases user stories is a matter of taste. In Scrum, user stories are usually use cases at the level of granularity and detail where it is feasible to order them in a production queue (backlog) ready for processing by Scrum teams [53, 17].

Use cases should be explicitly linked to *impact goals* elaborated in the business case for the development project. Impact goals are measurable goals which the project should reach. In general, we promote using a use case syntax in order to structure use cases and to obtain traceability of impact goals through all levels of refinement of use cases. For example,

> Use case: **As** <stakeholder A> **I can** <perform actions d in the domain> **by using** <functionality f in system S under development> **to** <perform actions s in S> **in order to** <reach impact goal I>.

Here, all stakeholders A, impact goals I, actions d should be defined in the business case, while functionality f, actions s, system S, are incrementally detailed through stepwise refinement. In agile settings, the business case itself is updated according to evolving knowledge in the project; so A, I, d may also be detailed and refined into more specific actors, goals and actions.

As mentioned above, use cases (epics and user stories) are requirements specifications and at the same time production units. They are the driving artefacts in an agile development project and they eventually get transformed into working software. More detailed specifications, such as UML diagrams or NAF diagrams, may be attached to use cases as desired.

3.2 Use Cases as requirements to operational context

The basic method for eliciting and handling requirements to CIS Capabilities (applications and services) may also be used for eliciting and handling requirements to the Operational Context. Similar systematic methods involving multiple stakeholders may be used to elicit operational epics. Then, for handling requirements, the same schema for elaborating and refining epics may be used. Through a series of elaboration and refinements, scenarios are eventually detailed into concrete operational tactics, techniques, procedures (TTPs); see Figure 3.2. Again, although Figure 3.2 shows a single layer of epics and scenarios, it is possible to detail epics into more detailed epics, and likewise to have several levels of scenarios. In requirements elicitation, detailing and refinement processes for software, the product owner is kept in the loop constantly and evaluates finished parts or prototypes of the system. For eliciting, elaborating and refining TTP requirements, similar methods for ensuring "product owner" feedback can be used; including prototyping in virtual worlds and static and dynamic modelling (i.e., simulation).

Requirements to CIS Capabilities (previous section) are also rooted in operational context. The important difference between handling requirements for the operational context and handling requirements to CIS Capabilities is that the focus of the latter is on using, and therefore on designing, some computerized system. In contrast, the focus of the former is on using and designing a human-based system; i.e., plans, routines, methods, modes for cooperation, etc. Although a human-based system may, of course involve computerized systems, there is a point in remaining in the Operational Context as long as possible, so as not to loose focus in favour of technology. The same use case syntax as above can be used, but with the understanding that the $<$functionality f in system S under development$>$ refers to functionality in a human-based system.

It is worth noticing that the Multilateral Interoperability Program (MIP) uses the term "Capability Package" to denote a manageable piece of capability functionality. Capability packages could take the role of epics. Capability Package Teams in the MIP community use Scrum to develop capability packages to the level of architectural model in NAF. There is also mention of TOGAF, see [52].

3.3 Use Cases as requirements using the C3 Taxonomy

The structure of the C3 Taxonomy can be used to discipline a combined approach to populating the taxonomy, using the methods in Sections 3.1 and 3.2. We propose to populate the taxonomy with, not only concrete artefacts (procedures, applications, services), but also with requirements (e.g., use cases) at various levels of elaboration and refinement during development of the concrete artefacts. And, we propose to place the requirements in step with elaborating and refining the taxonomy's categories.

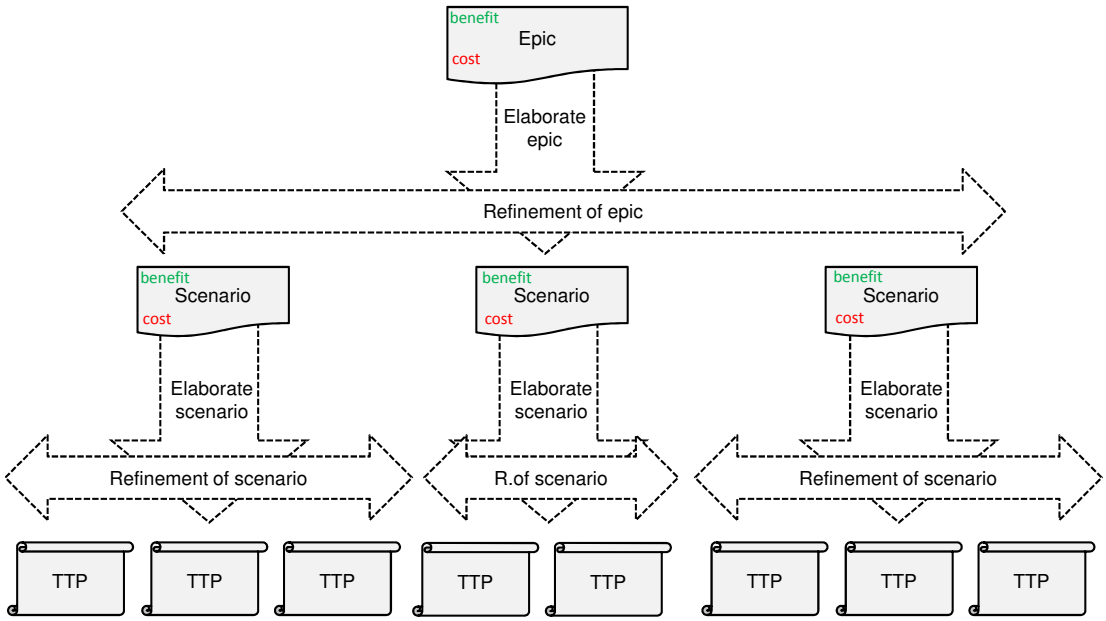

Figure 3.2 *Stepwise epic elaborate-refine tree for operational tactics, techniques, procedures (TTPs). Epics are elaborated into, possibly, several scenarios, which by separating concerns between them, represent a refinement of the epic. Likewise, scenarios are elaborated and refined (developed) into TTPs. Note that there may be several layers of both scenarios and epics.*

The key idea is that each top-level category in the C3 Taxonomy is a starting point for development using the use-case-based approach. An example is given in Figure 3.3: The C3 Taxonomy has a category named C3; see Figure 2.6. Epics at this abstract level describe on an abstract level what the requirements for C3 operational capabilities are. We place these epics in the C3 category. In the current state of the C3 Taxonomy as viewed via the EM wiki, the C3 category is sub-categorized into the three categories Consultation, Command & Control and (perhaps somewhat misplaced) Communications & Information Systems (see the EM wiki for descriptions of these subcategories). At the more elaborated and refined level that these three subcategories constitute, the epics are refined and detailed into more concrete scenarios for overall C3 capabilities. The scenarios are placed in their respective categories. In Figure 3.3, the Communications & Information Systems category is hypothetically detailed further onto a concrete category (denoted ?) populated by the concrete tactics, techniques, procedures (TTPs) that result from elaborating and refining the scenarios fully.

In Figure 3.3, we skip down to the User Applications layer, where, for M&S, the current state of the taxonomy subcategorizes the Modelling and Simulation COI Applications category into Scenario Preparation Applications, Federation Management Applications and Model Development Applications. Epics describe high-level requirements from the perspective of users of Modelling and Simulation COI Applications. The epics are then elaborated and refined until concrete applications populate the leaf nodes in the Modelling and Simulation COI Applications tree.

Similarly, epics in top-level categories are starting points for development at the COI Specific Services level; and so on at all levels of the C3 Taxonomy. As seen from Figure 2.5, it is envisioned that M&S

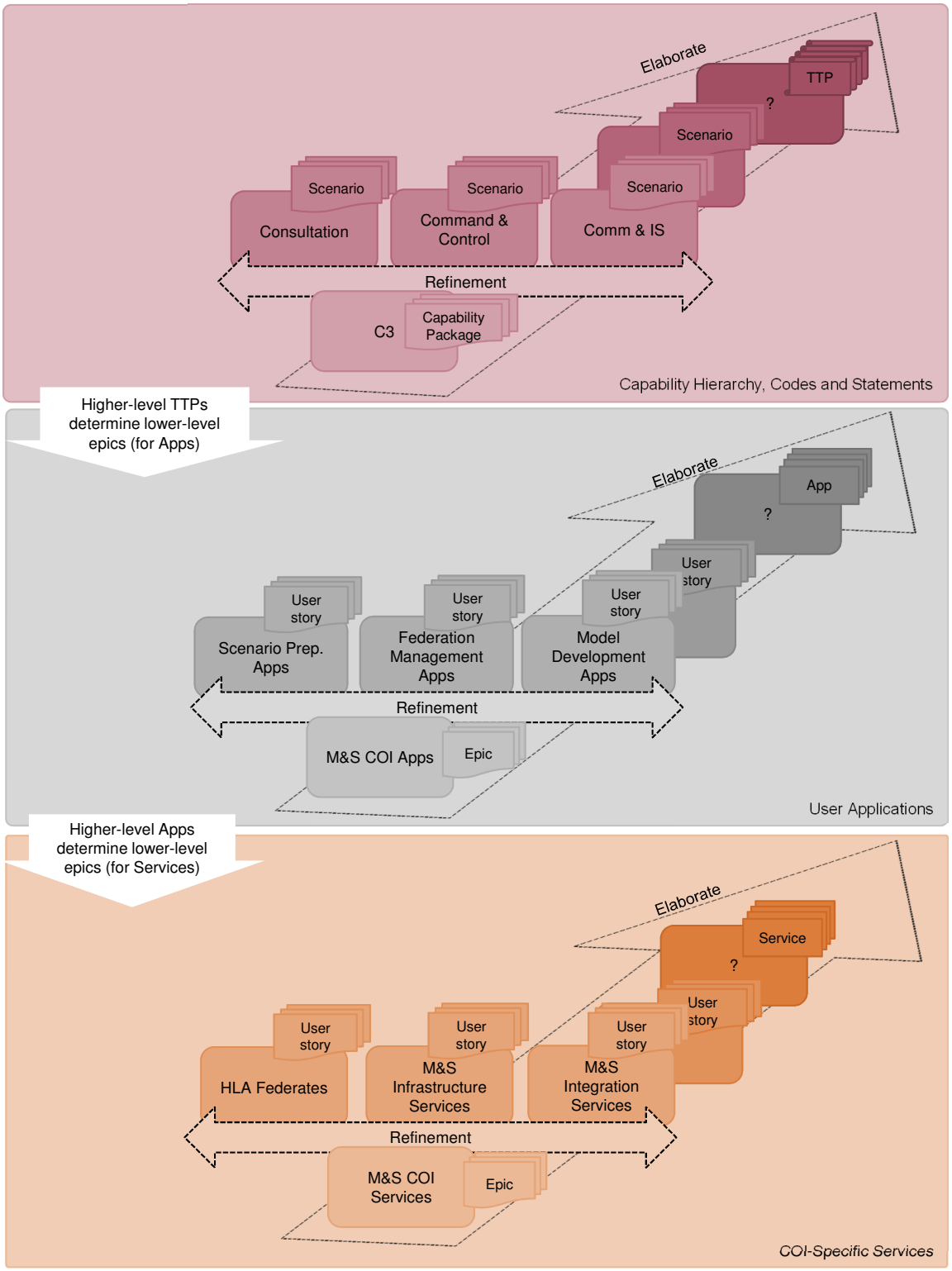

Figure 3.3 Uniform approach to elaboration and refinement of requirements in the C3 Taxonomy. The epic-elaborate-refine trees are placed horizontally away from the reader.

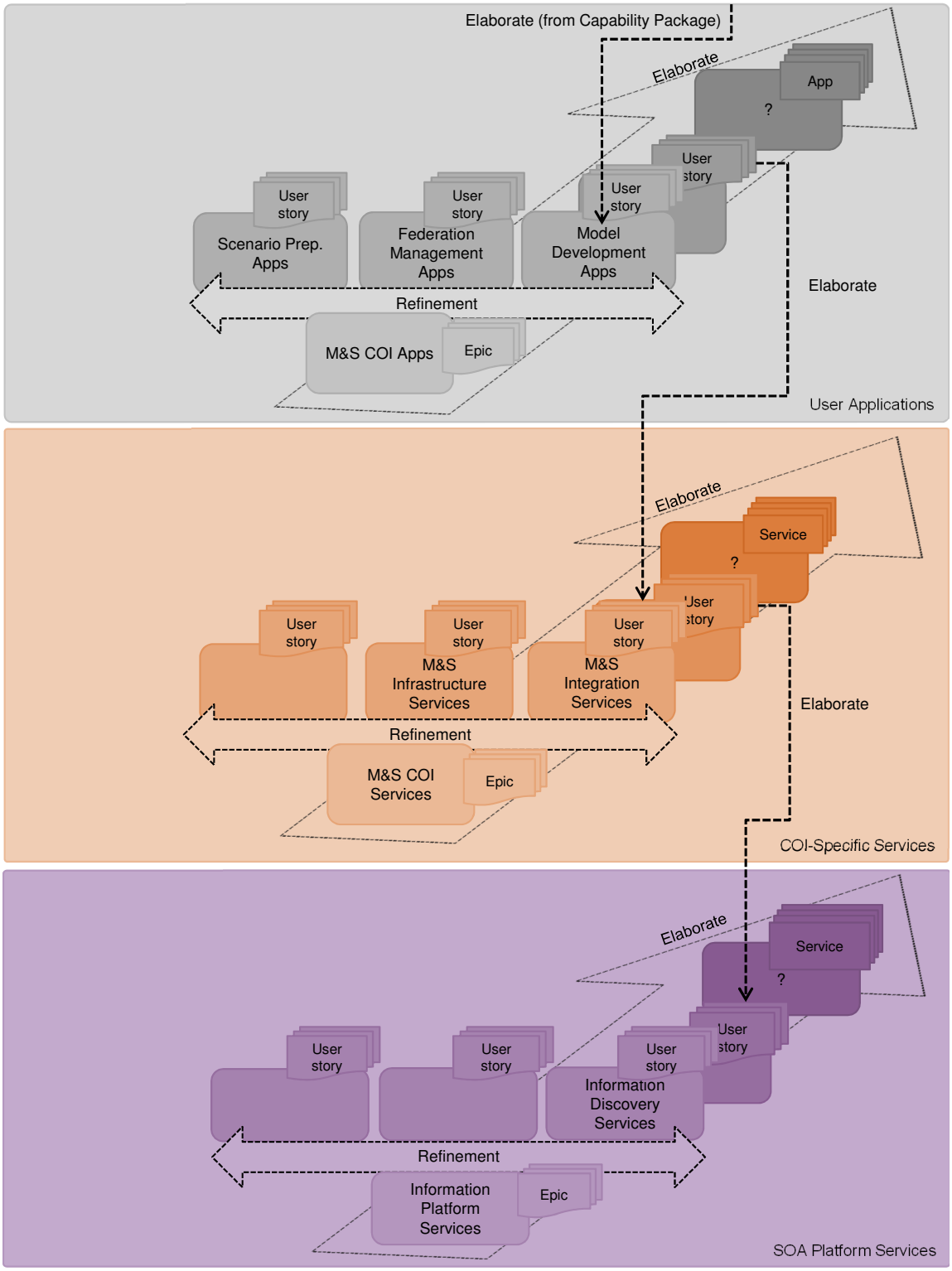

Figure 3.4 Uniform approach to elaboration and refinement of requirements in the C3 Taxonomy. Traditional development.

specificity ceases at the bottom of the Core Enterprise Services, on grounds that the layers below are generic for all higher services.

In general, the main point is:

> *Working in this manner, the C3 systems portfolio and its architecture are developed iteratively and incrementally, and they are developed in parallel in an interleaving manner, with production elements as the driving artefacts.*

In line with current views on large agile development; e.g., [20], we postulate that this is a viable method for gaining headway in developing the portfolio, including the NII and the INI, because both architecture development and systems development are captured in the same process and because a focus on production elements keeps one from the well-known tendency of loosing oneself in overly ambitious models and meta-models of architecture and system.

3.3.1 The ideal situation and the way there

At the future utopian limit, when the C3 systems portfolio and the NII and INI are perfectly service oriented, new user applications might be composed by orchestrating COI services on the fly. This will perhaps be done by using a user application for orchestration which consults a repository of service descriptions. Thus, epic-elaborate-refine trees in the User Applications layer will need only relate to its own layer (and indirectly to the layer below via the repository). And similarly for the other layers of the taxonomy. In other words, development can proceed (strictly) horizontally in Figure 3.3.

Moreover, the "people, then process, then technology" statement made for NNEC in Section 1 may be realized as promised by SOA: The concrete TTPs which are developed and populate the leaf nodes of the elaborate-refine trees of epics (capability packages) in the Operational Context lay the basis for epics on the next layer of the taxonomy; i.e., the User Applications layer. These epics are then, in turn, elaborated and refined until concrete applications populate the leaf nodes in their respective elaborate-refine trees; and so on through all layers of the C3 Taxonomy. This ensures the continuous guidance of operational context in developing the C3 portfolio.

We are not at this utopian limit. Nevertheless, it is important to keep the ideal situation in view as a guidance for developing the portfolio. It is not currently feasible to develop in a strictly top-down manner, in the sense that one cannot suspend developing core enterprise services until all business and operational processes have been defined and until all end-user applications have been developed. But the end results should nevertheless bear witness of the top-down structuring. In other words, the "people, then process, then technology" statement must be evident in the resulting portfolio. This is possible by following best practices regarding incremental development wherein not only requirements (use cases and scenarios) are refined and updated, but where also business cases are updated during development. Thus, each elaborate-refine tree undergoes revision and the C3 Taxonomy is constantly sanitized in accordance to current understanding of operational context.

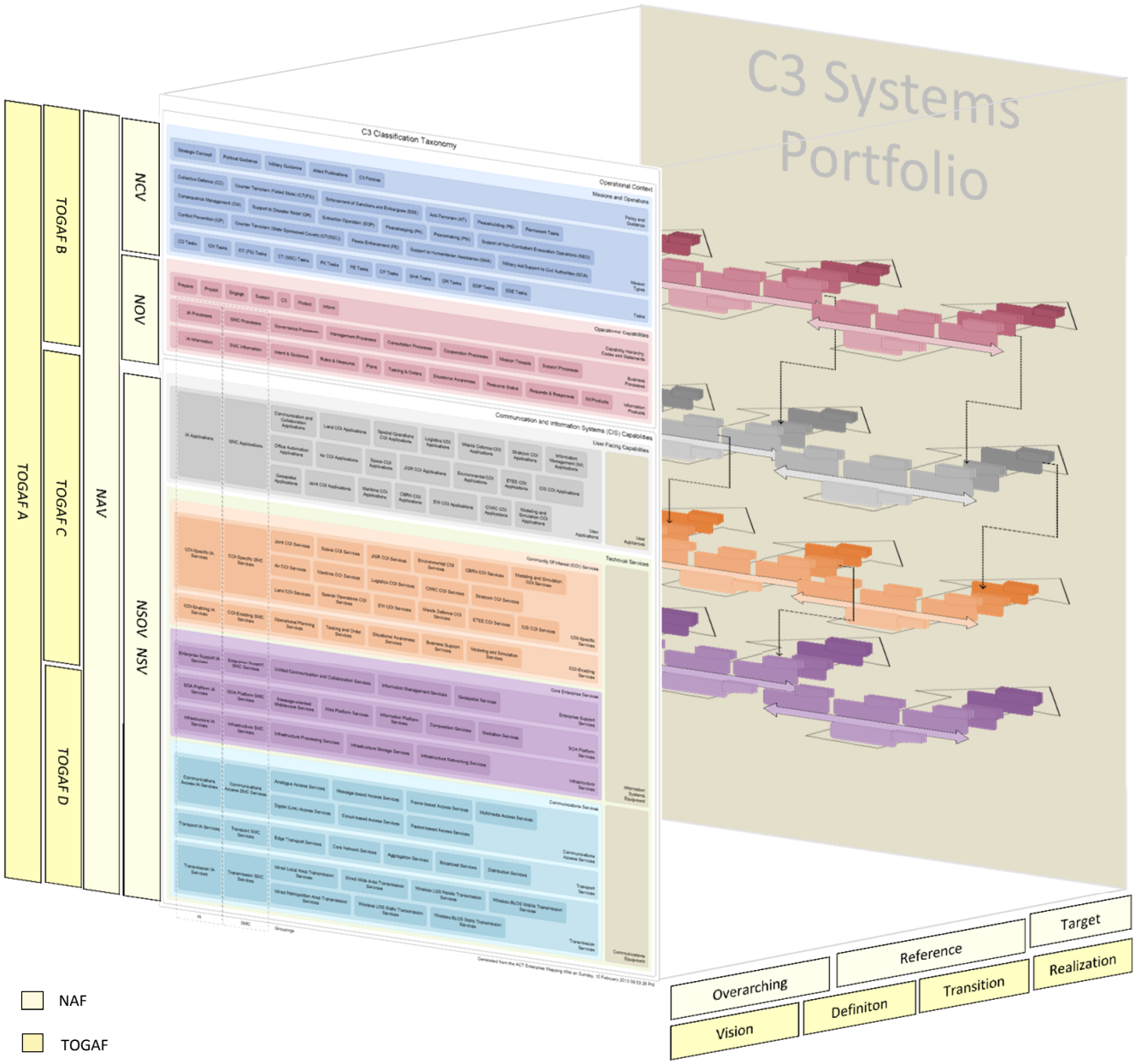

Figure 3.5 *Using the C3 Taxonomy to develop the C3 systems portfolio, including the NII and INI. Each category in the taxonomy is the starting point for elaborate-refine development trees which lead to the working systems portfolio.*

On the way to this future limit, software will be in a transition from traditional (legacy) software to service-oriented applications and services in the C3 Taxonomy sense. At early stages in this transition, traditional forms of software development will take place. This involves epic elaborate-refine trees that penetrate most technical layers of the taxonomy: To construct an application in the absence of orchestration, service repositories and suitable services, one has to involve software at all layers (the conventional three-layer interface-business-data architecture being a shallow example). Thus, the more detailed use cases will often be expressed at lower layers of the taxonomy. Although this temporarily compromises the ideal situation, this more realistic process should ensure cohesiveness between layers at early stages of development, which should gradually develop into loose coupling in the SOA sense. This situation is depicted in Figure 3.4, where development at higher layers introduces user stories at lower layers directly. In addition, development will certainly have to proceed at all layers in parallel, both to explore technical feasibility and give feed-back to higher layers in the taxonomy and to be ready once operational capabilities are fully developed.

Figure 3.5 summarizes our ideas in this section. The C3 Taxonomy can structure a unified product element approach to developing the C3 systems portfolio – and its architecture, where operational context guides development, even at stages of immature service orientation. The figure relates to NAF and TOGAF, which we cover in the next section.

3.3.2 NAF and TOGAF

The above approach is agile in spirit. Therefore, we promote the use of documentation and description not as driving artefacts, but as supporting artefacts. The NATO Architecture Framework (NAF) [74] is proposed as a description and documentation tool for Norway's INI. It provides a range of guidelines on how to describe and document an architecture. It suggests organizing development and results in views:

NAV NATO All View
NCV NATO Capability View
NOV NATO Operational View
NSOV NATO Service-Oriented View
NSV NATO Systems View
NTV NATO Technical View
NPV NATO Programme View

For example, in one of the sub-views in the Capability View one should describe business case and impact goals, while the Systems View has several sub-views giving guidelines for interfaces and data flow at the programmer's level. The layering of the NAF is compatible with the C3 Taxonomy. Although NAF is not agile in style, it would be useful to utilize NAF at all levels. The important point in our approach is that it is the use cases (epics and user stories/scenarios) which are the artefacts that go into production. NAF documents may then be attached to these artefacts as needed.

The Open Group Architecture Framework (TOGAF) is proposed as an architecture development method for Norway's INI. TOGAF comes with an Architecture Development Method (ADM) which "provides a tested and repeatable process for developing architectures. The ADM includes [...] developing architecture content, transitioning, and governing the realization of architectures. All of these activities are carried out within an iterative cycle of continuous architecture definition and realization that allows organizations to transform their enterprises in a controlled manner in response to business goals and opportunities" [45]. Phases within the ADM are as follows:

Preliminary Phase: Preparation activities required to create an Architecture Capability including customization of TOGAF and definition of Architecture Principles.

Phase A: Initial phase of an architecture development cycle. It includes defining the scope of the architecture development initiative, identifying the stakeholders, creating the *Architecture Vision*, and obtaining approval to proceed with the architecture development.

Phase B: Development of a *Business Architecture* to support the agreed Architecture Vision.

Phase C: Development of *Information Systems Architectures* to support the Architecture Vision.

Phase D: Development of the *Technology Architecture* to support the agreed Architecture Vision.

Phase E: Initial implementation planning and the identification of delivery vehicles; i.e., *Opportunities & Solutions* for the architecture defined in the previous phases.

Phase F: How to move from the Baseline to the Target Architectures by finalizing a detailed *Implementation and Migration Plan*.

Phase G: Gives *Implementation Governance* for architectural oversight of the implementation.

Phase H: Gives *Architecture Change Management* to establish procedures for managing change to the new architecture.

Requirements Management: Examines the process of managing architecture requirements throughout the ADM.

There is an explicit emphasis on iteration within and between phases, which makes TOGAF in line with current best practices. There have been efforts to integrate TOGAF and NAF; e.g., [44]. Both NAF and TOGAF are extensive frameworks, which require considerable expertise both to employ in full or to select sufficient parts to use. When using complex methods it is important to retain focus on and to keep track of the production elements. The overlying simplicity of epics and user stories/scenarios as the driving artefacts in developing the C3 systems portfolio is intended to enable one to keep this focus. Figure 3.5 indicates at a very high level how NAF and TOGAF may relate to the development approach suggested above. It is necessary to detail this picture in detail in order to give a viable process for developing the portfolio which integrates NAF, TOGAF, the C3 Taxonomy and modelling efforts within NATO such as MIP's capability package concept.

For M&S, several development methods have been devised. Notably, the Federation Development and Execution Process (FEDEP) IEEE 1516.3-2003 [81] is minted toward developing HLA federations (Section 6), and the Distributed Simulation Engineering and Execution Process (DSEEP) IEEE 1730-2010 [39] is a generalization of the FEDEP intended for developing (and utilizing) dis-

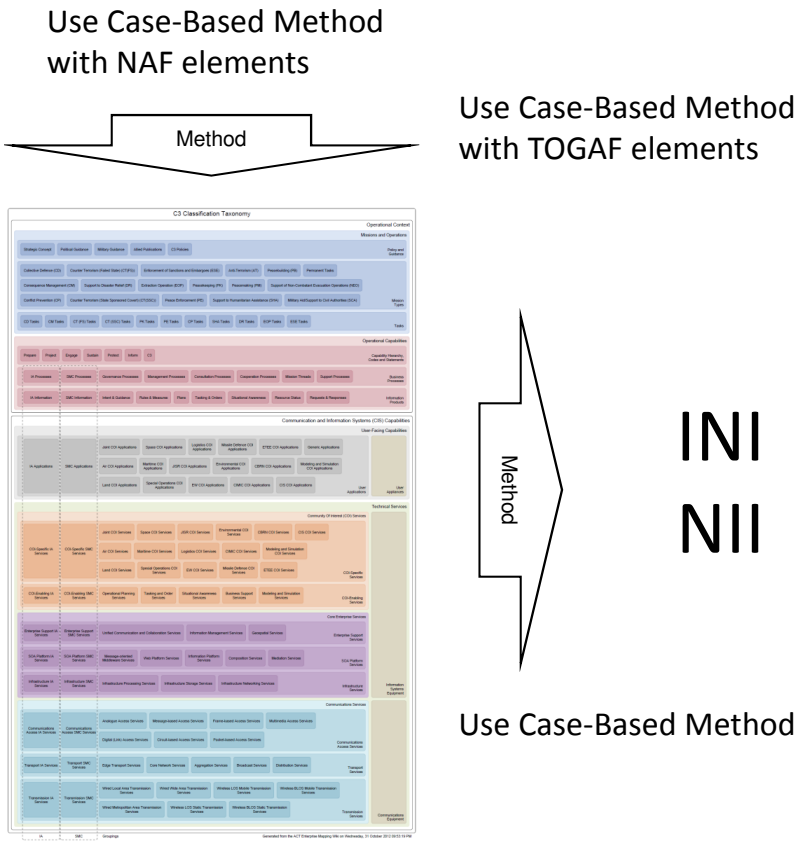

Figure 3.6 *Use-case-based method to develop architecture and information infrastructure using elements of NAF and TOGAF.*

tributed simulation systems in general; see [117] for an overview. Both methods are waterfall [51], but are still applicable in the elaborate-refine dimension in our approach.

To summarize, we propose to drive the development of the architecture and the C3 systems portfolio, including the INI or NII, by using use cases as production artefacts. We suggest to use elements of NAF and TOGAF as needed; see Figure 3.6.

4 Tentative use cases for modelling and simulation

We shall now put forth use cases that will help us to structure the discussion on how M&S should be integrated into the C3 systems portfolio and information infrastructures such as the INI and NII. We shall focus on the CIS Capabilities part of the C3 Taxonomy, and we will start at the User Applications layer. The Operational Context in the C3 Taxonomy has not yet been populated to any substantial degree, so the idea that the concrete operational capabilities residing in the taxonomy should be the base for defining epics belonging in the layers below is not realizable as such. We will therefore follow a tentative approach in this report, in order to get the discussion under way. There

are, however, documents that do their part in defining the operational context. For M& S, there is the NATO Modelling and Simulation Master Plan (Version 2.0) [77] which might reside in the C3 Policies category, but work on the operational context is outside the scope of this report.

Our discussion is intended as merely one of many preliminary efforts toward setting the stage for more structured processes. In our tentative approach, we will relate to a straight-forward categorization of operational capabilities that has arisen in informal discussions between defence personnel, research colleagues and ourselves on M&S and defence infrastructures: In order to get started at the User Applications layer, we will simply suggest use cases under the headings "Training", "Mission Rehearsal and Planning", "Missions and Operations", "Retrospective Analyses", and "Concept Development and Experimentation (CD&E)". These are somewhat in line with the NATO Modelling and Simulation Master Plan (Version 1.0) which state the following application domains for M&S [76], see also [119]:

- Defence Planning
- Training
- Exercises
- Support to Operations

In the present version (Version 2.0) of the Master Plan the application areas are [77]:

- Support To Operations (Operational Planning, Analysis, Decision-making)
- Capability Development (Defence Planning, Concept Development & Experimentation)
- Mission Rehearsal
- Training and Education
- Procurement

In other words, there are relevant application domains for M&S in an information infrastructure which we do not address in our tentative approach.

Our use cases are, at present, not elicited in a joint stakeholder forum; instead they are synthesized from literature and from our experience as researchers in the field. Moreover, we only present use cases that are specific to M&S integrated in an information infrastructure. Use cases are, in fact, requirement specifications. We must be precise about what these requirements are supposed to specify. The question of whether M&S should be integrated into defence information infrastructures is not our focus; that discussion has been resolved other places. Instead we focus on *how* M&S should be integrated; what should be applications, what should be services and at which level of granularity should applications and services be modularized and defined. Therefore, the use cases we propose are requirements for how users of the C3 systems portfolio, NII and INI, with M&S applications and services integrated, will use M&S in conjunction with other applications and services in the portfolio. Note that users of the portfolio include stakeholders that access the portfolio at various layers as defined by its architecture (expressed through the C3 Taxonomy); from operational personnel using User Applications to application/service users and providers, owners, developers and IT

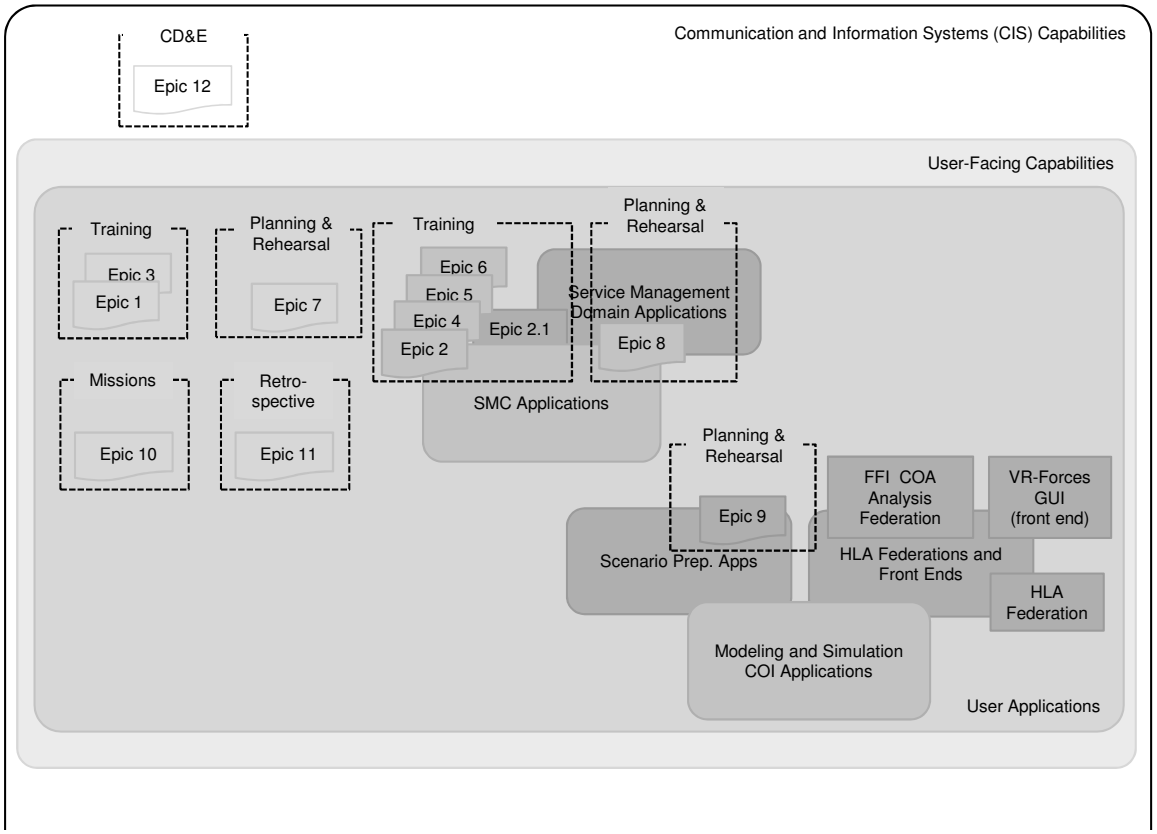

Figure 4.1 User stories tentatively placed in C3 taxonomy categories – User-Facing Capabilities part.

service personnel accessing the Technical Services. Users of the portfolio also include applications and services that use other applications and services.

We will illustrate by means of three stakeholder types: The "operational personnel" stakeholder represents end users with an interest in using applications and services for defence activities under the above headings. The various "operators" stakeholders represent personnel who is capable of combining services and applications (by using suitable applications); e.g.; a training operator for a Computer-Assisted Exercise (CAX). The "software developer" stakeholder represents IT expertise capable of designing and developing applications and services from lower-level services or from scratch. In relation to the CIS Capabilities, the first two types of stakeholder are typically users at the User Applications layer, while the third type of stakeholder is a user at the Community of Interest (COI) Services and the Core Enterprise Services layers. We will include impact goals for illustration only. These are not the results of a business case analysis.

We will tentatively place each use case in a category in the C3 Taxonomy. The appropriate category is determined in the "**by using**" clause in a use case, since this is were the direct demand on the system; i.e., the portfolio, is expressed. We summarize the location into categories in Figures 4.1 and 6.3. We allow ourselves to place use cases at all levels of specificity in the taxonomy.

4.1 Training

Training here means to perform tasks in order to gain procedural knowledge; i.e., to gain practical experience to some extent, prior to, or outside of, performing actual tasks in the actual job or performance situation. Procedural knowledge differs from declarative knowledge which is obtained from studying books or attending lectures or courses.

Training is common in many disciplines, for example in sports and the arts. It is less common in less manual disciplines, where it is less straight-forward to define practice tasks and perhaps difficult to define even the actual work tasks themselves. In civilian management, which involves judgement and decision making in a highly changing environment, the idea of training is typically undeveloped. Expertise is obtained over years of actual job practice through trial and failure in actual work life; often with catastrophic outcomes.

In contrast, the military has held a conscious focus on training on all levels of organization, from operational and decision-making levels down to the war fighter. Modelling and simulation play an essential role in this. In defence training, one plans to use the full range of Live-Virtual-Constructive simulation. The "live" aspects involves real personnel with real equipment, but where sensors detect e.g., movements and simulated hits (e.g., by laser, rather than live ammunition). The "virtual" aspect involves real personnel with simulated equipment; for example, pilots training in an aircraft simulator. The "constructive" aspect involves simulated personnel with simulated equipment, as when various simulation systems, such as Joint Theatre Level Simulation (JTLS), simulate troop and vehicle movements according to battlefield commands and terrain data; thus avoiding the logistics involved with live training. All LVC aspects can, in theory, be combined in various measures to cater for the training objective at hand; from small-scale personnel training in defence academia to large-scale joint defence exercises involving all of LVC or the constructive aspect only.

4.1.1 Train as you fight.

The maxim "Train as you fight" embodies the idea that training for a task should be undertaken on tasks as close to the actual task as possible in an environment as close to the actual environment as possible. If you undergo training with tools different from those you will use in an operation, you will be unprepared for the operational situation to the extent the tools differ in essential characteristics. The same goes for differing environments.

> Epic 1: **As** operational personnel **I can** train realistically **by using** User Applications **in order to** increase skills by a factor s and reduce time to proficiency by $x\%$.

M&S provides substitutes for actual tasks and environments where the latter are not feasible during training. However, it is important that participants view and manipulate the simulated battle situation through their regular applications used in actual operations (these are often complex); otherwise they will not be training a vital part of their operational task [116, 119].

Thus, in order to use M&S to train as you fight, the relevant simulation system must be integrated with, and front-ended by, the appropriate application(s) used in operations. For example, in a CAX involving computer-generated forces (CGF), simulated entities might be controlled through a command and control (C2) system; rather than directly in the simulation application itself (even when the simulation application has a better user interface). Another example is embedded training, in which on-board control systems in actual vehicles are stimulated with simulated data.

Further, the simulation application and operational application should be loosely coupled, in the sense that it should be transparent to the user of the operational application which simulation system it is that is feeding simulated data. Barring security issues, it should also be possible for the user to not see any difference between simulated and real data. Moreover, evolving combat structures demand new training regimes. Therefore, an essential capability lies in combining various operational applications with various simulation applications according to the training objectives at hand.

> Epic 2: **As** training operator **I can** compose realistic training regimes **by using** a SMC Application **to** combine M&S COI Applications with other User Applications **in order to** increase skills by a factor s and reduce time to proficiency by $x\%$.

The Service Management and Control (SMC) Applications manage, control and monitor services in all layers of the portfolio. In Epic 2, it is envisioned that there is an application in that category which can be used to combine systems at the User Applications level.

For illustration, we elaborate Epic 2 according to the current state of the C3 Taxonomy. We use a "dot"-notation and indentation to indicate subcategories.

> Epic 2.1: **As** training operator **I can** compose realistic training regimes **by using**
> SMC Applications **.**
> Service Management Domain Applications **.**
> Service Inventory Management Applications **and**
> Service Availability Applications **and**
> Service Portfolio Management Applications
> **to** find, request use of, and combine
> User Applications **.**
> M&S COI Applications **.**
> HLA Federations and Front Ends **with**
> Joint COI Applications
> **in order to** increase skills by a factor s and reduce time to proficiency by $x\%$.

The category User Applications **.** M&S COI Applications **.** HLA Federations and Front Ends does not exist at present. Our intention is that it contains concrete HLA-federations or their front ends (Section 6); i.e., simulation systems available to end-users.

4.1.2 "Train as you fight" is not enough.

Realistic training is important. However, research in various domains has shown that training that simply reflects actual circumstances is not sufficient. For the defence domain, Shadrick and Lussier remark: "The maxim 'Train as you fight' has risen to such a level of familiarity in the U.S. Army that the value of the notion goes almost unquestioned. Yet studies of the development of expertise clearly indicate that 'as you fight', [...], is neither the most effective nor efficient method of developing expertise" [102, p. 294]. For example, while it is pertinent to train using actual tools in an actual environment, it is important to focus training on the difficult parts of a task. Such targeted training is well-known: a musician will repeatedly focus on a difficult passage, thus engage in an artificial behaviour compared to an actual performance, while still using his actual instrument.

> Epic 3: **As** operational personnel **I can** train artificially **by using** User Applications **in order to** increase skills by a factor s and reduce time to proficiency by $x\%$.

In general, repetition frequency in task exposure should be designed from an understanding of risk, where *risk* = *likelihood* x *consequence*, rather than on likelihood alone [102]. Modelling and simulation enables risk-based task repetition. For example, flight simulators enable pilots to drill emergency scenarios in take-off and landing routines, which has very low likelihood, but extremely high consequence, compared to normal in-flight routines.

> Epic 4: **As** training operator **I can** compose risk-based training regimes **by using** a SMC Application **to** combine M&S COI Applications with other User Applications, during training **in order to** increase skills by a factor s and reduce time to proficiency by $x\%$.

In addition to repeating "hard passages", it may be necessary to engage in artificially enhanced tasks to heighten performance. Such tasks are the result of *task analyses* and depend on the actual task at hand. There are many ways to analyse tasks, but the important focus here is on differences between what a task demands in terms of observable behaviour and what a task demands in terms of cognitive processes [60, 103]. So-called *job-oriented task analysis* focuses on the activities (sub-tasks), tools, products, and outcomes tasks and subtasks that must be done in order to complete a task. There is a lot of research on this area; see e.g., [10, 125, 12]. On the other hand, *worker-oriented task analysis* focuses on the task doer in terms of knowledge, skills, abilities, and other characteristics; for example personality, intelligence, and expertise [1, 99, 98, 35]. However, for analysing the thought processes which are involved, one must use methods of *cognitive task analysis*. Particularly relevant for judgement and decision making tasks, such analysis uncovers a range of unconscious processes as well as how decision-makers of varying degrees of proficiency think [72, 30, 49, 47].

In tactical decision making (for which Shadrick and Lussier's remark above was made), it is not sufficient to engage in normal training, even if it involves both realism and repetitions, since this does not in itself focus on developing decision-making skills [102]. Tactical decision makers must engage also in training that triggers the explicit development of thinking skills. Moreover, judgement

and decision tasks are often so-called inconsistent (different people develop differing successful strategies) [14, 13] or ill-defined (hard even to define successful strategy) [43, 104, 95, 122]. For such tasks, teaching any one particular strategy has shown to be futile [49, 102]. Instead, the idea is to train adaptability; i.e., how to adapt to unknown and surprising situations. In line with this, the notion of *adaptive thinking* [93] has been adopted in the defence domain for tactical decision making [103, 102].

> Epic 5: **As** training operator **I can** compose training regimes for adaptive thinking and decision making **by using** a SMC Application **to** combine M&S COI Applications with other User Applications, during training **in order to** increase skills by a factor s and reduce time to proficiency by $x\%$.

Training for adaptive thinking requires artificial tasks (to trigger responses to unfamiliar or unforeseen events) [37, 112] or standard decision tasks in a deliberate practice setting [103, 102]. Deliberate practice [25] is a framework that takes the short-comings of "learning on the job" head on. Apart from a strong focus on difficult aspects and risk-based repetition, its essence lies in immediate and tailored feedback (by a coach or computer-adaptive system) followed by immediate tailored re-trials; and a focus on integrating parts which have been re-trained into the whole task.

Modelling and simulation is essential in both enabling the use of artificial tasks and enabling deliberate practice regimes. Artificial tasks have to be simulated by necessity, while deliberate practice demands flexibility and adaptability in the training regime. The active coaching and tailored re-trials in deliberate practice demands capabilities that allow the coach/trainer to call M&S components into play during training.

> Epic 6: **As** training operator **I can** compose deliberate practice regimes **by using** a SMC Application **to** combine M&S COI Applications with other User Applications, during training **in order to** increase skills by a factor s and reduce time to proficiency by $x\%$.

4.2 Mission rehearsal and planning

In contrast to training, which is a long-term activity, mission rehearsal and planning is a short-term activity for a specific mission or operation.Traditionally, round-the-table war gaming has been used in rehearsal and planning. This is cumbersome and time consuming and gives little time for playing through alternative scenarios and plans. Also, although the group process implied in war gaming is regarded as beneficial, there are issues of diverging perceptions of the situation which are not easily resolved in traditional settings. Simulation aids all the above in that it enables frequent replaying of plans and scenarios and the virtual playing out and testing of refined plans. Through role playing, simulation also enables building of common mental models of the mission or operation.

Epic 7: **As** operational personnel **I can** plan and prepare for missions **by using** User Applications **in order to** increase quality of plans by a factor q and increase common understanding by a factor c.

4.2.1 Rapid composability

A Commander of the U.S. Army Training Support Center states: "Nowhere is the need for rapid environment shaping more important than in mission rehearsal for real-world operations. In this case, both speed (i.e., rapid scenario generation) and accuracy are paramount to prepare military forces for imminent deployment and the conduct of operations. Current joint M&S is more suited to an 18-month JELC[1] vice the much shorter period required for rapid mission rehearsal (i.e., as short as 3–7 days). To achieve shorter planning cycles prior to training events, joint M&S solutions must be far more flexible than they have been – they must also be composable at the trainer level in order to provide units realistic rehearsal opportunities apart from simulation centers or established training capabilities that might not be able to respond in short time periods" [23].

Epic 8: **As** mission rehearsal operator **I can** prepare personnel for operations **by using** a SMC Application **to** combine component-size M&S COI Services and other COI-Specific Services rapidly and readily, prior to rehearsals and planning **in order to** reduce time to mission rehearsal readiness to t hours/days.

4.2.2 Scenario generation

The Commander continues: "Composability is more than just speed, however, it is also about framing the environment accurately and in so doing, limiting those parts of the environment that don't need to be modeled – that is, gaining efficiencies and reducing complexity in areas that aren't central to the training objectives. The trainer is closest to the unit and understands the training objectives best. He or she is best placed to select the geographical "playbox", the various factions in the battlespace, and the limiting factors that will shape the unit's operations. By allowing the trainer to frame the scenario directly and enabling rapid, and intuitive scenario generation, large-scale manpower savings can be achieved and unit training objectives are more likely to be best served. Composability also implies using only that which is needed, thus less M&S resources (computing power, bandwidth, etc.) are likely to be consumed when better environment precision is realized" [23].

Epic 9: **As** mission rehearsal operator **I can** construct scenarios at appropriate levels of detail without the aid of "M&S professionals" **by using** a M&S COI Applications**.** Scenario Preparation Application **to** combine component-size M&S COI Services and other COI-Specific Services rapidly and readily, prior to, and during, rehearsals and planning **in order to** improve mission rehearsal and planning quality by a factor y.

[1] Joint Exercise Life Cycle (JELC) is the series of planning conferences and meetings, along with database work, that frames training audience training objectives and formulates MSEL [Master Scenario Events Lists] and M&S environments to support the training objectives. The JELC for large-scale joint exercises can take 12–18 months prior to execution of the training event. [Footnote part of quote]

4.3 Missions

The capabilities given by M&S in the portfolio in the above epics transfer directly through to missions and operations. Plans devised during mission rehearsal and planning could be given new parameters during operations to reflect actual events, and the plan can then be re-simulated on this new knowledge. If the plan and simulation is platform-independent, the plan could be appended to communication between echelons; e.g., to actual battle orders, giving orders a new dimension. Replaying actual events in a simulation engine at high speed may also uncover trends in slow troop movement that would otherwise be hard to discern.

> Epic 10: **As** operational personnel **I can** update plans during missions with situational intelligence and then extrapolate to evaluate future best course of action **by using** a User Application **to** recombine and rerun plans developed in mission planning and rehearsal **in order to** improve quality of situational awareness for missions by a factor a.

4.4 Retrospective analyses

Systematic and targeted post-operation analysis enables learning, even in the complex system that warfare constitutes. Replaying actual events and analyzing these in relation to plans and scenarios (e.g., in terms of what-if analyses) should give valuable learning input.

> Epic 11: **As** operational personnel **I can** conduct post-operation analyses **by using** a User Application **to** rerun recorded events and recombine and rerun plans based on recorded or alternative events **in order to** improve learning from experience by a factor l.

4.5 Concept development and experimentation (CD&E)

During a demonstration of a piece of simulation software developed at FFI, a LtCol remarked to us the following insight, which exactly captures the motivation for service orientation and the need for agile usage: When clever people get their hands on powerful tools, they will inevitably find novel ways to use them and apply them to entirely new situations. This is an "emergent effect" [105] arising in a federation of systems; i.e., an effect that is not obvious from regarding the system *a priori*. By enabling increased flexibility, M&S should spur the generation of emergent effects, but M&S is also central in *planned* activities for experimentation due to increased possibilities for controlled experiments and other analytical and empirical studies [3, 2, 48].

> Epic 12: **As** a user of the portfolio **I can** conduct studies on Communication and Information Systems (CIS) Capabilities **by using** M&S with other Communication and Information Systems (CIS) **in order to** improve innovation by a factor v.

The "user" here covers any user of the portfolio at all levels. The "user" is required to be a "reflective practitioner" in the sense of [7, 8, 41], who may, or may not, be aided by academic researchers.

The epics in Section 4 are expressed at the User Applications layer and above. These use cases lay explicit and implicit demands on lower technical layers of the C3 Taxonomy. To understand what these requirements are, we must understand the characteristics of these lower layers. In the next sections we give an overview over Service-Oriented Architecture and the High Level Architecture.

5 Service-Oriented Architecture

It is explicitly stated at the strategic level that Norway's INI and NATO's NII, and the entire C3 systems portfolio should have service-oriented architectures. Service-orientation is perceived as instrumental in implementing a network-enabled defence regime.

Service-Oriented Architecture is an architectural software systems design and run-time approach which promotes abstraction, loose coupling, reusability, composability and discovery. It is easily seen that SOA inherits several ideas from object-oriented and component-based software and systems development. However, SOA extends, and in some important cases, deviates from the dogmas of classic object-orientation.

In use, SOA has three main actors: the *Service Provider*, the *Service Consumer* and the *Service Registrar*[2]; see Figure 5.1. A service provider registers a service it wishes to provide to the community with a registrar. A consumer looking for a service consults the registrar for suitable services, and if found, establishes a binding between a concrete *Client* within the consumer and a concrete *Service* hosted by the relevant provider, where the terms of this binding is agreed upon in a *Service contract*. For the purpose of our discussion, we will detail parts of this constellation below.

5.1 The Consumer-Provider relationship

Conceptually, SOA prescribes that services are developed, hosted and maintained by a service provider (who presumably is an expert on that service), whereas data is owned by a service consumer (who presumably is an expert on those data) and can be processed by requesting the hosted service. This is in direct contrast to classic object-orientation, where data and the functionality to process that data are packaged together in an abstract data type [33, 54, 92, 34].

The idea of an abstract datatype is that users of the datatype need no information on the internal data representation or functional implementation. The implementer is then free to choose the combination of data representation and functional implementation which optimizes computation, conceptual understanding, correctness reasoning, etc. While encapsulation as a principle is carried forward by SOA, the extremely tight coupling between data and functionality is not. SOA goes further on the encapsulation idea in that functionality and data are encapsulated separately. Not only need a consumer not be concerned about how functionality is implemented behind the scenes; the service provider need not concern itself with how data is represented internally at the customer.[3] In ad-

[2]The Registrar often goes by the name "Service Broker". There is unclarity around this terminology, but we will use "Broker" to denote a somewhat different entity later.

[3]Home media appliances have been used to exemplify SOA. For example, following the abstract data type paradigm,

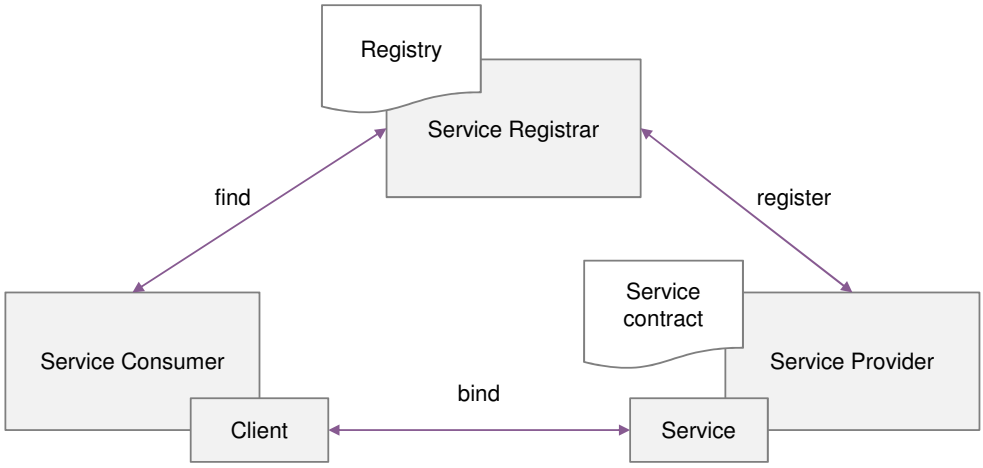

Figure 5.1 SOA Triangle (adapted from [26].

dition, both data and functionality may be complex and may viably be constructed by composing sub-data and sub-functionality using other providers. Further, to realize genuine decoupling of data and functionality, and of sub-components of both data and functionality, it must be possible for consumers and providers to be located at different places and on different platforms.

All this decoupling presents issues of communicating data and service requests between the consumer and provider; i.e., the bind between the specific client and service. The idea of message passing arose from object-orientated programming and allowed very late binding. This was further developed in the various component-based software techniques [114]. SOA does not prescribe any specific technology for implementing consumer-provider communication. However, there is an obvious need for community-wide agreements (standards) on how to communicate. This creates market opportunities for the vendor who can deliver a framework which enables such standardized communications, and there are therefore certain prominent technologies in terms of both features and market; see Figure 5.2. Current practice mostly uses WWW technology; in particular Web Service technology [126]. There are mainly two modes of Web Service. The oldest mode passes messages in terms of the Simple Object Access Protocol (SOAP) [127], using XML [130] documents. Interface specifications and thus service contracts are written using the Web Services Description Language (WSDL) [128]. The newer Representational State Transfer (REST) style [28] passes messages using leaner formats such as JavaScript Object Notation (JSON) [46], Plain Old XML (POX) [130], or Rich Site Summary (RSS). Interface specification are here written in the Web Application Description Language (WADL) [129]. On the underlying communications layer, SOAP may use several protocols, the most common being the Hypertext Transport Protocol (HTTP), the File Transfer Protocol (FTP) and the Simple Mail Transport Protocol (SMTP). On the other hand, REST mostly uses HTTP. Web services adhering to SOAP are often referred to as "Big Web ser-

every CD would come with its own CD player, whereas SOA would prescribe things as they are; you may use the same CD in multiple players of varying quality of service [9, 32]. The analogy has also been used to illustrate object-orientation (beyond abstract data types) versus SOA [36]. Discussions on the aptness of the analogy illustrates that connotations of "object-orientation" are many-faceted and often include the later developments of component-based programming [114].

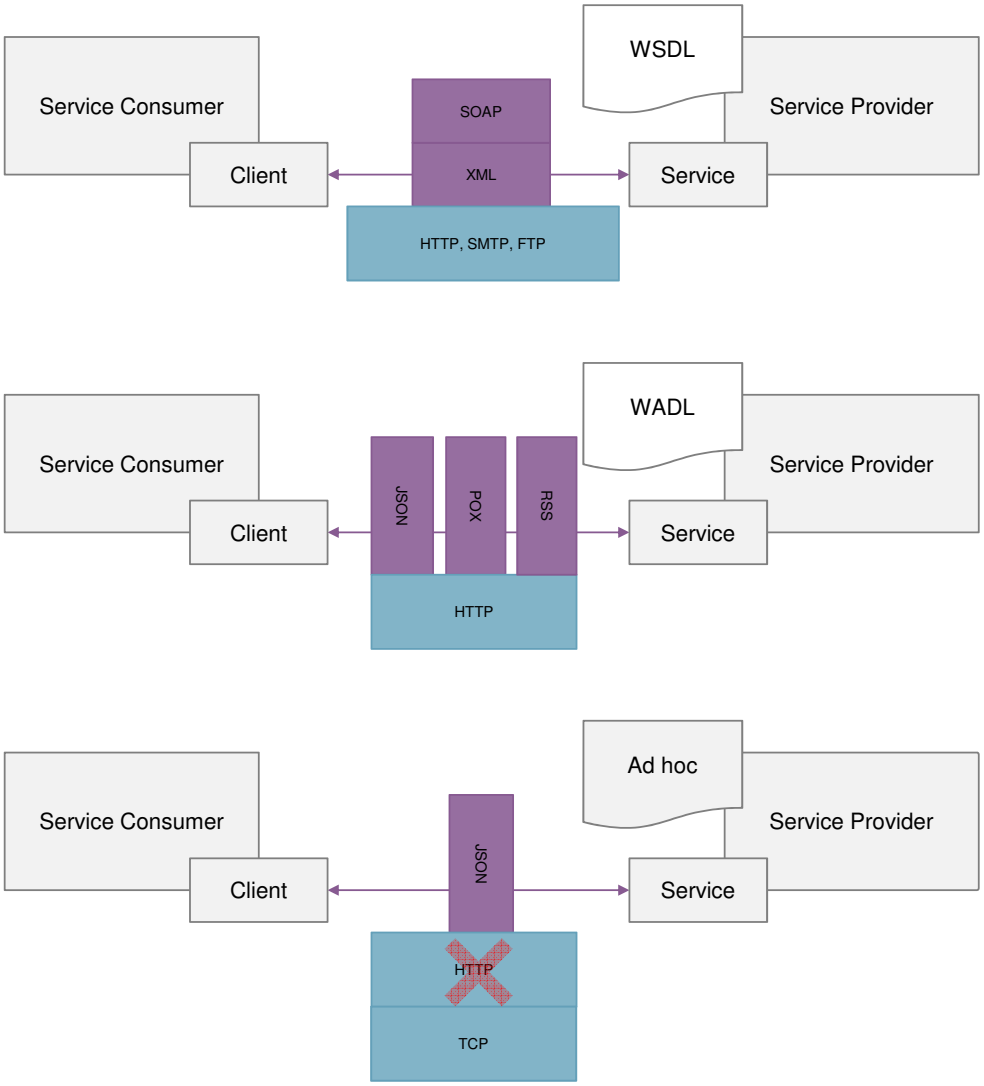

Figure 5.2 *Prominent technologies used for SOA. WebServices (WS-* or Big Web Services), REST (RESTful WebServices) and WebSockets.*

vices" or "WS-*" and web services adhering to REST often go under the name of "RESTful" web services. For a comparison of SOAP and REST and a guide to deciding between the two, see [90].

A recent development is the WebSocket Protocol [27]. Roughly, WebSocket APIs allow a user at the applications layer to bypass the layer of HTTP and do transactions directly on the layer below, using Transmission Control Protocol (TCP)-based connections. The WebSocket Protocol is part of the upcoming HTML5 specification and provides a full duplex, communications channel that operates via a single socket over the Web. In contrast, the HTTP protocol was not designed for real-time duplex communication, and one has to simulate duplex communication over two channels with substantial message transfer overhead. Websockets communicate binary data, unlike SOAP and REST which are text based. WebSockets are viewed by some as the next evolution in web communications and also essential for realizing SOA beyond the enterprise intranet [121] due to efficiency reasons. At the same time, others are sceptical that WebSockets will be useful beyond the

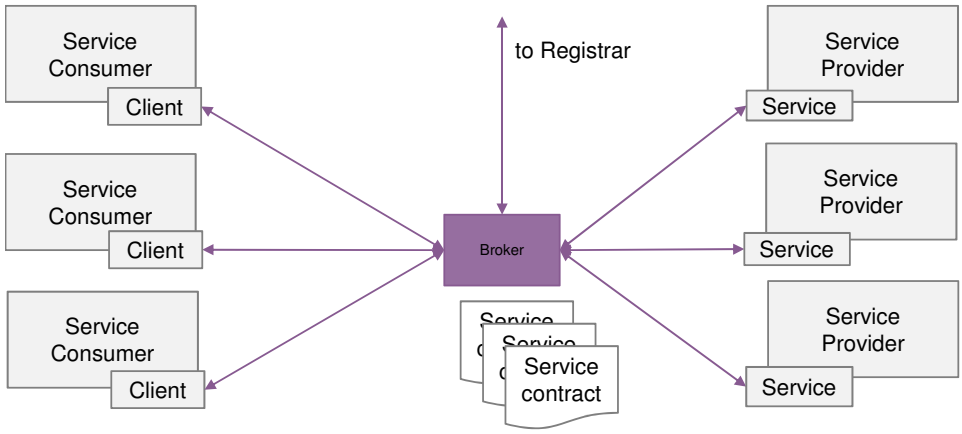

Figure 5.3 Broker technology.

intra-enterprise network, due to security reasons and the added load on other infrastructure when stripping messages of their HTTP headers, etc. [57]. Security can be provided by the web browser, but the lack of typing adds challenges when writing service contracts.

It may seem from Figure 5.1 that once a service is discovered, then binding and communication is always done directly between provider and consumer. This can often be the case. However, current technology also promotes the use of a Broker; see Figure 5.3. The broker facilitates publish/subscribe protocols, in which a consumer does not bind directly with a provider, but subscribes to services published by providers. The broker handles publication messages and routes service provider messages to those consumers who have subscribed to relevant services and routes consumer messages to appropriate publishers. The WS-BrokeredNotification standard (under the more general WS-Notification standard) [88] is a dominant technology for this. In this standard, the broker may, or may not interact with the registrar, and the broker may therefore take on various degrees of the registrars role.

Note that SOA's decoupling of data and functionality also means that there are no immediate data types or persistent objects that can administrate the system's state; i.e., the set of variables that record how the system processes its data over time. A loosely coupled service which is not aware of its potential consumer ahead of time, cannot necessarily update or keep track of the consumer's state. However, middleware – such as a broker – can be used as a central state-administration service. We will return to this later.

The need to integrate legacy software into new architectures is ever-present, also in SOA. A common way to service-enable existing software is to place a gateway (wrapper) in front of the software system, which translates communication from the software system to the standard communication used in the SOA; see Figure 5.4.

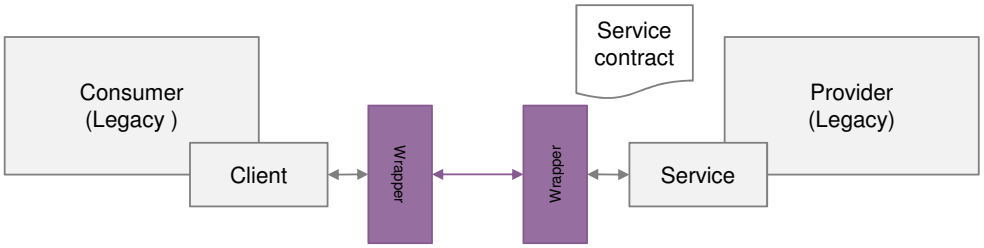

Figure 5.4 *Gateways (wrappers) are often used to service-enable software which was not designed service oriented at the outset.*

5.2 The Registrar role

SOA's principles may be used at all levels of technology, but also at all levels of organization. In fact, from a computing standpoint, SOA extends the ideas of object-orientation and component-based design to the organization. However, equally, if not more, important, is the converse influence of business process on software and systems design that SOA promotes.

Viewing software systems at the enterprise level already introduces integration issues when inter-department systems and old and new systems must communicate. Strategies such as developing *ad hoc* links between systems on the one hand or replacing the entire IT portfolio with an Enterprise Resource Planning (ERP) system on the other hand, have proven costly. Middleware solutions such as Enterprise Service Buses (ESB) which factor out the integration issues have often been developed in-house to cater for the technology of the organization's specific systems, thus effectively relocating integration problems instead of solving them; see e.g., [96] and Figure 5.5.

With the demands of network-enabled defence come cross-enterprise-demands on software services. Enterprise-specific integration solutions are no longer adequate. In addition, the unfortunate and common situation where an organization's business work-flows are dictated by the (usually bad) architectural design of its IT systems, becomes intolerable when multiple organizations are involved. Thus, the shift of focus toward facilitating globalized business processes necessitates abstraction, looser coupling, reuse, and composability on a whole new level.

Apart from decoupling data and functionality, SOA entails that services must be loosely coupled with respect to both location and time. This is especially relevant for network-enabled capabilities where mobile units move through disadvantaged grids [42]. The physical location and implementation details of each service must be transparent to consumers, and which services are available may vary over time. Thus, dynamic binding and dynamic service invocation are key capabilities which consumers and providers must support and be prepared for. This means that consumers cannot rely on knowing the names of services, or even their existence, at design time. Abstraction now also means abstracting away from programming language and run-time platform. Loose coup-

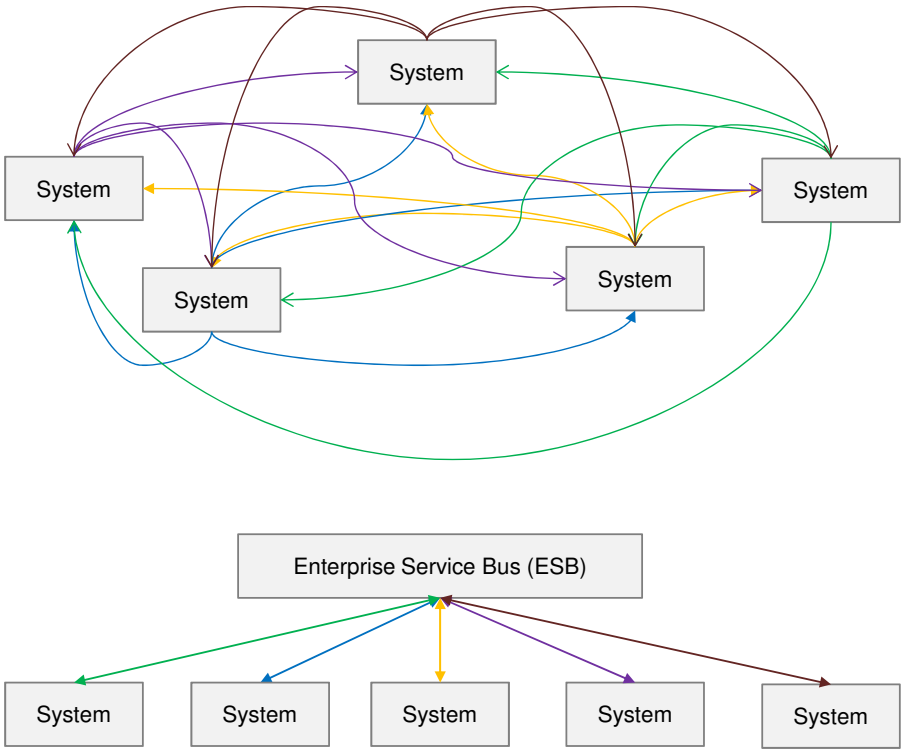

Figure 5.5 Integration headache. Enterprise Service Bus (ESB): Integration headache factored out.

ling, reusability, and composability now also mean interoperability, in the sense that services may dynamically discover and use each other, thereby generalizing the conventional client-server configuration. This inter-enterprise business-oriented demand on software systems is perhaps the largest conceptual leap that SOA offers from conventional component-based development [114].

The dynamic discovery mechanism through the Service Registrar is essential to the loose coupling of SOA. Without the discovery mechanism, services have to be known statically, thereby precluding the idea that services may be created and hosted independently of specific consumers. In fact, it has been argued that "SOA in practice" often does not include the discovery mechanism and only involves the lower part of the SOA triangle [63, 62]. For example, it is common to use WebService technology with no registrar and happily proclaim a service-oriented architecture. Without the discipline of using a service registrar, it is also tempting to revert to tighter coupling in other areas of the consumer-provider relationship, thereby making "SOA in practice" even less SOA.

For platform-independent and dynamic discovery, service interfaces have to be written in a uniform generic manner. This is a marked departure from Application Programming Interfaces (APIs) and Remote Procedure Calls (RPCs), where knowledge of the name and parameters of functionality must be known at design time. Again, SOA does not prescribe a specific technology for the service registrar or the relationships with it. In practice, it is the Web Services technology again that is the predominant technology. The Web Services Description Language (WSDL) [128] is an XML-based language for describing web services and how to access them. Data types can be specified using XML Schema in the WSDL file for a service, and a consumer that finds that a service is

appropriate by reading its WSDL file can subsequently call operations listed in the file using SOAP. The equivalent for RESTful services is the Web Application Description Language (WADL), also based on XML, but not a standard. However, the current version of WSDL, WSDL 2.0, also partially supports REST. Figure 5.2 incorporates the WSDL and WADL documents.

The registrar's role is to administrate a repository of service descriptions. Continuing the WWW technology regime, Universal Description, Discovery and Integration (UDDI) [87] is an XML-based registry designed to handle SOAP requests and provide access to WSDL documents describing web services. However, development and support for UDDI has by now ceased. Another standard is the WS-Discovery standard [89]. This approach is decentralized and based on mutlicasting advertisement and probe SOAP messages over the network.

Many of SOA's ideas are not possible to implement elegantly with today's technology. At present it is therefore important to be aware of the following distinction: Design-time discovery and runtime discovery. Today, most discovery is done by humans at design time [26]; in direct contrast to the true loose coupling ideal mentioned above. For example, a systems designer or software developer may consult a registrar to determine whether an appropriate service exists or needs to be developed. Runtime discovery in the ideal sense, would mean that a consumer could check at runtime whether an appropriate service is available and then automatically locate and consume that service – and even make the service inter-operate with other consumed services. This would require semantic interoperability and more [118]; technologies which are in their infancy. Runtime discovery and interoperability in the ideal sense is not, at present, wide-spread. Technologies such as UDDI and WS-Discovery are nonetheless geared toward runtime discovery. Currently, more static variants of the ideal scenario are possible. For example, a consumer may use WS-Discovery to probe for services, and consume services at runtime, provided that the consumer has been sufficiently prepared at design-time for consuming these services. Thus, run-time service discovery is more akin to a service availability check, where the actual services are known at design time. See [55] for an example, where a GUI viewer application is capable of displaying data form a variety of services. The viewer probes the network using WS-Discovery to check which services are on-line and to list available services for the user to choose from.

5.3 Implications on use cases

According to the ideal scenario, the concrete applications at the User Applications layer in the C3 Taxonomy should ultimately determine which services to develop at lower levels. And to develop services at a level one should start by defining epics at that level. Again, since the taxonomy is sparsely populated, we tentatively speculate what epics might be defined at lower levels.

The level of activity at the nether levels is a function of the level of maturity of service orientation. At advanced levels of maturity, one can envision end users composing and orchestrating systems by dragging and dropping services from a repository and organizing them in via a GUI, where after interoperability issues are automatically resolved behind the scenes. At modest levels of maturity, i.e., where we are at today, programmers have to manually set up communication between systems,

usually “SOA-fizing” systems on the fly. This often involves large design and programming efforts.

All this makes it clear that content in an information infrastructure is not static and that requirements at various levels will evolve over time. At present, there is a need for technical services that aid programmers in making things at all levels interoperable. In the distant future – when everything is loosely coupled and interoperable – software developers may be able to drag and drop technical services via a GUI to compose and orchestrate the necessary services to support user applications. End users may even be able to compose applications themselves from high-level services.

Our tentative higher-layer use cases from Section 4 are ignorant to the details or status of SOA. However, it is clear that they lay demands on the architecture in terms of (re-)composing applications and services readily and rapidly, in time for, and even during, operational activities. These use cases will enjoy diverse solutions according to technological maturity. We will now discuss implications of our use cases according to present maturity. Of course, one should still strive for solutions in more mature technology as it arrives.

Since SOA is not in a state where services can be composed on the fly, software developers must invest development efforts to realize these use cases. Among the Core Enterprise Services one must therefore find tools (software services) to aid developers in such efforts:

> Epic 13: **As** software developer **I can** develop User Applications **by using** SOA Platform Services **to** combine COI-specific Services readily and rapidly **in order to** reduce time to shippable code by $k\%$.

This is also applicable at the Users Applications layer, since users at this level cannot, at present, combine applications themselves:

> Epic 14: **As** software developer **I can** develop User Applications **by using** SOA Platform Services **to** combine User Applications readily and rapidly **in order to** reduce time to shippable code by $k\%$.

Indeed, while it may be conceptually tempting to regard services as the building blocks for developing higher-level services and user applications; and to regard user applications as situation-specific systems composed for the task at hand (and therefore not meant to function as building blocks themselves), this conceptually clean view is, at least at present, not realistic. Many of today’s systems are stove piped or involve thick (obese) clients with very few building blocks in common. At present, it is therefore necessary to view applications themselves as building blocks to make other applications, and the questions of loose coupling and interoperability apply to both services and applications.

Further, our tentative use cases suggest that M&S functionality must interoperate with other functionality at the User Applications layer and also that it must be possible to compose M&S functionality into M&S COI Applications. We here choose to use the impact goal part of the epic to point to the epics to which it contributes.

Epic 15: **As** software developer **I can** develop User Applications where M&S COI Applications interoperate with other User Applications **by using** SOA Platform Services **to** combine loosely coupled and interoperable M&S COI Services and M&S COI Applications with other loosely coupled and interoperable COI-Specific Services and User Applications **in order to** enable Epics 1, 2, 3, 4, 5, 6, 7, 10, 11, 12, 14.

Epic 16: **As** software developer **I can** develop M&S COI Applications where M&S COI Services interoperate **by using** SOA Platform Services **to** combine loosely coupled and interoperable M&S COI services **in order to** enable Epics 1, 3, 8, 9, 13.

Some of the use cases asked for applications (tentatively placed in the SMC Applications and Scenario Preparation Applications categories) which enable semi-IT technical personnel (operators) to compose and recompose applications readily and rapidly. At present-day maturity, it is possible to develop applications which fulfil this to a modest degree, by means of service-availability check and service import applications as mentioned in the previous section.

Epic 17: **As** software developer **I can** develop User Applications which invoke design-time specifiable COI-Specific Services and User Applications at runtime **by using** SOA Platform Services **to** combine COI-enabling Services **in order to** enable Epics 8, 9.

Epics 15, 16 and 17 illustrate the need to start epics at the Technical Services layers, as well as at the User-facing Capabilities layers.

The placement of the epics above in the C3 taxonomy is summarized in Figure 6.3. In order to be more specific about M&S technology, we now turn to M&S software architecture.

6 The High Level Architecture

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 Simulation Interoperability Standards Organization (SISO) coordinates many of the research and development efforts in these areas in close collaboration with NATO; specifically the NATO Modelling and Simulation Group, and other standardization organizations. As a result, several working solutions exist for M&S systems which enable loose coupling and interoperability. At present, the High Level Architecture (HLA) is adopted by a wide range of communities. It enables viable principles toward loose coupling and interoperability, and also refines principles developed earlier in other approaches. The HLA is a design, development, and runtime standard for distributed simulation software systems [91, 50]. The latest version, HLA Evolved IEEE 1516 – 2010, STANAG 4603, as well as the previous version IEEE 1516 – 2000, which is currently most in use, are defined in [38]. The present HLA standard has three parts :

1516-2010 Framework and Rules
1516.1-2010 Federate Interface Specification
1516.2-2010 Object Model Template (OMT) Specification

In addition, two older parts are still relevant:

1516.3-2003 Recommended Practice for HLA Federation Development and Execution Process (FEDEP)
1516.4-2007 Recommended Practice for Verification, Validation, and Accreditation of a Federation – An Overlay to the FEDEP

In HLA, the main simulation software modules that make up a simulation system are called *federates*. Federates may be combined to form a *federation*, coordinated by a runtime infrastructure (RTI); see Figure 6.1. Federates communicate with the RTI by means of APIs for Java or C++. In HLA Evolved, there is, in addition, a Web Service API which allows one to package federates as web services within the federation [69, 67, 70]. 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 coordinates these messages. Federates may also query the RTI on-the-spot for updates.

In HLA, the objects and interactions that are shared (i.e., whose attributes are published and subscribed) among federates in a federation are declared in a Federation Object Model (FOM), which is input to the RTI. Thus, all federates must relate to this data declaration during runtime, and also during design time. At run-time, the declarations in the FOM give rise to variables that constitute the shared state of the federation. The RTI thus administers the shared state of the federation. The RTI also administers time according to several time management schemes [29]. The federates themselves are ignorant with respect to the shared state; i.e., a federate is not aware of the states of other federates, except for the parts of the shared state that the federate subscribes to. For example, a federate may simulate an entity that fires a ground-to-air missile toward an entity at a specified location without needing to know which ground entities are in the simulation. Indeed, a federate need not know which other federates are present in the federation. Nevertheless, a federate who wishes to monitor if any of its entities will be hit by the missile can subscribe to the part of the shared state that reflects the relevant information.

A FOM is an object model based on inheritance according to the OMT. It is therefore possible to extend a FOM at modification time by declaring new subclasses of existing classes. New or modified federates may then relate to the extended FOM. In HLA Evolved, the concept of modular FOM enables one to extend and compose a FOM by using FOM modules at runtime [91]. Each FOM module describes a certain aspect of the information exchange [66]. While existing federates cannot relate to new FOM modules, new federates that relate to the extended FOM can dynamically join the federation.

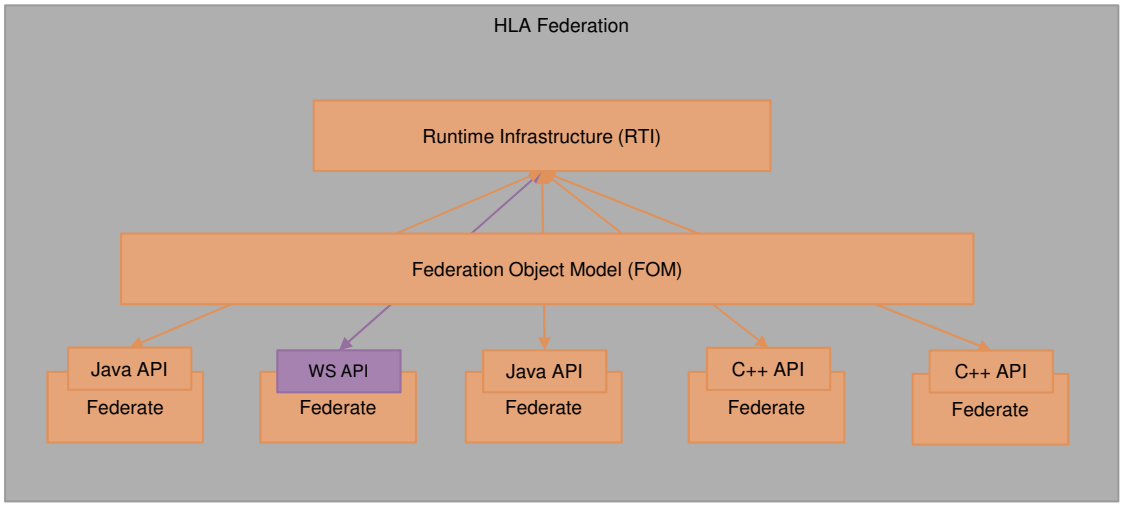

Figure 6.1 High-Level Architecture (HLA).

Several reference 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) [108] and its guide, the Guidance, Rationale, & Interoperability Modalities (GRIM) (Version 1.0) [107] SISO-STD-001.1-1999.[4] Work is currently under way on RPR-FOM and GRIM version 3.0. 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 1278 [40]. The DIS infrastructure 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 [118]. Another reference FOM is the NATO Education and Training Network (NETN) FOM [80].

There are also standard methods for communicating with a simulation system (i.e. a federation in HLA). The Coalition Battle Management Language (C-BML) is a formalized language under standardization to provide a common language for expressing a commander's intent, across C2 systems, simulation systems and also autonomous systems [79]. The Military Scenario Definition Language (MSDL) SISO-STD-007-2008 [109] is a standard for describing scenarios; e.g., for initializing simulation systems.

In terms of our remarks on architectures and methods for architectures in Section 2, HLA and associated methods live up to the requirements for both architecture development and system development, see Figure 6.2. The HLA standard as an architecture framework can be instantiated to concrete HLA architectures using FEDEP and DSEEP mentioned in Section 3.1. The FEDEP and DSEEP are federation lifecycle processes and cater for architectural work as well as systems development work. For developing an architecture, the OMT describes how to declare the various object models in HLA federations: In addition to a FOM for the federation, each individual federate has

[4] RPR is pronounced "reaper" to complete the allusion to Death.

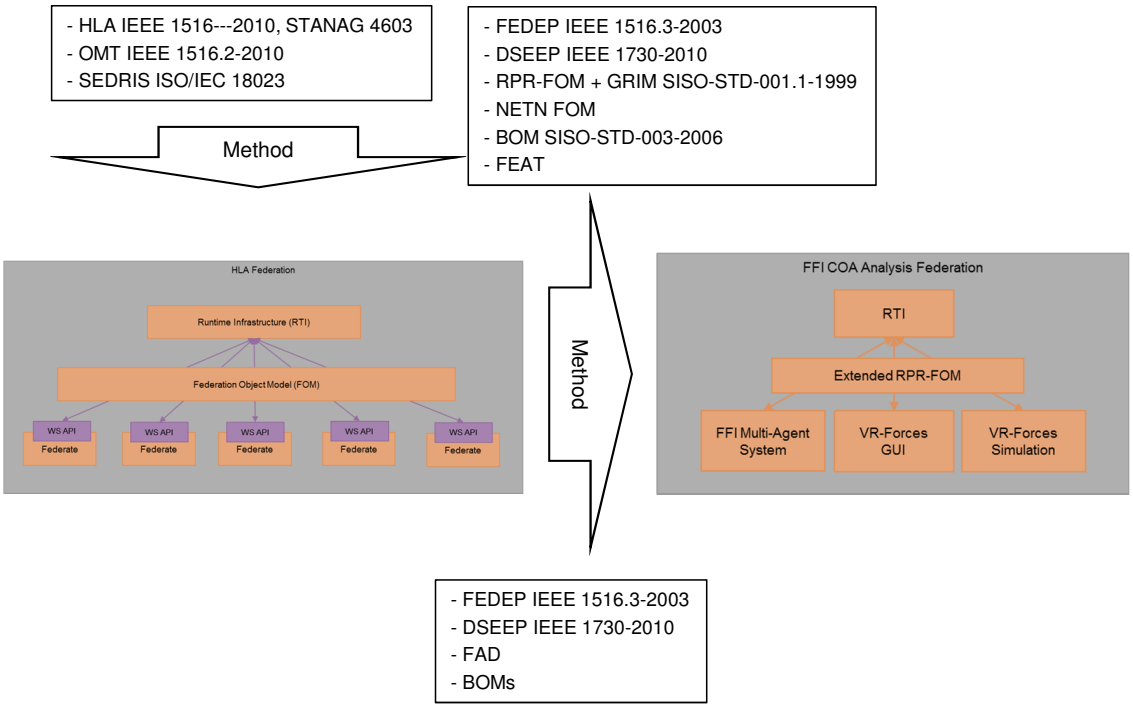

Figure 6.2 Architecture framework, architecture development method, and federation development method for HLA with federation example from FFI.

a Simulation Object Model (SOM) associated to it which declares the federate's state space, and the RTI is associated with a Management Object Model (MOM). The Base Object Model (BOM) standard SISO-STD-003-2006 provides a method for defining abstract object models of entities and interactions in a simulation system [106], which is useful in developing both the architecture and the federation. The Federation Agreement Template (FEAT) describes how to write a Federation Agreement Document (FAD). The former is part of the method for developing an architecture, the latter is part of the method for developing a federation. Other standards are also relevant for constructing a concrete HLA architecture, such as the Synthetic Environment Data Representation and Interchange Standard (SEDRIS) ISO/IEC 18023 [101]. Reference FOMs constitute, together with FEDEP and DSEEP, a method for developing a concrete HLA federation from a concrete architecture; exemplified in Figure 6.2 by a federation developed at FFI.

We have tentatively placed HLA functionality (APIs), FOMs, the RTI and the RPR-FOM in categories in the current state of the C3 Taxonomy; see Figure 6.3. We place HLA federates in a tentative HLA Federates category. This raises the question of how service enabled HLA federates are; which we discuss in Section 8. We place HLA federations in a HLA Federations category under User Applications; see Figure 4.1. The RTI is placed in Message-Oriented Middleware Services, but should eventually be relocated into a more specific category. Note that our placement of concrete artefacts does not imply that we think that they are services or service oriented. Rather, our placement reflects a desire that the artefacts should become services or service oriented in a future systems portfolio.

This focus on the future implies that we do not place DIS functionality in the taxonomy, because it is essential for interoperability that future development relates to standards; in this case HLA.

We place software, including machine-readable documents such as the RPR-FOM in the taxonomy. In our proposal, standards such as HLA are not first-class citizens of the taxonomy unless they actually play the role of use cases (a use case being a requirements specification and production element). On the other hand, standards are relevant as necessary references and complementary requirements specifications attached to the software. The EM wiki's structure gives opportunity to record such "second-class citizens" in the C3 Taxonomy.

In the following sections, we will discuss further implications of Epics 15 and 16 (Section 5.3). The reason for focusing on these two is that they represent two essential technical challenges: Loosely coupled and interoperable software systems at the User Applications level, and loosely coupled and interoperable M&S modules and components at the Community of Interest (COI) Services level.

7 Loosely coupled and interoperable software systems

Somewhat independently of concerted SOA initiatives, research has strived for interoperability and loose coupling for some time within both M&S and defence operational software systems. Many command and control systems support XML-based web technologies for communicating their data [116, 119]. Moreover, the Multilateral Interoperability Program (MIP) has developed the Joint Consultation, Command and Control Information Exchange Data Model (JC3IEDM), which sets out to specify the minimum set of data that needs to be exchanged in coalition or multinational operations. Software systems that operate and communicate in terms of JC3IEDM data structures therefore gain a certain level of interoperability since they, at least, operate in terms of the same data structure. The JC3IEDM is a NATO standard (STANAG 5525) and is an entity relationship model. The next generation of the JC3IEDM – the MIP Information Model (MIM) – is currently under development and is UML based.

Further, C-BML mentioned earlier is designed for exchanging battle order and report data based on JC3IEDM. Present C-BML implementations are XML-based, and a Web Service-based middleware repository offers post/retrieve services on C-BML messages. In addition, MSDL for describing scenarios is also XML based and uses parts of the JC3IEDM name space. Since there are no good reasons for having borders between initialization data and exchange data [116], there are initiatives toward aligning MSDL and C-BML; e.g., the NATO Modelling and Simulation Group Technical Activities MSG-079, MSG-085 and SISO activities.

Figure 7.1 gives an overview of a demonstration held at the Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC) 2011 conference [94], where inter-system collaboration was achieved using standards such as MSDL, C-BML, DIS and JC3IEDM. A Scripted BML (SBML) server coordinated both the joint MSDL initialization as well as the exchange of C-BML battle orders and reports. Conventional systems were service enabled using gateways (wrappers).

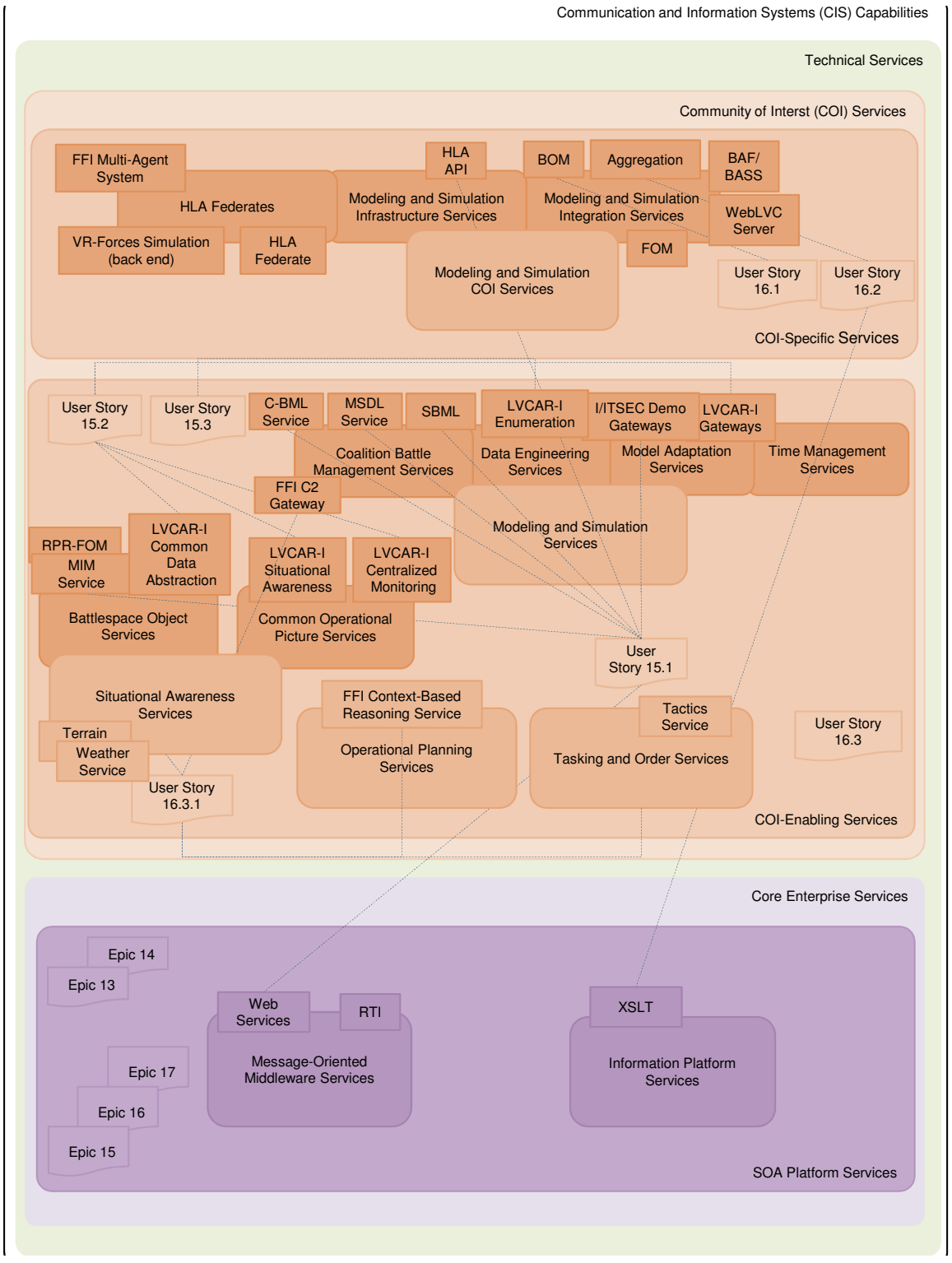

Figure 6.3 Use cases and concrete services tentatively placed in C3 taxonomy categories–Technical Services part.

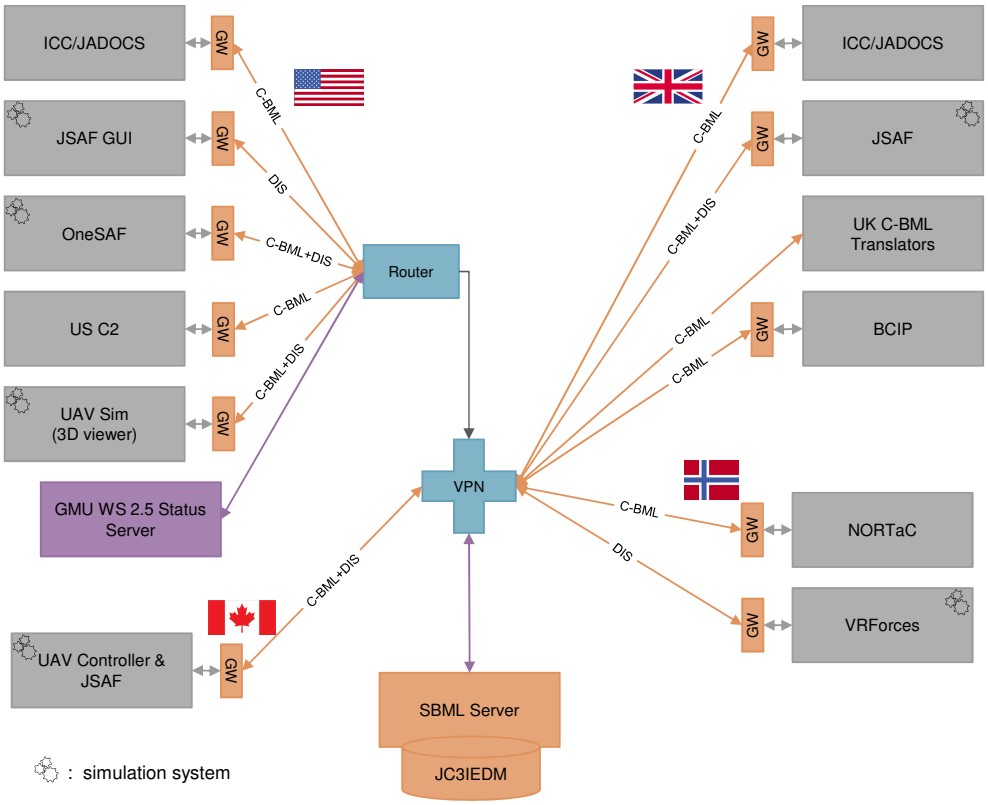

Figure 7.1 *Demonstration of inter-system collaboration using standards between C2 systems and simulation systems [94]. Systems are service enabled using gateways (yellow "GW").*

SOA does not hinge on the use of Web Service technology or other technologies currently associated with SOA. The use of other standards for interchanging data, such as C-BML or NATO Standardization Agreements (STANAGs) is in line with being a SOA. The useful aspect of SOA thinking in this context is that the software systems involved are loosely coupled and interoperable in the sense that they can be used meaningfully in various constellations and provide functionality to various consumers. Thus, if the systems involved in Figure 7.1 can be readily and rapidly re-combined, perhaps with other systems, and give meaningful functionality, then this is an aspect of service orientation which meets the requirements of Epic 15.

The I/ITSEC demonstration uncovers more detailed requirements at the technical level, related to data model and communication standards (JC3IEDM, MSDL, C-BML), gateways, and coordination functionality (SBML). We can express these in a use case toward the present state of the C3 taxonomy. We are now at a level of detail that includes concrete services and can formulate a user story under Epic 15. Note that we replace DIS by HLA in this use case in line with our remark on the future C3 systems portfolio above. Figure 6.3 summarizes the placement of artefacts in the following discussion.

User Story 15.1: **As** software developer **I can** increase loose coupling and interoperability of M&S COI Applications and other User Applications **by using**

COI-Enabling Services.
M&S Services.
Model Adaptation Services.I/ITSEC Demo gateways **and**
Coalition Battle Management Services.C-BML,MSDL,SBML **and**
Situational Awareness Services.Battlespace Object Services.JC3IEDM **and**
COI-Specific Services.
M&S COI Services.M&S Infrastructure Services.HLA **and**
Core Enterprise Services.
SOA Platform Services.Message-oriented Middleware Services.Web Services
to wrap and service-enable M&S COI Applications and other User Applications and provide common views of data and common modes of communication
in order to enable Epic 15.

User story 15.1 is an example of an elaboration and refinement of an epic which penetrates layers in the taxonomy as indicated in Figure 3.4.

Another demonstration was conducted as part of the LVC Architecture Roadmap Implementation (LVCAR-I) [5, 19, 4]. The demo – designated a SOA pilot – set out to develop a prototype interoperability layer to connect the Army Joint Land Component Constructive Training Capability (JLCCTC) Multi-Resolution Federation (MRF) and the JLCCTC Entity Resolution Federation (ERF) training system. Only 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 High Level Architecture (HLA) federations. Their stated value to the demo is that they have widely different object models (FOMs) and time management, and therefore challenge interoperability [5]: ERF uses an entity-centric DIS-based data model, while MRF is an aggregate unit-centric data model based on the Aggregate Level Simulation Protocol (ALSP) [123]. The MRF federation is time managed where all simulations coordinate the advancement of simulation time. The ERF federation 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 in a Universal Core (UCore) XML format (yet another standard for communicating battle information). In the demo, 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 all entities [5].

User Story 15.2: **As** software developer **I can** enable coordination of diverse M&S COI Applications **by using**
COI-Enabling Services.

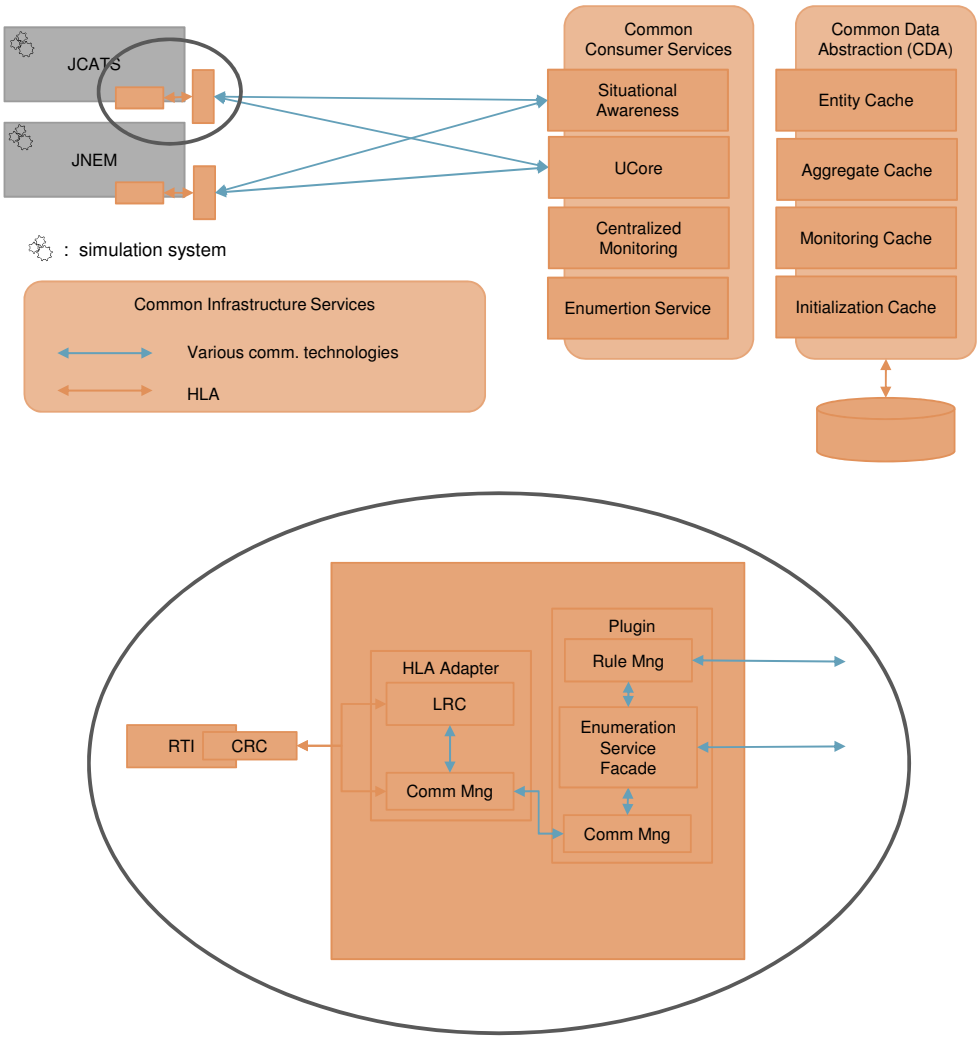

Figure 7.2 *Demonstration of inter-system collaboration involving two HLA federations, common consumer services and common data model [5]. Common infrastructure services are used for communication. Gateways (standing rectangles) enable the HLA federations to participate in the overall system; see more detailed view of gateway architecture in lower part of figure. The RTI is decomposed into CRC = Central Runtime Component, LRC = Local Runtime Component.*

Situational Awareness Services.
Common Operational Picture Services.
LVCAR-I Situational Awareness Service **and**
LVCAR-I Centralized Monitoring Service **and**
Battlespace Object Services.
LVCAR-I Common Data Abstraction Services
to maintain the minimal set of data required to share object state across all entities in the M&S COI Applications
in order to enable Epic 15.

The development of a so-called enumerations translation service was also under way in the demo. Enumerations pertain to entity identity and representation, and problems arise when different systems use different enumeration schemes. The service was going to provide a set of common methods for data producers and consumers to determine how to translate from native representations to common data abstraction representations. In this manner common functionality for gateways may be factored out and offered as services.

User Story 15.3: **As** software developer **I can** build Model Adaptation Services.gateways **by using** COI-Enabling Services.M&S Services.Data Engineering Services.LVCAR-I Enumeration **to** determine how to translate from native representations to common data abstraction representations **in order to** enable Epic 15.

The LVCAR-I demo illustrates the use of inter-M&S system coordination management offered as services. The main ideas concern factoring out common functionality; see [22, 21] for more details.

8 Loosely coupled and interoperable M&S components

In SOA terminology, the RTI in HLA federations functions as a state management deferral mechanism, since shared state management has been delegated away from the federates and to the RTI [26]. This gives the federates a high level of autonomy with respect to other federates, but a heavy dependency of federates on the RTI. The RTI could be seen as a utility service; i.e., middleware that intentionally violates the SOA ideal of statelessness in order that other services may enjoy their level of statelessness and state processing deferral [26]. However, as middleware goes, the RTI is relatively active, since it is capable of administering complicated time management schemes and since HLA promotes a “chatty” mode of data interchange.

In terms of SOA, what type of loose binding do federates within a federation enjoy? According to [32], the reliance on the FOM precludes a loose binding in the SOA sense. There is loose binding in the sense that federates may join or leave a federation during runtime, and there is loose binding in the sense that federates are not aware of other federates in the federation; all they need to do is to publish and subscribe to information. However, the data types (in the traditional sense of object

orientation) of that information has to be known at design time, and this is a tight coupling that contrasts with SOA's loosely coupled service contracts. So, although the federates do not manage the shared state, they nonetheless have to relate to relevant portions of the shared state variables at all times. Further, a federate cannot simply join a federation operating on a different FOM.

The idea of a federate as a service is therefore not straight-forward, and in light of the above discussion, it does not help that one can use Web Services technology to join a federate to a federation. A FOM represents the context for a specific orchestration (a federation). For a federate to function as a service in the SOA sense, it must be able to digest and adapt to various FOMs [32]. Today, FOMs are written in XML-style languages and are therefore in line with web technologies. However, producing federates that are capable of handling an arbitrary FOM on the fly seems unrealistic; particularly in light of the complexity of many FOMs in use.

Instead, one may consider the following approach [68, 110, 15, 16]: In HLA, one has several types of object model. Apart from FOMs, which are strictly necessary for federations to function, HLA also provides templates for conceptual models. The Base Object Model (BOM) template mentioned earlier describes how to model objects and interactions between them. In other words, BOMs are a way to provide interface information – in XML-based language. What is more, 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 interface specifications, 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 orchestration time using loosely coupled simulation components.

Figure 8.1 illustrates two ideas along this line of thought. On the lefthand side of the figure is an example from [68, 32] where the RPR-FOM is used as the basis for defining BOMs. The idea is that a BOM models states and behavior (interaction patterns) on an abstract level (independently of a specific FOM) between a defined set of entities, and thus functions as abstract service specifications for components as services. BOMs can be automatically translated into modular FOMs, so that with service composition (into federates) one also generates the FOM *du jour* on the fly. On the right-hand side of the figure, is an example from [110], where entity level BOMs are defined and used in entity aggregation and entity de-aggregation. 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 *du jour*

User Story 16.1: **As** software developer **I can** develop M&S components as M&S COI Services **by using** COI-Specific Services . M&S COI Services . M&S Integration Services . BOM **in order to** enable Epic 16.

User Story 16.2: **As** software developer **I can** compose HLA federations as M&S COI Applications **by using** COI-Specific Services . M&S COI Services . M&S Integration Services . Aggregation

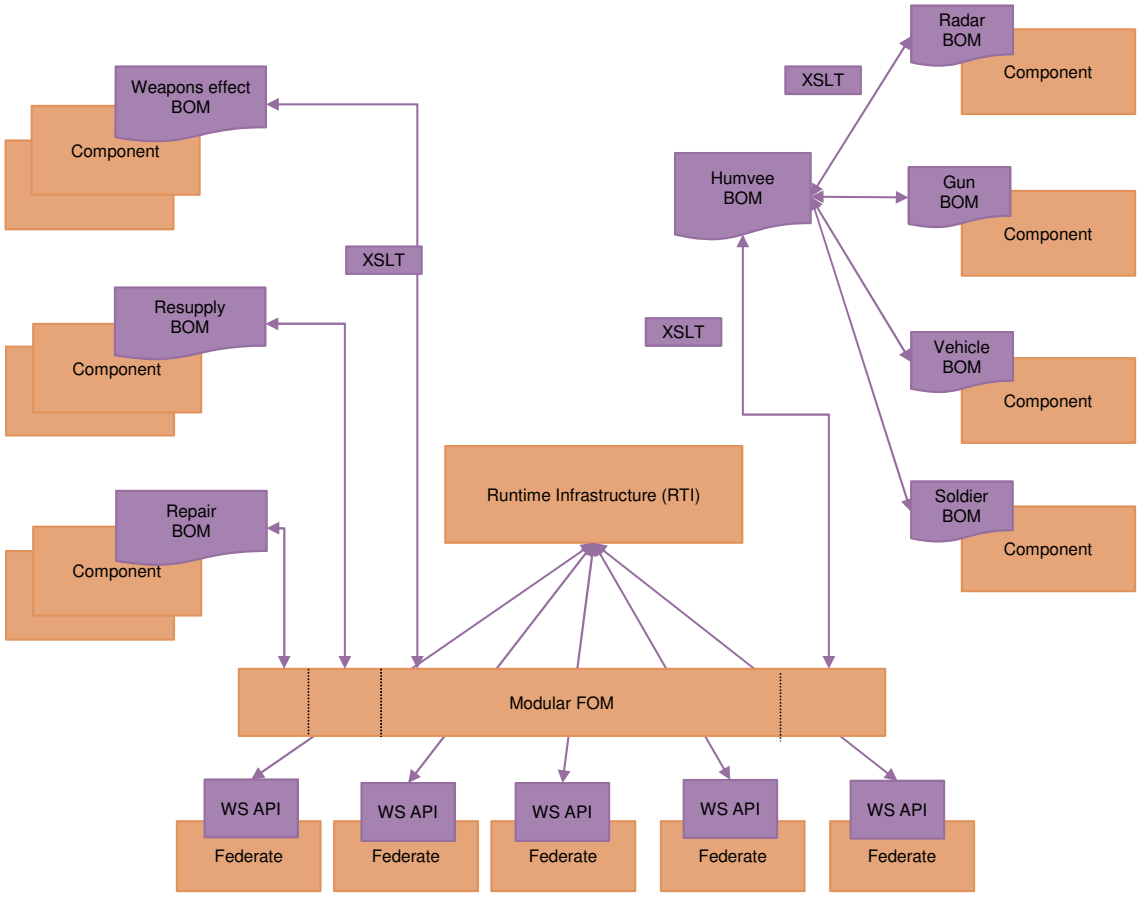

Figure 8.1 *BOMs as service contracts for simulation components. The RPR-FOM used as the basis for defining BOMs (left-hand side of figure) [68, 32]. Entity level BOMs defined and used in entity aggregation and entity de-aggregation (right-hand side of figure) [110].*

Service **and** Core Enterprise Services . SOA Platform Services . Information Platform Services . XSLT **in order to** enable Epic 16.

As appealing as this sounds, composing components at this level of granularity meets the same challenges as component-based software engineering in general. Years of experience has proven it hard to build software components which are reusable over a range of applications. This has both technological and organizational reasons. Harder still, is perhaps refactoring exercises, where large pieces of monolithic software are attempted modularized into components by factoring out common reusable functionality (e.g., the US JLVC-2020 project).

A challenge regardless of size is the composability of components based on their specifications and the verification and validation of the composition. For modelling and simulation, specific issues apply; see [120]. At lower levels of granularity components act increasingly detailed on the same state space which complicates things further.

This depends on the extent to which the components interact with each other. In the case of the

aggregation/de-aggregation example shown in Figure 8.1 the entities may simply be aggregated so as to represent one entity without components interacting; e.g., to save computation in the simulation while moving forward. When the aggregate has reached its goal, say, the entities may be deaggregated. SimVentions Inc. has developed their BOM Aggregation Framework (BAF) with a BOM Aggregation Support Server (BASS) which provides this type of aggregation/deaggregation functionality as run-time services into a HLA federation [110, 32]. See Figure 6.3 for a tentative placement of BAF and BASS in the C3 Taxonomy.

On the other hand, if components are to interact, 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, [58]. To amend these issues, semantic enhancements of BOMs have been developed [65, 64, 71], and methods for verifying the functionality of compositions have also been researched [58, 59].

A technological initiative which certainly presents a relevant use case, if not a direct step towards intra-system loose coupling and interoperability, is MÄK Labs' launch of its WebLVC platform. WebLVC provides a framework that enables one to join a hosted federation via web applications. The framework supports web applications written in JavaScript (simulating a fighter aircraft, say) which may then communicate with a hosted WebLVC Server using the WebSocket protocol. The Server then integrates the web applications with a hosted federation. The hosted federation is running on HLA using the RPR-FOM, or DIS. Federates are therefore still bound by these object models, even if they may communicate with the federation over the internet. See Figure 6.3 for a tentative placement of the WebLVC Server in the C3 Taxonomy.

At the moment, there might be a case for focusing on service-orientation with respect to a given object model (FOM), rather than service-orientation in general across object models. This is especially relevant since there exist several reference FOMs, some of which are even standardized. We will illustrate by referring to one more pilot study. In 2012, FFI demonstrated the feasibility of a *multi-agent framework*, where a multi-agent system receives C-BML orders from a C2 system and executes the orders by generating appropriate entity command movements in a simulation engine. A battle order originating from a C2 system is a general instruction which has to be executed according to knowledge about terrain, enemy actions, weather etc. The multi-agent system uses context-based reasoning [31] to compute appropriate entity commands according to contextual data available in the simulation; see Figure 8.2. The multi-agent system operates as a federate in an HLA federation running on a RPR-FOM extended to cater for the interactions with the multi-agent system; see [61, 6] for details on certain aspects of the multi-agent system.

One can take the following stance on loose coupling and interoperability issues: Rather than striving for a multi-agent system that can readily and rapidly be used as a service in an arbitrary FOM, one might concentrate on service-enabling the multi-agent system in the sense of offering varying battle order-execution strategies; e.g., in terms of reasoning algorithms (context-based reasoning, decision trees, neural nets, etc.), in terms of environment data to be considered (terrain, weather, season, mode of aggressiveness, etc.) and in terms of operational tactics. To realize our high-level epics it is

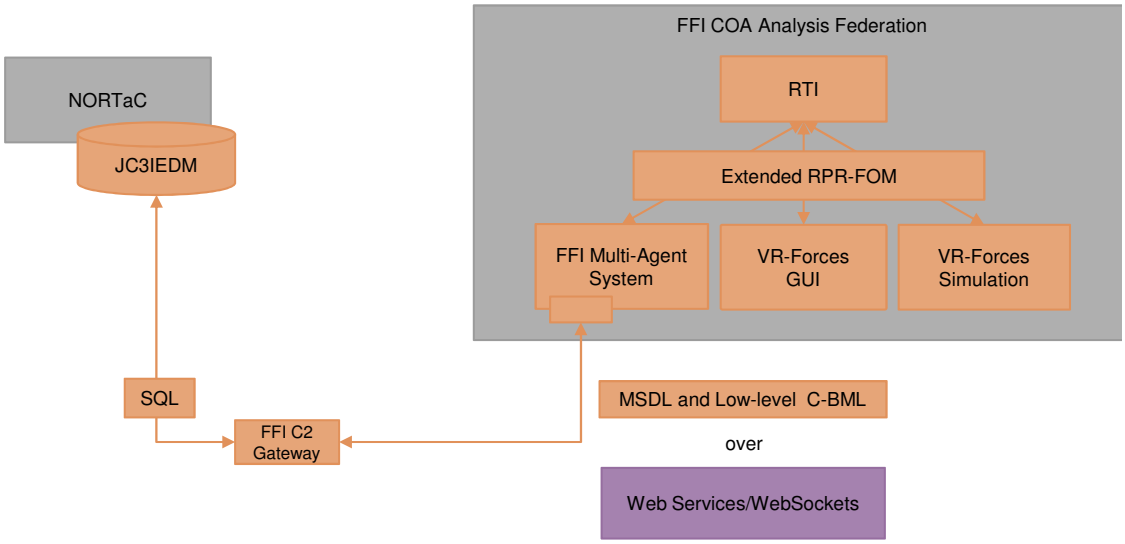

Figure 8.2 *Course of Action (COA) Analysis Pilot at Norwegian Defence Research Establishment (FFI). In this instance of the pilot, battle orders from a C2 system (NORTaC) are loaded into a JC3IEDM database, which are then processed by the multi-agent system which, in turn, sends more detailed battle orders to a VR-Forces back end. VR-Forces is a simulation framework kit developed by VT MÄK.*

relevant to be able to vary these parameters readily and rapidly during training, mission preparation, missions and retrospectives.

> User Story 16.3.1: **As** software developer **I can** develop a FOM-specific COI-Enabling Services . M&S Services . HLA Federates . Multi-Agent System **by using** COI-Enabling Services . Operational Planning Services **and** COI-Enabling Services . Tasking and Order Services **and** COI-Enabling Services . Situational Awareness Services **to** implement various battle order-execution strategies in terms of reasoning algorithms, defence tactics and environment data processing **in order to** enable Epic 16.

We can generalize this user story to a user story that expresses the principle of developing FOM-specific federates as services:

> User Story 16.3: **As** software developer **I can** develop FOM-specific M&S COI Services . HLA Federates **by using** COI-Enabling Services **in order to** enable Epic 16.

Note that the numbering of the previous user story (16.3.1) reflects that it is an elaboration and part of the refinement of User Story 16.3.

9 Discussion

From deliberating around our tentative use cases, it is evident that services and applications at various levels of granularity are relevant. There are cases for which larger application-size pieces of software must inter-operate and cases for which component-size pieces of software must inter-operate. The notion of service orientation and the architecture of the C3 taxonomy give structures which cater for these needs. However, as the pieces of inter-operating software get smaller, it is probable that the number of technical difficulties in achieving loose coupling and interoperability increases. Where to draw the line between what should be services and what should be traditional components, modules or merely pieces of software, is a question of both technical feasibility and cost-benefit. Further analyses and pilot studies are needed to arrive at more concrete conclusions.

The extent to which service orientation should penetrate HLA federations is an example in point. It is not clear at present to which extent it is technically feasible or cost-beneficial to develop federates, simulation components or simulation entities as services. It does, on the face of things, seem easier to defend developing entire federations as services. Within federations, literature seems to suggest that common infrastructure functionality could be factored out into services. The FOM data structure has been mentioned as an obstacle to true service orientation within federations. However, it is clear that, relative to a given FOM, federates enjoy high degrees of both interoperability and loose coupling, at least compared to solutions in other software domains. The existence of standard, albeit domain-specific, FOMs such as the RPR-FOM, pushes the issue of what more is needed in terms of interoperability and loose coupling, since such FOMs serve a multitude of federations. However, there are multiple instantiations and extensions of the RPR-FOM and also many other "standard" FOMs. A more detailed analysis on the need for interoperability and loose coupling between citizens of federations operating on such diverse FOMs is called for.

We have seen that the RTI in HLA can be seen as a state deferral mechanism in terms of SOA, and HLA federations have many of the interoperability and loose coupling characteristics sought after in SOA. As in many other areas of software development, it is easy to engage in dogmatic discussions on what is SOA and what is not SOA. We hold that such discussions are valuable to the extent that the dogma has been proven valuable as a whole in its own right; something which dogmas in software development are hardly ever. Therefore, we think that it is important to focus on the pragmatic aspects of the elements of SOA, and the extent to which they serve the needs for interoperability as analysed, in the outset, from operational requirements, e.g., in use cases such as those exemplified above.

It is important to gain headway in defining an architecture framework for M&S in a SOA information infrastructure. A good way to start would be to populate the C3 Taxonomy in a systematic manner, because it enables one to organize state-of-knowledge on the operational needs and the derived technical needs for service orientation. We hold that it is important to employ state-of-the art requirements elicitation and analysis on both operational and technical levels for this to succeed. This involves iterative and incremental refinement and detailing, along with pilots and demonstrators for empirical guidance through this process. Our tentative use cases should then be replaced by

properly elicited use cases, together with complete detailing and refinement into system specifications and concrete operations, applications and services.

10 Conclusion

We conclude by responding to our introductory research questions to the extent possible based on our discussion so far.

1. What are the requirements of users of a service-oriented defence information infrastructure that incorporates M&S software systems?

   Answering this question in full involves requirements elicitation and analysis on the scale of a very large IT developments project. We have tentatively suggested requirements in the form of well-structured use cases. We have suggested use cases at various levels of the NNEC C3 Taxonomy. The more detailed use cases – stemming from pilot studies – suggest concrete applications and services.

2. How can these requirements be used to build the information infrastructure?

   There are several ways to do this. We suggested a joint method based on requirements elicitation and analysis from best-practice software engineering. The method gives guidelines to populate the C3 Taxonomy in a systematic and traceable manner, based on incremental development at all layers of the taxonomy. We foresee that the content and the manner in which the taxonomy will be populated will evolve according to how service oriented the taxonomy is at any one point in time. At early stages, development of the taxonomy's content will follow traditional software development practice which involves software development (from scratch) through several layers of the taxonomy. At advanced stages where a lot of functionality is service oriented, loosely coupled and interoperable, software development can stay within one layer. Other layers are accessed through home-layer composition and orchestration applications or services. Architecture development is an integral process to this, rather than an isolated pre-process.

3. To what extent is it desirable and then possible to package application-size modelling and simulation software as loosely coupled and interoperable units in the context of a service-oriented defence information infrastructure?

   According to our use cases and according to the pilots we have referred to, this is certainly desirable and even necessary, since operational personnel must be able to access M&S functionality through their regular systems used in practice. Further, the centrality of M&S systems in operational activities implies that some M&S systems should evolve to become front-end members of the C3 systems in daily use by operational personnel. Packaging is possible, as

several pilot studies have demonstrated. Moreover, it is possible to specify technical use cases which give good indications for concrete functionality necessary for realizing this packaging.

4. To what extent is it desirable and then possible to use service-orientation internally in simulation software; hereunder,
   (a) can SOA be used as a simulation management framework beyond HLA?

   Certain elements of SOA are useful to enhance HLA; for example it is useful and possible to factor out common gateway and coordination functionality for integrating federates running under differing object models or time management schemes. It is also useful and possible to provide entity aggregation and de-aggregation as a service beyond the capabilities of HLA. As an aside, several features of HLA are definable in the SOA terminology; for example, the RTI (the coordinating middelware in HLA) can be seen as state-deferral service freeing the simulation modules (federates) from keeping track of the total state of the simulation federation.

   (b) can component/module-size M&S software be packaged as loosely coupled and interoperable units?

   In the context of a HLA simulation system's object model, M&S modules already are loosely coupled and interoperable. However, it is non-trivial to reuse these modules with a different object model (FOM); i.e.; in a substantially different simulation context. Solutions to the problem have been solved theoretically, but implementing these ideas requires advances in e.g., semantic technologies in order to conduct necessary verification and validation. At present, a more fruitful approach might be to develop services with respect to a specific (reference) object model (FOM).

In order to develop a defence information infrastructure within the context of a systems portfolio according to the C3 taxonomy, one should employ systematic methods for populating the taxonomy. This involves systematic requirements elicitation, analysis, detailing and refinement down to concrete operations, applications and services. We have outlined one such method which combines agile requirements handling with the structure of the C3 Taxonomy. Using the taxonomy's vertical structure (operational context to technical context) and the horizontal structure (detailing and refinement of categories), it is possible to populate the taxonomy with requirements in addition to concrete operations, applications and services. This should structure the development of the taxonomy and also give traceability of the process. The method should help the human process of "getting things started", since one now has the opportunity to populate the taxonomy with low-detailed preliminary insights on what eventually should be developed in terms of concrete operations, applications and services. Integrating M&S in a defence information infrastructure must be part of this general process of developing the taxonomy. The challenge remains, however, that M&S development cannot wait till the NII or INI, or the C3 systems portfolio are in place, and therefore, M&S systems must

somehow be developed to be “INI/NII/C3 systems portfolio ready”.

The way forward should include the following topics:

- A more detailed investigation into the architectural implications of HLA’s Object Model Template (OMT) and how those implications relate to service orientation.
- A more detailed investigation into the development processes of Federation Development and Execution Process (FEDEP) and Distributed Simulation Engineering and Execution Process (DSEEP) in the context of agile development, the C3 Taxonomy and enterprise and systems architecture frameworks such as NAF and TOGAF.
- A more complete classification of M&S functionality as services.

A first avenue for investigating the three points above, is to conduct a narrow pilot study using the methods suggested in this report: Start by defining a small capability package (meaningful at the operational level) for a piece of functionality which involves M&S functionality. Then, elaborate and refine incrementally and iteratively toward a working system for the capability package. Along the way, use NAF and TOGAF, and spawn other elaborate-refine processes as needed. Focus on service orientation.

References

[1] P. L. Ackerman and M. E. Beier. Methods for Studying the Structure of Expertise: Psychometric Approaches. In K. A. Ericsson, N. Charness, P. J. Feltovich, and R. R. Hoffman, editors, *The Cambridge Handbook of Expertise and Expert Performance*, chapter 9, pages 147–166. Cambridge Univ. Press, 2006.

[2] D. S. Alberts and R. E. Hayes. *Campaigns of Experimentation: Pathways to Innovation and Transformation*. Information Age Transformation Series.

[3] D. S. Alberts and R. E. Hayes. *Code of Best Practice for Experimentation*. CCRP Publication Series.

[4] G. W. Allen, R. Lutz, and R. Richbourg. Live, Virtual, Constructive, Architecture Roadmap Implementation and Net-Centric Environment Implications. *ITEA Journal*, 31(3), 2010.

[5] G. W. Allen and L. Schroeder. 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, 2011.

[6] A. Alstad and O. M. Mevassvik. Low-level BML. In *Proc. 2013 Spring Simulation Interoperability Workshop (SIW)*. Simulation Interoperability Standards Organization (SISO), 2013.

[7] C. Argyris. *Knowledge for Action*. Jossey-Bass Publishers, 1993.

[8] C. Argyris and D. A. Schön. *Organizational Learning II. Theory, Method, and Practice*. Addison-Wesley Publishing Company, Inc., 1996.

[9] D. K. Barry. *Web Services and Service-Oriented Architecture: The Savvy Manager's Guide*. Morgan Kaufmann, 2003.

[10] S. Bonner. A Model of the Effects of Audit Task Complexity. *Accounting Organization and Society*, 19:213–234, 1994.

[11] C4ISR Technology & Human Factors (THF) Branch, Allied Command Transformation (ACT). The C3 Classification Taxonomy. Technical report, 2012. Document generated from the ACT Enterprise Mapping Wiki on November 2012.

[12] D. J. Campbell. Task Complexity: A Review and Analysis. *Academy of Management Review*, 13(1):40–52, 1988.

[13] J. P. Campbell. Modeling the Performance Prediction Problem in Industrial and Organizational Psychology. In M. D. Dunnette and L. M. Hough, editors, *Handbook of Industrial and Organizational Psychology*, volume 1, pages 687–732. Consulting Psychologists Press, Inc., second edition, 1990.

[14] J. P. Campbell, R. A. McCloy, S. H. Oppler, and C. E. Sager. A Theory of Performance. In N. Scmitt and W. C. Borman, editors, *Personnel Selection in Organizations*, pages 35–70. Josey-Bass, 1993.

[15] T. Chase and P. Gustavson. 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 (SISO), 2005.

[16] T. Chase, P. Gustavson, and L. Root. From FOMs to BOMs and Back Again. In *Proc. 2006 Spring Simulation Interoperability Workshop (SIW)*. Simulation Interoperability Standards Organization (SISO), 2006.

[17] A. Cockburn. *Agile Software Development*. Addison-Wesley, 2002.

[18] M. Cohn and R. Martin. *Agile Estimating and Planning*. Prentice Hall, 2005.

[19] J. E. Coolahan and G. W. Allen. LVC Architecture Roadmap Implementation—Results of the First Two Years. In *Proc. 2012 Spring Simulation Interoperability Workshop (SIW)*. Simulation Interoperability Standards Organization (SISO), 2012.

[20] M. Denne and J. Cleland-Huang. *Software by Numbers: Low-Risk, High-Return Development*. Prentice Hall, 2003.

[21] D. L. Drake, I. X. Martins, R. A. Roca, and F. Carr. Live-Virtual-Constructive Service-Oriented Architecture. Service-Oriented Architecture Application to Live-Virtual-Constructive Simulation: Approach, Benefits, and Barriers. Technical Report NSAD-R-2011-025, National Security Analysis Department, The Johns Hopkins University, Applied Physics Laboratory, 2011.

[22] D. L. Drake and K. L. Morse. Use of SOA for Distributed Simulation: A Way Forward. In *Proc. 2012 Spring Simulation Interoperability Workshop (SIW)*. Simulation Interoperability Standards Organization (SISO), 2012.

[23] M. G. Edgren. Cloud-Enabled Modular Services: A Framework for Cost-Effective Collaboration. In *Proc. Conf. NATO M&S Group (MSG-094) Transforming Defence through Modelling and Simulation—Opportunities and Challenges*, 2012.

[24] H. Erdogmus. Let's Scale Agile Up. *Agile Times*, 2(1):6–7, April 2003.

[25] K. A. Ericsson. The Influence of Experience and Deliberate Practice on the Development of Superior Expert Performance. In K. A. Ericsson, N. Charness, P. J. Feltovich, and R. R. Hoffman, editors, *The Cambridge Handbook of Expertise and Expert Performance*, chapter 38, pages 683–703. Cambridge Univ. Press, 2006.

[26] T. Erl. *SOA principles of Service Design*. Prentice Hall, 2007.

[27] I. Fette and A. Melnikov. The WebSocket Protocol—Request for Comments: 6455. On Internet Engineering Task Force (IETF) pages, http://tools.ietf.org/pdf/rfc6455.pdf, 2011. Accessed September 2012.

[28] R. T. Fielding and R. N. Taylor. Principled Design of the Modern Web Architecture. *ACM Transactions on Internet Technology*, 2(2), 2002.

[29] R. M. Fujimoto. *Parallel and Distributed Simulation Systems*. Wiley-Interscience, 2000.

[30] G. Gigerenzer and P. M. Todd, editors. *Simple Heuristics that Make Us Smart*. Oxford University Press, 1999.

[31] A. J. Gonzalez, B. S. Stensrud, and G. Barret. Formalizing Context-Based Reasoning: A Modeling Paradigm for Representing Tactical Human Behavior. *Int'l J. Intelligent Systems*, 23:822–847, 2008.

[32] P. Gustavson, T. Chase, L. Root, and K. Crosson. Moving Towards a Service-Oriented Architecture (SOA) for Distributed Component Simulation Environments. In *Proc. 2005 Spring Simulation Interoperability Workshop (SIW)*. Simulation Interoperability Standards Organization (SISO), 2005.

[33] J. V. Guttag. *The Specification and Application to Programming of Abstract Data Types*. PhD thesis, Department of Computer Science, University of Toronto, 1975. Technical Report CSRG-59.

[34] J. E. Hannay. Abstraction Barrier-Observing Relational Parametricity. In M. Hofmann, editor, *Typed Lambda Calculi and Applications. Proceedings of the 6th International Conference, TLCA, Valencia (Spain)*, volume 2701 of *Lecture Notes in Computer Science*, pages 135–152. Springer Verlag, 2003.

[35] J. E. Hannay. Personality, Intelligence, and Expertise: The Impact on Software Development. In A. Oram and G. Wilson, editors, *Making Software: What Really Works and Why We Believe It*, chapter 6. O'Reilly, 2010.

[36] H. He. What is Service-Oriented Architecture. On O'Reilly xml.com, http://www.xml.com/pub/a/ws/2003/09/30/soa.html, 2003. Accessed September 2012.

[37] K. J. Holyoak. Symbolic Connectionism: Toward Third-Generation Theories of Expertise. In K. A. Ericsson and J. Smith, editors, *Toward a General Theory of Expertise: Prospects and Limits*, pages 301–335. Cambridge University Press, 1991.

[38] IEEE Standards Association. 1516-2010 – IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA), howpublished = http://standards.ieee.org/findstds/standard/1516-2010.html, note = Accessed September 2012, year = 2010.

[39] IEEE Standards Association. 1730-2010—IEEE Recommended Practice for Distributed Simulation Engineering and Execution Process (DSEEP). http://standards.ieee.org/findstds/standard/1730-2010.html, 2010. Accessed February 2013.

[40] IEEE Standards Association. Standard for Distributed Interactive Simulation (DIS). http://standards.ieee.org/develop/project/1278.2.html, 2012. Accessed September 2012.

[41] P. Jarvis. *The Practitioner-Researcher*. Jossey-Bass Publishers, 1999.

[42] F. T. Johnsen, T. H. Bloebaum, and M. R. Brannsten. Quality Aspects of Web Services. Technical Report FFI-rapport 2012/02494, Norwegian Defence Research Establishment (FFI), 2012.

[43] E. J. Johnson. Expertise and Decision under Uncertainty: Performance and Process. In M. T. H. Chi, R. Glaser, and M. J. Farr, editors, *The Nature of Expertise*, pages 209–228. Lawrence Erlbaum Associates, Inc., 1988.

[44] H. D. Jørgensen, T. Liland, and S. Skogvold. Aligning TOGAF and NAF—Experiences from the Norwegian Armed Forces. In P. Johannesson, J. Krogstie, and A. Opdahl, editors, *The Practice of Enterprise Modeling*, volume 92 of *Lecture Notes in Business Information Processing*, pages 131–146. Springer, 2011.

[45] A. Josey. TOGAF Version 9.1 Enterprise Edition—An Introduction. http://pubs.opengroup.org/architecture/togaf9-doc/arch/, 2011. Accessed January 2013.

[46] json.org. Introducing JSON. http://json.org, 2013. Accessed July 2013.

[47] D. Kahneman and S. Frederick. A Model of Heuristic Judgment. In K. J. Holyoak and R. G. Morrison, editors, *The Cambridge Handbook of Thinking and Reasoning*, pages 267–294. Cambridge Univ. Press, 2004.

[48] R. A. Kass. *The Logic of Warfighting Experiments*. The Future of Command and Control. DoD Command and Control Research Program, 2006.

[49] G. Klein. Developing Expertise in Decision Making. *Thinking & Reasoning*, 3(4):337–352, 1997.

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

[51] R. Landaeta and A. Tolk. Project Management Challenges for Agile Federation Development: A Paradigm Shift. In *Proc. 2010 Fall Simulation Interoperability Workshop (SIW)*. Simulation Interoperability Standards Organization (SISO), 2010.

[52] B. Lang, M. Gerz, O. Meyer, and D. Sim. An Enterprise Architecture for the Delivery of a Modular Interoperability Solution. In *Semantic and Domain-based Interoperability: Proc. RTO Information Systems Technology Panel (IST) Symposium*. NATO Research and Technology Organisation, 2011.

[53] C. Larman and B. Vodde. *Scaling Lean & Agile Development: Thinking and Organizational Tools for Large-Scale Scrum*. Addison Wesley, 2008.

[54] B. H. Liskov and S. N. Zilles. Programming with Abstract Data Types. *ACM SIGPLAN Notices*, pages 50–59, 1974.

[55] K. Lund, F. T. Johnsen, T. H. Bloebaum, and E. Skjervold. SOA Pilot 2011—Service Infrastructure. Technical Report FFI-rapport 2011/02235, Norwegian Defence Research Establishment (FFI), 2012.

[56] L. A. Maciaszek. *Requirements Analysis and Systems Design*. Addison Wesley, third edition, 2007.

[57] L. Macvittie. SPDY versus HTML5 WebSockets. On SOAWorld Magazine, `http://soa.sys-con.com/node/2296338`, 2012. Accessed September 2012.

[58] I. Mahmood, R. Ayani, V. Vlassov, and F. Moradi. Fairness Verification of BOM-Based Composed Models Using Petri Nets. In *Proc. 11th IEEE Workshop on Principles of Advanced and Distributed Simulation*, pages 1–8. IEEE Computer Society, 2011.

[59] I. Mahmood, R. Ayani, V. Vlassov, and F. Moradi. 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)*, pages 250–257. IEEE Computer Society, 2012.

[60] E. J. McCormick, P. R. Jeanneret, and R. C. Mecham. A Study of Job Dimensions as Based on the Position Analysis Questionnaire. *J. Applied Psych.*, 56(4):347–368, 1972.

[61] O. M. Mevassvik and A. Alstad. Stridsledelsesspråk for koalisjonsoperasjoner—en teknologi for å integrere kommando og kontroll og simulering. Technical Report FFI-rapport 2012/00176, Norwegian Defence Research Establishment (FFI), 2012.

[62] A. Michlmayr, F. Rosenberg, P. Leitner, and S. Dustdar. End-to-End Support for QoS-Aware Service Selection, Binding, and Mediation in VRESCo. *IEEE Transactions on Services Computing*, 3(3):193–205, 2010.

[63] A. Michlmayr, F. Rosenberg, C. Platzer, M. Treiber, and S. Dustdar. Towards Recovering the Broken SOA Triangle—A Software Engineering Perspective. In *Proc. 2nd Int'l Workshop Service Oriented Software Engineering (IW-SOSWE'07)*, pages 22–28. ACM, 2007.

[64] V. Mojtahed, B. Andersson, V. Kabilan, and J. Zdravkovic. BOM++, a Semantically Enriched BOM. In *Proc. 2008 Spring Simulation Interoperability Workshop (SIW)*. Simulation Interoperability Standards Organization (SISO), 2008.

[65] V. Mojtahed, E.-O. Svee, and J. Zdravkovic. Semantic Enhancements when Designing a BOM-based Conceptual Model Repository. In *Proc. 2010 European Simulation Interoperability Workshop (SIW)*. Simulation Interoperability Standards Organization (SISO), 2010.

[66] B. Möller, F. Antelius, M. Johansson, B. Löfstrand, and Å. Wihlborg. Processes and Tools for Management and Reuse of FOM Modules. In *Proc. 2010 Fall Simulation Interoperability Workshop (SIW)*. Simulation Interoperability Standards Organization (SISO), 2010.

[67] B. Möller and C. Dahlin. A First Look at the HLA Evolved Web Service API. In *Proc. 2006 European Simulation Interoperability Workshop (SIW)*. Simulation Interoperability Standards Organization (SISO), 2006.

[68] B. Möller, P. Gustavson, R. Lutz, and B. Löfstrand. Making Your BOMs and FOM Modules Play Together. In *Proc. 2007 Fall Simulation Interoperability Workshop (SIW)*. Simulation Interoperability Standards Organization (SISO), 2007.

[69] B. Möller and S. Löf. A Management Overview of the HLA Evolved Web Service API. In *Proc. 2006 Fall Simulation Interoperability Workshop (SIW)*. Simulation Interoperability Standards Organization (SISO), 2006.

[70] B. Möller, K. L. Morse, M. Lightner, R. Little, and R. Lutz. HLA Evolved—A Summary of Major Technical Improvements. In *Proc. 2008 Fall Simulation Interoperability Workshop (SIW)*. Simulation Interoperability Standards Organization (SISO), 2008.

[71] F. Moradi, R. Ayani, S. Mokarizadeh, G. H. A. Shahmirzadi, and G. Tan. 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)*, pages 145–155. IEEE Computer Society, 2007.

[72] T. Mussweiler. Comparison Processes in Social Judgment: Mechanisms and Consequences. *Psych. Review*, 110(3):472–489, 2003.

[73] NATO Communications and Information Agency (NCIA). The C3 Classification Taxonomy. `http://www.ncia.nato.int/ourwork/Pages/Coherence/C3-Classification-Taxonomy.aspx`, 2011. Accessed August 2012.

[74] NATO Consultation, Command and Control Board. NATO Architecture Framework Version 3. `http://www.nhqc3s.nato.int/ARCHITECTURE/_docs/NAF_v3/ANNEX1.pdf`, 2007. Accessed January 2013.

[75] NATO Consultation, Command and Control Board (C3B). Core Enterprise Services Standards Recommendations—The Service Oriented Architecture (SOA) Baseline Profile, Version 1.7. On Tidepedia, 2011. Accessed September 2012.

[76] NATO Modelling and Simulation Group. NATO Modelling and Simulation Master Plan (version 1.0), 1998.

[77] NATO Modelling and Simulation Group. NATO Modelling and Simulation Master Plan (version 2.0). http://ftp.rta.nato.int/Public/Documents/MSG/NATO_MS_Master_Plan_Web.pdf, 2012. Accessed January 2013.

[78] NATO Network Enabled Capability (NNEC). NATO Network Enabled Capability. http://www.nato.int/cps/en/natolive/topics_54644.htm, 2010. Accessed October 2012.

[79] NATO Research and Technology Organisation. Coalition Battle Management Language (C-BML). Technical Report RTO-TR-MSG-048, 2012.

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

[81] NATO Research and Technology Organisation Modelling and Simulation Group (NMSG) Task Group MSG-005/TG-005. Federation Development and Execution Process (FEDEP) Tools in Support of NATO Modelling & Simulation (M&S) Programmes. Technical Report RTO-TR-MSG-005, 2004.

[82] Norwegian Ministry of Defence. Policy for militær tilpasning og anvendelse av informasjons- og kommunikasjonsteknologi i Forsvaret. http://www.regjeringen.no/upload/FD/Reglement/Policy_militaer_tilpasning_IKT_2oppl.pdf, 2005. Accessed August 2012.

[83] Norwegian Ministry of Defence. Forsvarssjefens plan for utvikling av nettverksbasert forsvar, Del I—Strategi, 2010.

[84] Norwegian Ministry of Defence. Forsvarssjefens plan for utvikling av nettverksbasert forsvar, Del II—Plan, 2011.

[85] Norwegian Ministry of Defence. Et forsvar for vår tid. Prop. 73S (2011–201). Proposisjon til Stortinget (forslag til stortingsvedtak). http://www.regjeringen.no/pages/37583840/PDFS/PRP201120120073000DDDPDFS.pdf, 2012. Accessed January 2013.

[86] Norwegian Ministry of Defence. Forsvarets IKT-strategi, 2013.

[87] Organization for the Advancement of Structured Information Standards (OASIS). UDDI Spec Technical Committee Draft. http://uddi.org/pubs/uddi_v3.htm, 2004. Accessed September 2012.

[88] Organization for the Advancement of Structured Information Standards (OASIS). Web Services Brokered Notification (WS-BrokeredNotification) 1.3. https://www.

oasis-open.org/committees/tc_home.php?wg_abbrev=wsn, 2006. Accessed October 2012.

[89] Organization for the Advancement of Structured Information Standards (OASIS). Web Services Dynamic Discovery (WSDiscovery) Version 1.1. http://docs.oasis-open.org/ws-dd/discovery/1.1/os/wsdd-discovery-1.1-spec-os.pdf, 2009. Accessed October 2012.

[90] C. Pautasso, O. Zimmermann, and F. Leymann. RESTful Web Services vs. Big Web Services: Making the Right Architectural Decision. In *17th Int'l World Wide Web Conference (WWW2008)*, pages 805–814. ACM, 2008.

[91] M. D. Petty and P. Gustavson. Combat Modeling with the High Level Architecture and Base Object Models. In A. Tolk, editor, *Engineering Principles of Combat Modeling and Distributed Simulation*, chapter 19, pages 413–448. Wiley, 2012.

[92] G. D. Plotkin and M. Abadi. A Logic for Parametric Polymorphism. In M. Bezem and J. F. Groote, editors, *Typed Lambda Calculi and Applications. Proceedings of the International Conference, TLCA'93, Utrecht (The Netherlands)*, volume 664 of *Lecture Notes in Computer Science*, pages 361–375. Springer Verlag, 1993.

[93] E. D. Pulakos, S. Arad, M. A. Donovan, and K. E. Plamondon. Adaptibility in the Work Place: Development of a Taxonomy of Adaptive Performance. *J. Applied Psychology*, 85(4):612–624, 2000.

[94] J. M. Pullen, D. Corner, A. Brook, R. Wittman, O. M. Mevassvik, and A. Alstad. MSDL and C-BML Working Together for NATO MSG-085. In *Proc. 2012 Spring Simulation Interoperability Workshop (SIW)*. Simulation Interoperability Standards Organization (SISO), 2012.

[95] W. Reitman. *Cognition and Thought*. Wiley, 1965.

[96] H. Rognerud and J. E. Hannay. Challenges in Enterprise Software Integration: An Industrial Study Using Repertory Grids. In *Proc. 3rd Int'l Symp.Empirical Software Engineering and Measurement (ESEM)*, pages 11–22. IEEE Computer Society, 2009.

[97] A. San Jose Martin. MSG-ET-34 M&S as a Service MSaaS—First Exploratory Team meeting. Master Presentation, 2012.

[98] F. L. Schmidt, J. E. Hunter, and A. N. Outerbridge. Impact of Job Experience and Ability on Job Knowledge, Work Sample performance, and Supervisory Ratings of Job performance. *J. Applied Psychology*, 71(3):432–439, 1986.

[99] F. L. Schmidt, J. E. Hunter, A. N. Outerbridge, and S. Goff. Joint Relation of Experience and Ability with Job Performance: Test of Three Hypotheses. *J. Applied Psychology*, 73(1):46–57, 1988.

[100] K. Schwaber. *Agile Project Management with Scrum*. Microsoft Press, 2004.

[101] SEDRIS Technologies. SEDRIS. http://www.sedris.org/ab_4trpl.htm, 2006. Accessed January 2013.

[102] S. B. Shadrick and J. W. Lussier. Training Complex Cognitive Skills: A Theme-based Approach to the Development of Battlefield Skills. In K. A. Ericsson, editor, *Development of Professional Expertise*, chapter 13, pages 286–311. Cambridge University Press, 2009.

[103] S. B. Shadrick, J. W. Lussier, and R. Hinkle. Concept Development for Future Domains: A New Method for Knowledge Elicitation. Technical Report 1167, U.S. Army Research Institute for the Behavioral and Social Sciences, 2005.

[104] H. A. Simon. The Structure of Ill-structured Problems. *Artificial Intelligence*, 4:181–201, 1973.

[105] H. A. Simon. *The Sciences of the Artificial*. MIT Press, third edition, 1996.

[106] Simulation Interoperability Standards Organization (SISO). Base Object Model (BOM) Template Specification.

[107] Simulation Interoperability Standards Organization (SISO). Guidance, Rationale, & Interoperability Modalities for the RPR FOM (GRIM 1.0). http://www.sisostds.org/DigitalLibrary.aspx?Command=Core_Download&EntryId=30822, 1999. Accessed January 2013.

[108] Simulation Interoperability Standards Organization (SISO). Real-time Platform Reference Federation Object Model (RPR FOM 1.0). http://www.sisostds.org/DigitalLibrary.aspx?Command=Core_Download&EntryId=30823, 1999. Accessed January 2013.

[109] Simulation Interoperability Standards Organization (SISO). Standard for: Military Scenario Definition Language (MSDL). http://www.sisostds.org/DigitalLibrary.aspx?Command=Core_Download&EntryId=30830, 2008. Accessed August 2012.

[110] B. Sisson, P. Gustavson, and K. Crosson. Adding Aggregate Services to the Mix: An SOA Implementation Use Case. In *Proc. 2006 Spring Simulation Interoperability Workshop (SIW)*. Simulation Interoperability Standards Organization (SISO), 2006.

[111] M. Sliger and S. Broderick. *The Software Project Manager's Bridge to Agility*. Addison Wesley, 2008.

[112] E. M. Smith, J. K. Ford, and S. W. J. Kozlowski. Building Adaptive Expertise: Implications for Training Design. In M. A. Quinones and A. Dudda, editors, *Training for 21st Century Technology: Applications of Psychological Research*, pages 89–118. APA Books, 1997.

[113] T. Sulaiman, B. Barton, and T. Blackburn. AgileEVM— Earned Value Management in Scrum Projects. In *Proc. IEEE AGILE 2006*, pages 7–16. IEEE Computer Society, 2006.

[114] C. Szyperski, D. Gruntz, and S. Murer. *Component Software, Beyond Object-Oriented Programming (Second Edition)*. Addison-Wesley, 2002.

[115] H. Takeuchi and I. Nonaka. The New New Product Development Game. *Harvard Business Review*, pages 137–146, January/February 1986.

[116] A. Tolk. Integration of M&S Solutions into the Operational Environment. In A. Tolk, editor, *Engineering Principles of Combat Modeling and Distributed Simulation*, chapter 15, pages 295–327. Wiley, 2012.

[117] A. Tolk. Modeling and Simulation Development and Preparation Processes. In A. Tolk, editor, *Engineering Principles of Combat Modeling and Distributed Simulation*, chapter 13, pages 243–262. Wiley, 2012.

[118] A. Tolk. Standards for Distributed Simulation. In A. Tolk, editor, *Engineering Principles of Combat Modeling and Distributed Simulation*, chapter 12, pages 209–241. Wiley, 2012.

[119] A. Tolk. Terms and Application Domains. In A. Tolk, editor, *Engineering Principles of Combat Modeling and Distributed Simulation*, chapter 4, pages 55–78. Wiley, 2012.

[120] A. Tolk. Verification and Validation. In A. Tolk, editor, *Engineering Principles of Combat Modeling and Distributed Simulation*, chapter 14, pages 263–294. Wiley, 2012.

[121] P. Vogel. Building a Simpler WebSockets Service. On Visual Studio Magazine, `http://visualstudiomagazine.com/articles/2012/06/22/building-a-simpler-websockets-service.aspx`, 2012. Accessed September 2012.

[122] J. F. Voss and T. A. Post. On the Solving of Ill-Structured Problems. In M. T. H. Chi, R. Glaser, and M. J. Farr, editors, *The Nature of Expertise*, pages 261–285. Lawrence Erlbaum Associates, Inc., 1988.

[123] A. L. Wilson and R. M. Weatherly. The aggregate level simulation protocol: an evolving system. In *Simulation Conference Proceedings, Winter 1994*, pages 781–787, Dec. 1994.

[124] E. Winjum, O. I. Bentstuen, A. Eggen, K. Lund, R. H. MacDonald, N. A. Nordbotten, R. Rasmussen, B. Reitan, and J. E. Voldhaug. Forslag til innretting av perspektivplan materiell (PPM) for programområde NbF-systemer. Technical Report FFI-rapport 2012/02075, Norwegian Defence Research Establishment (FFI), 2012.

[125] R. E. Wood. Task Complexity: Definition of the Construct. *Behaviour and Human Decision Processes*, 37:60–82, 1986.

[126] World Wide Web Consortium (W3C). Web Services Architecture. `http://www.w3.org/2002/ws/arch`, 2004. Accessed September 2012.

[127] World Wide Web Consortium (W3C). SOAP Version 1.2 Part 0: Primer (Second Edition). http://www.w3.org/TR/2007/REC-soap12-part0-20070427, 2007. Accessed July 2013.

[128] World Wide Web Consortium (W3C). Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. http://www.w3.org/TR/wsdl20, 2007. Accessed September 2012.

[129] World Wide Web Consortium (W3C). Web Application Description Language (WADL) Submission. http://www.w3.org/Submission/wadl, 2009. Accessed July 2013.

[130] World Wide Web Consortium (W3C). Extensible Markup Language (XML). http://www.w3.org/XML, 2013. Accessed July 2013.

Abbreviations

ACT	NATO Allied Command Transformation
ADM	TOGAF Architecture Development Method
API	Application Programming Interface
BOM	Base Object Model
C2	Command and Control
C3	Consultation, Command and Control
CAX	Computer-Assisted Exercise
C-BML	Coalition Battle Management Language
CD&E	Concept Development and Experimentation
CIS	Communications and Information Systems
COI	Community of Interest
COP	Common Operating Picture
DIS	Distributed Interactive Simulation
DSEEP	Distributed Simulation Engineering and Execution Process
EM	Enterprise Mapping
ESB	Enterprise Service Bus
FAD	Federation Agreement Document
FEAT	Federation Agreement Template
FEDEP	Federation Development and Execution Process
FFI	Forsvarets forskningsinstitutt (Norwegian Defence Research Establishment)
FOM	Federation Object Model
FTP	File Transfer Protocol
GRIM	Guidance, Rationale, & Interoperability Modalities
GUI	Graphical User Interface
HLA	High-Level Architecture
HTML	Hypertext Markup Language
HTTP	Hypertext Transport Protocol
IA	Information Assurance
IEEE	Institute of Electrical and Electronics Engineers
INI	Norwegian Defence Information Infrastructure
JC3IEDM	Joint Consultation, Command and Control Information Exchange Data Model
JSON	avaScript Object Notation
LVCAR	LVC Architecture Roadmap
LVCAR-I	LVC Architecture Roadmap Implementation
M&S	Modelling and Simulation
MIM	MIP Information Model
MIP	Multilateral Interoperability Program
MSaaS	M&S as a Service

MSDL	Military Scenario Definition Language
MSG	Modelling and Simulation Group
NAF	NATO Architecture Framework
NATO	North Atlantic Treaty Organization
NC3B	NATO Consultation, Command and Control Board
NCIA	NATO Communications and Information Agency
NDPP	NATO Defence Planning Process
NETN	NATO Education and Training Network
NII	Networking and Information Infrastructure
NNEC	NATO Network-Enabled Capability
OMT	Object Model Template
OSI	Open Systems Interconnection
REST	Representational State Transfer
RPC	Remote Procedure Call
RPR-FOM	Realtime-Platform-Reference Federation Object Model
RSS	Rich Site Summary
RTI	Runtime Infrastructure
SISO	Simulation Interoperability Standards Organization
SMC	Service Management and Control
SMTP	Simple Mail Transport Protocol
SOA	Service-Oriented Architecture
SOAP	Simple Object Access Protocol
SOM	Simulation Object Model
STANAG	Standardization Agreement
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
TIDE	Technology for Information, Decision and Execution superiority
TOGAF	The Open Group Architecture Framework
TTP	Tactics, Techniques, Procedures
UDDI	Universal Description, Discovery and Integration
WADL	Web Application Description Language
WSDL	Web Services Description Language
WWW	World Wide Web
XML	Extensible Markup Language
XSLT	Extensible Stylesheet Language Transformations