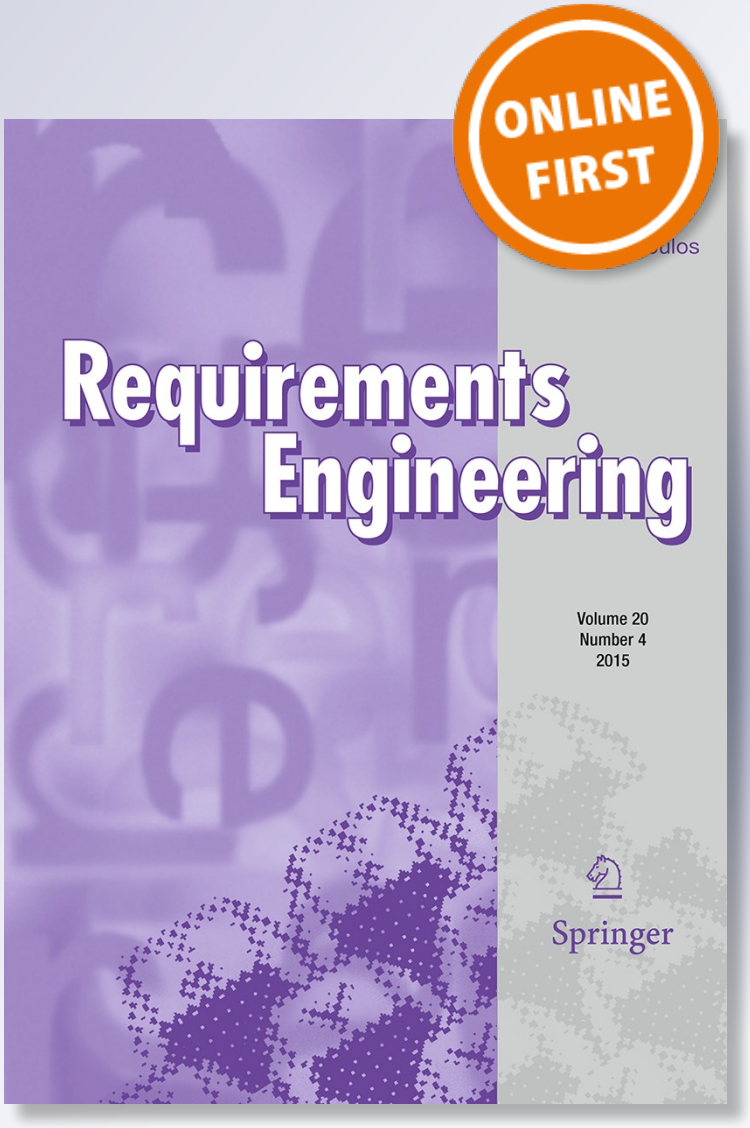
Agile requirements handling in a service-oriented taxonomy of capabilities

Jo Erskine Hannay, Karsten Brathen & Ole Martin Mevassvik

Requirements Engineering

ISSN 0947-3602

Requirements Eng
DOI 10.1007/s00766-016-0244-8



Springer

Requirements Eng
DOI 10.1007/s00766-016-0244-8

ORIGINAL ARTICLE

Agile requirements handling in a service-oriented taxonomy of capabilities

Jo Erskine Hannay[1] · **Karsten Brathen**[1] · **Ole Martin Mevassvik**[1]

Received: 26 June 2015 / Accepted: 4 January 2016

Abstract To get to grips with information systems portfolio development, strategic decisions tend towards service orientation and cloud deployment. Functionality should be presented as services that can be consumed from secure clouds in a range of contexts, and service-oriented architectures should enable one to build and rebuild systems portfolios readily and rapidly. However, there is little practical guidance on how to organize and coordinate the multiple lines of work that developing, or modernizing to, a service-oriented portfolio entails. We outline a method framework that uses the structure of a service-oriented taxonomy of capabilities to organize requirements and development in terms of elaboration and refinement of requirements. The method compiles several best practices and supports independent, but integral, lines of work that can be organized in small-scale projects. We illustrate the framework on three cases that involve computer- and simulation-assisted business processes. We conclude that service-oriented capability taxonomies can be used to structure and discipline requirements handling at all levels, from enterprise strategy to technical systems. We suggest that our framework supports the development of capabilities and services that are persistent in the service-oriented sense relative to each other and to implementation. We suggest further that the framework supports collaborative work by facilitating shared conceptions across lines of work. We emphasize that empirical studies should be conducted to evaluate and refine the framework.

Keywords Requirements handling · Service orientation · Architecture · Capability Taxonomy · Agile management and development

1 Introduction

IT systems portfolio management and development involve the architectural and managerial coordination of several projects. These projects may involve diverse activities, such as development, integration and modernization. Recent disasters in large portfolio modernization and development projects, with considerable loss to both enterprise and society, illustrate that there are inherent challenges in the portfolio setting that exacerbate those already present in single-project settings.

Further challenges arise due to growing flexibility demands on portfolios. Global businesses, acquisitions, offshoring and rapidly changing markets entail that functionality must be accessible anywhere and readily and rapidly adaptable to changing business processes.

To tackle architectural challenges, service orientation and cloud computing have been leveraged as means not only to design more flexible, reusable and adaptable systems, but also to enable cross-project architectural work within a portfolio. However, for several portfolios, there has been considerable architectural work to and fro at very high levels of abstraction with little effect on further development, and in some instances, it seems hard to even get started [8, 27].

Besides architectural challenges, several management issues arise in multi-line development settings [14, 22]:

✉ Jo Erskine Hannay
jo.hannay@ffi.no

Karsten Brathen
karsten.brathen@ffi.no

Ole Martin Mevassvik
ole-martin.mevassvik@ffi.no

[1] FFI (Norwegian Defence Research Establishment), PB. 25, 2027 Kjeller, Norway

Published online: 30 January 2016

1. No link between strategy and project selection
2. Reluctance to kill projects
3. Selection of short-term and easy projects
4. Lack of adequate information for decisions
5. Inadequate project-level activities
6. Lack of resources, competencies and methods
7. Unclear roles and responsibilities
8. Inadequate portfolio-level activities
9. Inadequate information management

Two main themes in this are project selection [4, 35] and actually having a means for portfolio management [16].

In the ensuing discussion, we will present a method framework for managing portfolio projects at the requirements level. We will elaborate the method in the context of defence systems portfolios, but the principles apply to any domain. We will show how the structure of a capability taxonomy can help to partition and discipline requirements handling and systems development, in conjunction with agile principles.

By design, our framework addresses the issues above: the capability taxonomy that we will use in this discussion explicitly declares and defines capabilities across enterprises at strategic, operational and technical levels. It can therefore be used to explicate links between strategy and projects. Further, the division of functionality into capabilities and services enables the definition of small manageable projects with less risk [38] and clear scope, while retaining a cohesive portfolio relationship. By using value management methods from agile practices, projects can be evaluated on their benefit/cost ratio, based on simple but adequate metrics, on the same scale, thereby facilitating project prioritization, selection and termination. This enables sound and consensual decisions within and between projects.

Viable requirements handling is vital in the complex landscape of portfolios. For example, acquisition and development process models in the defence domain tend to prescribe early planning and are rooted in explicit and unambiguous statements of user/sponsor needs, followed by traditional waterfall or V-Model development. While apt in many cases, the development of a loosely coupled systems portfolio demands a substantially different process model at the macro-level. The overwhelming tasks that projects face due to ambitions that entail too early planning are likely a disabler for getting started with eliciting useful requirements and getting under way with development of large portfolios [27].

Indeed, development work can get stranded in the architecture development phase, if architecture is perceived as an artefact to be completed prior to commencing systems development. This poses similar obstacles to effective development as did the view that requirements should be fully elicited prior to development, and which motivated the rise of agile management and development methods. The method framework we propose is iterative and incremental and avoids the massive wall of architectural preplanning by adhering to service-oriented, incremental architecture development.

We do not introduce any fundamentally new methods. Rather, we extend and combine existing best practices to a viable method for developing service-oriented systems portfolios, using adequate groupings of requirements as product elements. We see this effort as extremely important, since strategic decisions to use methods give no guidelines on how to use them, and in particular how to use them in combination.

In Sect. 2, we present the concept of capability taxonomy, and in Sect. 3, we extend relevant agile management and development practices to large development and value management. We propose how to use a capability taxonomy to structure development in Sect. 4. Sections 5, 6 and 7 outline three cases applying the method framework. We conclude in Sect. 8.

2 Capability taxonomies

The Open Group Service-Oriented Architecture (SOA) Reference Architecture (RA) [58] prescribes to organize functionality into layers of *capabilities*. A capability defines an ability to perform certain actions to achieve certain outcomes [58, 59] and are expressed in general and abstract terms. To implement a capability, a combination of organization, people, processes and technology must be coordinated into a so-called *capability configuration*. But importantly, capabilities as such are persistent and independent of their various implementations. A capability can therefore be seen as an abstract requirement that has a life time in terms of strategic periods, rather than in terms of, say, a development project. Capabilities should therefore be viewed as essential to portfolio definition, since they lend themselves to cross-project and long-term planning in an altogether better manner than does a focus on concrete assets or system-specific requirements.

Capabilities as a foundation for bridging the gap between enterprise modelling and service specification and design have lately received attention [63]. In the Capability as a Service in Digital Enterprises (CaaS) project, a capability is defined as the ability and capacity that enable an enterprise to achieve a business goal in a certain context [47]. The CaaS project develops a Capability-Driven Design (CDD) method, which aims to respond to the dynamic nature and changing context of modern business environments. Central to CDD is enterprise architecture modelling and IT service-oriented architectures [15].

Similar approaches are employed in the defence community with, e.g. the NATO Architecture Framework (NAF) [41] and its capability and service-oriented views.

The Open Group SOA RA is an abstract template that needs to be instantiated into domain-specific reference architectures. To support the transition to a service-oriented portfolio, the North Atlantic Treaty Organization (NATO) Communications and Information (NCI) Agency are developing the Consultation, Command and Control (C3) Taxonomy [40]; see Fig. 1 for an abstract view. Figure 2 shows a more detailed view (the point is to show the overall structure, so full legibility of the figure is not important). In defence terms, C3 encompasses military management capabilities, which are also pertinent for civilian authorities with crisis management tasks, such as emergency health authorities, police, fire brigades and coast guard. On a more fundamental level, the management capabilities embody principles that are central in managing any kind of business. The C3 Taxonomy is a C3 instantiation of The Open Group SOA RA at the level of capabilities.

The strategic vocabulary of defence has shifted from a focus on assets (specific material and resources) to a focus on capabilities. The taxonomy enables the defence

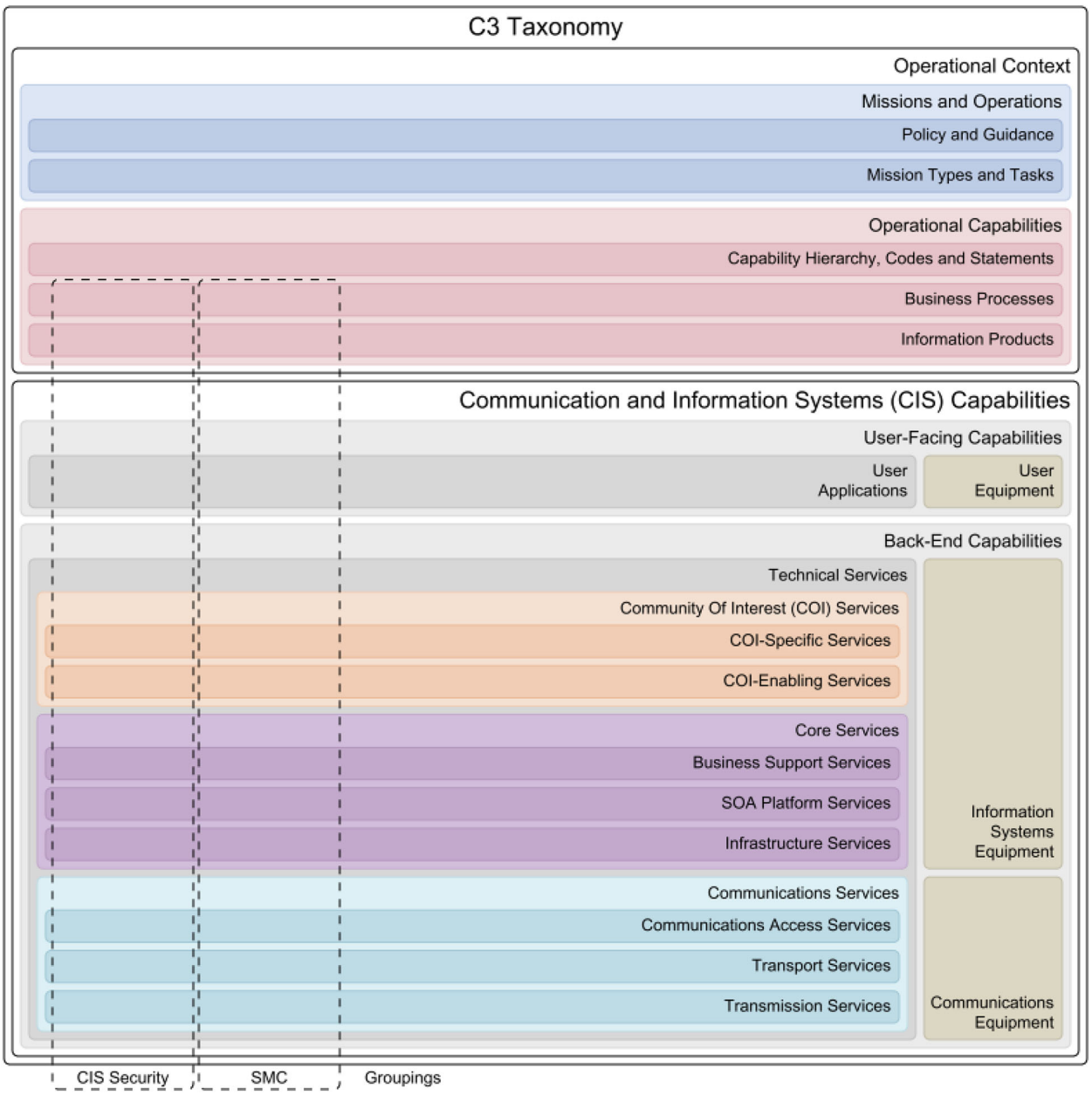


Fig. 1 C3 taxonomy—*top-level* view [40]

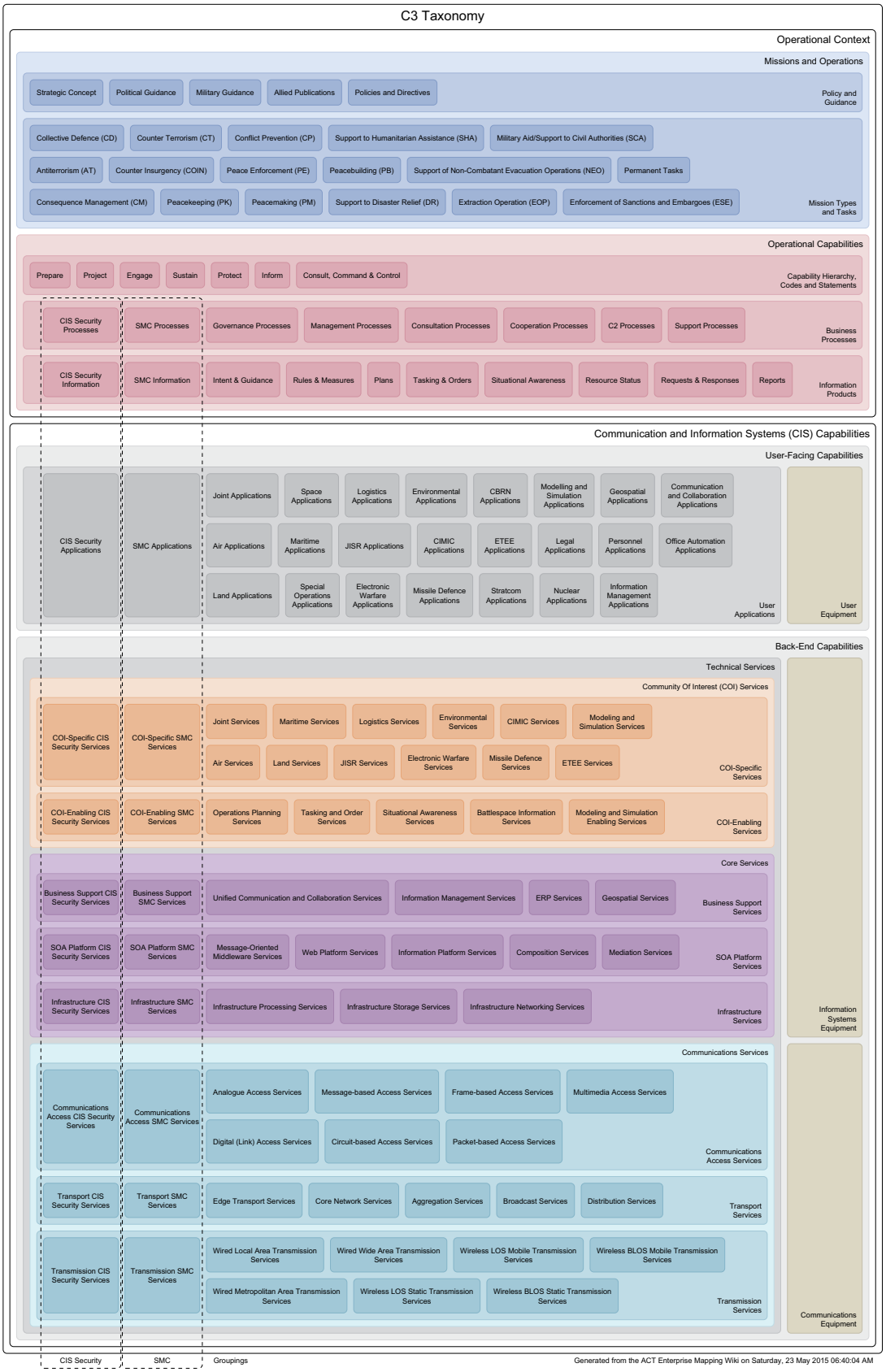


Fig. 2 C3 Taxonomy (more detailed view) [40]

community to sort all its doings into capabilities. In line with the layering of The Open Group SOA RA, it explicitly includes, in the same picture, the operational context (Operational Context frame) and the computing context (Communication and Information Systems (CIS) Capabilities frame).

The Operational Context is divided into Missions and Operations and the enabling Operational Capabilities. The Operational Capabilities section is further layered into Capability Hierarchy, Codes and Statements, which is a mapping out of operational capabilities together with abstract requirements describing what the capabilities are, or should be. Essentially, capability statements are formulated as abstract user stories describing what an organization, unit or person with a given capability is capable of doing, by using the two layers below; Business Processes and Information Products.

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

Two cross-cutting concerns are declared. The Information Assurance (IA) grouping holds functionality for safety and security. The Service Management and Control (SMC) grouping holds functionality for managing and federating a SOA, such as service discovery, mediation and quality of service management.

The service concept purports to meet the demands of today's systems portfolios: Loosely coupled services applicable across the portfolio entail that systems can be assembled and combined readily and rapidly according to the particular information handling demands at hand and are ready to be reassembled rapidly to meet upcoming and unforeseen needs. A service is persistent and may be realized by various service implementations administrated by various service providers; indeed, a service persists only through its abstract service description. In terms of their descriptions, or requirements specifications, capabilities and services are at the same level of abstraction. A natural grouping of services can be seen as a capability to support other capabilities.

The taxonomy is a "chest of drawers" in which to categorize capabilities. It is maintained via an Enterprise Mapping (EM) semantic wiki. Figure 2 shows only the outermost drawers. Each category (rectangle) in the figure can be elaborated and refined into sub-categories which themselves can be elaborated and refined and so on. We propose that each rectangle in Fig. 2 is therefore the root of an elaborate-refine tree into which one may place ever more specific requirement artefacts. This will be the topic in Sect. 4.

3 Agile management

Large IT development projects increasingly adopt agile management and development methods [23, 48, 55, 57], with the intention to manage the inevitable high levels of complexity and uncertainty in such projects. Although such projects have inherent challenges [7, 25], productivity is comparatively high, and production can be monitored and adjusted continuously so that the project can be managed based on the knowledge and experience acquired in the project. Agile requirements handling entails that knowledge be managed when available, minimizing the strain and wasted effort of eliciting information that is not yet available.

Appropriate requirements elicitation and handling are of crucial importance when using such methods. Nominees from each stakeholder group should engage in systematic methods for determining, formalizing, and content analysing requirements. Three key agile requirements handling principles are central to this discussion:

- 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 elements for production.

3.1 Product elements

In agile development, requirements are formulated in *user stories*, 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, user stories should be abstract placeholders, often called *epics*. During the course of the project epics get elaborated and refined into more detailed user stories, often called, simply, *stories*, and ultimately end up as concrete development tasks (*sprint tasks* in

Scrum) which produce shippable code; see Fig. 3. User stories (epics and stories) are units of production, or *product elements*. They are the driving artefacts in an agile development project.

In line with the second point above, epics should be 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 epics reflects the system as seen from the business context. Moreover, in line with the third point above, an epic represents a piece of the system which, from the vantage point of business context, makes sense to produce as a subsystem.

Epics should be explicitly prioritized according to estimated benefit and estimated cost [12, 17, 18, 56]. To this end, each epic should be given a cost estimate and a benefit estimate. While cost estimates are common, providing benefit estimates with the same level of rigour is rare. However, to deliver value for customer, it is important that not only project cost be managed but also project benefit. Benefit estimates should reflect epics' contribution to *project objectives* or *impact goals* elaborated in the business case for the project.

As an epic becomes elaborated and refined, technical details become relevant and therefore other stakeholders become involved. Stories and sprint tasks thus represent a partitioning of the system meaningful for more technical concerns. Stories inherit their epic's benefit and cost estimates according to the respective proportions of benefit and cost the story represents.

In general, we promote using a syntax in order to structure user stories and to obtain traceability of project objectives through all levels of refinement of user stories. For example,

> User story: **As** ⟨stakeholder A⟩ **I can** ⟨perform actions d in domain D⟩ **by using** ⟨functionality f in system S⟩ **to** ⟨perform actions s in S⟩ **in order to** ⟨reach project objective I⟩.

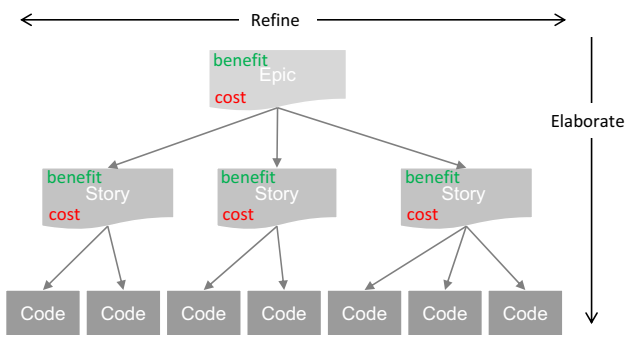


Fig. 3 Stepwise epic-elaborate-refine tree for user stories. Epics are elaborated into stories, which by separating concerns between them, represent a refinement of the epic. Likewise, stories are elaborated and refined (developed) into code

Here, all stakeholders A, project objectives I, actions d should be defined in the business case, while functionality f, actions s and system S are incrementally detailed through stepwise refinement.

User stories are requirements specifications and, at the same time, product elements. User stories can be incorporated into larger system descriptions and detailed using UML-style diagrams in conjunction with descriptive and methodological frameworks such as the NATO Architecture Framework (NAF) [41] and The Open Group Architecture Framework (TOGAF) [59].

3.2 Process

In large agile projects, iteration and increments may occur at all levels of planning. The agile fractal diagram [55] in Fig. 4 illustrates the point. At the project management level, epics are ordered into a product backlog, which is distributed into releases according to priorities and dependencies in the Project Vision. The stories in epics, in turn, are distributed to sprints according to priorities and dependencies uncovered in release plans. At the development level, sprint tasks resulting in code are organized into daily work in sprint plans.

4 C3 Taxonomy structure for agile management

Failures in large portfolio modernization and development suggest strongly that a single-system view, where the totality of (re-)development tasks is managed in one go under one project, gives a series of intractable tasks at all levels of the project. Moreover, when funding has been given for the total system, there is an overwhelming risk that a large part of the funding gets spent even after things start going wrong.

In contrast, issuing small projects on well-defined parts of a portfolio with appropriately limited funding is associated with much less risk. This demands an ability to organize portfolio development into manageable parts. We now suggest how a capability taxonomy can structure lightweight, controllable and documentable development, based on agile requirements handling. We will outline how to do this in several stages.

4.1 Extending the agile fractal

First, we will relate the agile fractal in Fig. 4 to the C3 Taxonomy. In Fig 5, the original fractal has been extended into the enterprise domain and colour coded according to

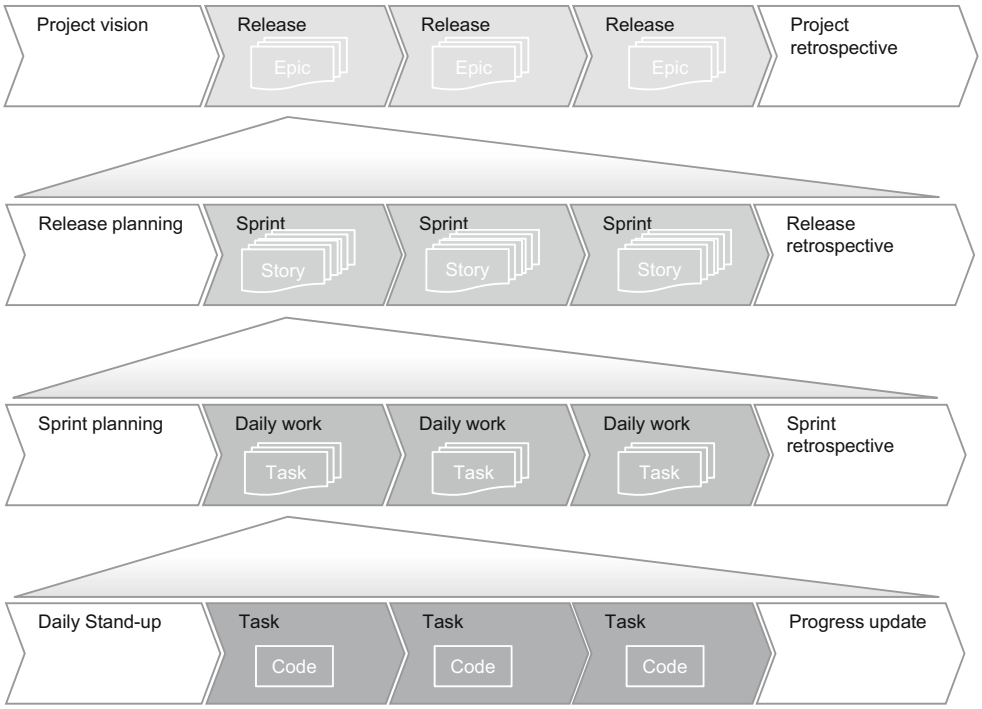


Fig. 4 Agile fractal (adapted from [55]) with product elements

the C3 Taxonomy. In the blue portion of Fig. 5, iterative business cases following the Business Vision drive initiatives or projects. The business cases hold objectives for the projects. In turn, business case objectives are linked to return or revenue in strategic periods in line with the enterprise's strategic policies in the upper light blue portion of Fig. 5.

This works as follows: the enterprise, in a certain strategic period, finds that to meet its goals it must initiate an initiative, perhaps in the form of an IT development project. Prior to launching the project, the enterprise develops a business case to mandate the project. The business case holds project objectives which the project must meet in order to be of value.

4.2 Product elements

In Fig. 5, product elements have been placed according to stakeholder focus and concerns, in relation to the C3 Taxonomy. We now take this idea further.

When designing a system to support business processes, it is, of course, vital to elicit information about those processes. In agile management and development, the product owner has an explicit role along the entire course of development so as to ensure timely feedback and input regarding business processes. However, in software development (even the agile kind) requirements elicitation and handling usually start some time around project inception and is focused around the system to be developed. As a consequence, one often confounds eliciting enterprise requirements for the system and understanding the enterprise process in itself. If the business process is not well understood, as may well be the case, the focus on eliciting requirements for a system may, in fact, define the business process in terms of that system. This is not desirable and goes counter to the intention of liberating business processes from the clutches of information systems.

We therefore propose that the structure of a capability taxonomy can aid in compartmentalizing process and systems development at all levels. A top-level category in the C3 Taxonomy can function as a starting point for development: In Fig. 6, *super epics* are very abstract epics that get refined into epics and stories in increasingly elaborated and refined categories of the taxonomy. Multiple levels of epics and stories may be called for. Here, the elaborate-refine trees (white) go horizontally inward into the taxonomy, rather than vertically down through layers as would "traditional" development, as depicted in Fig. 5.

This horizontal structure supports the development of capabilities which are independent and persistent relative to capabilities in surrounding layers; a prerequisite for loose coupling prescribed by SOA. Still, all capabilities have to be integral to the portfolio as a whole.

Notice how product elements in Fig. 6 retain the main colour (red, grey, orange) of their respective levels. This reflects that focus is on capabilities at that level. At the

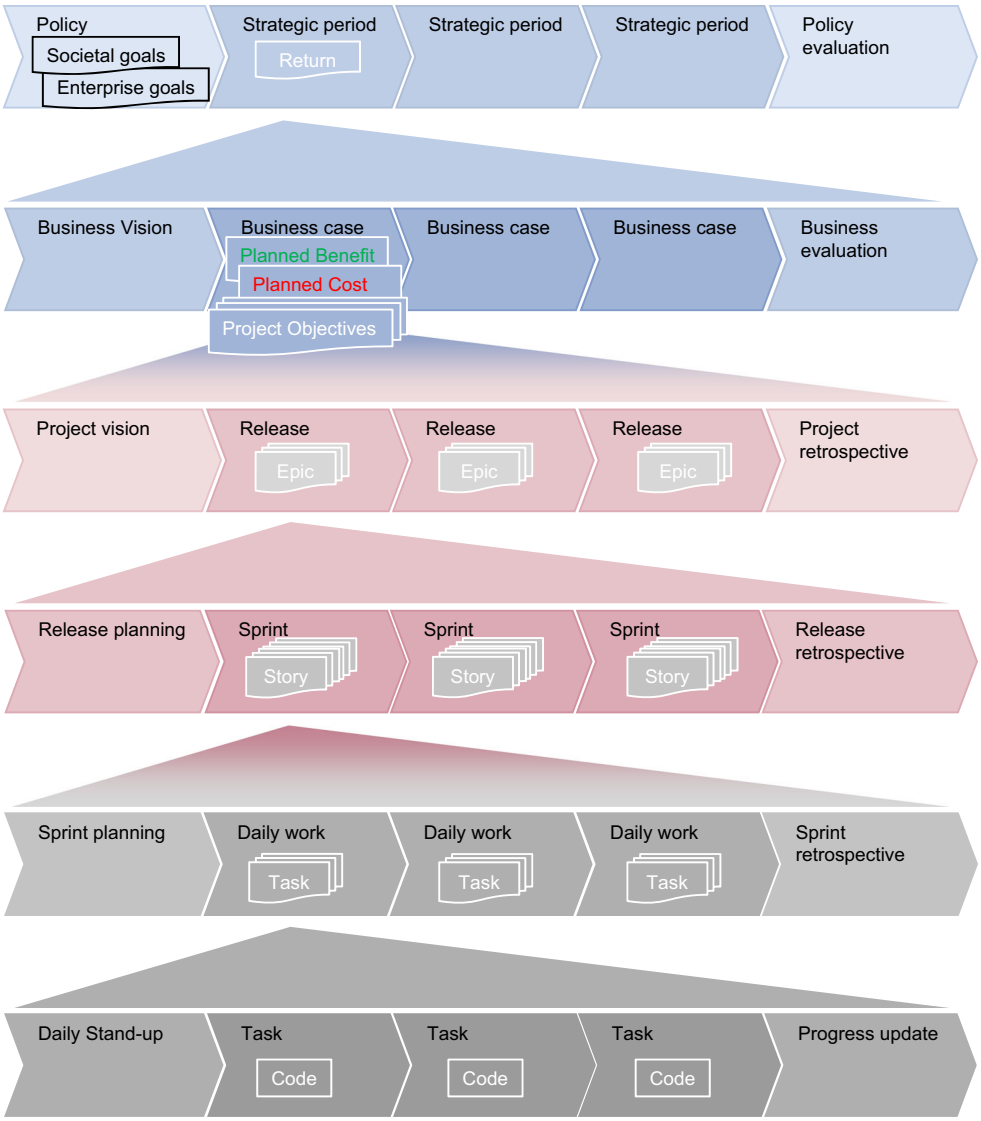


Fig. 5 Agile fractal extended into enterprise with C3 Taxonomy colour coding (adapted from [55] and extended)

operational context level, this gives the opportunity to develop operational/business processes without having to think about IT. In contrast, product elements in Fig. 5 are all grey, reflecting their traditional orientation towards being requirements for the system or user application.

4.3 Requirements for operational context

At the Operational Context level, elaborate-refine trees constitute development of operational capabilities. In fact, capability statements at this level are phrased in the form "Capable of (performing) actions [in domain] . . .", which fits our user story syntax well, with the understanding that the ⟨functionality *f* in system *S*⟩ refers to functionality in a human-based system.

An explicit focus on operational context is very much in line with the motivation behind SOA. A human-based system may, of course, involve computerized systems, but there is a point in remaining in the Operational Context as long as possible, so as not to loose focus inadvertently to technology. The focus on enterprise-driven service definition shifts the definition of an organization's enterprise processes back to where they belong: in the business process domain, and away from the IT department. Although user requirements are in focus in traditional systems and software development practices, it is still technologists who define the architecture of the system and who develop the system. The resulting IT system will possibly fulfil stakeholder needs at the point of deployment, but will likely become an obstacle when the organization's strategy and processes inevitably change. What is needed is systems that are agile to an organization's strategy and processes also after deployment. Loosely coupled layers and lines of development are means to accomplish that.

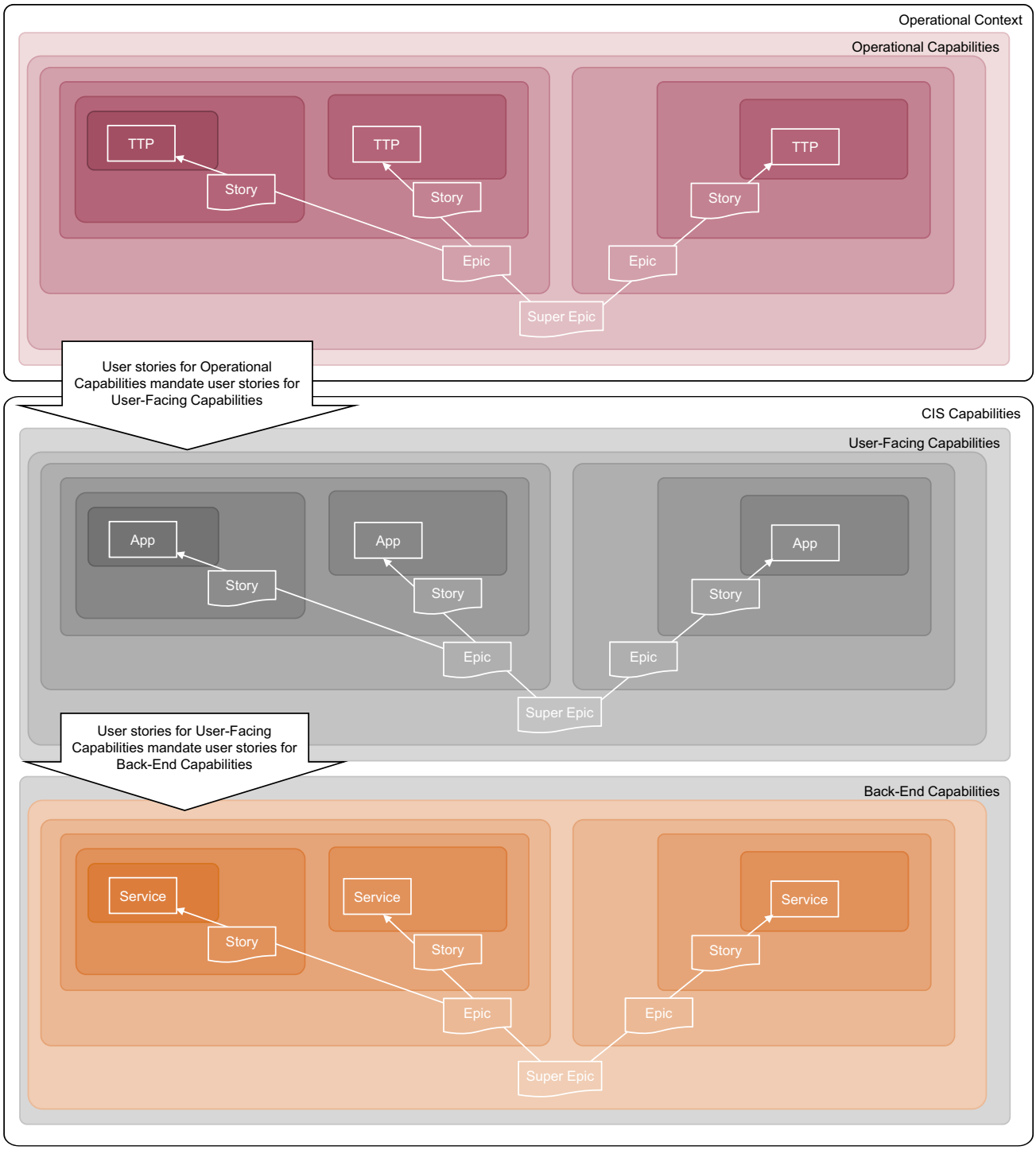


Fig. 6 Uniform approach to elaboration and refinement of requirements in the C3 Taxonomy. The epic-elaborate-refine trees (*white*) are placed horizontally away from the reader

Elaborate-refine trees within the Operational Context level can be developed using similar systematic methods found in requirements handling best practices for software and systems engineering. Multiple stakeholders may be used to elicit operational epics, which are elaborated and refined and eventually detailed into concrete operational tactics, techniques and procedures (TTPs). In agile requirements handling processes for software and systems engineering, 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 and modelling in virtual worlds.

Diagrams in, for example, Business Process Model and Notation (BPMN 2.0) and SoaML ordered in NAF views can be used to consolidate epics and stories at this level. However, the stratification that we promote here must be consciously adhered to, because the structure of many tools tend towards the traditional IT systems-focused approach in Fig. 5.

4.4 Requirements for user-facing capabilities

Requirements for User-Facing Capabilities should be mandated in Operational Context. The important difference between handling requirements for the Operational Context and handling requirements to IT systems, or CIS Capabilities in the C3 Taxonomy, 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 and modes for cooperation.

In turn, the horizontal structure evident in Fig. 6 implies that development of user applications may be undertaken with a steady focus on user interaction with IT, without delving too soon into technical detail. This supports the development of very thin user applications. Moreover, the reference to this user interaction level as capabilities in their own right supports user applications that are as independent and persistent with respect to underlying implementation as possible.

Epics for user applications go under various names in various frameworks. For example, Denne and Cleland-Huang use the term "Minimum Marketable Feature" to denote epics that are requirements to IT systems, and the Multilateral Interoperability Program (MIP) uses the term "Capability Package" to denote a manageable piece of capability functionality. Capability Package Teams in MIP use Scrum to develop epics to the level of architectural model in NAF [34].

Diagrams in, for example, SoaML, SysML and UML ordered in NAF views can consolidate epics and stories at the User Applications level. Again, stratification is supported somewhat but must be consciously adhered to. We read the SOA paradigm as prescribing thin-client user applications relying on services to perform tasks. The loose coupling and genericity inherent in the service concept entail that service development should be a separate process from developing a specific user application.

4.5 Requirements for back-end capabilities

Requirements for Back-End Capabilities should be mandated in User-Facing Capabilities. Elaborate-refine trees at this level support the development of (classes of) services as capabilities in themselves. In fact, the concepts of "service" and "capability" are, in many ways, similar, as mentioned in our introductory remarks. Notations such as SoaML, SysML and UML ordered in service-oriented and technical views in NAF can consolidate epics and stories at the Technical Services level.

4.6 Maintaining cohesive story lines

Above, we advocated the development of independent and persistent capabilities layer-wise in the C3 Taxonomy. However, these capabilities must be integral parts of a portfolio. It is therefore crucial to trace the story line through taxonomy layers. Further it is important to trace how requirements mandate requirements at levels below, since, without this traceability, it is hard to use the requirements to develop and compose architecture and systems according to intention.

We propose to use the syntax for user stories to help ensure traceability. Figure 7 shows how this may be done: the actions performed in the system become actions performed in the domain at the level below. This goes for levels internal to elaborate-refine trees and also for levels between elaborate-refine trees.

It is important to note that Fig. 7 represents a logical structure, in that it ensures cohesiveness in story lines

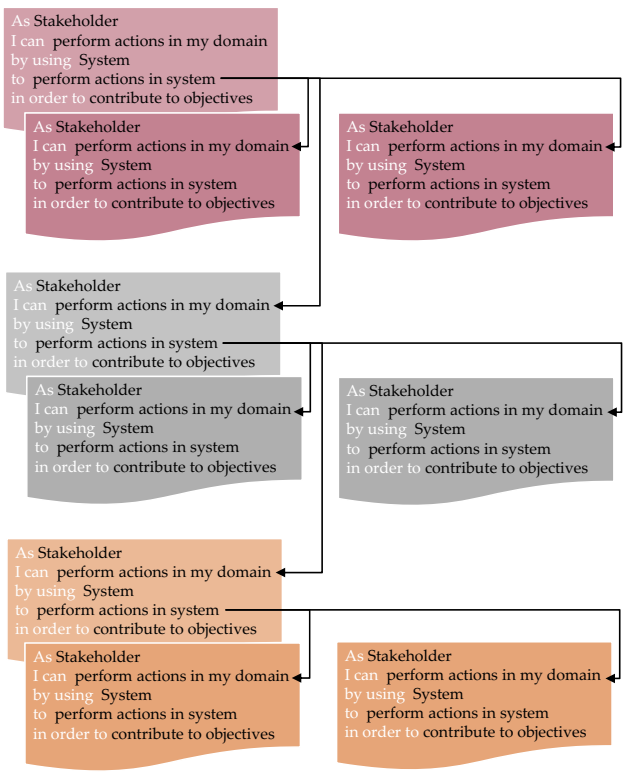


Fig. 7 Tracing elaboration and refinement through taxonomy layers

through layers in a taxonomy. There are two remarks to be made to this: The first is that although story lines should be cohesive, the architecture of the resulting federation of systems (systems portfolio) should still be loosely coupled. Second, although the structure in Fig. 7 shows a cohesive story line, the development of such a story line may follow other paths than those indicated by the relationship arrows in Fig. 7.

4.7 The ideal situation and the way there

At the future utopian limit, with perfect service orientation, new systems might be composed by orchestrating capabilities and services on the fly. This will perhaps be done by using a user application for orchestration, which consults a registry of capability and service descriptions. Thus, epic-elaborate-refine trees in the three main capability levels will need only relate to its own level (and indirectly to the layer below via the registry). In other words, development can proceed (strictly) horizontally in Fig. 6, and capabilities and services are "just there" to be consumed.

In an ideal world, development towards this utopian limit could proceed literally in line with the "people, then process, then technology" focus of SOA: Operational Context epics are elaborated and refined into concrete business processes and TTPs. These lay the basis for User-Facing Capabilities epics, which are then, in turn, elaborated and refined into concrete applications, which, in turn lay the basis for Back-End Capabilities epics, and so on through all layers of the C3 Taxonomy. This ensures the continuous guidance of operational context in developing a systems portfolio.

We are neither at this utopian limit, nor is it realistic to develop according to this ideal world scenario. Nevertheless, it is important to keep the ideal situation in view as a guidance for developing portfolios. To proceed towards this goal, development must proceed at all layers of the taxonomy, with extensive cross-layer communication between stakeholders. And, since there is a focus on extremely loose coupling this development can proceed in parallel runs, in small tractable projects.

Thus, it is not feasible to proceed in a strictly top-down manner, in the sense that one cannot suspend developing, e.g. core services until all business and operational processes have been defined and until all end-user applications have been developed. But the end results, the logical model, 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 are refined and updated, but where also objectives and architecture are updated during development. In that manner, each elaborate-refine tree undergoes revision, and the taxonomy is constantly sanitized in accordance with current understanding of portfolio needs.

Furthermore, on the way to this future limit, software will be in a transition from traditional (legacy) software to service-oriented applications and services. 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, registries and suitable services, one has to develop software at all layers. Thus, the more detailed user stories will often be expressed at lower layers of the taxonomy. This temporarily compromises the ideal situation, but embedding requirements and development in the structure of a capability taxonomy in the way we propose should ensure cohesiveness between layers at early stages of development, while maintaining the vision of loose coupling. This situation is depicted in Fig. 8, where development at higher layers introduces user stories at lower layers directly. Development will have to proceed at all layers in parallel, both to explore technical feasibility and give feedback to higher layers in the taxonomy and to ensure readiness for when operational capabilities reach maturity.

4.8 Whole product thinking in manageable parts

Our method framework as proposed above can be summarized in the following five principles:

(A) Use a capability taxonomy to structure requirements partitioning
(B) Use requirements in the form of user stories that are linked to project objectives
(C) Elaborate and refine user stories in step with the structure of the capability taxonomy
(D) Keep a focus on the horizontal layer of the taxonomy as far as possible
(E) Maintain explicit story line cohesiveness

Although the method framework must be detailed to suit particular development/integration/modernization efforts, we propose that a focus on these overlying principles will help portfolio developers to meet the managerial issues (1–9) outlined earlier.

Specifically, requirements can be structured according to persistent capabilities and services. This benefits persistence and stability in requirements handling as well. This gives a basis for defining small manageable projects within the total portfolio project (Issue 5, 8). Further, the capability taxonomy arches over the entire portfolio. This benefits requirements and project definition based on a shared view of the portfolio, which benefits cross-project prioritization and coordination (Issue 7, 8, 9). By using

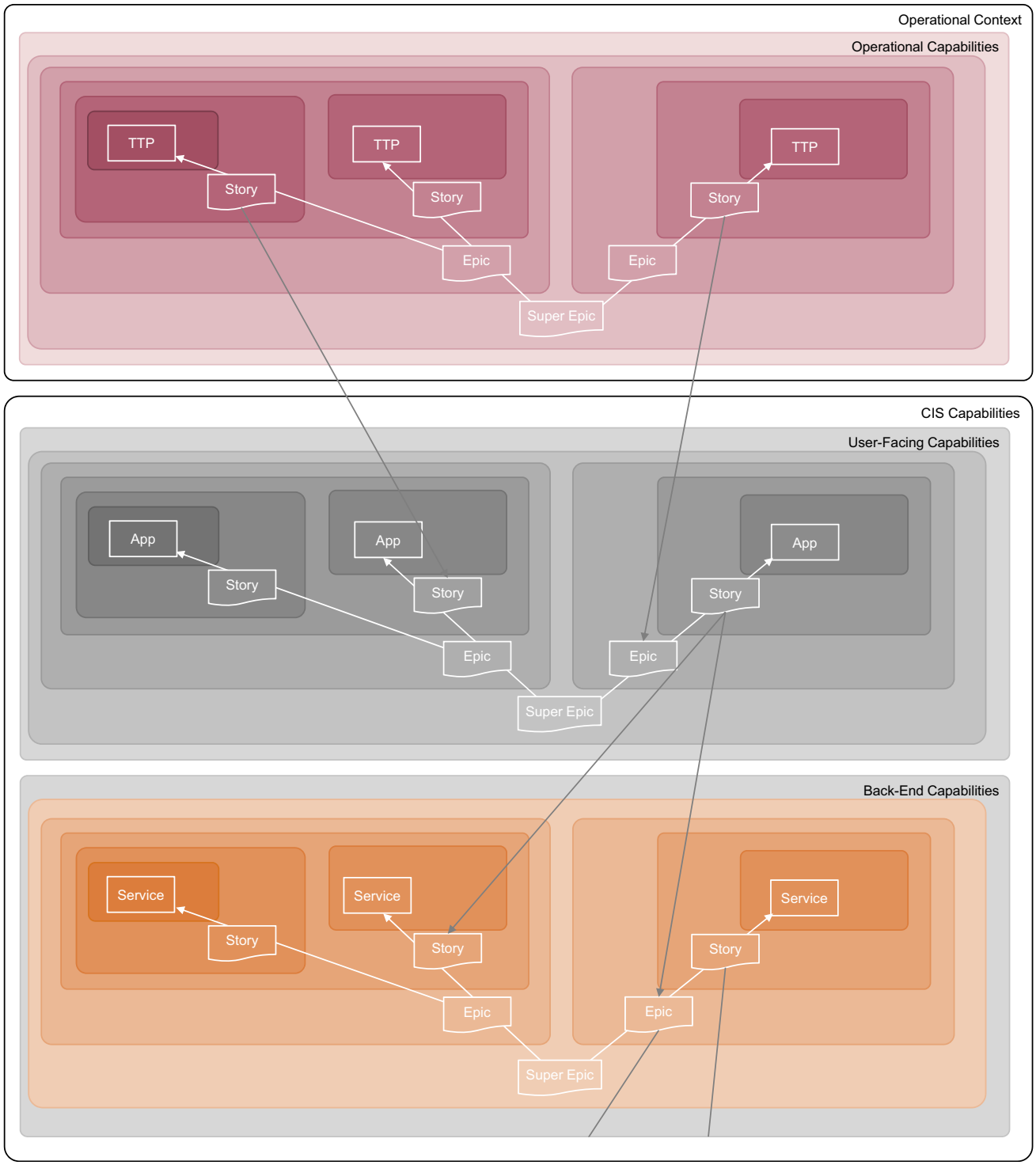


Fig. 8 Uniform approach to elaboration and refinement of requirements in the C3 Taxonomy. Horizontal development (*white*) interspersed with traditional vertical development (*grey*)

user stories endowed with both benefit and cost estimates, product elements can be prioritized both within and between projects (Issue 2, 5). Further, the accumulated benefit and cost for user stories as product elements give a benefit/cost index for each project, which gives a basis for sound strategically based decisions on project initiation, prioritization and termination (Issue 1, 2, 3, 4, 9). Importantly, this basis can now be shared throughout the team of portfolio project managers, facilitating common rules and consensual agreement (Issue 7, 9).

In elaborating and refining epics, the structure of capability refinement and elaboration in the taxonomy can be used to keep epics and stories on track with capability structure. By staying within a layer as long as possible, it is possible to develop capabilities that are persistent and independent (loosely coupled) with respect to other capabilities. By using our approach to maintain cohesive story lines amidst loose coupling, it is possible to achieve functional traceability from business processes, through user applications, to back-end services. The framework gives methods that enable project learning and competence building (Issue 6).

The method framework also allows the underlying service-oriented architecture to be developed incrementally together with systems implementing the capabilities. Development of both architecture and systems is captured in the same process, and a focus on product elements helps to keep one from loosing oneself in overly ambitious early models of architecture and system.

Figure 9 illustrates the idea. A capability taxonomy front-end is used to structure and coordinate a product element-driven approach to developing a systems portfolio and its architecture in an interleaving manner. Categories and sub-categories of the taxonomy reflect stages in requirements elaboration and refinement. Horizontal work within layers enables the development of capabilities that are independent and persistent relative to other capabilities in terms of SOA.

We postulate that this is a viable method for gaining headway in developing portfolios, due to the simplicity of its five principles. Empirical studies must evaluate the approach. Since the context is real-life large development/integration/modernization efforts, challenges include finding willing cases and designing studies to gain valid results in a relatively short time span. We now exemplify parts of the framework on three cases.

5 Case: Simulation-based training

We demonstrate the framework's horizontal development approach with a case. The case is an elaboration of a Concept Development and Experimentation (CD&E) activity for army training [28, 29]. Live, Virtual, Constructive (LVC) simulation combines fully computer-simulated units, personnel operating in a virtual environment and live players operating in the field with live equipment (but with, e.g. laser systems in place of real ammunition). LVC simulation is a technical capability intended to enable an increased and enhanced operational training capability.

The CD&E activity focused on developing a technical capability prototype. In Fig. 10, development that was done is indicated by solid arrows, while identified future development is indicated by dashed arrows. Here capability packages are top-level epics.

The training capability should be mandated from directives at the strategic level (blue). The NATO Task List (blue outlined box) declares the tasks that NATO forces should be capable of performing. It is a tool for developing training and for initiating operations. Then, the Collective Training And Exercise Directive (blue outlined box) gives comprehensive guidelines on how to plan for, execute and assess a NATO military training exercise. It is held that national task lists and training directives should encompass (LVC) simulation training explicitly, and we therefore indicate future development starting from a Training Directives capability package.

At the operational level (red), the LVC training capability package is elaborated and refined into requirements for collective training and exercise (CTE) processes for specifying, planning, conducting and assessing LVC simulation-supported exercises. Processes agnostic to LVC simulation are already declared in the indicated categories. The LVC processes shown are innovations to those processes that use LVC simulation to enhance the existing training capability.

At the user-facing level (grey), The LVC training capability mandates the development of the LVC technical capability. In Fig. 10, this is illustrated by the technical LVC capability package being elaborated and refined into requirements for applications for setting up scenarios, controlling LVC simulations, integrating simulation events into common visualizations for training audiences and analysing training. These thin applications consume simulation services and gateway services so that all LVC simulation entities and events are reflected in all systems; in particular, in Battle Management Systems (BMS) used by personnel in the field for situational awareness and Command and Control Information Systems (C2IS) used by operational command. Thus, training audiences can use their regular tools when undergoing LVC simulation-supported training. No epics at the service level were written. Rather, stories were elaborated from the user-facing epic.

The structuring into capabilities and layers of development enabled an explicit scope and definition for the CD&E activity. Moreover, the layered approach relating to the C3 Taxonomy facilitates a clear and structured picture of further necessary development efforts.

6 Case: Simulation-supported planning

This case is an elaboration of a development effort to produce a system prototype for simulation-supported planning [9]. The service-oriented system links C2IS with a simulation system. The C2IS is used by a brigade

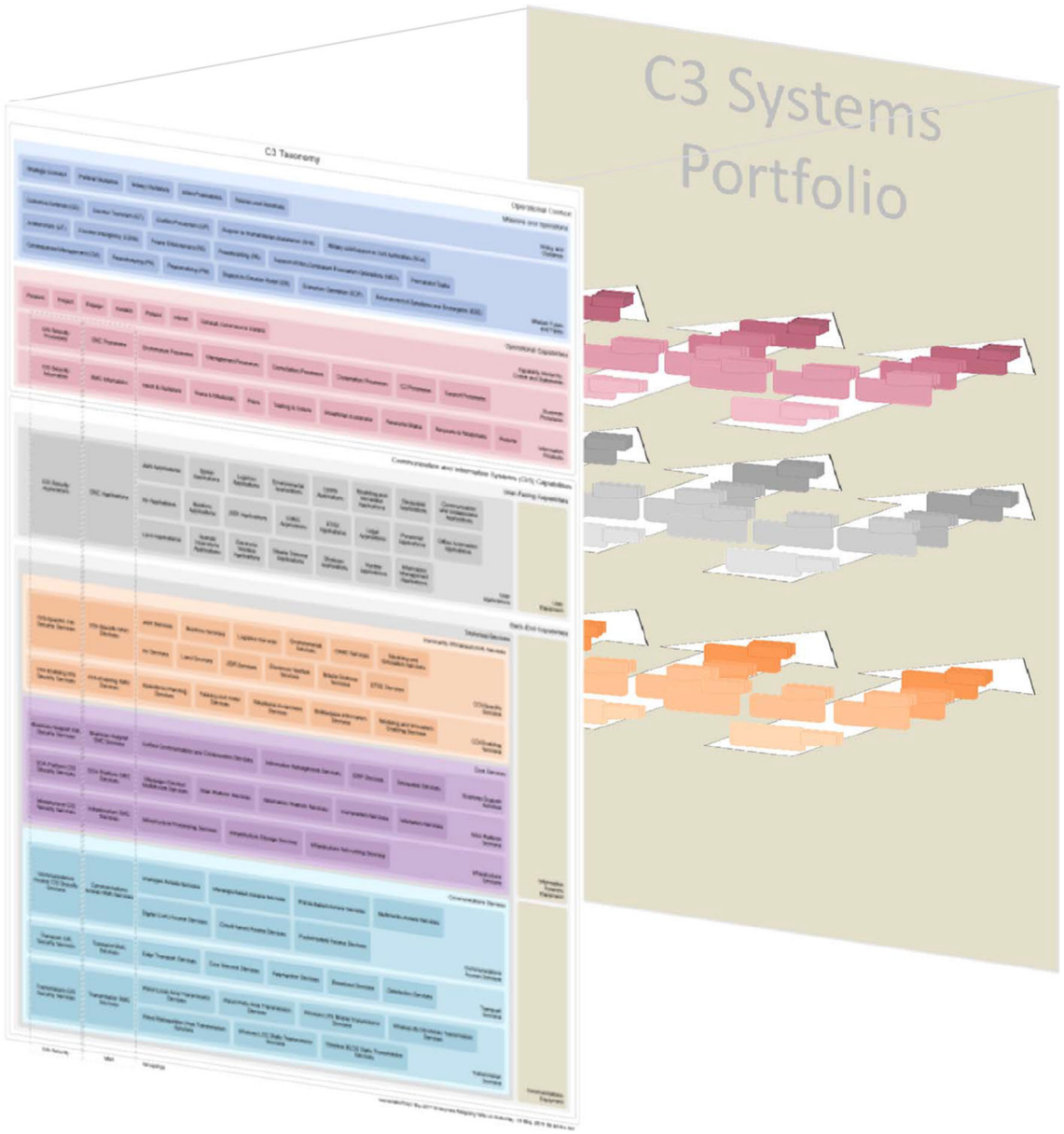


Fig. 9 Using the C3 Taxonomy to develop the C3 systems portfolio. Each category in the taxonomy is the starting point for elaborate-refine development trees which lead to the working systems portfolio

operations planning group (OPG) to develop plans for a military operation. The C2IS has functionality for drawing plan graphics with military symbols and arrows indicating directions of operation, etc. In developing a plan, various *Course of Action*s (COAs) are proposed. Traditionally, two, or at most three COAs are played out by the OPG war gaming the plan on a flat table with toy models to visualize the plan and assess its feasibility. Computer-supported simulation increases the amount of war gaming the OPG may indulge in and also gives more flexibility to try out alternative COAs with metrics to compare alternatives. Key to any plan is the *synchronization matrix*—a tool for coordinating resources and events. Computer simulations may be used to derive better synchronization matrices quicker.

Figure 11 shows the horizontal elaborate-refine trees in white. Strategically, the Comprehensive Operations Planning Directive (COPD) guides planning at various levels of operations. For this case, the COPD has a national counterpart, so there is no development indicated at the strategic level. In the Concept of Operations (CONOPS) Development Process category, sub-processes for COAs development, comparison and analysis are already present (red outlined boxes). New development aims to define

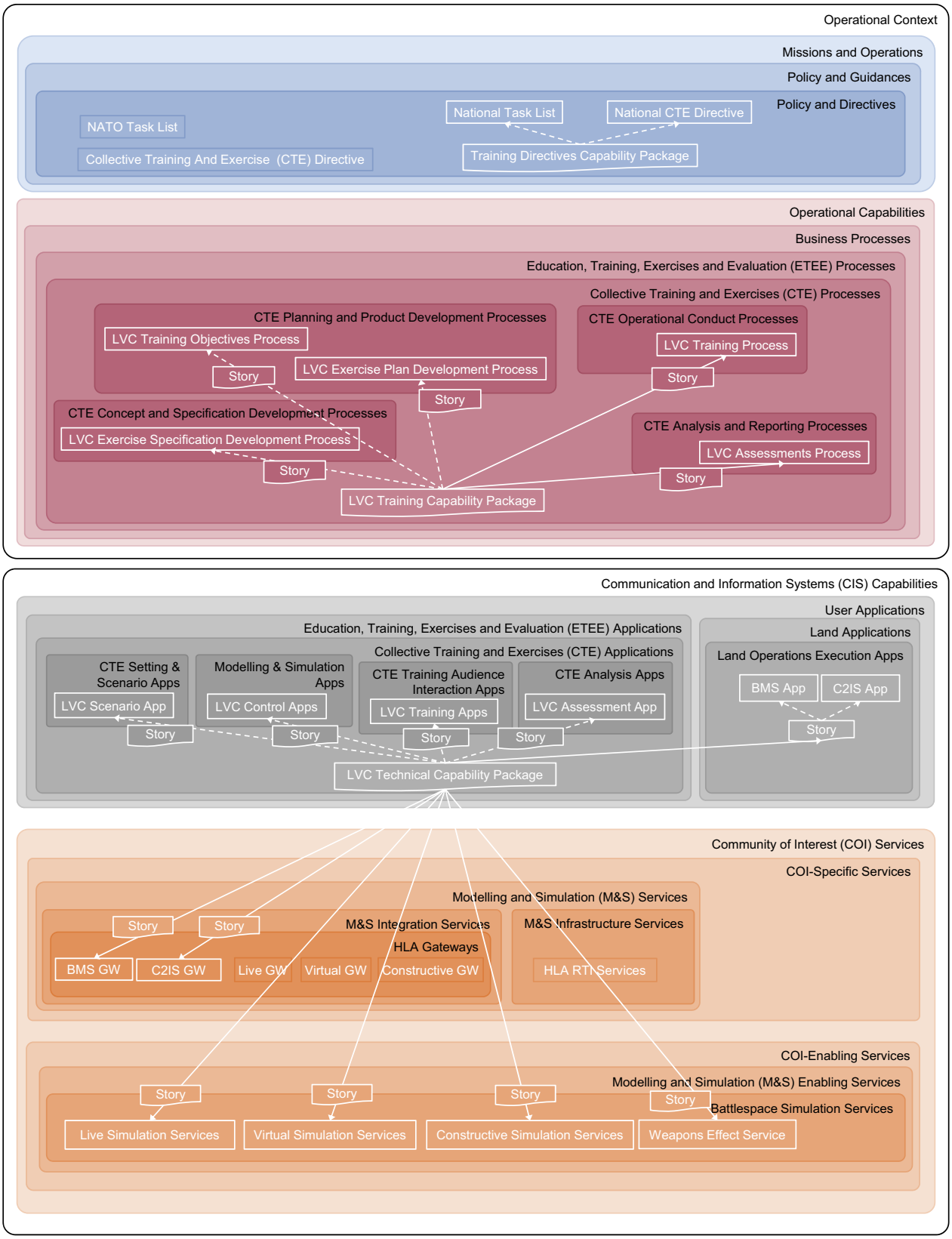


Fig. 10 Capability development: the technical capability enables a (new/improved) training capability, which is mandated by strategic directives. Horizontal development in the form of elaborate-refine trees (*white*). Initiated development (*solid arrows*) and future development (*dashed arrows*)

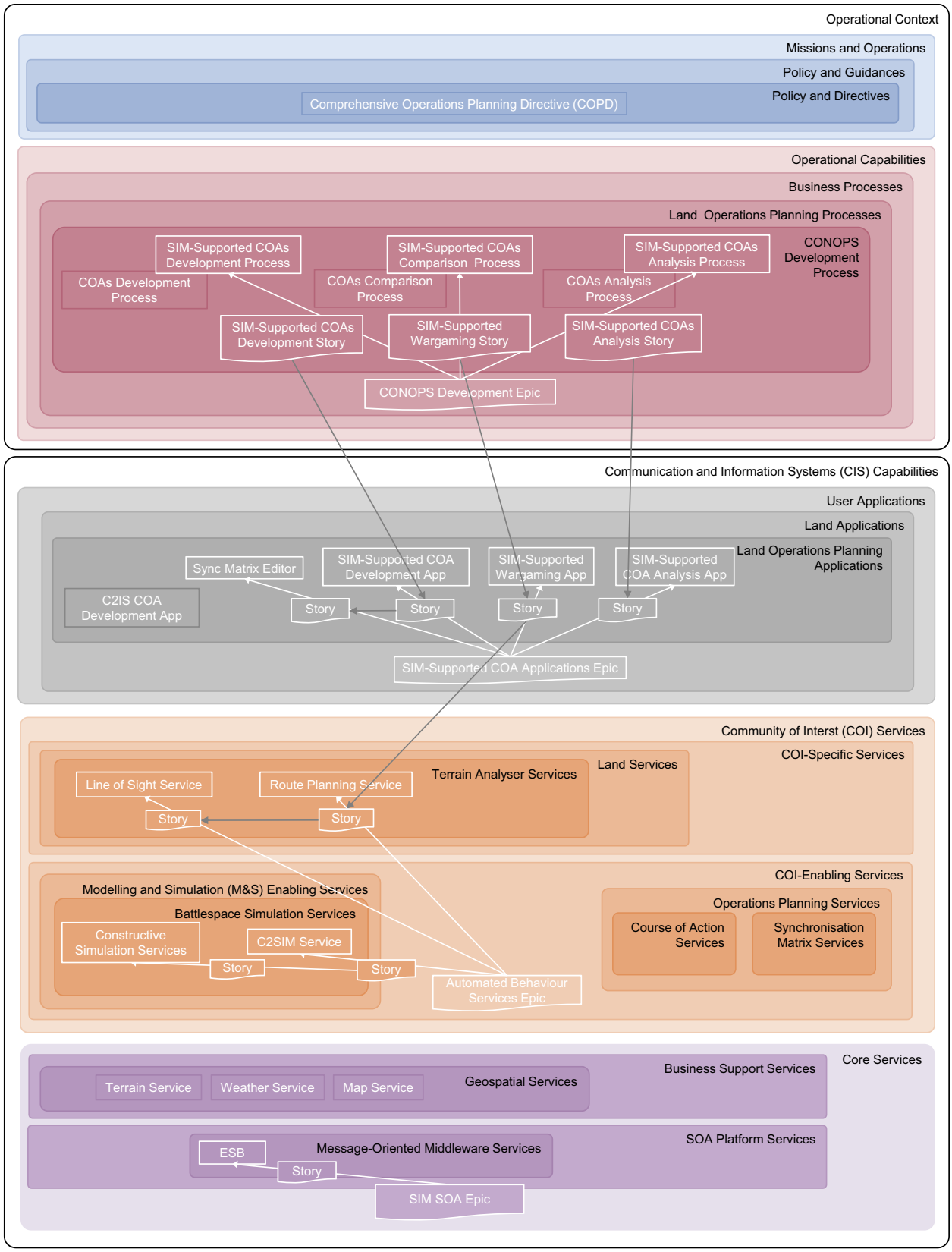


Fig. 11 Development of simulation-supported planning capabilities. Elaborate-refine trees (*white*) with a thread of traditional vertical development shown (*grey*)

simulation-supported enhancements to these starting from a CONOPS Development Epic.

At the User Applications level, a SIM-Supported COA Applications Epic initiates development of applications to support the parts of the CONOPS development process that pertain to COAs. In the Land operations Planning Applications category, we place a C2IS COAs Development Application (grey outlined box), to indicate that the C2IS in question has functionality for COA development, even though that functionality is not given through a thin client as prescribed by SOA. In general, legacy systems can be included in the C3 Taxonomy by their representing applications in this way. The understanding is that, in time, these should be refactored into or replaced by thin application-based clients calling on services. In the meantime, such systems may have to be service enabled via wrappers.

For this case, development at the service level starts appropriately with a designated epic, in recognition of the ideas of our framework. The C2SIM service uses behavioural models to break down high-level command and control orders at brigade level to orders for battalions and companies for execution in the constructive simulation running the war gaming. The Route Planning service is used by the C2SIM service to decide how forces should move according to terrain and threats.

To ensure loose coupling between all components, an enterprise service bus was developed to handle messaging. This was initiated by an abstract requirement (SIM SOA Epic) for simulations (adhering to specialized architectures) running in an encompassing SOA.

The basis of this case is a prototype development effort at the technical level. Since services were not in place and the operational simulation-supported processes were not defined in advance, development skipped through levels using traditional paths of requirements development in line with Fig. 8. The grey lines in Fig. 11 show the direction of requirements development in contrast to the horizontal structure strived for ultimately. For example, development did not proceed all the way to the operational SIM-Supported COAs Development Process, but skipped from a story down to a story for the development of the corresponding user application.

Figure 12 shows the epic and stories at the Operational Capabilities level, the epic and three stories at the User Applications level and the epic and three stories at the COI Services level. Figure 13 shows part of the cohesive story line of the user stories in Fig. 12. The grey lines in Fig. 13 show the direction of requirements development corresponding to the grey lines in Fig. 11. Although the sequence of actual development does not follow the structure of the cohesive story line, the latter is still the overall result of the requirements development steps.

Even though actual requirements development by necessity had traditional vertical aspects, the development team had the horizontal layered approach in mind. This enabled clarification to both developers and stakeholders as to at what level requirements were being developed and at what level innovations were targeted.

7 Case: Tentative user stories for M&S COI

We now illustrate the framework in a multi-project context. To provide starting points for developing modelling and simulation (M&S) capabilities, we put forth abstract requirements for M&S in a service-oriented systems portfolio. For space reasons, we show technical levels (CIS Capabilities), but it will be apparent that much of the motivation we state for technical-level requirements is material for strategic and operational user stories. More details of this case can be found in [27].

User stories are requirement specifications from the viewpoint of users of the portfolio. Users access the portfolio at various layers of the C3 Taxonomy, from operational personnel using User Applications to application/service users and providers, owners, developers and IT service personnel accessing the Technical Services. Users of the portfolio also include applications and services that use other applications and services.

We here focus on three human stakeholder types: The "personnel" stakeholders represents end-users with an interest in using applications and services for defence activities. The various "operators" stakeholders represent semi-technical staff who are capable of combining services and applications (by using suitable applications), e.g. a training operator tending 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.

As in the previous cases, we place user stories in the C3 Taxonomy. The appropriate category is determined in the "**by using**" clause in a user story, since this is where the direct demand on the portfolio is expressed. The placement into categories is summarized in Fig. 14.

There are several initiatives to populate the C3 Taxonomy from the M&S point of view, but with no method attached. For simulation, the NATO Distributed Networked Battle Labs (DNBL) Service Description for M&S Services summarizes important work [43].

Fig. 12 A thread of user stories for simulation-supported planning capabilities

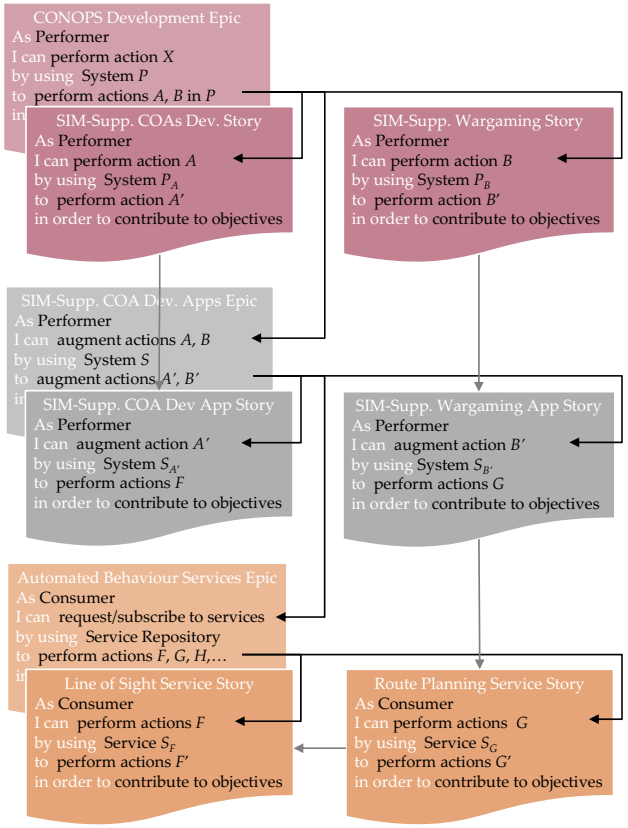


Fig. 13 Cohesive story line in user stories for computer-aided planning capabilities

We start at the User Applications layer, where we suggest user stories under the headings "Training", "Operations Planning and Rehearsal ", "Operations", "Retrospective Analyses", and "Concept Development and Experimentation (CD&E)". These are in line with the NATO Modelling and Simulation Master Plan [42]—in effect a strategic document outlining the modelling and simulation part of NATO systems portfolio.

7.1 Training

Training is 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. M&S plays an essential role in this.

7.1.1 Train as you fight

The maxim "Train as you fight" embodies the idea that training for a task should be undertaken on tasks and in environments as close to the actual task and environment as possible. If not, you will be unprepared for the operational situation to the extent the task and environment differ in essential characteristics.

M&S provides substitutes for actual tasks and environments for better benefit/cost ratio in training. However, participants should view and manipulate the simulated battle situation through their applications used in operations; otherwise, they will not be training a vital part of their operational task [60, 61].

> Epic 1: **As** operational personnel **I can** train realistically **by using** C2 Applications **to** employ simulation capabilities in training **in order to** increase capabilities for training effect by *X*.

The epic "LVC Technical Capability Package" (Sect. 5) can be placed as a sub-epic to Epic 1.

Further, simulations and operational applications should be loosely coupled, in the sense that it should be transparent to the user of operational applications 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 simulations 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 C2 Applications and CTE Services and M&S Services **in order to** increase training capabilities by *T*.

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.

7.1.2 "Train as you fight" is not enough

Research has shown that training that reflects actual circumstances is not sufficient alone. For example, while it is pertinent to train using actual tools in an actual environment, it may be necessary to engage in artificially enhanced tasks to heighten performance [26].

> Epic 3: **As** operational personnel **I can** train artificially **by using** C2 Applications **to** employ simulation capabilities in training **in order to** increase capabilities for training effect by *X*.

Epic "LVC Technical Capability Package" (Sect. 5) would be a sub-epic to Epic 3 and/or Epic 1.

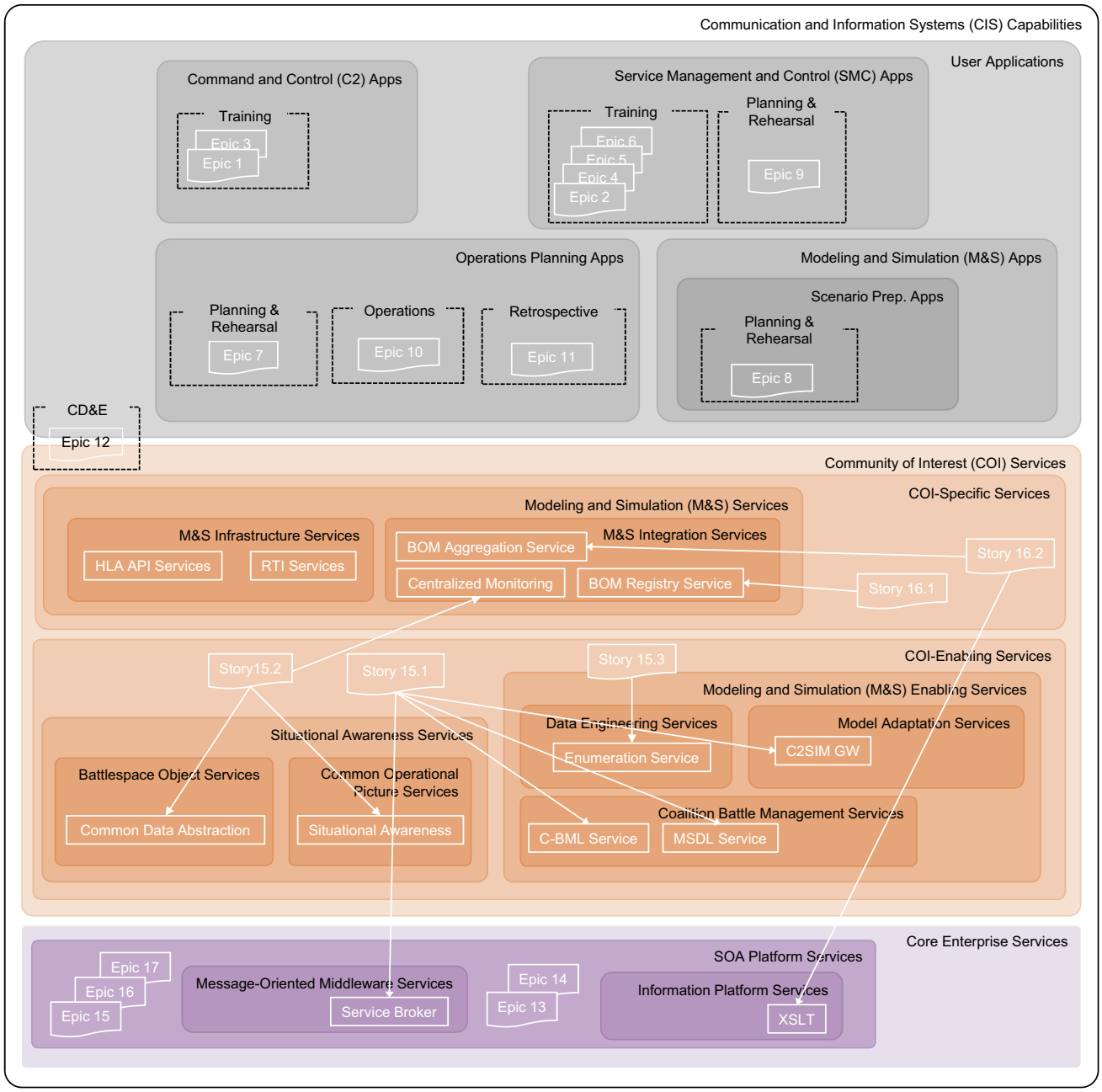


Fig. 14 Entry points for M&S COI development. *White arrows* indicate development in terms of elaboration and refinement

In general, repetition frequency in task exposure should be determined from an understanding of risk, where *risk* = *likelihood* x *consequence*, rather than on likelihood alone [49].

Epic 4: **As** training operator **I can** compose risk-based training regimes **by using** a SMC Application **to** (re-)combine C2 Applications and CTE Services and M&S Services **in order to** increase training capabilities by *T*.

For tactical decision-making, teaching any one particular task strategy has shown to be futile [32, 49]. Instead, the idea is to train adaptability, i.e. how to adapt to unknown and unpredictable situations. For this, the notion of *adaptive thinking* [45] has been adopted in the defence domain for tactical decision-making [49, 50].

Epic 5: **As** training operator **I can** compose training regimes for adaptive thinking **by using** a SMC Application **to** (re-)combine C2 Applications and CTE

Services and M&S Services **in order to** increase training capabilities by *T*.

Deliberate practice [24] is a framework which enables one to incorporate the above aspects. Its essence lies in immediate and tailored feedback followed by immediate tailored re-trials.

Epic 6: **As** training operator **I can** compose deliberate practice regimes **by using** a SMC Application **to** (re-)combine C2 Applications and CTE Services and M&S Services **in order to** increase training capabilities by *T*.

Simulation is essential for the use of artificial tasks, risk-based task repetition and deliberate practice regimes.

7.2 Operations planning and rehearsal

In contrast to training, which is a long-term activity, planning and mission rehearsal are a short-term activity for a specific operation. Traditionally, round-the-table manual war gaming has been used in planning and rehearsal. 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 enables the building of common mental models of the operation.

Epic 7: **As** operational personnel **I can** plan and prepare for missions **by using** Operations Planning Applications **to** employ simulation capabilities in planning **in order to** increase quality of plans by *Q* and increase common understanding by *C*.

The epic "SIM-Supported COA Applications Epic" (Sect. 6) can be placed as a sub-epic to Epic 7.

7.2.1 Scenario generation

To save time, mission rehearsal personnel should be able to construct simulation scenarios without help from technical personnel. By enabling direct scenario development, large-scale manpower savings can be achieved, and rehearsal objectives are more likely to be met [21].

Epic 8: **As** rehearsal coordination personnel **I can** construct simulation scenarios for the mission at at hand without the aid of M&S experts **by using** a Scenario Preparation Application **to** rapidly and readily orchestrate simulated entities from LVC simulations **in order to** improve mission rehearsal and planning support by *Y*.

7.2.2 Rapid composability

Once a scenario has been constructed, the simulations for running the scenario must be composed rapidly and accurately. According to [21], current simulation capabilities are more suited to 18-month planning cycles, rather than the 3–7 days allowed for rapid mission rehearsal. To achieve this, M&S solutions must become far more flexible and composable.

Epic 9: **As** mission rehearsal operator **I can** combine simulations and implement a scenario **by using** a SMC Application **to** (re-)combine C2 Applications and CTE Services and M&S Services readily and rapidly **in order to** reduce time to mission rehearsal readiness to *t* hours.

7.3 Operations

The capabilities enhanced by M&S in the above epics transfer to operations. Plans developed in planning phases could be given new parameters during operations to reflect actual events, and the plans re-simulated on this new knowledge. Replaying actual events in a fast-forward simulation 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 according to observations and extrapolate to evaluate future best course of action **by using** Operations Planning Applications **to** re-simulate plans on my system developed in operations planning **in order to** improve quality of situational awareness by *A*.

This allows plans to be updated and simulated frequently according to available information, across multiple battle management systems.

7.4 Retrospective analyses

Systematic and targeted post-operation analysis enables learning, even in the complex system that warfare constitutes. Replaying actual events and analysing 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** Operations Planning Applications **to** rerun recorded events and recombine and rerun plans based on recorded or alternative

events **in order to** improve learning from experience by *L*.

That is, we envision that the same applications can support operations planning, conducting operations and retrospective analyses. This demands applications that are flexible, composable and interoperable.

7.5 Concept development and experimentation

During a demonstration for the Army of a simulation system for planning support, the following insight was offered by the audience, which captures a key capability from service orientation and 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 would be an "emergent effect" [51] arising in a federation of systems, i.e. an effect that is not obvious from regarding the system a priori. By enabling increased flexibility, a loosely coupled portfolio might give rise to emergent effects, but it is also central in *planned* activities for experimentation due to increased possibilities for controlled experiments and other analytical and empirical studies [1].

Epic 12: **As** a user of the portfolio **I can** conduct studies on Communication and Information Systems (CIS) Capabilities **by using** Communication and Information Systems (CIS) **to** to explore new ways of using applications and services **in order to** improve innovation by *V*.

The "user" here covers any user of the portfolio, but required to be a "reflective practitioner" in the sense of [5, 6, 31], who may, or may not, be aided by researchers.

7.6 Back-end capability user stories

The User Applications epics above are ignorant to the details or status of SOA. However, it is clear that they lay demands on Back-End Capabilities layers of the C3 Taxonomy in terms of (re-)composing applications and services readily and rapidly, in time for, and even during, operational activities. These epics will employ solutions according to technological maturity.

The level of activity at technical layers is a function of the maturity of service orientation. At advanced levels of maturity, one can foresee end-users composing and orchestrating systems in a user application by dragging and dropping services from a cloud repository, with interoperability issues being automatically resolved behind the scenes. At today's modest level of maturity, programmers have to set up communication between systems manually, service enabling systems at need.

Since SOA is not in a state where services can be composed on the fly, software developers must invest considerable effort to realize these epics. Among the Core Enterprise Services, one must therefore find tools 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*.

Many of today's systems are stove piped or involve obese clients with very few building blocks in common. One cannot shut down a portfolio in transition phases, so systems must be packaged temporarily as application and service providers until "true" applications and services as promoted by SOA can be developed.

Epic 14: **As** software developer **I can** expose functionality as User Applications and Technical Services **by using** SOA Platform Services **to** build gateways and interfaces readily and rapidly **in order to** integrate legacy systems into the portfolio.

Our epics 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 Applications.

Epic 15: **As** software developer **I can** make M&S Applications interoperate with other User Applications **by using** SOA Platform Services **to** combine loosely coupled and interoperable M&S Services and M&S 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 Applications where M&S Services interoperate **by using** SOA Platform Services **to** combine loosely coupled and interoperable M&S services **in order to** enable Epics 1, 3, 8, 9, 13.

Some of the epics request applications (tentatively placed in the SMC Applications and Scenario Preparation Applications categories) that 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.

Epic 17: **As** software developer **I can** develop User Applications that plug in known COI-Specific Services at runtime **by using** SOA Platform Services **to**

discover and combine available COI Services **in order to** enable Epics 8, 9.

The epics in this section illustrate the need to start epics at the Technical Services layers.

We now discuss further implications of Epics 15 and 16, because they represent two essential technical challenges: Loosely coupled and interoperable software systems at the User Applications level, and loosely coupled and interoperable M&S components at the Community of Interest (COI) Services level.

7.6.1 Interoperability between user applications

The decoupling prescribed by SOA necessitates standardized communication of data and service requests. SOA does not prescribe any specific technology for this, but there are several industry standards in use, one being the Web Services (WS-*) suites of standards [62].

The simulation community has its own interoperability standards. The High-Level Architecture (HLA) [30] is an architecture standard for distributed simulation systems [33, 44]. It enables practices for loose coupling and interoperability. In HLA, the main software modules that make up a simulation system are termed federates, which are combined (federated) to form a HLA federation, coordinated by a runtime infrastructure (RTI). The federates must adhere to a federation object model (FOM) which declares the data structure of objects and interactions shared between federates.

There are standards for communicating with a HLA federation. The Coalition Battle Management Language (C-BML) [53] is a formal language for expressing a commander's plans, orders and reports across C2IS, simulation systems and autonomous systems. The Military Scenario Definition Language (MSDL) [52] is a standard for describing scenarios, e.g. for initializing simulation systems. Present C-BML and MSDL implementations are XML-based, and a Web Service-based broker offers post/retrieve services on C-BML messages.

Several projects have explored the viability of linking simulation federations with C2IS using the standards above. In one project [46], a scripted BML (SBML) server provided services for MSDL initialization and exchange of C-BML battle orders and reports. Legacy systems were service enabled using gateways. This uncovers requirements which can be formulated as a story under Epic 15.

> Story 15.1: **As** software developer **I can** increase loose coupling and interoperability of M&S Applications and C2 Applications **by using** C2SIM gateways, MSDL Service, C-BML Service **to** service-enable M&S Applications and C2 Applications and provide common views of data and common modes of communication **in order to** enable Epic 15.

Story 15.1 is an example of elaboration and refinement that penetrates layers in the taxonomy (Sect. 4.7).

Another project [2, 3, 13] developed an interoperability layer to combine two federations with widely different FOMs and time management [3]. A service offered centralized monitoring of the status of the federations, and a second service offered common situational awareness for all entities in the simulations that could also be consumed and displayed in Google Earth. To support these services, common data abstraction services maintained the minimal set of data required to share object state across all entities [3].

> Story 15.2: **As** software developer **I can** enable coordination of diverse M&S Applications **by using** Situational Awareness Service, Centralized Monitoring Service, Common Data Abstraction Services **to** maintain the minimal set of data required to share object state across entities in the M&S Applications **in order to** enable Epic 15.

So-called enumerations pertain to the identity and representation of entities in simulations, and problems arise when different systems use different enumeration schemes. An enumeration service was set to offer common methods for data producers and consumers to determine how to translate from native representations to common data abstraction representations.

> Story 15.3: **As** software developer **I can** build Common Enummeration Gateways **by using** Enumeration Service **to** determine how to translate from native representations to common data abstraction representations **in order to** enable Epic 15.

This illustrates inter-M&S system coordination offered as services. The main ideas concern factoring out common functionality; see [19, 20] for more details.

7.6.2 Interoperability between simulation services

Federates in a HLA federation enjoy interoperability and loose binding with respect to the federation they participate in. How service-oriented are HLA federates? HLA Base Object Models (BOMs) are a way to provide interfaces 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 [10, 11, 39, 54]. In this way, federates can provide services in any federation as long as the BOMs are consistent with the required FOM. We can state the following elaboration and refinements of Epic 16:

> Story 16.1: **As** software developer **I can** develop HLA federates as FOM-specific M&S Services providers **by using** BOM Repository Service **to** find and generate functionality according to BOM interfaces **in order to** enable Epic 16.
>
> Story 16.2: **As** software developer **I can** compose HLA federations **by using** BOM Aggregation Service and XSLT Services **to** aggregate BOM interfaces and generate corresponding FOM **in order to** enable Epic 16.

The cases in Sects. 5 and 6 represent two development projects in a portfolio, while the case in Sect. 7 shows entry points for several projects. The requirements in the three cases have obvious shared elements. Our method framework allows stakeholders and developers to describe and coordinate requirements across cases (projects) in a shared common format.

8 Conclusion

To meet both architectural and managerial challenges in systems portfolio development activities, we propose a method framework designed with emphasis on:

- Facilitating the definition of small manageable projects
- Facilitating cross-project evaluation, communication and prioritization
- Facilitating the design and development of persistent capabilities

The framework is intended to be simple. It is based on five overarching principles for work as guidances in the midst of project and portfolio activities.

The proposed framework augments current best practices in agile management and development. It is front-ended by a capability taxonomy. Such taxonomies can be used to structure requirements elaboration and refinement. Elaborate-refine trees that go horizontally inward in taxonomy levels are key to retaining focus on a given level and support the service-oriented idea of developing capabilities that are persistent with respect to its surroundings and with respect to implementations. The stratification of such taxonomies can help discipline development into smaller manageable sub-projects. Key elements of the proposed framework are user stories as product elements with both benefit and cost estimates, a user story syntax which can be used to ensure cohesive story lines through a loosely coupled systems portfolio, and the idea that elaborate-refine trees can start at any level, initiating a relatively independent development process with a focus on the domain of that level. The agile principles of balancing planning and development, together with capability and service orientation, should diminish the barriers of massive ahead-planning and provide practical starting points for activity that many portfolio projects lack.

We have applied elements of the method framework and found that it clarified definition, scope and areas of responsibility. We outlined starting points for development for the modelling and simulation community of interest as an example of initiating portfolio development for one leaf of a large systems portfolio. The endeavour of applying the method framework in full has to be undertaken by a portfolio development management committed to leveraging the method consistently in the organization, together with committed stakeholders.

Empirical studies (case studies, action research, controlled experiments) must be conducted to evaluate, adjust and elaborate the framework outlined here. Those features that we claim for our method framework by design are obvious candidates for empirical evaluation. However, we emphasize that having shared notions of both methodology and product [25, 36, 37] is extremely important in large projects, perhaps more important than exactly what method is used. We propose that our method is a framework that by design benefits shared mental models of process and product and that further studies might start with this claim.

References

1. Alberts DS, Hayes RE (2005) Campaigns of experimentation: pathways to innovation and transformation. Information age transformation series, DoD command and control research program
2. Allen GW, Lutz R, Richbourg R (2010) Live, virtual, constructive, architecture roadmap implementation and net-centric environment implications. ITEA J 31(3)
3. Allen GW, Schroeder L (2011) Utilization of service oriented architecture (SOA)-based commercial standards to address live, virtual, constructive (LVC) interoperability challenges. In: Proceedings of interservice/industry training, simulation, and education conference (I/ITSEC) 2011. National Training and Simulation Association
4. Archer NP, Ghasemzadeh F (1999) An integrated framework for project portfolio selection. Int J Proj Manag 17(4):207–216
5. Argyris C (1993) Knowledge for action. Jossey-Bass Publishers, San Franscisco
6. Argyris C, Schön DA (1996) Organizational learning II. Theory, method, and practice. Addison-Wesley Publishing Company Inc, Boston
7. Benestad HC, Hannay JE (2011) A comparison of model-based and judgment-based release planning in incremental software projects. In: Proceedings of 33rd international conference on software engineering (ICSE 2011). ACM, New York, pp 766–775

8. Bloebaum TH, Hannay JE, Hedenstad O-E, Haavik S, Lillevold F (2013) Architecture for the Norwegian defence information infrastructure (INI)—remarks on the C3 Taxonomy. FFI-rapport 2013/01729, Norwegian Defence Research Establishment (FFI)
9. Bruvoll S, Hannay JE, Svendsen GK, Asprusten ML, Fauske KM, Kvernelv VB, Løvlid RA, Hyndøy JI (2015) Simulation-supported wargaming for analysis of plans. In: NATO modelling and simulation group symposium on M&S support to operational tasks including War Gaming, Logistics, Cyber Defence (MSG-133)
10. Chase T, Gustavson P (2005) RPR-BOM initiative: Providing a set of applicable BOMs to the M&S community. In: Proceedings of 2005 spring simulation interoperability workshop (SIW). Simulation Interoperability Standards Organization (SISO)
11. Chase T, Gustavson P, Root L (2006) From FOMs to BOMs and back again. In: Proceedings of 2006 spring simulation interoperability workshop (SIW). Simulation Interoperability Standards Organization (SISO)
12. Cohn M, Martin R (2005) Agile estimating and planning. Prentice Hall, New Jersey
13. Coolahan JE, Allen GW (2012) LVC architecture roadmap implementation–results of the first two years. In: Proceedings 2012 spring simulation interoperability workshop (SIW). Simulation Interoperability Standards Organization (SISO)
14. Cooper RG, Edgett SJ, Kleinschmidt EJ (1998) Portfolio management for new products. Perseus Books, New York
15. Danesh MH, Yu E (2015) Analyzing IT flexibility to enable dynamic capabilities. In: Advanced information systems engineering workshops, volume 215 of lecture notes in business information processing. Springer, New York, pp 53–65
16. De Reyck B, Grushka-Cockayne Y, Lockett M, Calderini SR, Moura M, Sloper A (2005) The impact of project portfolio management on information technology projects. Int J Proj Manag 23(7):524–537
17. Denne M, Cleland-Huang J (2003) Software by numbers: low-risk, high-return development. Prentice Hall, New Jersey
18. Denne M, Cleland-Huang J (2004) The incremental funding method: data-driven software development. IEEE Softw 21(3):39–47
19. Drake DL, Martins IX, Roca RA, Carr F (2011) 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
20. Drake DL, Morse KL (2012) Use of SOA for distributed simulation: a way forward. In: Proceedings of 2012 spring simulation interoperability workshop (SIW). Simulation Interoperability Standards Organization (SISO)
21. Edgren MG (2012) Cloud-enabled modular services: a framework for cost-effective collaboration. In: NATO modelling and simulation group symposium on transforming defence through modelling and simulation-opportunities and challenges (MSG-094)
22. Elonen S, Artto KA (2003) Problems in managing internal development projects in multi-project environments. Int J Proj Manag 21:395–402
23. Erdogmus H (2003) Let's scale agile up. Agile Times 2(1):6–7
24. Ericsson KA (2006) The influence of experience and deliberate practice on the development of superior expert performance. In: Ericsson KA, Charness N, Feltovich PJ, Hoffman RR (eds) The Cambridge handbook of expertise and expert performance, chapter 38. Cambridge University Press, Cambridge, pp 683–703
25. Hannay JE, Benestad HC (2010) Perceived productivity threats in large agile development projects. In: Proceedings of 4th international symposium on empirical software engineering and measurement (ESEM). IEEE Computer Society, New York, pp 1–10
26. Hannay JE, Brathen K, Hyndøy JI (2015) On how simulations can support adaptive thinking in operations planning. In NATO modelling and simulation group symposium on M&S support to operational tasks including War Gaming, Logistics, Cyber Defence (MSG-133)
27. Hannay JE, Brathen K, Mevassvik OM (2013) Simulation architecture and service-oriented defence information infrastructures—preliminary findings. FFI-rapport 2013/01674, Norwegian Defence Research Establishment (FFI)
28. Hannay JE, Mevassvik OM, Skjeltorp A, Brathen K (2014) Live, virtual, constructive (LVC) simulation for land operations training: concept development & experimentation (CD&E). In NATO modelling and simulation group symposium on integrating modelling and simulation in the defence acquisition lifecycle and military training curriculum (MSG-126)
29. Hannay JE, Skjeltorp A, Holen JE, Blix JE, Mevassvik OM, Olafsen HK (2014) Live, virtual, constructive (LVC) simulation for land training: system description and technical evaluation. FFI-rapport 2014/01597, Norwegian Defence Research Establishment (FFI)
30. IEEE Standards Association. 1516–2010 - IEEE Standard for modeling and simulation (M&S) High Level Architecture (HLA). http://standards.ieee.org/findstds/standard/1516-2010.html, 2010. Accessed Sept 2012
31. Jarvis P (1999) The practitioner-researcher. Jossey-Bass Publishers, New Jersey
32. Klein G (1997) Developing expertise in decision making. Think Reason 3(4):337–352
33. Kuhl F, Weatherly R, Dahmann J (1999) Creating computer simulations—an introduction to the high level architecture. Prentice Hall PTR, Upper Saddle River
34. Lang B, Gerz M, Meyer O, Sim D (2011) An enterprise architecture for the delivery of a modular interoperability solution. In: Semantic and Domain-based Interoperability: Proceedings of RTO information systems technology panel (IST) symposium. NATO Research and Technology Organisation
35. Lee JW, Kim SH (2001) An integrated approach for interdependent information system project selection. Int J Proj Manag 19(2):111–118
36. Levesque LL, Wilson JM, Wholey DR (2001) Cognitive divergence and shared mental models in software development project teams. J Organ Behav 22:135–144
37. Mathieu JE, Goodwin GF, Heffner TS, Salas E, Cannon-Bowers JA (2000) The influence of shared mental models on team process and performance. J Appl Psychol 85(2):273–283
38. McAfee A (2012) When too much IT knowledge is a dangerous thing. MIT Sloan Management Review
39. Möller B, Gustavson P, Lutz R, Löfstrand B (2007) Making your BOMs and FOM modules play together. In: Proceedings of 2007 fall simulation interoperability workshop (SIW). Simulation Interoperability Standards Organization (SISO)
40. NATO Communications and Information Agency. The C3 Taxonomy. http://www.act.nato.int/article-8a, 2015. Accessed Jan 2015
41. NATO Consultation, Command and Control Board. NATO Architecture Framework Version 3. http://www.nhqc3s.nato.int/ARCHITECTURE/_docs/NAF_v3/ANNEX1, 2007. Accessed Jan 2013
42. North Atlantic Treaty Organization 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. 2012
43. Oberndorfner M, van Gest J (2011) Modelling and simulation events enabled by the distributed networked battle labs

framework. In: NATO modelling and simulation group symposium on enhance or replace: finding the right live vs. synthetic balance (MSG-087)
44. Petty MD, Gustavson P (2012) Combat modeling with the high level architecture and base object models. In: Tolk A (ed) Engineering principles of combat modeling and distributed simulation, chapter 19. Wiley, New York, pp 413–448
45. Pulakos ED, Arad S, Donovan MA, Plamondon KE (2000) Adaptability in the workplace: development of a taxonomy of adaptive performance. J Appl Psychol 85(4):612–624
46. Pullen JM, Corner D, Brook A, Wittman R, Mevassvik OM, Alstad A (2012) MSDL and C-BML working together for NATO MSG-085. In: Proceedings of 2012 spring simulation interoperability workshop (SIW). Simulation Interoperability Standards Organization (SISO)
47. Sandkuhl K, Koc H (2014) On the applicability of concepts from variability modelling in capability modelling: experiences from a case in business process outsourcing. In: Advanced information systems engineering workshops, volume 178 of lecture notes in business information processing. Springer, New York, pp 65–76
48. Schwaber K (2004) Agile project management with scrum. Microsoft Press, New York
49. Shadrick SB, Lussier JW (2009) Training complex cognitive skills: a theme-based approach to the development of battlefield skills. In: Ericsson KA (ed) Development of professional expertise, chapter 13. Cambridge University Press, Cambridge, pp 286–311
50. Shadrick SB, Lussier JW, Hinkle R (2005) Concept development for future domains: a new method for knowledge elicitation. Technical report 1167, US Army Research Institute for the Behavioral and Social Sciences
51. Simon HA (1996) The sciences of the artificial, 3rd edn. MIT Press, New York
52. Simulation Interoperability Standards Organization. Standard for: Military Scenario Definition Language (MSDL). http://www.sisostds.org/DigitalLibrary.aspx?Command=Core_Download&EntryId=30830, 2008. Accessed Aug 2012
53. Simulation Interoperability Standards Organization. Standard for Coalition Battle Management Language (C-BML) phase 1. http://www.sisostds.org/DigitalLibrary.aspx?Command=Core_Download&EntryId=41767, 2014. Accessed July 2014
54. Sisson B, Gustavson P, Crosson K (2006) Adding aggregate services to the mix: an SOA implementation use case. In: Proceedings of 2006 spring simulation interoperability workshop (SIW). Simulation Interoperability Standards Organization (SISO)
55. Sliger M, Broderick S (2008) The software project manager's bridge to agility. Addison Wesley, Boston
56. Sulaiman T, Barton B, Blackburn T (2006) AgileEVM—earned value management in scrum projects. In: Proceedings of IEEE AGILE 2006. IEEE Computer Society, New York, pp 7–16
57. Takeuchi H, Nonaka I (1986) The new new product development game. Harvard business review, pp 137–146
58. The open group. SOA reference architecture technical standard. https://www2.opengroup.org/ogsys/catalog/C119, 2011. Document number C119
59. The open group. TOGAF Version 9.1 Enterprise Edition–an introduction. http://pubs.opengroup.org/architecture/togaf9-doc/arch/, 2011. Accessed Jan 2013
60. Tolk A (2012) Integration of M&S solutions into the operational environment. In: Tolk A (ed) Engineering principles of combat modeling and distributed simulation, chapter 15. Wiley, New Jersey, pp 295–327
61. Tolk A (2012) Terms and application domains. In: Tolk A (ed) Engineering principles of combat modeling and distributed simulation, chapter 4. Wiley, New Jersey, pp 55–78
62. World Wide Web consortium. Web services architecture. http://www.w3.org/2002/ws/arch, 2004. Accessed Sept 2012
63. Zdravkovic J, Pastor O, Loucopoulos P (2015) Preface to the 2nd int'l workshop on advances in services DEsign based on the notion of CApabiliy—ASDENCA 2015. In: Advanced information systems engineering workshops, volume 215 of lecture notes in business information processing. Springer, New York, pp 53–65