Cover Page

Revised Draft for Review Only
(last edit: 20 September 2017)

Title: Link-Autonomy and Autonomics for Resilient Cross-Layer Communications Services

Primary Topic: Interoperability/Integration and Security
Alternate: Methodological Development, Experimentation, Analysis, Assessment and Metrics

Authors: Jayson Durham, SSC Pacific, San Diego, CA 92152-5001
Joan Kaina, SSC Pacific, San Diego, CA 92152-5001
Jason Landsborough, SSC Pacific, San Diego, CA 92152-5001
Minh Vuong, SSC Pacific, San Diego, CA 92152-5001
Zeeshan Barkatullah, SSC Pacific, San Diego, CA 92152-5001
Ryan Gabrys, SSC Pacific, San Diego, CA 92152-5001

Point of Contact: Jayson Durham, SSC Pacific, Code 56150, San Diego, CA 92040
(email) jayson.durham@navy.mil; (office) 619-553-2344; (cell) 619-971-9619

Keywords: Network Integration and Interoperability (Network I&I); Model-Based Systems Engineering (MBSE); System of Systems Engineering (SOSE); Autonomic Computing

Link-Autonomy and Autonomics for Resilient Cross-Layer Communications Services

Jayson Durham, Joan Kaina, Jason Landsborough, Minh Vuong, Zeeshan Barkatullah, Ryan Gabrys
SSC Pacific, San Diego, CA 92152-5001

Point of Contact: Jayson Durham, SSC Pacific, Code 56150, San Diego, CA 92040
(email) jayson.durham@navy.mil; (office) 619-553-2344; (cell) 619-971-9619

Keywords: Network Integration and Interoperability (Network I&I); Model-Based Systems Engineering (MBSE); System of Systems Engineering (SOSE); Autonomic Computing

Abstract: For supporting distributed network-centric operations, the development of a cloud-based experimentation framework has continued to be explored and further developed. The overall objective of the experimentation framework focuses on developing reliable, robust, and resilient tactical Intelligence, Surveillance, and Reconnaissance (ISR) communications and networking support capabilities that can be trusted to operate within a variety of adverse conditions. Thus, a cloud-computing oriented experimentation framework is being further explored and developed for supporting ongoing model-based system-of-systems engineering (MBSE/SOSE) experimentation capabilities.

Current work focuses on the utilization of autonomic computing frameworks (e.g. Rainbow) for developing and assessing a variety of strategies (i.e. probes, effectors, decision logic) that enable wireless line-of-sight (LOS) data-links to adapt to adverse conditions that may impact the ability to meet information exchange requirements for respective net-centric distributed operations.

Within the context of autonomic computing technologies and respective adaptation strategies, the work discussed in this paper concurrently addresses the need for extending and augmenting such adaptation services to include model-based integration of emerging standardized architectures/frameworks, open source resources, and their respective source-level build capabilities. For example, the work described herein focuses on how a growing assortment of commercial-off-the-shelf and government-off-the shelf (COTS/GOTS) resources, which includes the Rainbow framework, can support self-adaptive cross-layer cross-platform content delivery/distribution networking (CDN) and respective endpoint-management requirements within the context of emerging data-modeling and enterprise-computing standards. Throughout the paper, footnotes are included to assist readers with more specialized and domain-specific topics.[a]

[a] Note to the reader: Technical references are enclosed in square brackets and listed at the end of the paper. Additionally, superscripted endnotes are provided to help facilitate an introductory understanding of concepts and topics that may be more common in one technology domain but otherwise not familiar or as applicable to the population at large. For accessing links and enlarging graphics as desired or needed, viewing a soft-copy on a large screen is recommended.

1. Introduction and Executive Summary

1.1 High-Level Introduction and Overview

As has been highlighted for a government-wide inter-agency initiative, called the Information Sharing Environment (ISE), there is an ongoing cross-cutting objective to "provide the right information to the right stakeholder at the right place and time" [1].[1] Such information-sharing capabilities are especially critical for Information Dominance and Network Centric Warfare (ID/NCW).[2]

Much progress has been made towards an evolutionary transformation to an operational information technology infrastructure (ITI) framework that addresses emerging concerns while supporting ID/NCW capabilities. Within this context of maturing ITI framework capabilities, there are a growing number of challenges that include both the need for addressing functional requirements for improved functional capabilities, while also addressing the non-functional requirements (e.g. "-ilities") that help to further improve both systems-engineering and enterprise-engineering support, responsiveness, and agility.[3]

A key consideration for ID/NCW is the model-based design, configuration, and provisioning of information-sharing services specifically tailored, managed, and optimized to support well-informed maneuvering of geographically-dispersed forces.[4] Other network-centric initiatives and methodologies share similar characteristics (e.g. network-enabled capability, network-centric organization).[5] Thus, information and communications technology (ICT) and unified communications are key enablers for providing critical infrastructure components for information-sharing and related ID/NCW services.[6]

An ongoing focus of work, as described herein, is to further explore and survey applicable technologies, reference models, architectures, resources, and capabilities (e.g. cloud-based distributed computing).[7] Of particular interest are models and resources that can be utilized for incremental continuous-improvement of a persistent converged-infrastructure with standardized generically-defined shared-services.[8] The goal is to provide a model-based system-of-systems engineering (MBSE/SOSE)[9] oriented approach that can help to better identify/discover, characterize, and minimize the operational impact of such ID/NCW related events that can degrade ID/NCW ITI frameworks and cause Denied – Disrupted/Disconnected, Intermittent, and Low-bandwidth (D-DIL) conditions which impact overall mission performance.

Thus, such applicable models, methods, and techniques help to identify, represent, and optimize the respective value-chains and enable performance engineering capabilities that address both functional and non-functional requirements.[10] Application performance engineering is particularly of interest, due to the ongoing focus on methods "to develop and test application performance in various settings, including mobile computing, the cloud, and conventional information technology (IT)".[11]

Figures 1 and 2 highlight the type of ITI framework components and respective communication services that have been the focus of our ongoing work. Of particular interest are long-haul radio frequency (RF) and microwave line-of-sight (LOS) components of land-air-sea data-links deployed within an area of responsibility (AOR) and operating within a distributed ITI framework (i.e. ad-hoc network of airborne/surface networking-backbone and tactical-edge nodes).[12] As highlighted by the upper block of the diagram in figure 1 (from [2]), such infrastructure components focus on the Internet Protocol (IP) based "off-board communication links" that support LOS platform-to-platform communication and networking capabilities.[13]

Figure 2, from the Joint Communication Simulation System (JCSS) home page, highlights another example GOTS/COTS resource.[14] As highlighted, JCSS enables stakeholders to assess performance, relative to operational scenarios and expected loads (i.e. traffic generation models).[15] Note that existing data-transport and data-synchronization oriented tools, such as JCSS, typically have limited support for mission scenarios that include potential D-DIL related events and associated degradation of the NCW/ID infrastructure that may impact overall operations. Furthermore, the incorporation of integration and interoperability (I&I) and cloud-based ITI framework challenges, are typically addressed to a somewhat limited extent.[16] Such emerging I&I and ITI oriented challenges are the focus of the ongoing work described herein. In particular, this survey/overview and supporting discussion works to address the question of how to best develop a mission-driven experimentation-based assessment/validation framework that includes the respectively applicable use-case oriented measures of performance/effectiveness (MOP/MOE) metrics/indicators and enables continuous improvement of ITI framework services.[17]

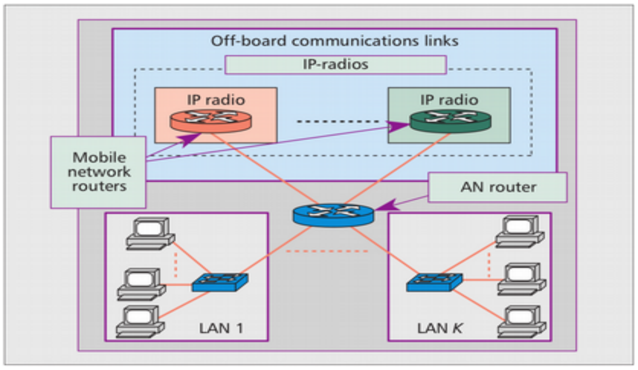


Fig. 1 Platform Architecture: ALN Example [2]

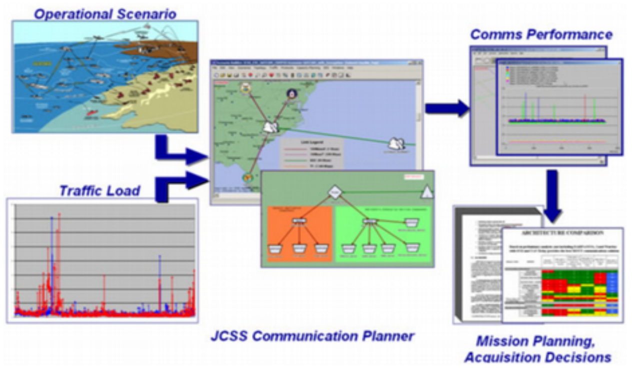

Operational Scenario
Comms Performance
Traffic Load
JCSS Communication Planner
Mission Planning, Acquisition Decisions

Fig. 2 Operational Workflow: JCSS Example [endnote 14]

1.2 Outline of Paper and Executive Summary

The overarching goal of this ongoing research effort is to incrementally leverage the longstanding ongoing trends toward service-oriented enterprise-computing and subsequent virtualization of system functions (e.g. abstraction, encapsulation, virtual function, generic function, function overloading, late/dynamic binding, duck typing, multiple dispatch, dynamic dispatch).[18] This trend towards virtualization is complicated by the various types of capabilities and methods that support the decoupling of implementation details from more application-specific and hardware-specific constraints. A variety of hypervisors and virtual machine (VM) managers provide a broader scope and basis for hardware virtualization.[19] Network functions virtualization (NFV), which typically includes network virtualization and input/output (I/O) virtualization, further incorporates software-defined networking (SDN) and related technologies (e.g. software-defined environments).[20] Thus, the need to continue to explore applicable emerging technologies, best practices, and standards for developing experimentation frameworks, with a particular focus on providing reliable, robust, and resilient tactical-ISR communications and networking support capabilities that can be trusted to operate within a variety of operating conditions.

As highlighted within the following sections, much progress has been made within a range of domains and areas of specialization (i.e. domains of discourse), wherein each separately addresses various aspects of the associated ID/NCW ITI science and technology (S&T) needs.[21]

Section two provides background information that helps to provide a working introduction to the need and associated payoff for developing the type of capabilities described herein. Related work with multi-tier goal-oriented workflow management is covered within this context of ISR mission-support.

Within section two, complementary connectivity and link management efforts are also discussed in some detail. For supporting D-DIL conditions within challenging environments, various aspects of inter-related command and control (C2) data-synchronization services (C2SS) and directional line-of-sight (LOS) ad-hoc networking (e.g. connectivity/link management) efforts are surveyed and reviewed.

Section three covers standardized architectures and frameworks within both application and technology domains. For application-domain related architectures, a full motion video (FMV) example highlights the incorporation of enterprise service bus (ESB) reference architectures for coordinating and orchestrating FMV dataflow between producer-consumer endpoints. Within this context of ESB architectures, examples of adaptive ESB architecture frameworks are introduced and discussed. Of particular interest are the logical architecture and associated adaptation-strategy mechanisms.

With this initial introduction to adaptive enterprise computing components (e.g. adaptive ESB frameworks), self-optimizing enterprise architecture frameworks such as software defined networks (SDNs) and software defined environments (SDEs) are also discussed to further emphasize the emerging trend towards highly-virtualized service-oriented cloud-computing infrastructures. The continuous-optimization and continuous-assurance (i.e. enterprise services) capabilities of emerging SDN/SDE architecture frameworks are also covered to highlight the emerging trend towards intrinsically adaptive, self-healing, and resilient/robust enterprise infrastructures. Finally, within section three, autonomic-computing models and associated frameworks, such as Rainbow, are introduced and discussed within this context of adaptive enterprise computing.

Overall, the combination of a standardized model-based information and communications technology (ICT) infrastructure (e.g. enterprise architecture, service oriented architecture, cloud computing), with the additional capabilities of self-awareness, adaptivity, and self-healing (e.g. autonomic computing), creates an order of complexity that includes not only the breadth and depth of the respective technologies, but also the socio-technical challenges of cross-disciplinary communication, alignment, cooperation, and collaborative life-cycle support.

Section four covers metrics, indicators, and capability assessment models, within applicable technology domains. Examples from wireless networking and quality of service (QoS) focused best-practices and standards are covered within this section.

Section five covers emerging technologies and architectures for self-adaptive and self-optimizing systems. Applicable streaming-video and wireless networking examples are highlighted and discussed. An example of a self-optimizing wireless-networking architecture that focuses on the multi-objective optimization of inter-related objectives (e.g. coverage, quality of service, throughput, latency/delay, energy efficiency, cost), is noted as particularly applicable to the focus of the work discussed herein.

Section six reviews applicable examples of risk management, reliability, and safety-engineering: models, standards, and best practices. Within this section, a number of standardized processes are highlighted, with a particular emphasis on the opportunity to re-purpose and tailor/augment such standards for development/certification of adaptive LOS data-links that demonstrate link-autonomy.

Section seven provides a high-level review of example change and adaptation models that focus on particular features of interest for developing link-autonomy: (i) design-pattern oriented approaches for developing adaptation strategies and tactics; (ii) latency-awareness oriented methods and techniques that address timing, coordination, and proactive adaptation needs.

Section eight further discusses the various aspects of technology domains and respective examples that were reviewed and surveyed within the respective sections. The need for adoption and tailoring of the surveyed models, architectural frameworks, standards, and best practices, is also discussed. The ongoing need for conceptual-level I&I models and standardized practices are also discussed.

Sections nine and ten respectively provide a summary/conclusion and short discussion of future work. The remainder of the paper includes sections for acknowledgements and a list of technical references.

2. Background: Previous/Ongoing Tactical-ISR Support Efforts

2.1 Multi-Tier Goal-Oriented Workflow Management

The diagram in figure 3 illustrates how generic baseline performance indicators (e.g. key performance indicators - KPIs) can be associated with standardized baseline widgets to develop mission-specific business activity monitoring resources (e.g. dashboards) tailored to a specific stakeholder context.[22] Such constructs can, in principle, have baseline default configurations for DoD Architecture Framework (DoDAF) activities, standardized mission tasks (e.g. Universal Joint Task List – UJTL), mission essential task lists (METLs), and associated master scenario event lists (MSELs).[23]

Figure 4 illustrates an adapted version of the generic Goal, Question, Indicator, Metric (GQIM) process that is a variant of the Goal Question, Metric (GQM) methodology.[24] The diagram has been tailored to illustrate the applicability to ID/NCW oriented applications. GQM, GQIM, and more recent variants, such as GQM+Strategies, are examples of methods that provide concepts and actionable steps for creating the links between goals, objectives, workflow, and measurement-based decision-making.[25] Thus, baseline templates can be developed and deployed as stakeholder-specific dashboard elements.

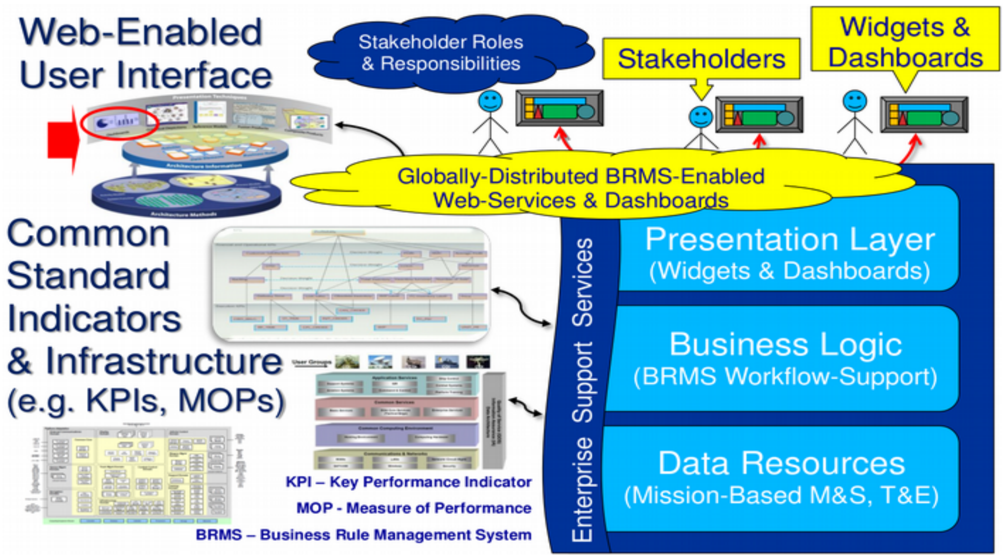

Web-Enabled User Interface
Stakeholder Roles & Responsibilities
Stakeholders
Widgets & Dashboards
Globally-Distributed BRMS-Enabled Web-Services & Dashboards
Common Standard Indicators & Infrastructure (e.g. KPIs, MOPs)
Enterprise Support Services
Presentation Layer (Widgets & Dashboards)
Business Logic (BRMS Workflow-Support)
Data Resources (Mission-Based M&S, T&E)
KPI – Key Performance Indicator
MOP - Measure of Performance
BRMS – Business Rule Management System

Fig. 3 Standardized Dashboards and Widgets: Example

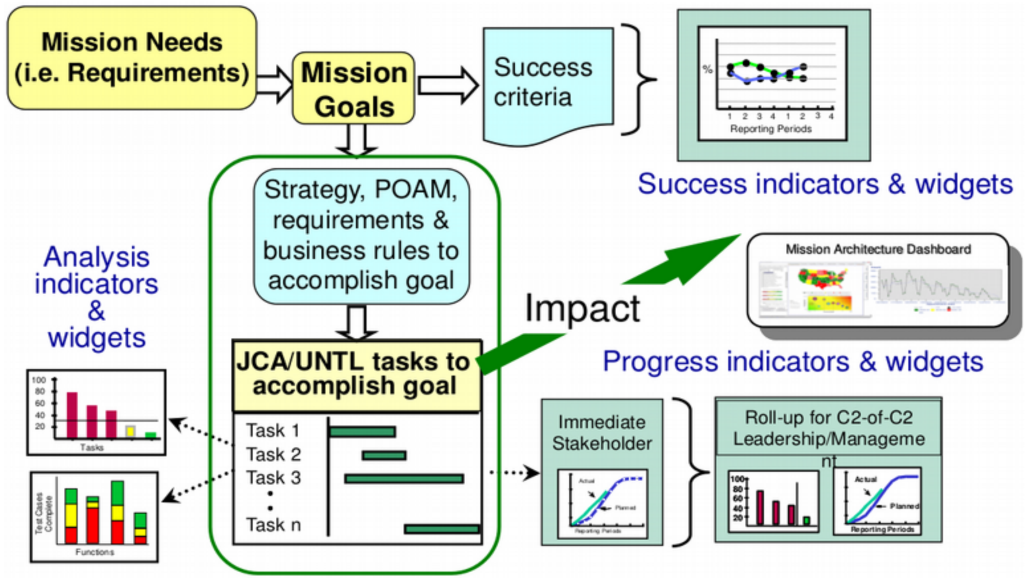

Mission Needs (i.e. Requirements)
Mission Goals
Success criteria
Reporting Periods
Success indicators & widgets
Strategy, POAM, requirements & business rules to accomplish goal
Analysis indicators & widgets
Impact
Mission Architecture Dashboard
Progress indicators & widgets
JCA/UNTL tasks to accomplish goal
Task 1
Task 2
Task 3
Task n
Tasks
Test Cases Complete
Functions
Immediate Stakeholder
Roll-up for C2-of-C2 Leadership/Management
Actual
Planned
Reporting Periods

Fig. 4 Standardized Widgets and Indicators: Example (POAM – Plan of Action and Milestones; JCA – Joint Capability Area; UNTL - Universal Naval Task List)

Furthermore, such mission-thread performance metrics and indicators can be dynamically tailored to better reflect and support D-DIL variants of typical mission-execution profiles. This type of business-logic based workflow-support capability has also continued to mature. For example, rule-based management systems (RBMS) represent such business-logic as business-rules within enterprise architecture or service oriented architecture (EA/SOA) business-rule engine resources (e.g. Wildfly, Drools).[26] Improved capabilities for semantic interoperability, content management, lexicon services, and advanced wiki-style user-interfaces, are examples of rapidly maturing technologies that further enable dynamically-adaptive approaches [3-6].[27] Thus, applicable parameters of MOPs, MOEs, and related performance assessment models can be represented, managed, and dynamically tailored during the execution of the respective mission-threads and scenarios. The development of this type of capability is an ongoing S&T objective of this effort.

2.2 Data-Synchronization Services

Figures 5 and 6 are diagrams from the command and control (C2) synchronization services (C2SS) research and exploratory development effort [7]. Figure 5 illustrates the concept of using dependency-injection, adapter, and other applicable patterns for in-line insertion and weaving of C2SS aspects and respective functionality.[28] In principle, this includes insertion of context-dependent functions that help to continuously improve data-synchronization capabilities. Note that aspect-oriented software development and programming techniques were originally developed to address the need to logically-encapsulate (i.e. virtualize) cross-cutting areas of concern and functionality (e.g. reporting, data logging).[29] For service-oriented architectures/models, this translates into conceptually managing similar types of methods (e.g. data synchronization) that may be associated with a variety of ITI services.

Figure 6 illustrates how functional data paths can be traced from the mission-threads and activities of a given mission-area, to the specific widgets that provide the necessary information to the respective stakeholders. This type of dataflow oriented mapping of information-support requirements can be utilized for tracing information-exchange requirements of the respective widgets.

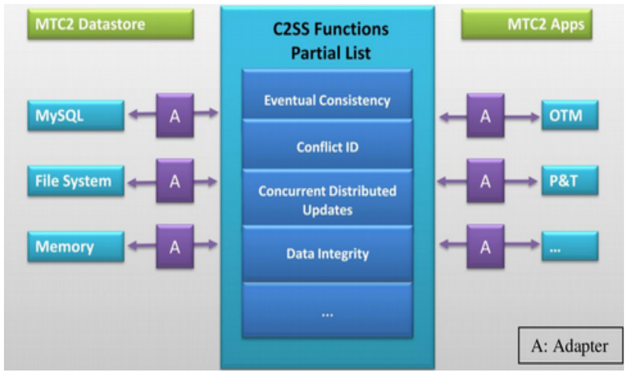


Fig. 5 C2 Data-Synchronization Services (C2SS) [7]

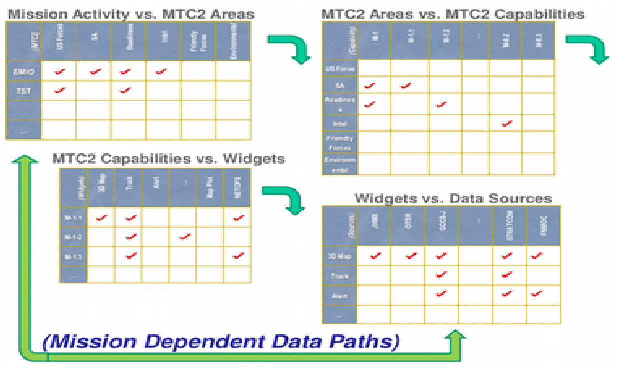

Mission Activity vs. MTC2 Areas
MTC2 Areas vs. MTC2 Capabilities
MTC2 Capabilities vs. Widgets
Widgets vs. Data Sources
(Mission Dependent Data Paths)

Fig. 6 Dataflow and Data-Path Mapping/Tracing [7]

Overall, C2SS types of services are intended to augment existing ITI frameworks with data-synchronization functions that resolve capability gaps and S&T challenges. Due to the open number of possible types of mission-execution environments, different types of solution implementations are anticipated and expected. Thus, as highlighted, the definition and use of generic interfaces and abstract methods (i.e. services) hide implementation and context-dependent details, enabling continuous improvement through incremental incorporation of alternative context-dependent variants.[30]

Thus, the technical details of the ITI framework dependencies (e.g. physical/logical network topology, schedulability, worst case execution time) can be factored into the logical dataflow constraints and information exchange requirements (IERs) that are more directly associated with potential operational impact (e.g. total latency, operator response).[31] The work discussed herein is considered a continuation of an ongoing effort that works to further address MBSE/SOSE S&T challenges associated with developing this type of mission-driven performance assessment capability. Note that a primary goal of this research is to identify algorithms and methods that enable the resulting ITI framework to be as reliable, dependable, available, robust, fault tolerant, and resilient as possible, while also supporting an ability to gracefully degrade, if necessary.[32]

2.3 Directional LOS Ad-Hoc Networks: Connectivity and Link Management

Figures 7-10 provide a high-level overview and help to illustrate the layering of logical subdomains of the multilayer (i.e. multitier) models of the Open Systems Interconnection (OSI) protocol stack, EA/SOA stack, and cloud-services stack.[33] From an object-oriented distributed processing perspective (e.g. Reference Model of Open Distributed Processing - RM-ODP), the highly-desirable feature of "information hiding" is provided by encapsulation of generic functionality within the respectively defined layers. Thus, enabling the hiding of more implementation-specific details by definition of the respective generic (i.e. standardized) application programming interfaces (APIs).[34] This also enables opportunities to simultaneously optimize the degree of coupling and cohesion between inter-dependent areas of system functionality.[35]

For the type of directional LOS networking support that has been the focus of ongoing efforts, the connectivity and link management mission-support conceptually map into such layered architectures and respective domain-specific "services stacks" (i.e. OSI, EA/SOA, cloud). Thus, the ongoing focus towards generically defined cross-layer/cross-platform IP-based connectivity and link-management capabilities. Figure 7 highlights the logical-layering of directional line-of-sight (LOS) ad-hoc networking support, in terms of the layering of subdomains of types of functionality (i.e. logical scoping and encapsulation) that span multiple layers of the OSI protocol stack.[36]

As highlighted within the more detailed block diagram in figure 8 (from [8]), there are a number of logical-layer services that are needed for facilitating mission-driven connectivity management and logical IP-based dataflow management. Thus, within these middleware layers, the respective application-layer tasks, such as Warfighter tasks (e.g. C2; intelligence, surveillance and reconnaissance - ISR) and mission-support tasks (e.g. data-synchronization, entity management) are supported.[37] Such logical-layer services collaboratively work with physical-layer oriented RF/microwave LOS link-management tasks to establish objective and threshold values for link-management MOPs/MOEs that can most effectively support Warfighter policies and needs (i.e. QoS, IERs). Within the context and perspective of the OSI protocol stack, the diagram (figure 8) illustrates RF/microwave LOS comms/networking link-management challenges.[38]

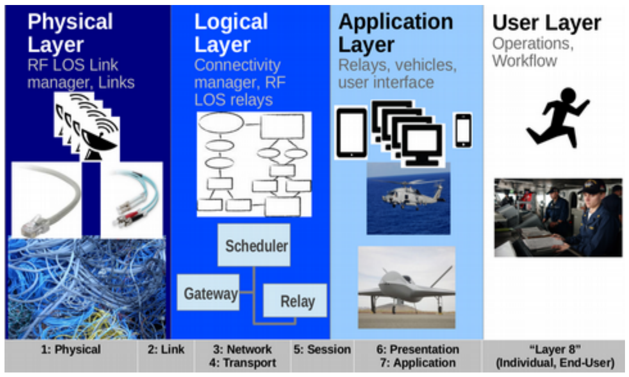


Fig. 7 ISO Model (Services Stack): ISR-Support Ex. [8]

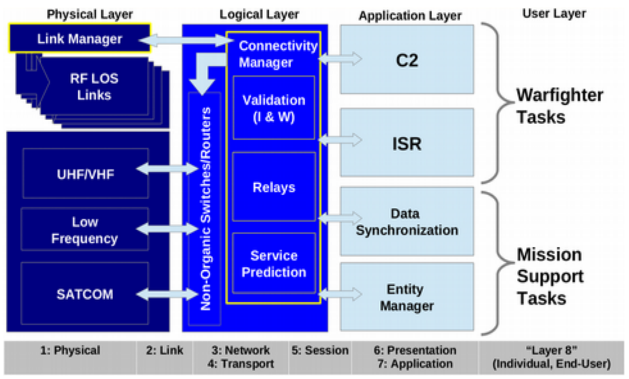


Fig 8 ISR Communication Services: Block Diagram [8]

Figure 9 is a high-level cloud-based conceptual view that highlights the mapping of the connectivity-manager and link-manager functionality (i.e. services) into their respective layers of a cloud stack. As highlighted in figure 9, the EA/SOA/Cloud oriented services stacks parallel the comms/networking oriented OSI model (i.e. protocol stack).[39] Within a cloud-computing model of computation, service developers and providers would typically interact with a variety of ITI framework oriented service domains, such as the following: Platform-as-a-Service (PaaS), Infrastructure-as-a-Service (IaaS), Networking-as-a-Service (NaaS), "Metal/Hardware-as-a-Service (MaaS)".[40] Note that from a cloud-computing perspective, the functionality of software-intensive systems can be mapped into more highly virtualized domains, such that there can be a number of different logically defined domains that support the respective "as-a-Service (aaS)" areas of functionality.[41]

As illustrated in figure 10, standardized sequencing diagrams such as Unified Modeling Language (UML) interaction/sequence diagrams, enable the representation of the operational and technical details for information-exchange events and their respective ordering.[42] From a standardized task management perspective, the representation and management of IER related information (e.g. latency distributions for prototypical information-exchange events) helps to identify and assess the parameters and respective objective/threshold values that determine the risk status for a specific mission task or collection of tasks (e.g. METL). This also addresses network-scheduling and business-logic aspects of battle management. This type of task planning and workflow management is, in principle, supported within recently developed standards and battle management language (BML) capabilities [10-11].[43]

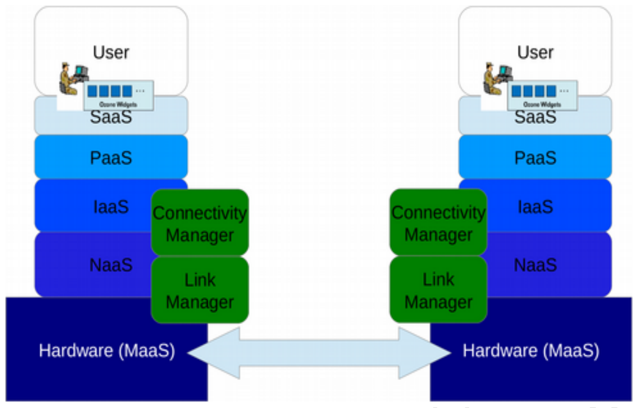


Fig. 9 ISR Communication Services: Block Diagram [8]

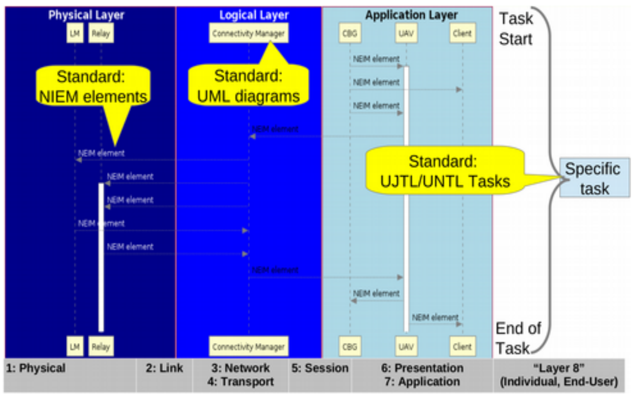


Fig. 10 ISR Comms Services: Cloud-Based View [8]

Figures 11 and 12 highlight the technical aspects of mission-task IERs, and their respective execution profiles, in terms of latency. As illustrated by the horizontal dashed-lines in figure 11, a baseline mission-thread may have identified relatively fixed objective and threshold levels for the respective information-exchange latencies. During mission execution, the respective EA/SOA/Cloud based communications services may in fact have a highly-variable timing profile (e.g. red-line in figure) that determines under which time-intervals the IERs may be met for a given task. Due to the nonstationary stochastic nature of such profiles, time-indexed estimates of the respective probability distributions are necessary for potentially tracking such data-synchronization variability and associated risks.

Figure 12 illustrates a more desirable situation where the estimates of the dynamically changing information-exchange timing-profiles are tracked and enable the dynamic self-adaptive management of the respective communications services. In this case, the communications services (e.g. traffic shaping, network scheduling) are able to better schedule the respective IERs, relative to the anticipated time-intervals where such requirements can be expected to be satisfied.

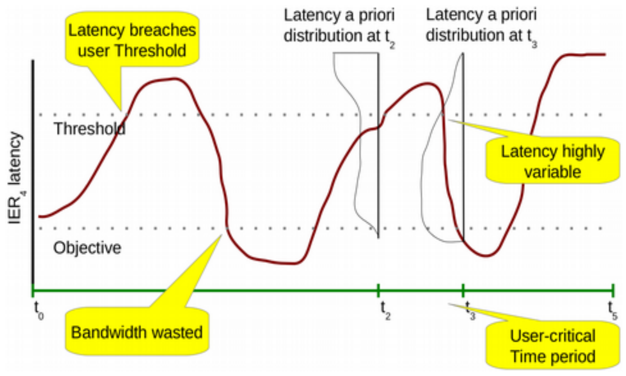


Fig. 11 Connectivity Management: Ex. Need (As-Is) [8]

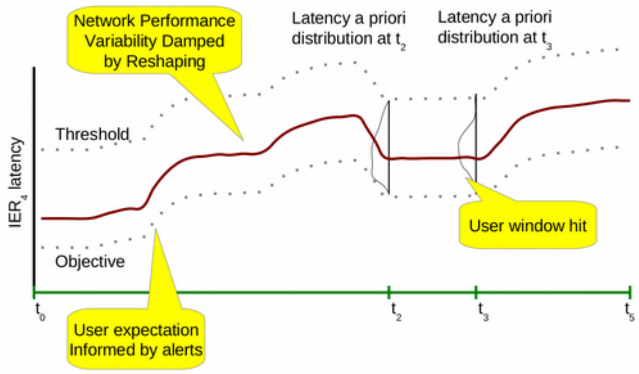


Fig. 12 Connectivity Management: Objective (To-Be) [8]

Thus, the overall capacity planning, provisioning, and dynamic management of the physical layer (via the link-manager) and logical layer (via the connectivity-manager) enables the self-adaptive and collaborative tuning/optimization of the communications services and associated physical resources.[44] This includes, in principle, supply chain management (SCM) related techniques/methods that incorporate rapidly-deployable sea-launched unmanned-platforms, each with multiple high-bandwidth RF/microwave LOS links. Thus, the potential risk of not being able to meet IER constraints during mission-execution, is further minimized.

Most importantly, much needed MOP/MOE estimates need to be generated within the respective design phases of the inter-dependent subsystems (e.g. RF/microwave LOS link management, unmanned-systems). Thus, the respective IER related statistics can provide improved working baselines for collaborative assessment and validation of mission-driven link-management MOPs/MOEs.

For example, risks due to information-exchange delays and compromised data-integrity (e.g. D-DIL conditions), can be represented and managed at the stakeholder workflow-level of mission-support services. Within the commercial sector, business-rules approaches are another example of this type of management of dynamic work environments. Business rules are especially useful for establishing and managing policies for how to manage situations that are outside normal (i.e. baseline) operating conditions. From a systems design perspective, this type of information is useful for assertion oriented and process calculus methodologies (e.g. design-by-contract, pi-calculus, process calculi) that utilize conditional-execution specifications (e.g. precondition, postcondition, invariant).[45] In addition, rule-based systems are examples of event-driven and declarative programming types of systems designs that are commonly utilized for time-sensitive distributed-control applications and cyber-physical systems.[46]

3. Standardized Architectures and Frameworks: Examples

3.1 Application-Oriented Architectures: Full Motion Video (FMV) Example

Figures 13-15 (from [12]) illustrate a recently developed ITI oriented architecture that focuses on Navy FMV (NFMV) data-transport and data-mediation needs. Figure 13 is the high-level operational view (OV-1) for the overarching architecture model. Figure 14 is a DoDAF V2.0 Capability View (CV) diagram, called the Capability Taxonomy (CV-2). Note that this diagram highlights where the NFMV functionality and respective ISR communications services fit within the context of a standardized capability taxonomy (i.e. Joint Capability Areas - JCA). Figure 15 depicts the "NFMV rich services enterprise service bus (ESB)" that resulted from the architecture modeling effort.

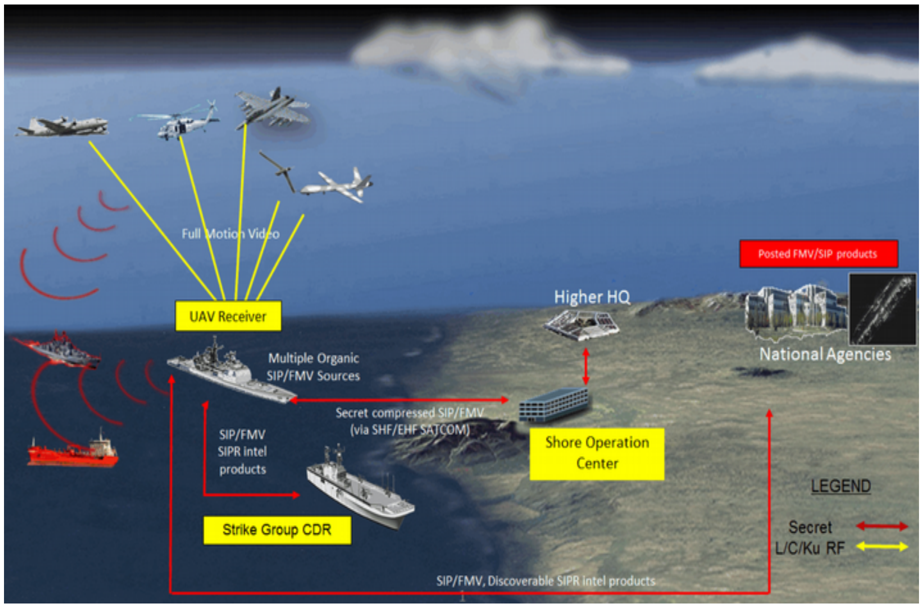

Full Motion Video
UAV Receiver
Posted FMV/SIP products
Higher HQ
National Agencies
Multiple Organic SIP/FMV Sources
Secret compressed SIP/FMV (via SHF/EHF SATCOM)
SIP/FMV SIPR intel products
Shore Operation Center
Strike Group CDR
LEGEND
Secret
L/C/Ku RF
SIP/FMV, Discoverable SIPR intel products

Fig. 13 FMV Example: High-Level View (OV-1) [12]

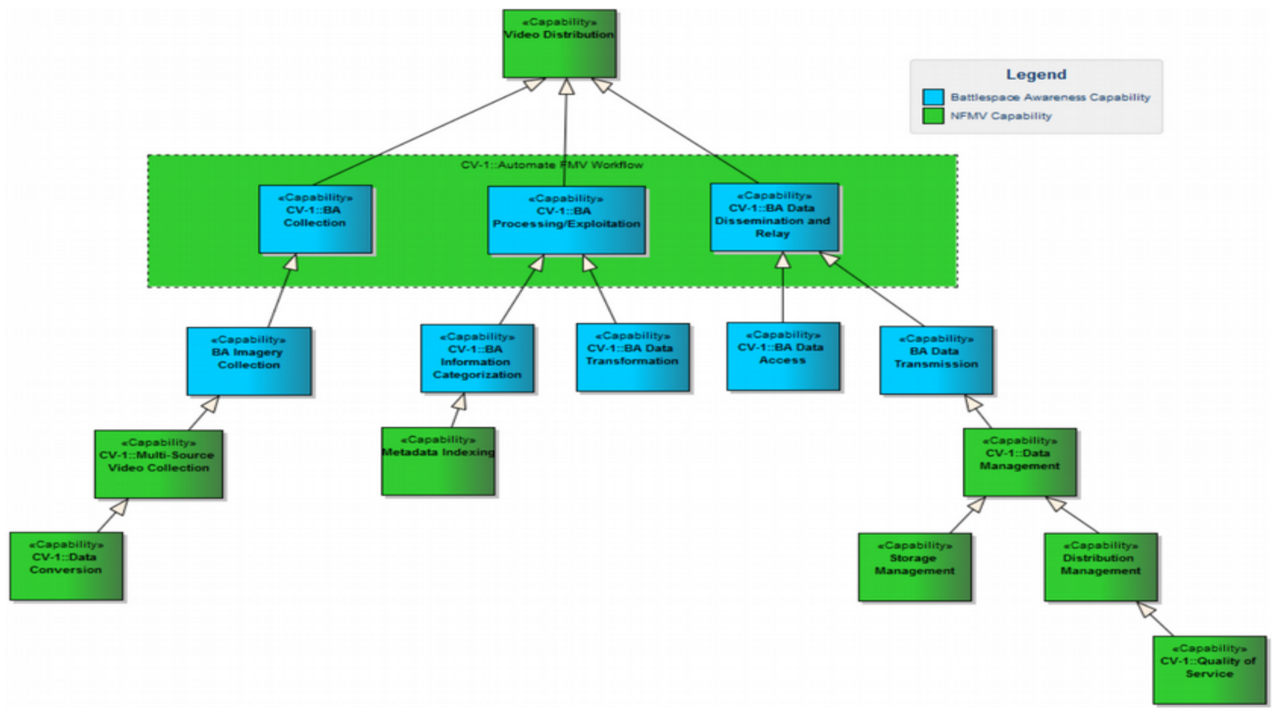

«Capability» Video Distribution
Legend
Battlespace Awareness Capability
NFMV Capability
CV-1::Automate FMV Workflow
«Capability» CV-1::BA Collection
«Capability» CV-1::BA Processing/Exploitation
«Capability» CV-1::BA Data Dissemination and Relay
«Capability» BA Imagery Collection
«Capability» CV-1::BA Information Categorization
«Capability» CV-1::BA Data Transformation
«Capability» CV-1::BA Data Access
«Capability» BA Data Transmission
«Capability» CV-1::Multi-Source Video Collection
«Capability» Metadata Indexing
«Capability» CV-1::Data Management
«Capability» CV-1::Data Conversion
«Capability» Storage Management
«Capability» Distribution Management
«Capability» CV-1::Quality of Service

Fig. 14 FMV Example: Capability Taxonomy (CV-2) [12]

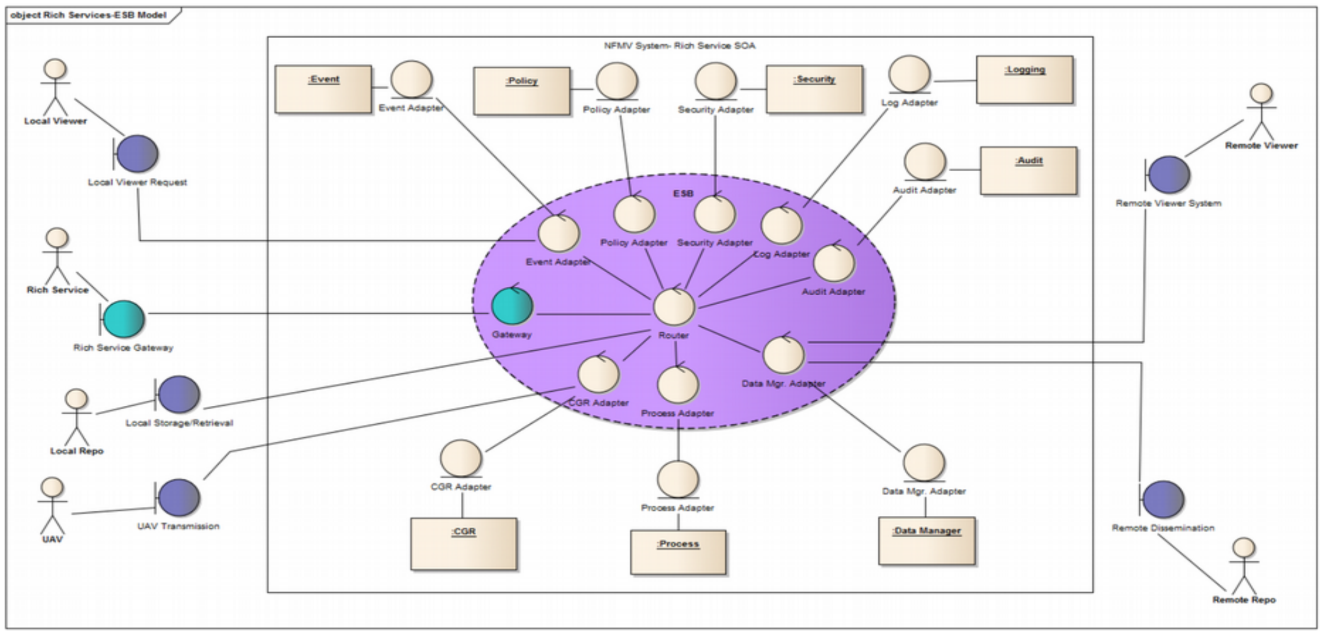


Fig. 15 FMV Example: Rich Services Enterprise Service Bus (ESB) [12]

Ideally, this type of rich-services ESB can be integrated and aligned with an overarching meta-model that more comprehensively includes the breath and depth of the ITI related technology domains. In particular, enterprise computing (e.g. software-defined environments), connectivity/link management, and autonomic-computing aspects of data-transport, data-mediation, and data-synchronization services (i.e. areas of concern) need to be more explicitly addressed, represented, and incorporated.

3.2 Adaptive Enterprise Service Bus (ESB): Example Framework

Figures 16 and 17 (from [21]) illustrate an ESB-based logical architecture that incorporates adaptation and monitoring within the context of workflow management and automation (e.g. Business Process Model and Notation – BPMN; XML Process Definition Language - XPDL).[47] Note that the ESB is working in concert with an "adaptation and monitoring engine (decision mechanisms)" and Web Services Business Process Execution Language (WS-BPEL) engine. As highlighted in figure 17, adaptation requirements are achieved by monitoring of events that trigger adaptation mechanisms implemented for supporting the respective adaptation strategies.

Autonomic computing and other self-managed architectures tend to address similar QoS issues within a much broader scope than is within the context of enterprise information portal (EIP) focused ESB-based architectures.[48] Due to the rapid maturity of such adaptive-workflow capabilities, these are additional areas of work that may provide the flexibility/agility needed to address D-DIL conditions.

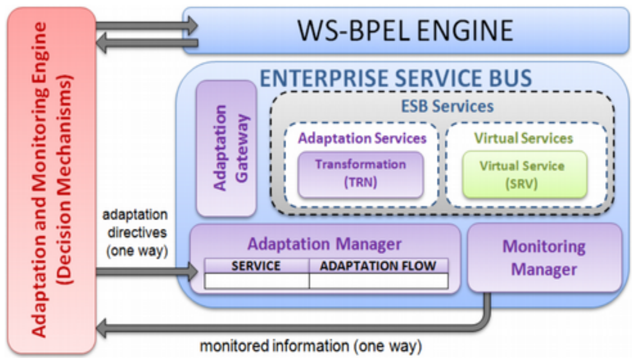


Fig. 16 Adaptive ESB: Logical Architecture [21]

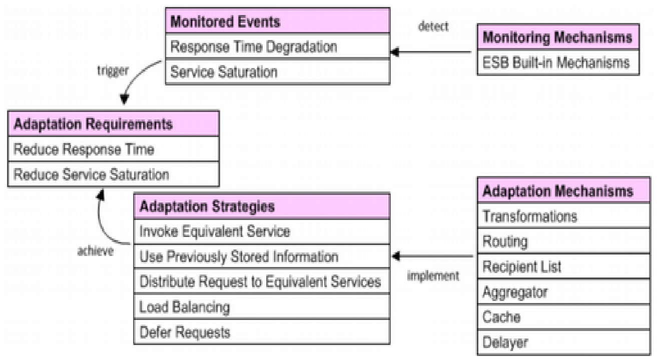


Fig. 17 Adaptive ESB: Strategies and Mechanisms [21]

From an architectural model-based systems engineering (i.e. MBSE/SOSE) perspective, the logical separation of ESB concerns (i.e. data-transport, data-mediation, data-synchronization) versus autonomic computing services, can in principle, be independently represented and merged/weaved into application-specific and platform-specific configurations.[49] Architecturally, this is another example of the types of open research challenges that are the focus of ongoing work. From a practical and pragmatic perspective, the issue is ultimately a question of where and how the adaptation and reconfiguration logic is to be represented and executed, and at which level of abstraction (e.g. modeling-level vs implementation-level).

3.3 Self-Optimizing Networks: Software Defined Environment (SDE)

Figures 18 and 19 (from [14]) illustrate the role of continuous optimization, as a value-added service and integral component of highly virtualized EA/SOA/Cloud-based frameworks, such as software defined environments (SDE). Note that software quality parameters and key performance indicators (KPIs) help to parameterize the stakeholder (e.g. user) utility functions and drive the optimization process that dynamically manages the virtualized infrastructure. Figure 19 illustrates the integration of continuous dynamically-managed assurance for maximizing resiliency and security for SDE frameworks. For current work, this is a key enabler for developing data-link focused adaptation-strategies that empower such data-link terminals to be more robust/resilient (i.e. autonomous).

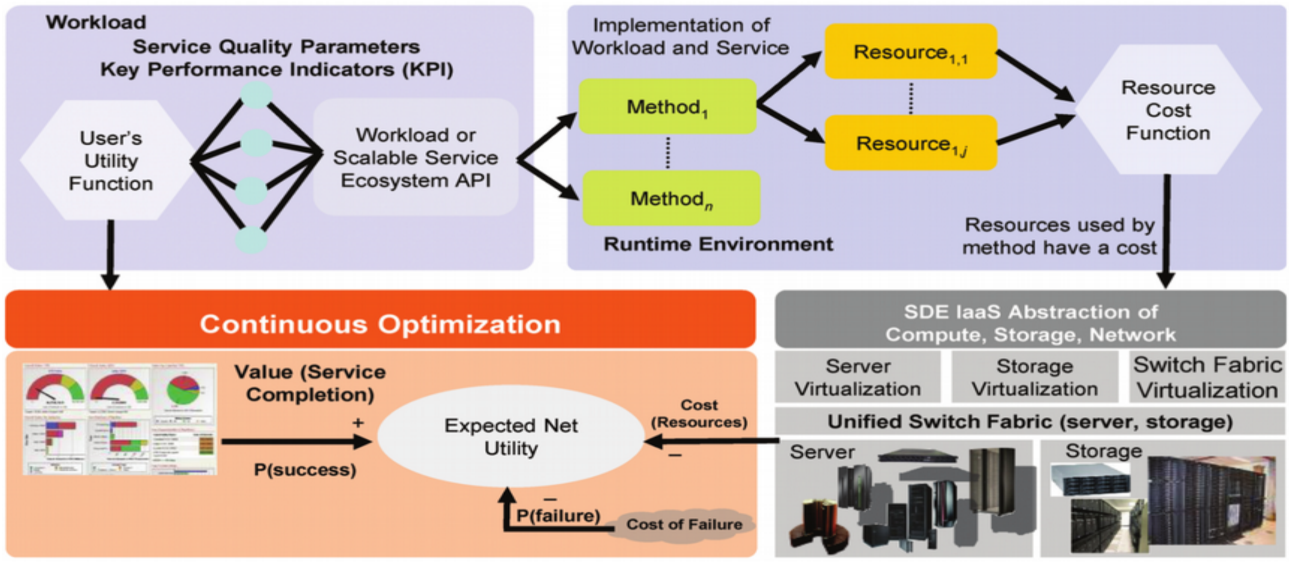

Workload
Service Quality Parameters
Key Performance Indicators (KPI)
User's Utility Function
Workload or Scalable Service Ecosystem API
Implementation of Workload and Service
Method$_1$
Method$_n$
Resource$_{1,1}$
Resource$_{1,j}$
Resource Cost Function
Runtime Environment
Resources used by method have a cost
Continuous Optimization
Value (Service Completion)
P(success)
Expected Net Utility
Cost (Resources)
P(failure)
Cost of Failure
SDE IaaS Abstraction of Compute, Storage, Network
Server Virtualization
Storage Virtualization
Switch Fabric Virtualization
Unified Switch Fabric (server, storage)
Server
Storage

Fig. 18 Outcome-Optimized Framework for Software Defined Environments: Example [14]
(SDE – Sofware Defined Environment; IaaS – Infrastructure as a Service)

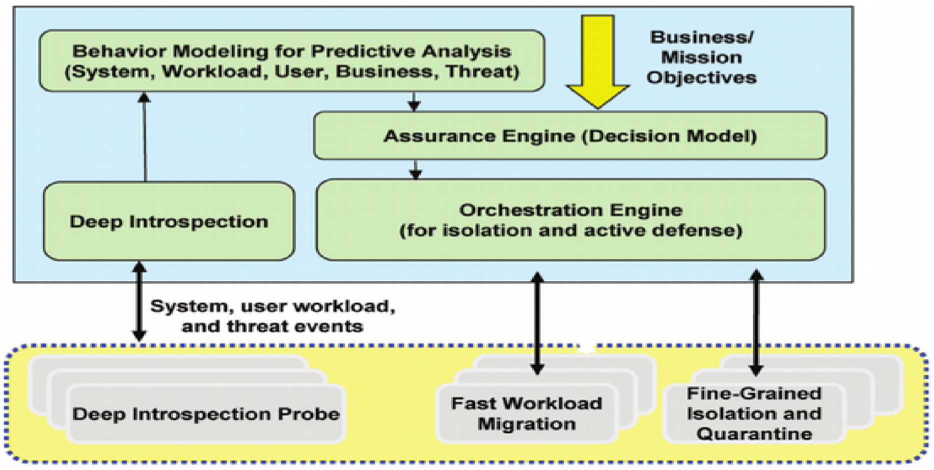

Behavior Modeling for Predictive Analysis (System, Workload, User, Business, Threat)
Business/ Mission Objectives
Assurance Engine (Decision Model)
Deep Introspection
Orchestration Engine (for isolation and active defense)
System, user workload, and threat events
Deep Introspection Probe
Fast Workload Migration
Fine-Grained Isolation and Quarantine

Fig. 19 Continuous Assurance for Resiliency and Security within Software Defined Environments: Example [14]

Figure 20 (from [19]) is an example of a resilience management framework that utilizes a SDE/SDN type of network infrastructure. The use of management patterns, resilience management functions, and event monitoring/correlation provide a patterns-based event-driven approach. Also, note that the resilience targets are determined by service level agreements (SLAs), wherein allowable deviations are a function of the role assigned to a given managed object. Thus, much of the desired or necessary characteristics of the previously discussed data-transport, data-mediation, and data-synchronization related capability-needs are also addressed within this example resilience-management framework.

Figures 21-23 (from [20]) highlight an autonomic-computing type of hierarchical feedback model within a web-based multi-layered architecture framework. In this case, the focus is on development of Hypertext Transfer Protocol (HTTP) Adaptive Streaming (HAS) and associated mechanisms for quality of experience (QoE) optimization of video traffic, which is an applicable use-case for ITI communication services.

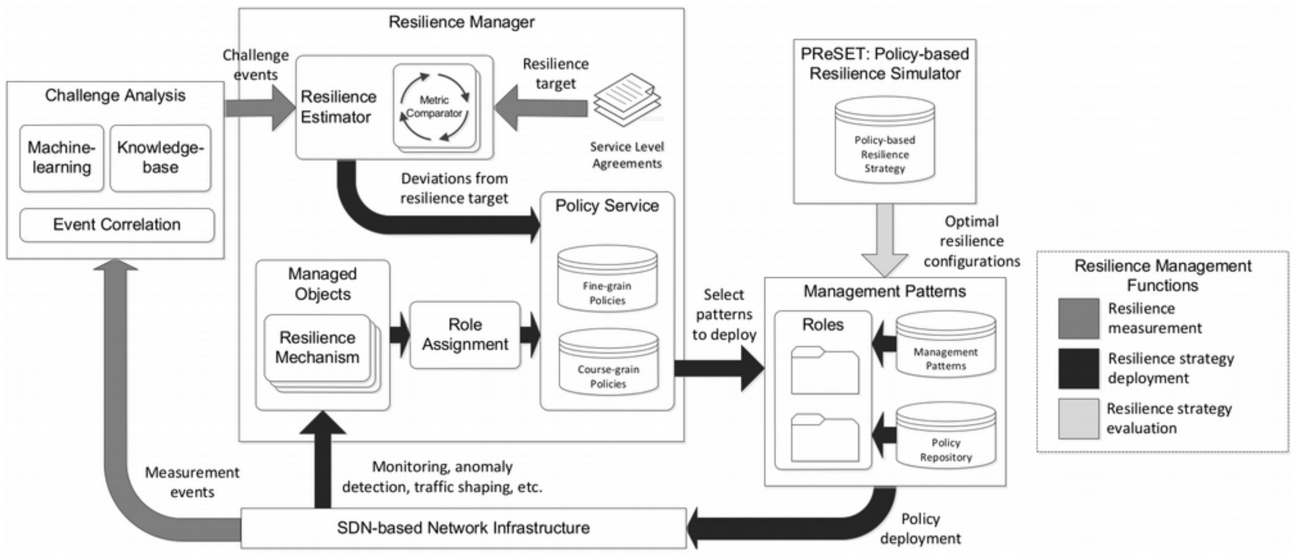


Fig. 20 Resilience Management Framework: Example [19]

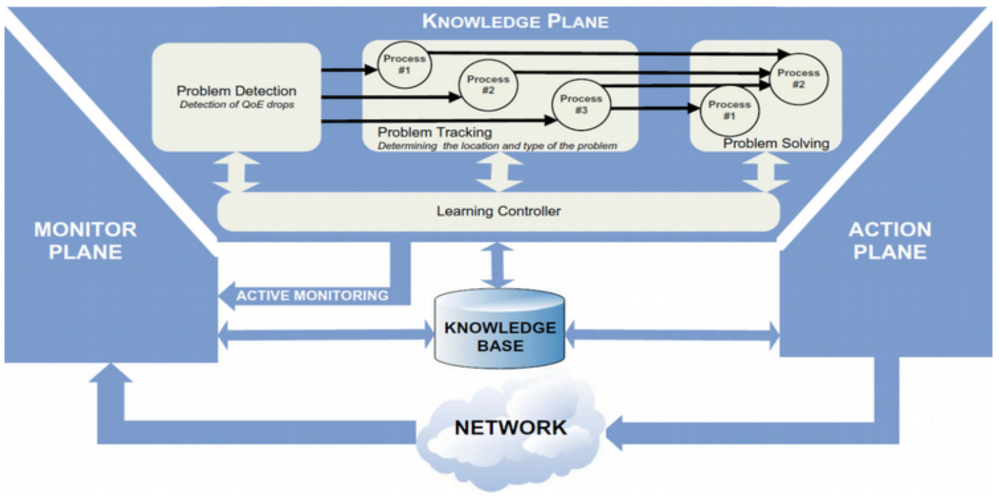


Fig. 21 Dynamic QoE Optimization: Example Autonomic-Computing Architecture Model [20]

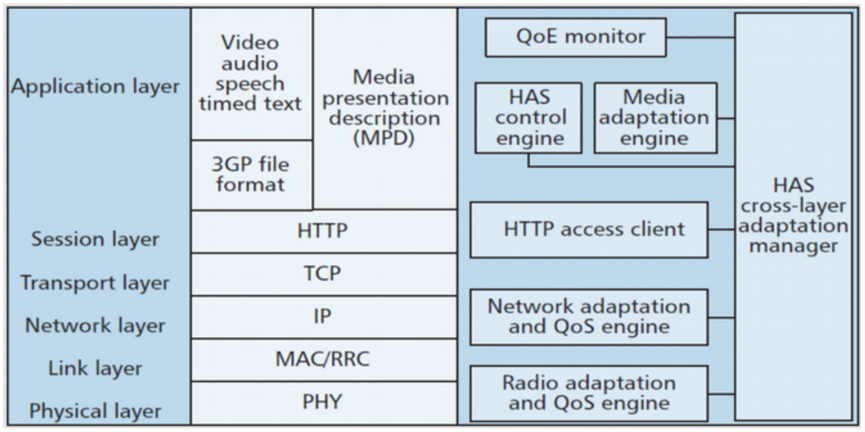


Fig. 22 Dynamic QoE Optimization: Ex. Architecture [20]

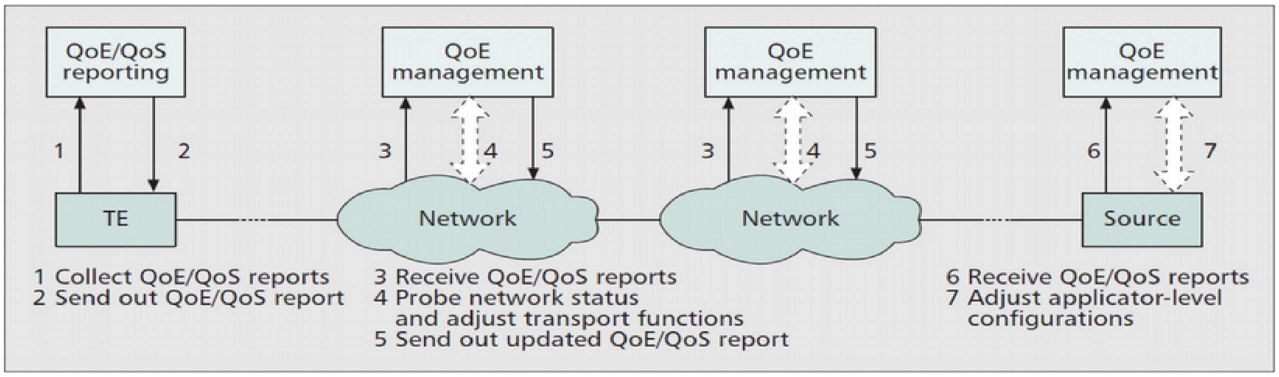

QoE/QoS reporting
QoE management
QoE management
QoE management
1
2
3
4
5
3
4
5
6
7
TE
Network
Network
Source
1 Collect QoE/QoS reports
2 Send out QoE/QoS report
3 Receive QoE/QoS reports
4 Probe network status and adjust transport functions
5 Send out updated QoE/QoS report
6 Receive QoE/QoS reports
7 Adjust applicator-level configurations

Fig. 23 Distributed End-to-End QoE Assurance System: Example [20]

Thus, from a design patterns and respective meta-modeling perspective, there may be opportunities for developing platform-independent models (i.e. meta-models) that capture the similarities and differences between a variety of domains, which include reconfigurable video-streaming, reconfigurable networks, wireless networking, resilient networking, autonomic computing, and ESB architecture models.

3.4 Autonomic Computing Frameworks: Rainbow Example

Figure 24 (from [25]) illustrates a high-level conceptual model of an autonomic manager that primarily executes a Monitor-Analyze-Plan-Execute (MAPE) loop, relative to a given managed element. Figure 25 (from [49]) is a high-level block diagram of the Rainbow Framework, which is an example of an architecture-based variant of the autonomic-computing MAPE model.

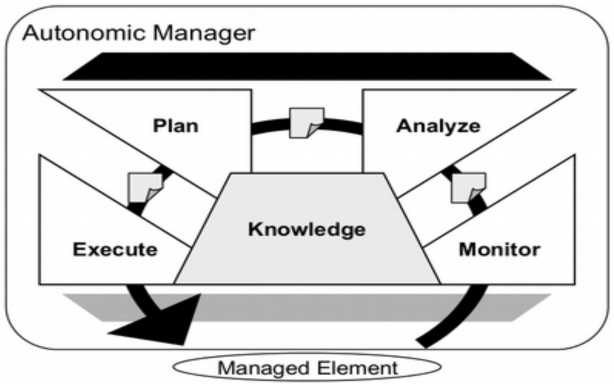


Fig. 24 Monitor-Analyze-Plan-Execute (MAPE) Model [25]

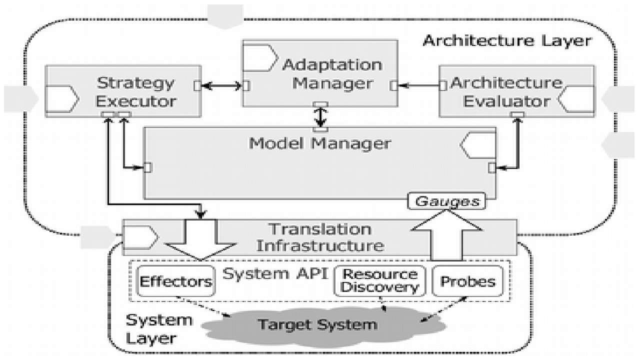


Fig. 25 Rainbow: Example of Autonomic Management [49]

Other conceptual models, such as Boyd's observe-orient-decide-act (OODA) loop and Deming's plan-do-check-act (PDCA) circle/cycle, provide similar types of adaptive meta-modeling capabilities that also enable dynamic self-healing behaviors and help to enable improved overall performance.[50]

As highlighted within [26], Rainbow is a particular instance of an architecture-based self-adaptation framework, wherein probes are utilized to monitor the target system and gauges are utilized to aggregate and abstract monitored information to update the respective control models (maintained with the model manager). These models consist of system architecture and environment models that are used to reason about the target system and its execution context. Rainbow uses an architecture evaluator to detect when a target-system is in a state suitable for repair, an adaptation manager to select an appropriate repair strategy, and a strategy executor to carry out the actions in the strategy, along with appropriate infrastructure for effecting changes in the target system. This type of autonomic-computing framework is of interest due to the architecture-based self-adaptation capabilities that are supported.

The above examples are a sampling of resources that can be adapted and tailored towards a more highly integrated and interoperable family of systems and respective capabilities. This type of ITI experimentation framework also facilitates model-integration and co-design between EA/SOA/Cloud based functionality (e.g. enterprise services). Furthermore, this type of unified lifecycle support process can potentially provide a common basis for a more cohesive workforce development and training process that also addresses variance reduction needs within challenging budgetary climates.

4. Capability Assessment: Standardized Metrics and Models

4.1 Example EA/SOA/Cloud-Based Framework

Figures 26-28 (from [29]) are a set of diagrams generated by an effort that is working towards the development of a capability-assessment framework. Figure 26 illustrates the taxonomy of the characteristics of the service measurement index (SMI).[51] Note that the SMI defines a framework and method for the calculation of a relative index, which may be used to compare IT services against one another, or to track services over time.

Figure 27 is a table that summarizes a number of the main components (i.e. elements) which typically need to be considered when assessing an ITI (e.g. enterprise computing) framework. This helps to provide a more well defined structure for an assessment process that measures specific characteristics and best captures the data needed for assessing capabilities. Figure 28 is a diagram that provides an overview of the overarching phases of a typical capability assessment process.

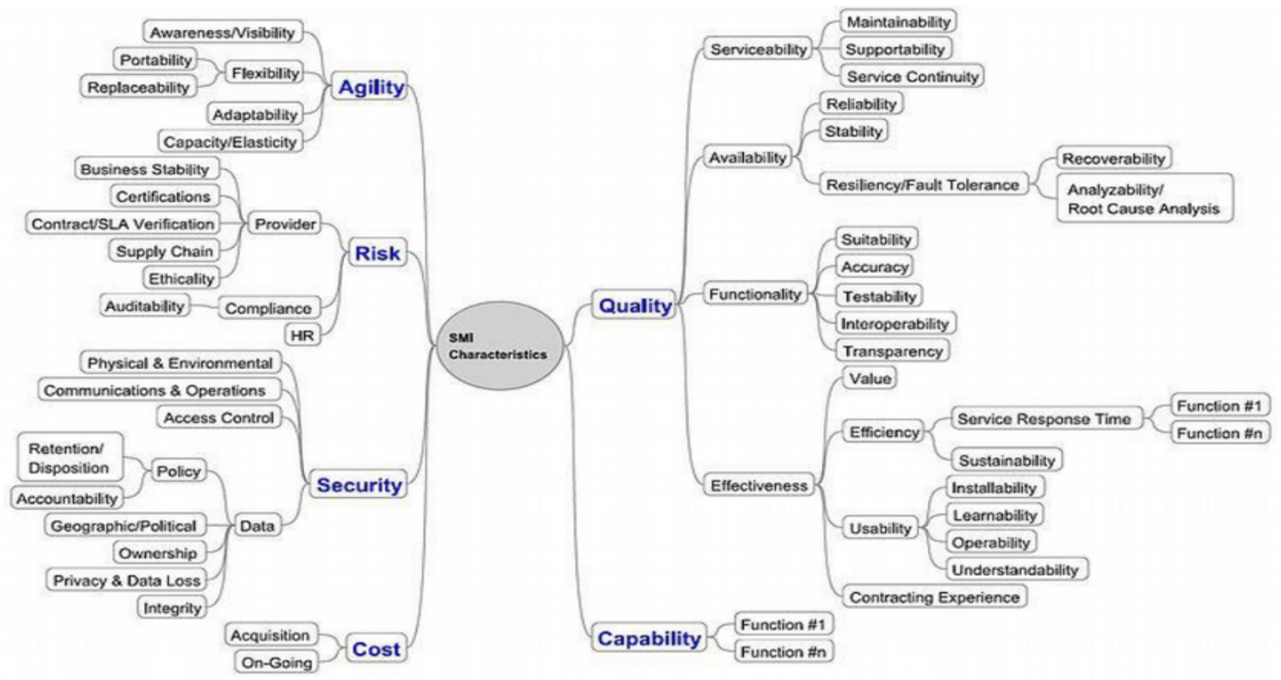
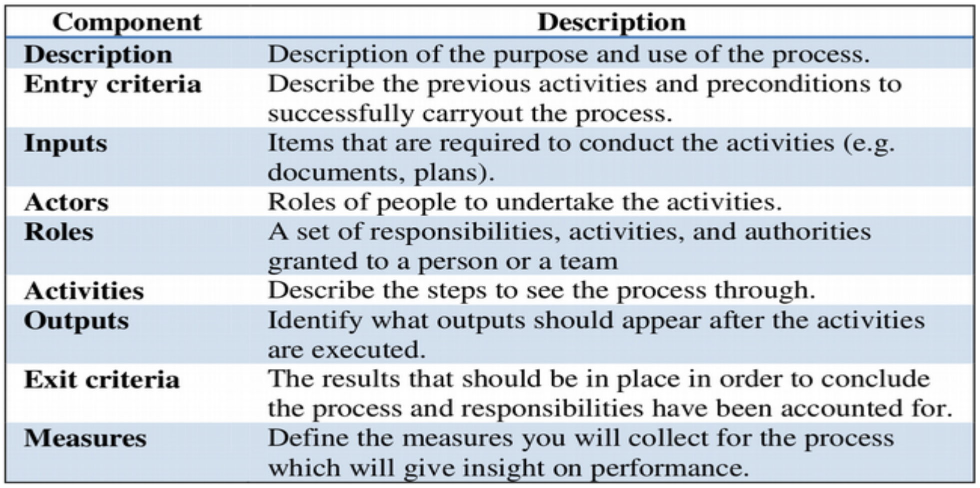


Fig. 26 Service Measurement Index (SMI): Taxonomy of Standardized Srvc Characteristics and Metrics/Indicators [29]



Component	Description
Description	Description of the purpose and use of the process.
Entry criteria	Describe the previous activities and preconditions to successfully carryout the process.
Inputs	Items that are required to conduct the activities (e.g. documents, plans).
Actors	Roles of people to undertake the activities.
Roles	A set of responsibilities, activities, and authorities granted to a person or a team
Activities	Describe the steps to see the process through.
Outputs	Identify what outputs should appear after the activities are executed.
Exit criteria	The results that should be in place in order to conclude the process and responsibilities have been accounted for.
Measures	Define the measures you will collect for the process which will give insight on performance.

Fig. 27 Main Components (i.e. elements) of Standardized Assessment Process: Example [29]

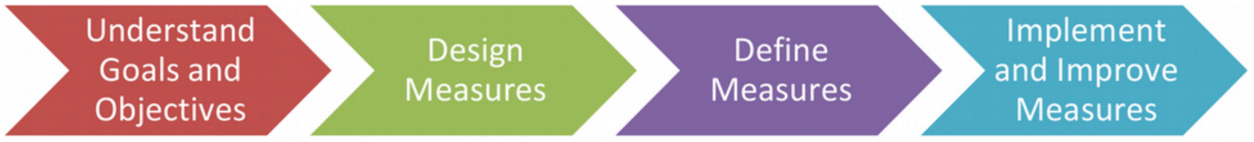

Understand Goals and Objectives
Design Measures
Define Measures
Implement and Improve Measures

Fig. 28 Capability Assessment Process: Overview [29]

4.2 Wireless Networks: Quality of Service (QoS) Focused Examples

Figure 29 (from [30]) is a more specific example of a wireless-networking focused taxonomy that provides a classification system and data model for wireless-communications oriented quality of service (QoS) metrics that can be incorporated into a unified ITI framework.[52] Figure 30 (also from [30]) is an example taxonomy of the types of QoS based enhancements for which QoS metrics can assess the respective contribution to achieving QoS goals and objectives.

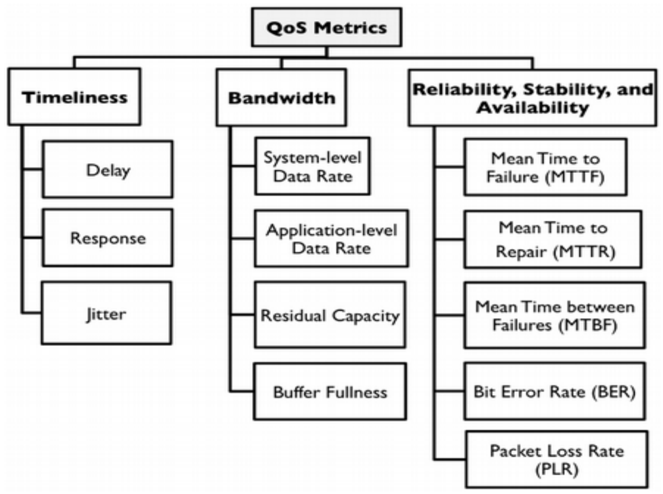

QoS Metrics
Timeliness
Delay
Response
Jitter
Bandwidth
System-level Data Rate
Application-level Data Rate
Residual Capacity
Buffer Fullness
Reliability, Stability, and Availability
Mean Time to Failure (MTTF)
Mean Time to Repair (MTTR)
Mean Time between Failures (MTBF)
Bit Error Rate (BER)
Packet Loss Rate (PLR)

Fig. 29 QoS Taxonomy for Wireless Networks: Example [30]

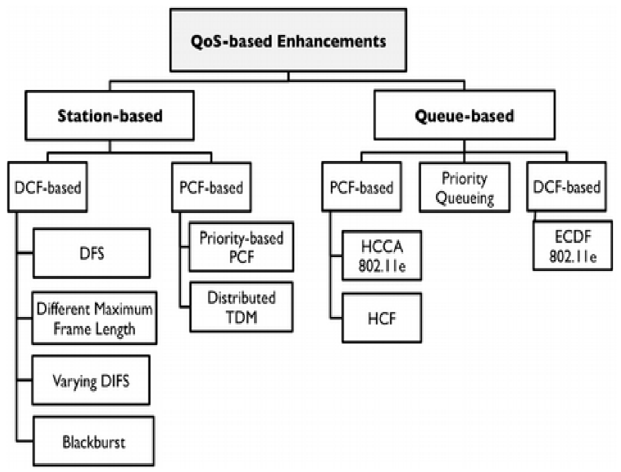


Fig. 30 MAC Layer QoS Enhancement Schemes: Ex. [30]

5. Self-Adaptive and Self-Optimizing Systems: Example Models

5.1 Requirements Modeling & Analysis

Figures 31 and 32 are from [32]. As discussed in [32], other investigators have previously proposed classification of modeling dimensions and optional degrees for self-adaptive software systems. The classification of modeling dimensions illustrated in figure 31 is an example that draws on (and derived from) such previous work [33]. Each modeling dimension describes a particular facet of the system that is relevant to self-adaptation. As noted within [32], “by considering these dimensions during modeling requirements and systems, the produced models will be more powerful.”

Figure 32 similarly draws upon and is derived from previous work [34]. In this case, the diagram illustrates and highlights that other investigators have previously proposed assurance characteristics (i.e. dimensions) for self-adaptive software systems. As noted within [33], the original authors of [34] proposed 17 assurance criteria for self-adaptive systems. From the perspective of three basic categories (i.e. effectiveness, effects and consumption,) a set of eight criteria and three extended aspects should be considered as the assurance dimensions for runtime adaptation (figure 32). Effectiveness captures the validity of the adaptation. Effects characterize the impact of adaptation upon the system. Consumption is concerned with the consumed time and resources. By illustrating these dimensions, researchers can justify the trustworthiness of their approaches and compare their results with others’.

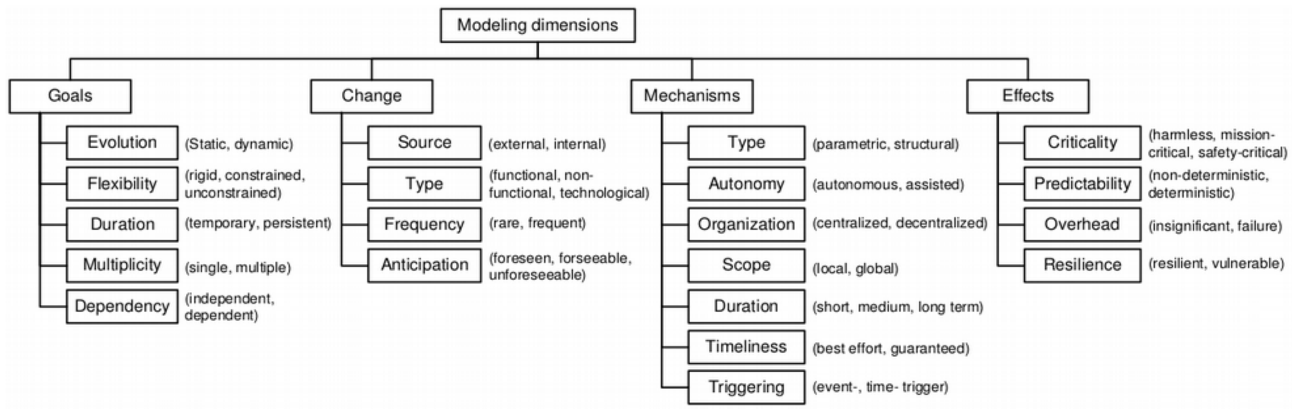
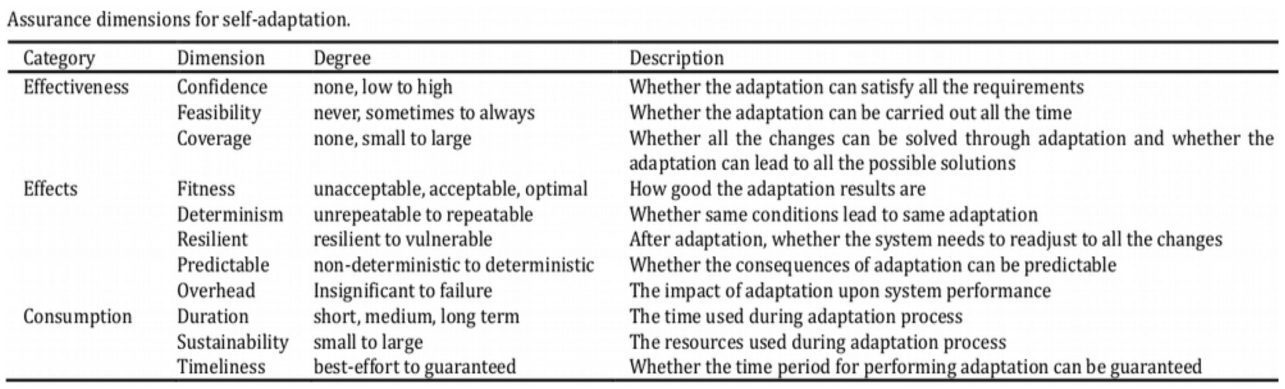


Fig. 31 Requirements Modeling for Self-Adaptive Software-Intensive Systems: Example Taxonomy [32]


Assurance dimensions for self-adaptation.

Category	Dimension	Degree	Description
Effectiveness	Confidence	none, low to high	Whether the adaptation can satisfy all the requirements
	Feasibility	never, sometimes to always	Whether the adaptation can be carried out all the time
	Coverage	none, small to large	Whether all the changes can be solved through adaptation and whether the adaptation can lead to all the possible solutions
Effects	Fitness	unacceptable, acceptable, optimal	How good the adaptation results are
	Determinism	unrepeatable to repeatable	Whether same conditions lead to same adaptation
	Resilient	resilient to vulnerable	After adaptation, whether the system needs to readjust to all the changes
	Predictable	non-deterministic to deterministic	Whether the consequences of adaptation can be predictable
	Overhead	Insignificant to failure	The impact of adaptation upon system performance
Consumption	Duration	short, medium, long term	The time used during adaptation process
	Sustainability	small to large	The resources used during adaptation process
	Timeliness	best-effort to guaranteed	Whether the time period for performing adaptation can be guaranteed

Fig. 32 Assurance Dimension for Self-Adaptation: Example [32]

5.2 Self-Healing Technologies: Video-Streaming Example

Figure 33, from [35] is a block diagram that illustrates the generic dataflow for this type of data-streaming application. Note that LOS data-links are critical elements within the network, due to the ability to provide freespace (e.g. wireless) information exchange capabilities that can support higher data-rate streaming needs. As noted by [35], services that deliver video-streaming content, typically have the following characteristics: (i) Time delivery requirements, such that the timing of the delivery of the video is critical for providing satisfactory customer experience; (ii) The relationship between the transmission of data with their consumption is different in download and streaming models, such that in the typical download model, the client application (e.g., web browser) retrieves the data objects from the server (e.g., web server) and decodes/displays them to the user after completely receiving them; and (iii) The choice of the data delivery protocol adopted for video-streaming services dictates the flow of data and client-server interactions during the data transmission period. Note that for the types of safety-critical scenarios that are the focus of this effort, the above services include time-critical store-and-forward capabilities (e.g. threat alert, intruder detection).

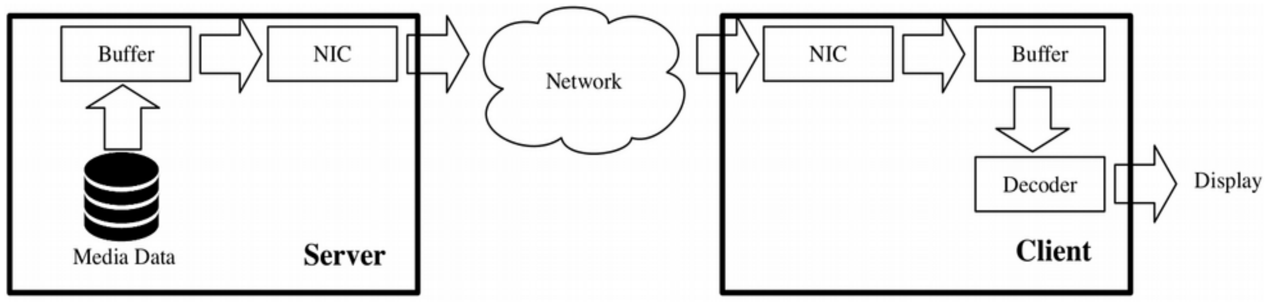


Fig. 33 Elements of a Video-on-Demand (VoD) video-streaming service: Example [35]

Figure 34, also from [35], highlights a taxonomic model of the problem space for self-healing, as applied towards developing self-healing video-streaming applications (e.g. FMV). As reviewed and discussed in a previous section, this type of application often utilizes RF/microwave LOS data-links. Thus, the applicability and potential for utilization of this type of adaptation (i.e. self-healing) model.

The taxonomy in figure 34 covers four categories of aspects for the self-healing problem:

• **Fault-model**. The fault-model states what faults or injuries should be self-healed, which includes the fault duration, fault source (e.g., operational or implementation errors and defective system requirements) and other fault characteristics;

• **System response**. System response aspects consider fault detection, specification of the degree of degraded operation provided by self-healing, fault response and fault recovery issues. Fault detection is a broad area that includes techniques such as application semantics-driven assertions, supervisory checks, computer answers examination, comparison of replicated components, and online self-testing, among others. On the other hand, recovery can be performed partially or completely. The effectiveness of recovery depends on the built-in redundancy and fault response techniques implemented (e.g., fault masking, retry, rollback and rollforward operations);

• **System completeness**. The incompleteness of specifications and designs and the limited knowledge of systems have implications on their recoverability. A thorough understanding of application semantics and about the system behavior in the absence and presence of faults is required to develop self-healing systems. However, there are many challenges to obtain the complete knowledge about the system, such as, the large frequency of updates and the use of COTS components by several modern systems. Self-knowledge and system evolution are important research topics akin to this problem;

• **Design-context**. The design-context addresses abstraction-level problems, such as component-level heterogeneity, behaviors, system scope, system linearity and user involvement degree.

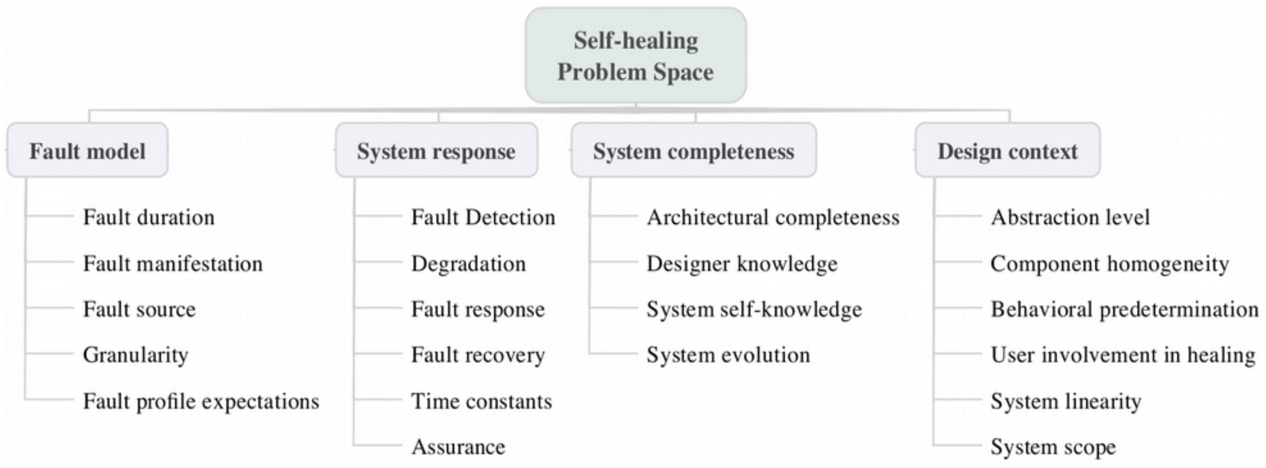


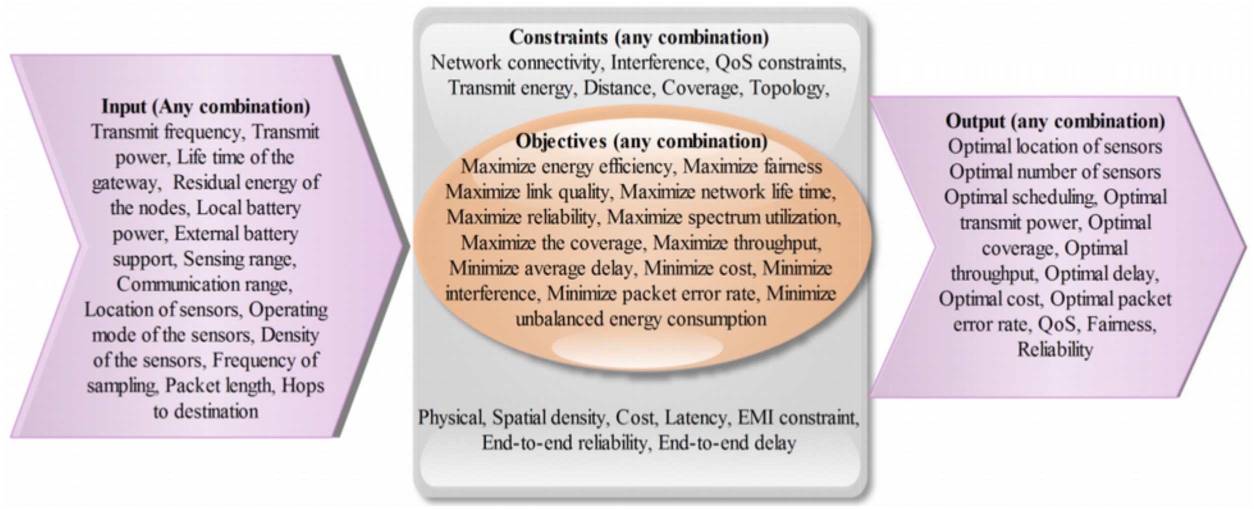
Fig. 34 Self-healing problem space: Example [35]

Note that when the supporting data-links (e.g. freespace LOS communication channels) are not able to provide the necessary information-exchange support, this type of situation is generically considered to be a type of network anomaly. The goal of the effort described herein, is to minimize the possibilities of such network-related data-link anomalies, to the greatest extent possible. Also, in terms of cross-layer (i.e. LAN) and cross-platform (i.e. client-server) dataflow, the goal is to provide a fully integrated self-healing solution whereby an autonomic computing infrastructure (e.g Rainbow) incorporates cross-layer cross-platform strategies/tactics that more naturally adapt to such situations.

Thus, the primary goal is to continue to provide sufficient solutions that meet operational requirements, while working to also simultaneously optimize the system performance to the extent possible that the minimum requirements (i.e. threshold conditions) can be exceeded (i.e. objective thresholds). In the video-streaming context, an example of this type of self-healing strategy would be to optimize the richness of the media, relative to the mission (i.e. scenario, use-case) IERs. Ideally, the predictive-analytic capabilities of an autonomic-computing infrastructure can enable proactive latency-aware adaptation strategies that further optimize overall performance. Thus, the need for self-healing can be avoided by the improvements in dynamically managing the spatial-temporal scope and respective trade-space of the overall mission and context dependent IERs.

5.3 Self-Optimizing Wireless Networks: Example Model

Figures 35 and 36 (from [31]) provide high-level views of system characteristics and associated metrics/indicators, as they apply to wireless sensor network optimization. Figure 35 highlights a time-ordered taxonomic representation of a number of parameters that respectively apply to the inputs, constraints, and outputs of such self-adaptive systems. Figure 36 provides a illustration of the inter-relationships between multiple optimization objectives for wireless systems.



Input (Any combination)
Transmit frequency, Transmit power, Life time of the gateway, Residual energy of the nodes, Local battery power, External battery support, Sensing range, Communication range, Location of sensors, Operating mode of the sensors, Density of the sensors, Frequency of sampling, Packet length, Hops to destination
Constraints (any combination)
Network connectivity, Interference, QoS constraints, Transmit energy, Distance, Coverage, Topology,
Objectives (any combination)
Maximize energy efficiency, Maximize fairness Maximize link quality, Maximize network life time, Maximize reliability, Maximize spectrum utilization, Maximize the coverage, Maximize throughput, Minimize average delay, Minimize cost, Minimize interference, Minimize packet error rate, Minimize unbalanced energy consumption
Physical, Spatial density, Cost, Latency, EMI constraint, End-to-end reliability, End-to-end delay
Output (any combination)
Optimal location of sensors Optimal number of sensors Optimal scheduling, Optimal transmit power, Optimal coverage, Optimal throughput, Optimal delay, Optimal cost, Optimal packet error rate, QoS, Fairness, Reliability

Fig. 35 Generic multi-objective optimization problem in wireless networks [31]

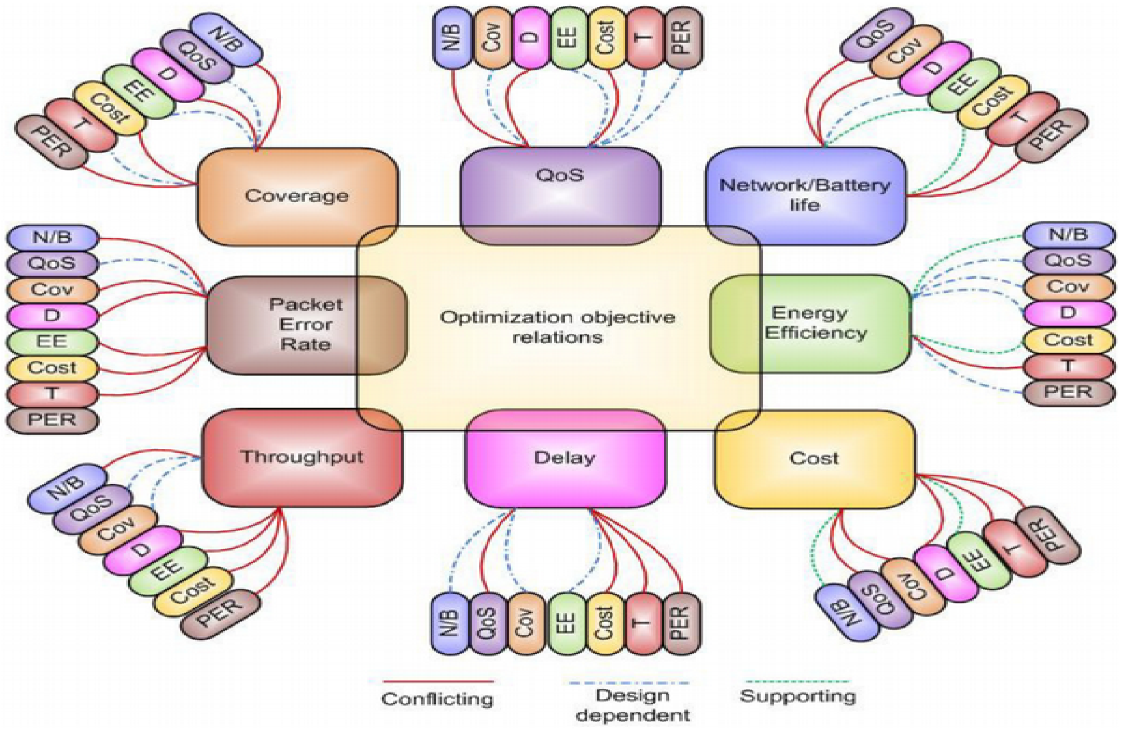


Fig. 36 Relation between desirable objectives in wireless networks [31] (Cov -Coverage; QoS – Quality of Service; NB – Network/Battery Life; EE – Energy Efficiency; Cost – Cost; D – Delay; T – Throughput; PER – Packet Error Rate)

6. Risk Management and Safety Engineering: Models/Standards

6.1 Software Risk/Reliability Modeling: Example ISO/IEC Standards

Figure 37 (from [35]) illustrates that dependability is a general concept associated to critical systems. Thus, system dependability is defined by the ability to deliver a service that can justifiably be trusted. Dependability is an integrative concept that encompasses most security attributes. Like dependability, security includes integrity and availability attributes, but requires additionally confidentiality which means the absence of unauthorized disclosure of information.

Figure 38 (from [37]) is from a recent literature survey/review that focuses on the knowledge area of software product quality.[53] As highlighted within the survey, the state-of–the-art within this domain is led by international standard proposals such as those from International Organization for Standardization / International Electrotechnical Commission (ISO/IEC) or Institute of Electrical and Electronics Engineers (IEEE) and other more operationally-specific standards, such as military standards (MIL-STD) or European Cooperation for Space Standardization (ECSS). Within this context, the ISO/IEC 25000 "Software Product Quality Requirements and Evaluation" (SQuaRE) series of standards were selected as a reference framework. Within the figures, note that fault tolerance plays a key role within the context of software reliability modeling and associated standards. Figure 39 (also from [37]) highlights the reliability characteristics of the SQuaRE series of standards. Note that fault tolerance and recoverability are also considered critical features of such reliability standards.

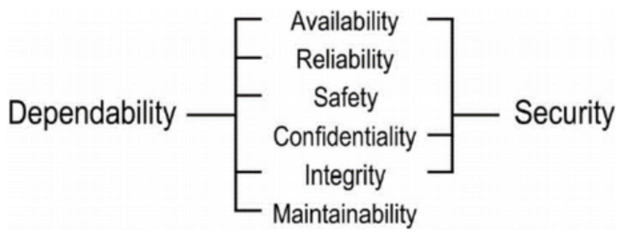


Fig. 37 Dependability and security attributes [35]

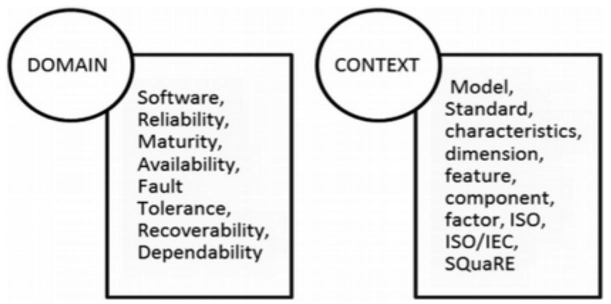
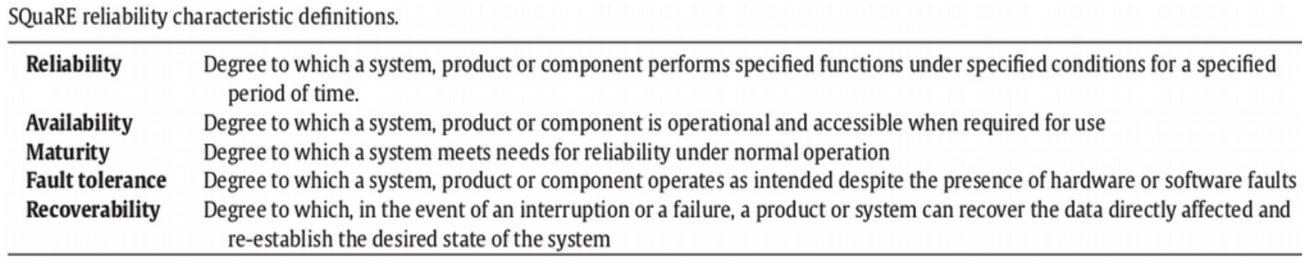

DOMAIN
Software, Reliability, Maturity, Availability, Fault Tolerance, Recoverability, Dependability
CONTEXT
Model, Standard, characteristics, dimension, feature, component, factor, ISO, ISO/IEC, SQuaRE

Fig. 38 Reliability Modeling: Domain/Context Keywords [37]



SQuaRE reliability characteristic definitions.

Reliability	Degree to which a system, product or component performs specified functions under specified conditions for a specified period of time.
Availability	Degree to which a system, product or component is operational and accessible when required for use
Maturity	Degree to which a system meets needs for reliability under normal operation
Fault tolerance	Degree to which a system, product or component operates as intended despite the presence of hardware or software faults
Recoverability	Degree to which, in the event of an interruption or a failure, a product or system can recover the data directly affected and re-establish the desired state of the system

Fig. 39 Reliability Characteristics: Standardized Definitions (SquaRE) [37]

6.2 Software-Systems Safety-Engineering: Example

Figures 40-42 (from [40]) are example views (i.e. diagrams) from the Joint Software Systems Safety Engineering Handbook (JSSSEH). These diagrams provide a more specific example of a set of best practices and associated standards that can be applied toward developing a model-based systems of systems engineering (MBSE/SOSE) approach for enabling physical-layer freespace communication links to provide more robust/resilient information-exchange (e.g. FMV) support services.

Figure 43 (from [41]) further illustrates availability of standardized processes that can be augmented and tailored towards developing a lifecycle approach for engineering more robust/resilient IP-based directional (i.e. RF LOS) data-streaming and respective communications/networking support. Other references similarly work to achieve similar standards-based capabilities [42-45].

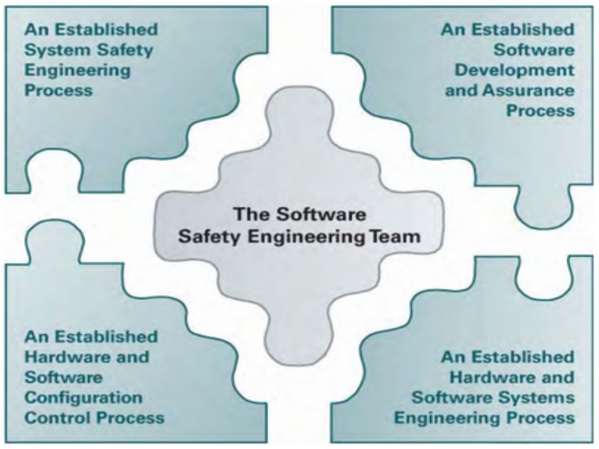
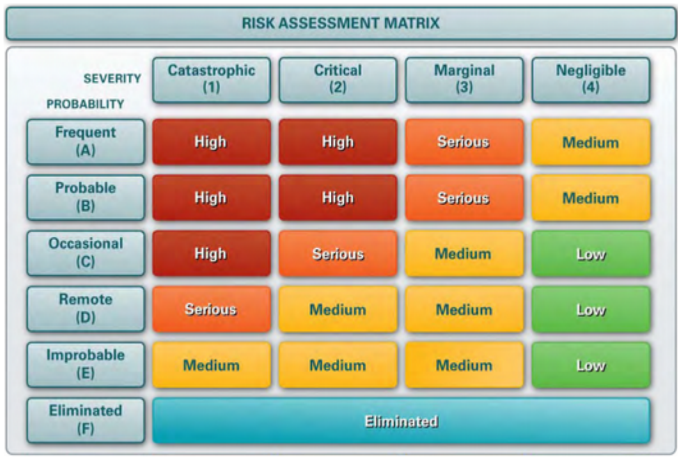

An Established System Safety Engineering Process
An Established Software Development and Assurance Process
The Software Safety Engineering Team
An Established Hardware and Software Configuration Control Process
An Established Hardware and Software Systems Engineering Process

Fig. 40 Integration of Eng. Personnel & Processes [40]



RISK ASSESSMENT MATRIX

SEVERITY / PROBABILITY	Catastrophic (1)	Critical (2)	Marginal (3)	Negligible (4)
Frequent (A)	High	High	Serious	Medium
Probable (B)	High	High	Serious	Medium
Occasional (C)	High	Serious	Medium	Low
Remote (D)	Serious	Medium	Medium	Low
Improbable (E)	Medium	Medium	Medium	Low
Eliminated (F)	Eliminated			

Fig. 41 Risk Assessment Matrix [40]

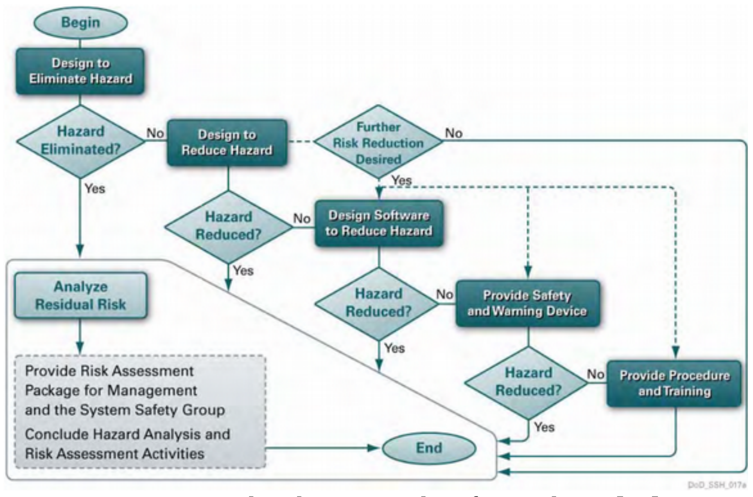

Begin
Design to Eliminate Hazard
Hazard Eliminated?
No
Yes
Design to Reduce Hazard
Further Risk Reduction Desired
No
Yes
Hazard Reduced?
No
Yes
Design Software to Reduce Hazard
Hazard Reduced?
No
Yes
Provide Safety and Warning Device
Hazard Reduced?
No
Yes
Provide Procedure and Training
Analyze Residual Risk
Provide Risk Assessment Package for Management and the System Safety Group
Conclude Hazard Analysis and Risk Assessment Activities
End

Fig. 42 Hazard Reduction: Order of Precedence [40]

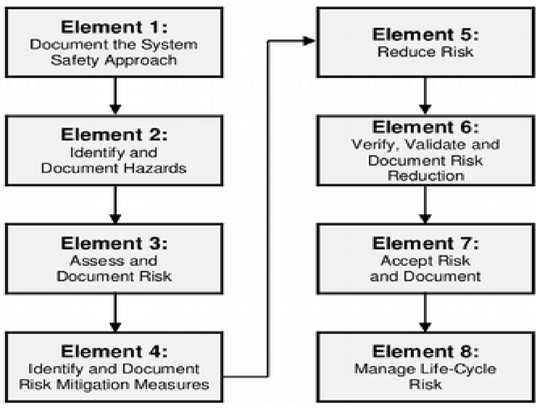

Element 1: Document the System Safety Approach
Element 2: Identify and Document Hazards
Element 3: Assess and Document Risk
Element 4: Identify and Document Risk Mitigation Measures
Element 5: Reduce Risk
Element 6: Verify, Validate and Document Risk Reduction
Element 7: Accept Risk and Document
Element 8: Manage Life-Cycle Risk

Fig. 43 System-Safety Process: Example [41]

6.3 Functional Safety

Figures 44-49 (from [39]) are example views of the lifecycle characteristics and associated definitional elements that facilitate an architectural approach to assuring functional safety within SOSE lifecycles.

Figure 44 provides an overview of the four principle classes of counter measures: fault forecast, fault prevention, fault removal, and fault tolerance. Regarding the counter measures used in practice as well as the respective demands in safety standards, these four classes cover all possible measures. Note that the diagram in figure 44 shows at which point in the fault-error-failure cascade, such counter measures can be applied. Figure 45 highlights the standards-based elements of safety-related development. Figure 46 highlights the coupling of the MBSE/SOSE development lifecycle with the safety-assurance lifecycle. Figure 47 highlights the central role of dependability and the associated categorical characteristics of dependability attributes, means and threats.

Figure 48 is an example of an architectural synthesis of these complementary concerns of functional capabilities (e.g. SOSE) and nonfunctional capabilities (e.g. safety assurance, dependability). Figure 49 highlights the incorporation and respective role of safety models within the safety assurance lifecycle.

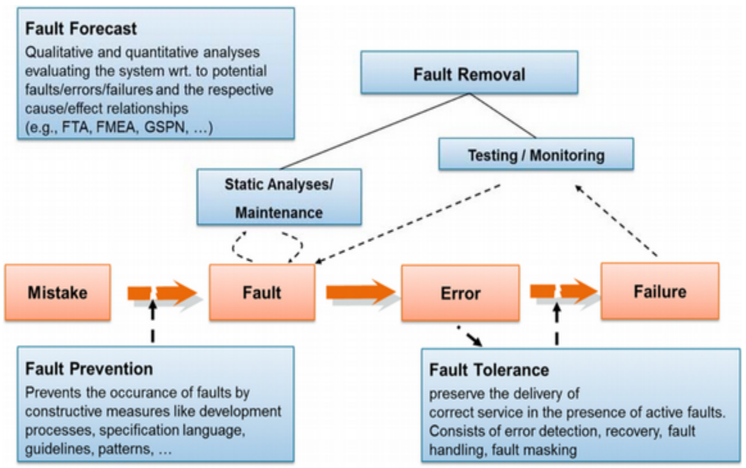

Fault Forecast
Qualitative and quantitative analyses evaluating the system wrt. to potential faults/errors/failures and the respective cause/effect relationships (e.g., FTA, FMEA, GSPN, ...)
Fault Removal
Testing / Monitoring
Static Analyses/ Maintenance
Mistake
Fault
Error
Failure
Fault Prevention
Prevents the occurance of faults by constructive measures like development processes, specification language, guidelines, patterns, ...
Fault Tolerance
preserve the delivery of correct service in the presence of active faults. Consists of error detection, recovery, fault handling, fault masking

Fig. 44 Safety-Engineering Counter Measures: Ex. Model [39]

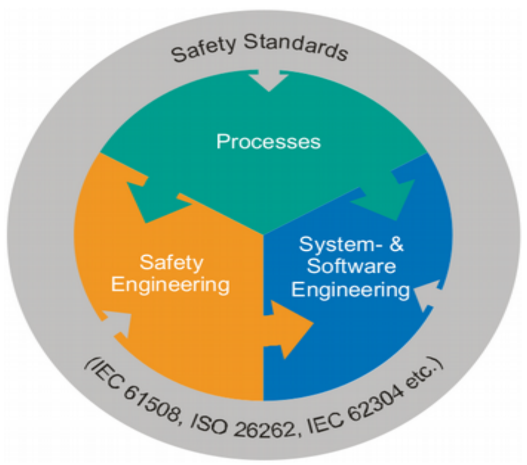

Safety Standards
Processes
Safety Engineering
System- & Software Engineering
(IEC 61508, ISO 26262, IEC 62304 etc.)

Fig. 45 Safety-Related Dev.: Ex. Elements [39]

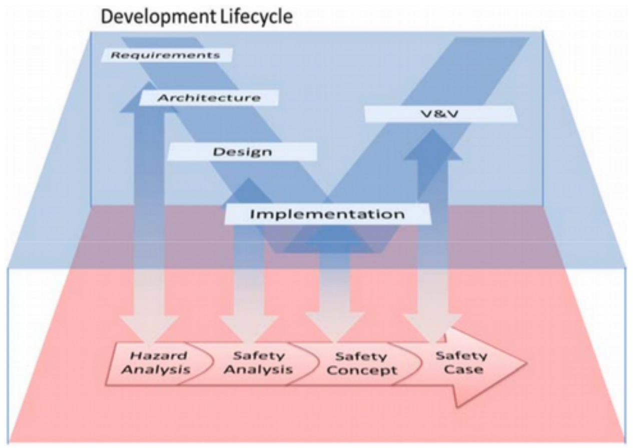


Fig. 46 Development and Safety Eng. Lifecycles [39]

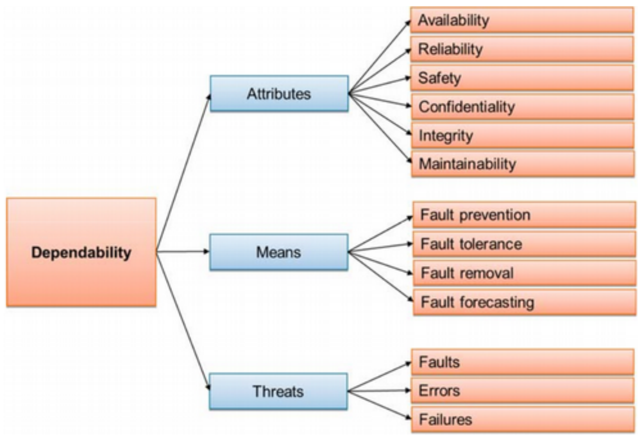


Fig. 47 Dependability: Definitional Elements [39]

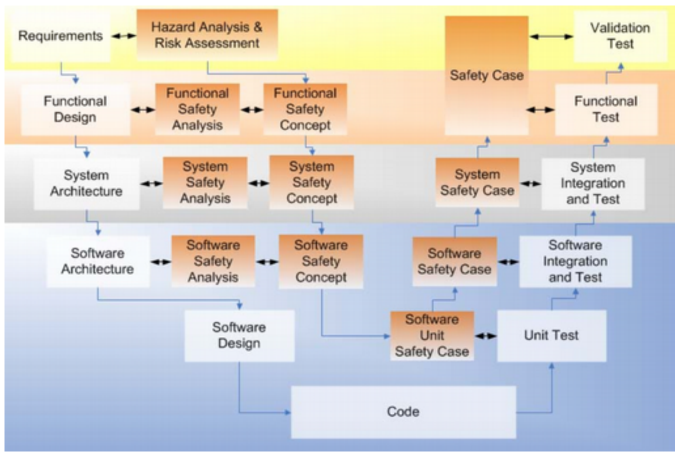


Fig. 48 Safety Engineering Lifecycle [39]

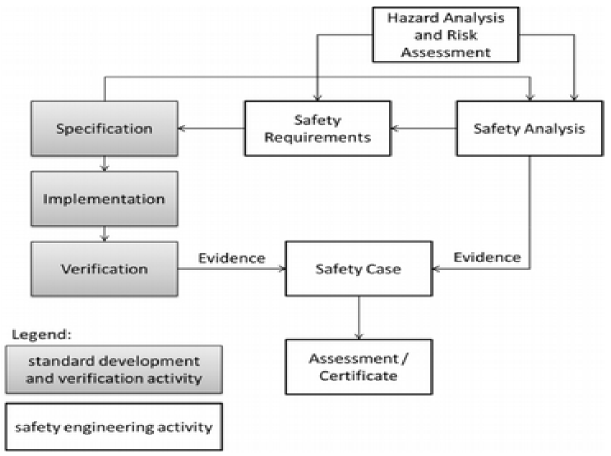


Fig. 49 Safety Models in Safety Assurance Lifecycle [39]

The above referenced examples highlight why safety models are considered critical architectural components for developing strategies that enable freespace LOS data-links to be more robust/resilient, while also optimizing overall performance. Within this context, a number of assurance oriented models/architecture, standards, and best practices can be readily applied and tailors

6.4 Multiple-Viewpoint Models: Additional Examples

Figures 50-52 (from [52]) highlight a methodology for the systematic ENgineering of TRUstworthy Self-adaptive sofTware (ENTRUST). ENTRUST uses a combination of the following: (i) design-time and runtime modelling and verification; and (ii) industry-adopted assurance processes to develop trustworthy self-adaptive software and assurance cases arguing the suitability of the software for its intended application. As shown in Figure 50, the closed-loop control-system paradigm involves using an external software controller to monitor the system and to adapt its architecture or configuration after environmental and internal changes. ENTRUST incorporates autonomic-computing and goal-oriented assurance argument patterns for supporting this type of dynamic control functionality.

As highlighted in figure 51, the assurance cases are represented in the Goal Structuring Notation (GSN), a community standard widely used for assurance case development in industry. Strategies are used to partition an assurance argument and describe the nature of the inference that exists between a goal and its supporting goal(s). The rationale (assumptions and justifications) for individual elements of the argument can be captured, along with the context (e.g. to describe the operational environment) in which the claims are stated.

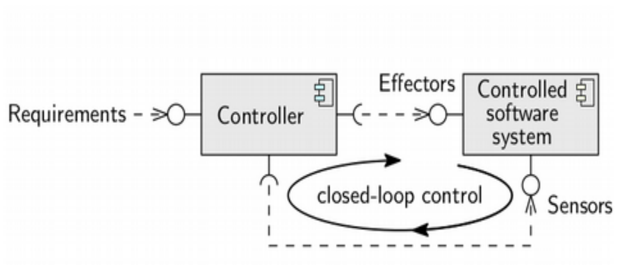


Fig. 50 Closed-Loop Control: Ex. Block Diagram [52]

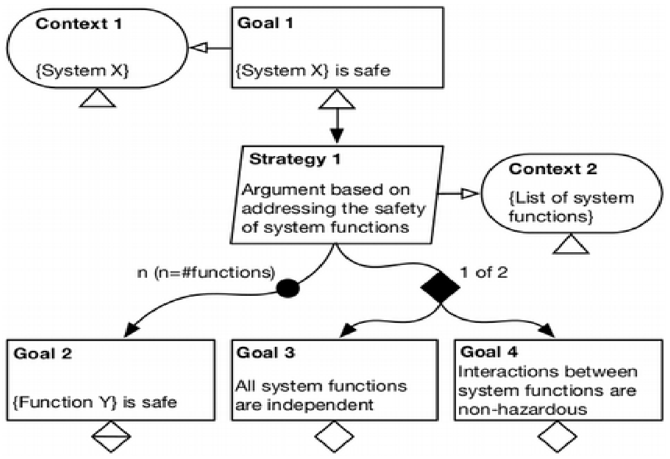

Context 1
{System X}
Goal 1
{System X} is safe
Strategy 1
Argument based on addressing the safety of system functions
Context 2
{List of system functions}
n (n=#functions)
1 of 2
Goal 2
{Function Y} is safe
Goal 3
All system functions are independent
Goal 4
Interactions between system functions are non-hazardous

Fig. 51 Example of an assurance argument pattern [52]

Figure 52 illustrates the stages and key artefacts of the ENTRUST methodology. In line with the two principles underpinning the methodology, its first stage involves the development of verifiable models for the controller, controlled system and environment of the self-adaptive system used throughout the remaining stages, and multiple stages reuse application-independent software and assurance artefacts.

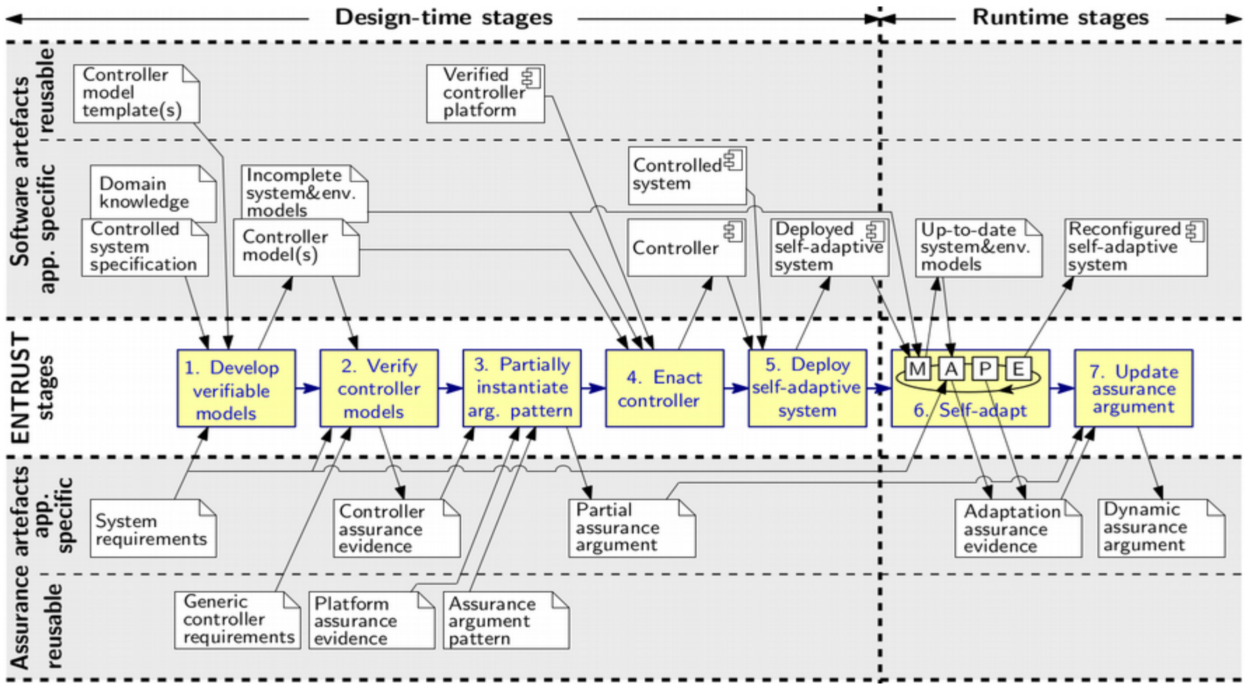
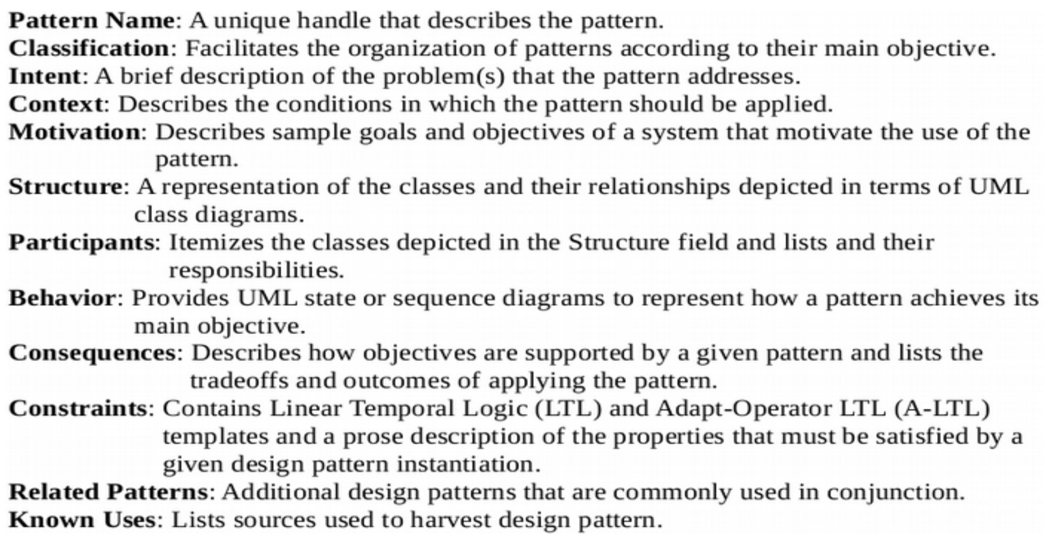

Design-time stages
Runtime stages
Software artefacts
reusable
app. specific
ENTRUST stages
Assurance artefacts
app. specific
reusable
Controller model template(s)
Verified controller platform
Domain knowledge
Incomplete system&env. models
Controlled system
Controlled system specification
Controller model(s)
Controller
Deployed self-adaptive system
Up-to-date system&env. models
Reconfigured self-adaptive system
1. Develop verifiable models
2. Verify controller models
3. Partially instantiate arg. pattern
4. Enact controller
5. Deploy self-adaptive system
M
A
P
E
6. Self-adapt
7. Update assurance argument
System requirements
Controller assurance evidence
Partial assurance argument
Adaptation assurance evidence
Dynamic assurance argument
Generic controller requirements
Platform assurance evidence
Assurance argument pattern

Fig. 52 Stages and Key Artefacts of the ENTRUST Methodology [52]

7. Change and Adaptation Models: Examples

7.1 Adaptation Strategies: Design-Pattern Oriented Approach

Figures 53 (from [54]) lists characteristics and attributes of an example adaptation design-pattern template. Figure 54 (also from [54]), highlights similarities and differences between a model-based autonomic-computing (e.g. Rainbow), versus a more design-pattern oriented approach to developing and representing adaptation strategies. Thus, highlighting architectural variants of adaptation strategies.



Pattern Name: A unique handle that describes the pattern.
Classification: Facilitates the organization of patterns according to their main objective.
Intent: A brief description of the problem(s) that the pattern addresses.
Context: Describes the conditions in which the pattern should be applied.
Motivation: Describes sample goals and objectives of a system that motivate the use of the pattern.
Structure: A representation of the classes and their relationships depicted in terms of UML class diagrams.
Participants: Itemizes the classes depicted in the Structure field and lists and their responsibilities.
Behavior: Provides UML state or sequence diagrams to represent how a pattern achieves its main objective.
Consequences: Describes how objectives are supported by a given pattern and lists the tradeoffs and outcomes of applying the pattern.
Constraints: Contains Linear Temporal Logic (LTL) and Adapt-Operator LTL (A-LTL) templates and a prose description of the properties that must be satisfied by a given design pattern instantiation.
Related Patterns: Additional design patterns that are commonly used in conjunction.
Known Uses: Lists sources used to harvest design pattern.

Fig. 53 Adaptation Design-Pattern Template: Example [54]

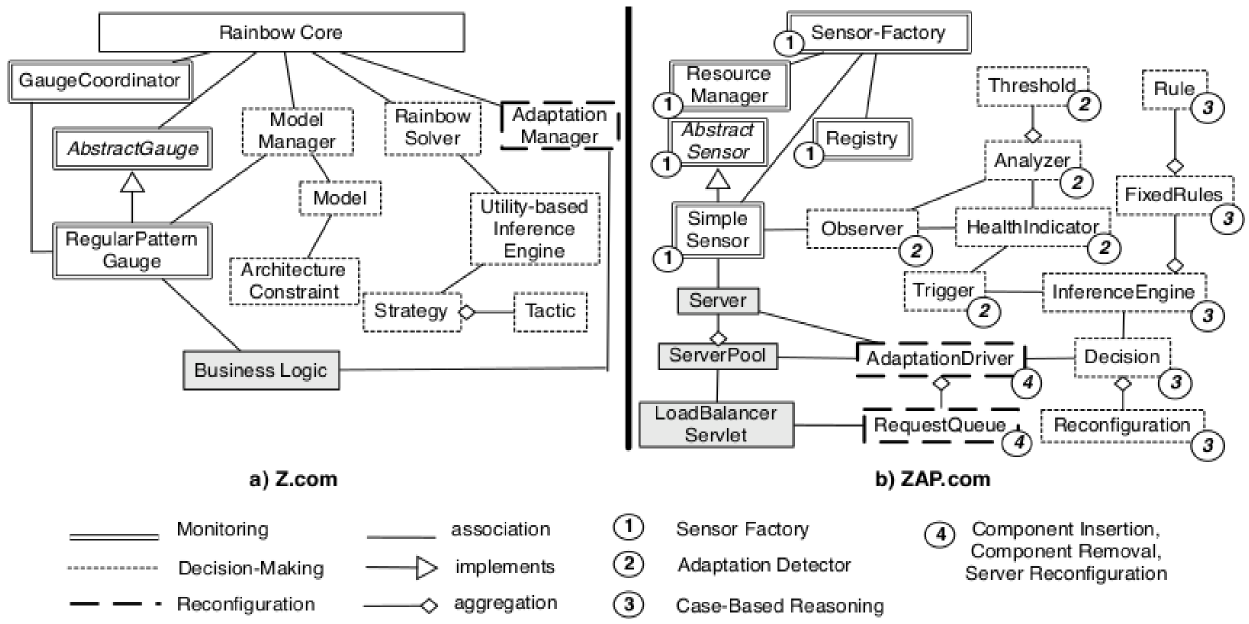


Fig. 54 Elided structural models: Rainbow-framework built (Z.com) versus pattern-based (ZAP.com) [54]

7.2 Latency Awareness: Timing, Coordination, & Proactive Adaptation

Figures 55 and 56 (from [46]) provide an example of an safety-assurance oriented approach that implements a supervisory-control and respective hazard monitoring capability. As highlighted in figure 55, this type of architecture is functionally similar to architecture-based autonomic-computing frameworks, such as Rainbow. Figure 56 is an example of timing and hazard-level dynamics that apply to the type of IERs and respective latency requirements discussed in previous sections.

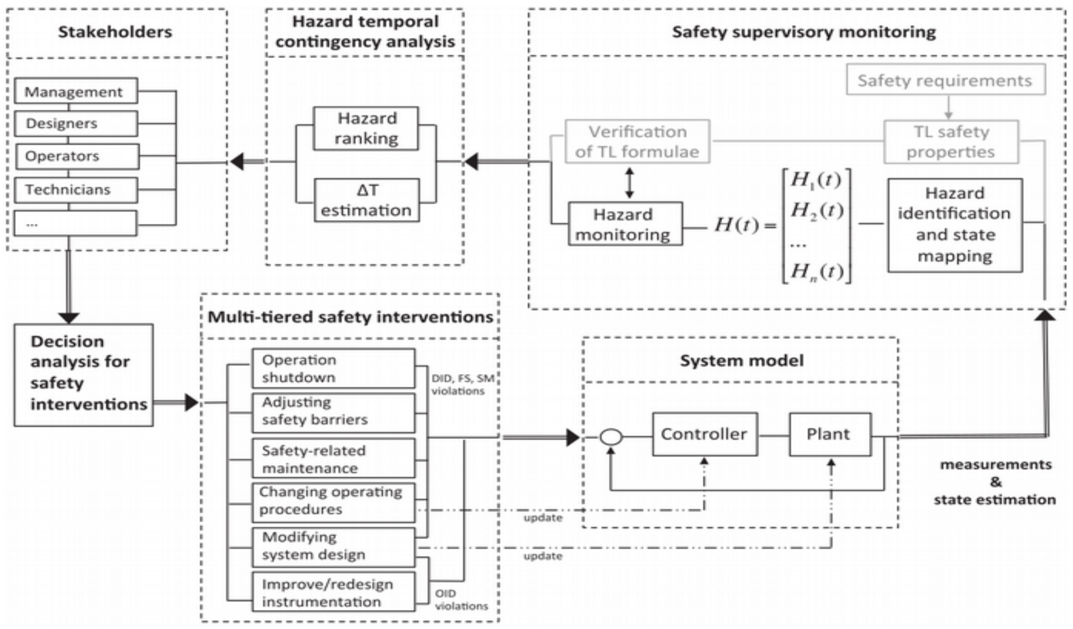
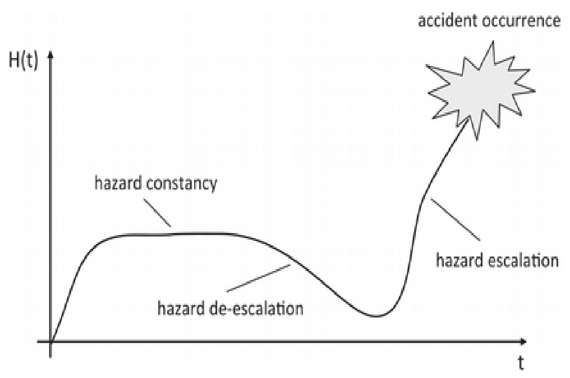

Stakeholders
Management
Designers
Operators
Technicians
...
Hazard temporal contingency analysis
Hazard ranking
ΔT estimation
Safety supervisory monitoring
Safety requirements
Verification of TL formulae
TL safety properties
Hazard monitoring
$H(t) = \begin{bmatrix} H_1(t) \\ H_2(t) \\ ... \\ H_n(t) \end{bmatrix}$
Hazard identification and state mapping
Decision analysis for safety interventions
Multi-tiered safety interventions
Operation shutdown
Adjusting safety barriers
Safety-related maintenance
Changing operating procedures
Modifying system design
Improve/redesign instrumentation
DID, FS, SM violations
OID violations
update
update
System model
Controller
Plant
measurements & state estimation

Fig. 55 Overview of model-based safety supervisory control, & dynamic hazard monitoring for safety interventions [46]



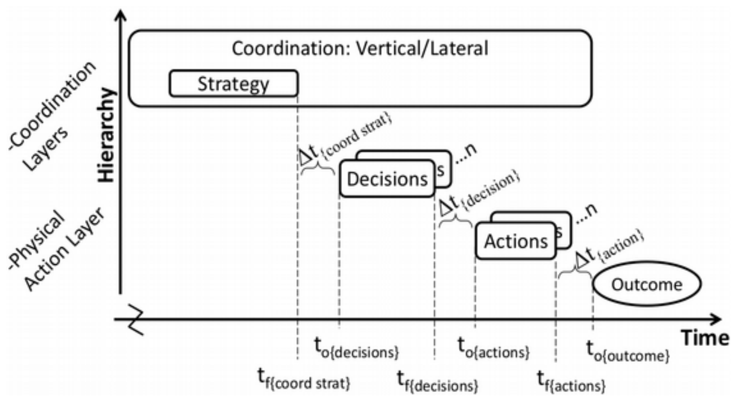
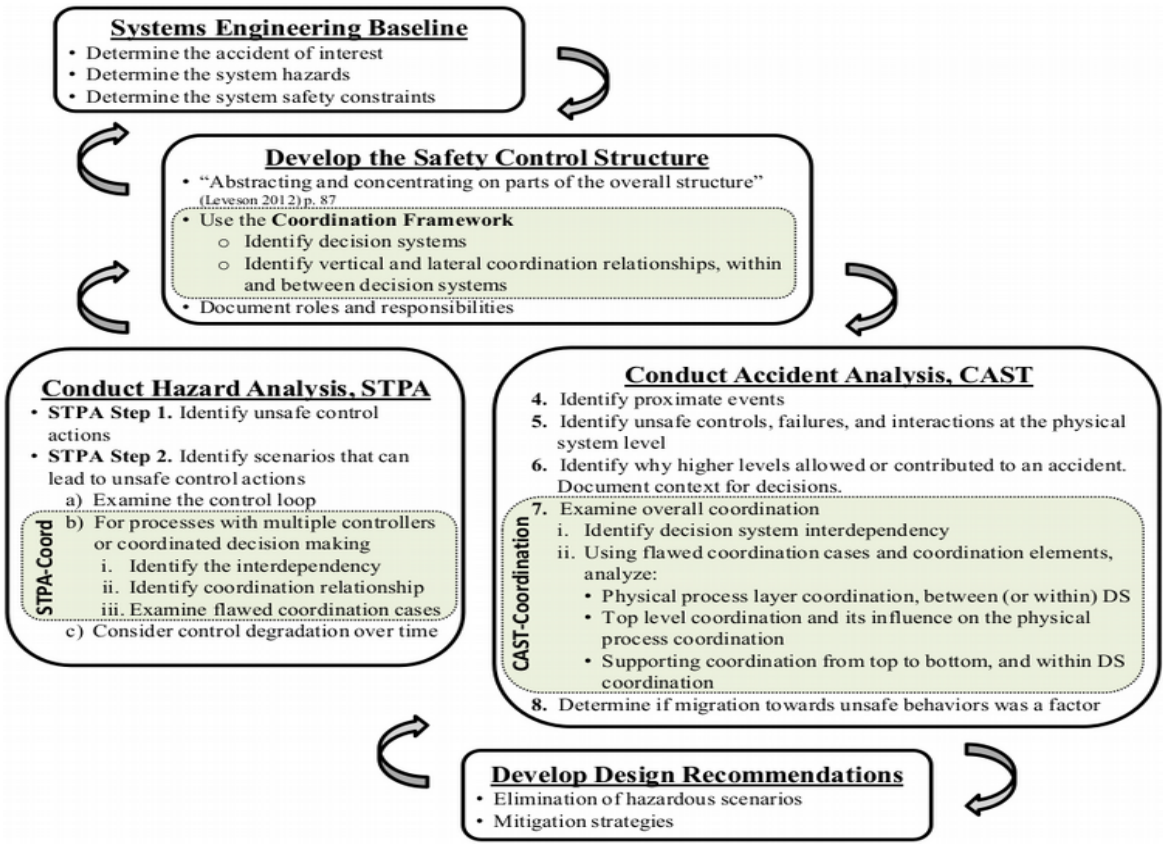


Fig. 56 Illustration of hazard level dynamics [46]   *Fig. 57 Coordination of Interdependent Activities: Ex. Timeline [47]*

Figures 57 and 58 (from [47]) highlight the extension of systems-theoretic safety analyses to incorporate the need for improved coordination across the hierarchy of inter-dependent activities and events (i.e. strategies, decisions, actions,, outcomes) that are inherent to monitoring and managing safety related concerns (i.e. potential hazards).



Systems Engineering Baseline
- Determine the accident of interest
- Determine the system hazards
- Determine the system safety constraints

Develop the Safety Control Structure
- "Abstracting and concentrating on parts of the overall structure" (Leveson 2012) p. 87
- Use the **Coordination Framework**
  - o Identify decision systems
  - o Identify vertical and lateral coordination relationships, within and between decision systems
- Document roles and responsibilities

Conduct Hazard Analysis, STPA
- **STPA Step 1.** Identify unsafe control actions
- **STPA Step 2.** Identify scenarios that can lead to unsafe control actions
  - a) Examine the control loop
  - STPA-Coord
  - b) For processes with multiple controllers or coordinated decision making
    - i. Identify the interdependency
    - ii. Identify coordination relationship
    - iii. Examine flawed coordination cases
  - c) Consider control degradation over time

Conduct Accident Analysis, CAST
4. Identify proximate events
5. Identify unsafe controls, failures, and interactions at the physical system level
6. Identify why higher levels allowed or contributed to an accident. Document context for decisions.

CAST-Coordination

7. Examine overall coordination
   - i. Identify decision system interdependency
   - ii. Using flawed coordination cases and coordination elements, analyze:
     - Physical process layer coordination, between (or within) DS
     - Top level coordination and its influence on the physical process coordination
     - Supporting coordination from top to bottom, and within DS coordination
8. Determine if migration towards unsafe behaviors was a factor

Develop Design Recommendations
- Elimination of hazardous scenarios
- Mitigation strategies

Fig. 58 Systems Approach to Safety: Incorporation of Coordination Capabilities [47] (STPA - (System-Theoretic Process Analysis; CAST - Causal Analysis based on STAMP; STAMP - Systems-Theoretic Accident Model and Processes)

The proactive latency-aware (PLA) adaptation is an approach for self-adaptive systems that improves over reactive adaptation by considering both the current and anticipated adaptation needs of the system, and taking into account the latency of adaptation tactics so that they can be started with the necessary lead time, as illustrated by figure 59 (from [57]). Figure 60 (from [57]) is an illustrative example of the model-based latency-aware proactive adaptation algorithm. Using dynamic programming and relying on a prediction of the environment for the duration of a system run, their algorithm can find the adaptation decision that at each time step maximizes the future aggregate utility, while accounting for the penalty of switching configurations. This is another example of an emerging capability that can help address timing and coordination related challenges.

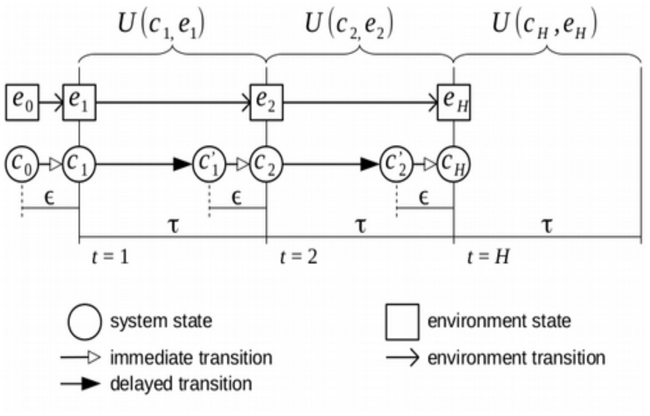
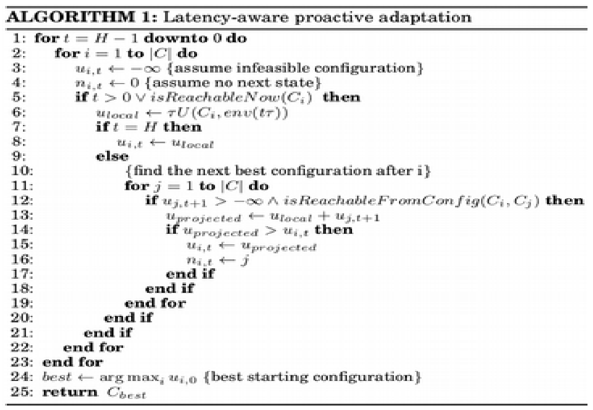


Fig. 59 Adaptation Transition Patterns: Example [57]



```
ALGORITHM 1: Latency-aware proactive adaptation
for $t = H - 1$ downto 0 do
    for $i = 1$ to $|C|$ do
        $u_{i,t} \leftarrow -\infty$ {assume infeasible configuration}
        $n_{i,t} \leftarrow 0$ {assume no next state}
        if $t > 0 \lor isReachableNow(C_i)$ then
            $u_{local} \leftarrow \tau U(C_i, env(t\tau))$
            if $t = H$ then
                $u_{i,t} \leftarrow u_{local}$
            else
                {find the next best configuration after i}
                for $j = 1$ to $|C|$ do
                    if $u_{j,t+1} > -\infty \land isReachableFromConfig(C_i, C_j)$ then
                        $u_{projected} \leftarrow u_{local} + u_{j,t+1}$
                        if $u_{projected} > u_{i,t}$ then
                            $u_{i,t} \leftarrow u_{projected}$
                            $n_{i,t} \leftarrow j$
                        end if
                    end if
                end for
            end if
        end if
    end for
end for
$best \leftarrow \arg\max_i u_{i,0}$ {best starting configuration}
return $C_{best}$
```

Fig. 60 Latency-Aware Adaptation: Ex. Algorithm [57]

Figures 61 and 62 (from [55]) illustrate how both the PLA and the Control-based Requirements-oriented Adaptation (CobRA) framework map into the MAPE-K (i.e. monitor – analyze – plan – execute – knowledge) model of autonomic-computing. As highlighted in the daigram, the different MAPE stages share a knowledge base that integrates them, and cover the activities that are carried out by the control loop, namely: (i) monitoring the system and the environment; (ii) analyzing the information monitored and deciding whether the system has to adapt; (iii) planning the best course of action for adaptation; and (iv) executing adaptation. The CobRA approach also follows the MAPE-K model, combining principles from Requirements Engineering and Control Theory. The architecture of CobRA is depicted in figure 62, where each component corresponds to one of the MAPE stages.

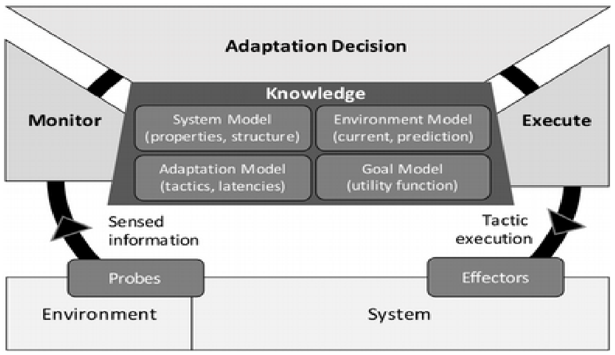

Adaptation Decision
Knowledge
Monitor
Execute
System Model (properties, structure)
Environment Model (current, prediction)
Adaptation Model (tactics, latencies)
Goal Model (utility function)
Sensed information
Tactic execution
Probes
Effectors
Environment
System

Fig. 61 Proactive Latency-Aware (PLA) Adaptation Loop [55]

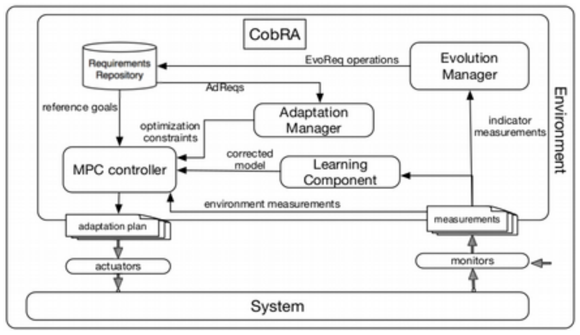


Fig. 62 Control-based Requirements-oriented Adaptation (CobRA) Framework [55]

This correspondence of the MAPE-K model with both PLA and CobRA, demonstrates an ability to map autonomic-computing oriented adaptation-strategies across various levels of dynamic control. For developing link-autonomy capabilities, this is especially useful due to the cross-layer cross-platform aspects of the respective freespace LOS data-link connectivity. Note that the respective communications terminals primarily reside within the physical and data layers of the protocol stack.

8. Discussion

For developing link-autonomy and associated experimentation support for data-transport, data-mediation, and data-synchronization, the previous sections provided representative examples of the areas of work that need to be addressed. Much of the discussion was from a MBSE/SOSE perspective and position of adopting, tailoring, and merging models/methods that are available within their respective application domains. In contrast to the more immediate incremental focus of the previous sections, this section focuses on the long-term end-goals, objectives, and desired outcomes.

As indicated within the previous sections, much of the technologies surveyed can incorporate and immediately leverage available models, frameworks, and resources, while also providing the basis for a roadmap for longer lead-time objectives. A key distinction of this effort is the intent to incorporate taxonomies of functions, algorithms, and their generic context-dependent utilization. With this context, an added objective is to more explicitly incorporate conceptual and abstract models that are otherwise more removed from the engineering and implementation details of ITI frameworks. The end-goal is to at least provide an explicitly defined conceptual accounting and mapping for MBSE/SOSE based development of ITI frameworks. This type of conceptual milestone for MBSE/SOSE enables the ability to more directly work through and further develop the foundational aspects of the respective end-goals.

The initial steps are underway for developing this type of virtualization of algorithmic and ITI services capabilities. Development of virtual networks of representative self-adaptive RF/microwave LOS nodes provides a basis and foundation for assessing how to best accommodate and support development/experimentation for such highly-virtualized performance-metric oriented services.

9. Summary and Conclusion

Agile network-enabled C2 requires a distributed-computing infrastructure that is tolerant to D-DIL communication/networking conditions. This is especially the case for time-sensitive mission-tasks operating within a variety of environments. Thus, there is a critical need for more explicitly modeling and understanding the different types of dynamically changing sources of degradation that, in turn, can cause different types of D-DIL conditions that impact mission success. The overall focus of this paper has been to further survey the applicable technologies, models, and resources, while addressing the practical challenges of implementing a MBSE/SOSE based framework that focuses on the need for improved adaptive self-healing (i.e. autonomic) link-autonomy and model-level I&I capabilities.

As highlighted within the previous sections, much progress has been made within a range of domains and areas of specialization (i.e. domains of discourse), wherein each separately addresses various aspects of the associated ID/NCW ITI S&T needs.

Section two provided background information that helped to provide a working introduction to the need and associated payoff for developing the type of capabilities descussed herein. This background information included discussion of previous work with multi-tier goal-oriented workflow management is covered within this context of ISR mission-support. Within section two, complementary connectivity and link management efforts are also discussed in some detail. For supporting D-DIL conditions within challenging environments, various aspects of inter-related data-synchronization services (e.g. C2SS) and directional LOS ad-hoc networking (e.g. connectivity management, link management) efforts were also surveyed and reviewed.

Section three covered examples of standardized architectures and frameworks within applicable application and technology domains. For application-domain related architectures, a FMV example highlighted the incorporation of ESB reference architectures for coordinating and orchestrating FMV dataflow between producer-consumer endpoints. Within this context of ESB architectures, examples of adaptive ESB architecture frameworks were introduced and discussed.

With this initial introduction to adaptive enterprise computing components (e.g. adaptive ESB frameworks), self-optimizing enterprise architecture frameworks such as SDNs and SDEs were also discussed to further emphasize the emerging trend towards highly-virtualized service-oriented cloud-computing infrastructures. The continuous-optimization and continuous-assurance (i.e. enterprise services) capabilities of emerging SDN/SDE architecture frameworks were also covered to highlight the emerging trend towards intrinsically adaptive, self-healing, and resilient/robust enterprise infrastructures. Finally, within section three, autonomic-computing models and associated frameworks, such as Rainbow, were introduced and discussed within this context of adaptive enterprise-computing infrastructure.

Section four covered metrics, indicators, and capability assessment models, within applicable technology domains. Examples from wireless networking and quality of service (QoS) focused best-practices and standards are covered within this section.

Section five covered emerging technologies and architectures for self-adaptive and self-optimizing systems. Applicable streaming-video and wireless networking examples are highlighted and discussed. An example of a self-optimizing wireless-networking architecture that focuses on the multi-objective optimization of inter-related objectives (e.g. coverage, quality of service, throughput, latency/delay, energy efficiency, cost) was noted as particularly applicable.

Section six reviewed applicable examples of risk management, reliability, and safety-engineering. Within this section, a number of standardized processes were highlighted, with a particular emphasis on the opportunity to re-purpose and tailor/augment such standards for development/certification of adaptive LOS data-links that demonstrate link-autonomy.

Section seven provided a high-level review of example change and adaptation models that focus on particular features of interest for developing link-autonomy: (i) design-pattern oriented approaches for developing adaptation strategies and tactics; (ii) latency-awareness oriented methods and techniques that address timing, coordination, and proactive adaptation needs.

Section eight further discussed the various aspects of technology domains and respective examples that were reviewed and surveyed within the respective sections. The need for adoption and tailoring of the surveyed models, architectural frameworks, standards, and best practices, was also discussed.

While incrementally addressing such challenges, the goal is to more effectively address the need to establish S&T roadmaps that incorporate possibly more disruptive and transformational technology development increments. In principle, the desired MBSE/SOSE approach should anticipate both a near-term, as well as, longer-term evolution towards more fully virtualized self-adaptive distributed-computing infrastructures. Furthermore, such infrastructures need to better represent and incorporate the elements of conceptual models and architectural components of resilient systems, autonomic frameworks, software defined environments, and cognitive networking technologies.

10. Future Work and The Way-Ahead

This ongoing technology survey has been motivated by the desire to develop a MBSE/SOSE process that addresses conceptual I&I needs for EA/SOA/Cloud and related reference models, patterns, best practices, standards, and available resources. In particular, recently open-sourced resources are recognized as examples of inter-related, as well as inter-dependent, code bases that can be potentially co-evolved and tailored to address data-transport, data-mediation, and data-synchronization needs.

The role and importance of self-adaptive time-sensitive data-transport, data-mediation, and data-synchronization services, relative to information-exchange and interaction with unmanned systems is area of future work. For tactical ISR, the impact of D-DIL conditions on machine-to-machine, machine-to-human, and human-to-human interactions (and collaboration) are of particular interest. Working with robotics oriented projects and respective subject matter experts, such S&T challenges and associated capability gaps can be further addressed.

Acknowledgements

The ONR visiting faculty and NREIP/SEAP student internship program has provided a basis of academic outreach that has also facilitated this ongoing multi-disciplinary research effort. Peer-review and suggestions from collaborators, colleagues, and other stakeholders, have helped to significantly improve early drafts through ongoing active dialog and collaboration. The authors want to also acknowledge support from the SPAWAR Systems Center Pacific (SSC Pacific) Naval Innovative Science and Engineering (NISE) program. Finally, the authors most particularly thank the program sponsors and respective project managers who have provided continued support. We are especially thankful for the opportunity to develop S&T results into net-enabled operations-support services that address high-priority Warfighter needs and capability gaps. The authors also want to thank the reviewers of the original draft paper and their thoughtful comments and suggestions that helped to further improve the final draft.

Technical References

[1] Heald, K., "Whole of Government Information Sharing & Safeguarding Priorities," Information Sharing Environment (ISE) Presentation, Enterprise Architecture Conference, 2011 (http://goveaconference.com/Events/2011/Sessions/Tuesday/Session-2.5.aspx)
[2] Kwak, K. J., et al., "Airborne network evaluation: challenges and high fidelity emulation solution", Comms Magazine, 2014 (http://www.i-a-i.com/public/rfnest/docs/airborne_network_evaluation.pdf)
[3] Durham, J., and A. Daswani, "Enterprise Lexicon Services (ELS): Pilot Tools and Web-Services for Acquisition and Mission Support", DoD Enterprise Architecture Conference, 13 May 2010 (afei-www.kzoinnovations.com/afei/ppt/durham.pdf)
[4] Durham, J., "Enterprise Lexicon Services (ELS)", The 13th Business Rules & Decision Forum, November 2009 (http://www.businessrulesforum.com/abstracts.php?id=301)
[5] Durham, J., L. McLauchlan, and R. Yuster. "Enabling a common and consistent enterprise-wide terminology: An initial assessment of available tools", Web Intelligence and Intelligent Agent Technology Conference, 2008. (http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4740506)
[6] McGirr, S. C., R. J. Wroblewski, and E. L. Dorman, "Active Wiki Knowledge Repository", DTIC, 2012 (http://www.dtic.mil/dtic/tr/fulltext/u2/a584803.pdf)

[7] Perkins, R., F. Dejesus, J. Durham, R. Hastings, and J. McDonnell, "C2 Data Synchronization in Disconnected, Intermittent, and Low-bandwidth (DIL) Environments", ICCRTS, 2013 (http://www.dodccrp.org/events/18th_iccrts_2013/post_conference/papers/097.pdf; http://www.dodccrp.org/events/18th_iccrts_2013/post_conference/presentations/097.pdf)
[8] Durham, J., R. Zeller-Townson, L. McLauchlan, M. Mehrubeoglu, R. Cardenas, F. Dejesus, J. McDonnell, "Network-Centric Operations Support: Lessons Learned, Status, and Way-Ahead" (Paper), 19th ICCRTS, 2014 (http://dodccrp.org/events/19th_iccrts_2014/track_chair/papers/055.pdf)
[9] Durham, J., and Zeller-Townson, R., "ISR Communication Services: Description of Connectivity Management", working slides, 2014
[10] Blais, C., "Application of coalition battle management language (C-BML) and C-BML services to live, virtual, and constructive (LVC) simulation environments", Winter Simulation Conference, 2011 (http://calhoun.nps.edu/public/bitstream/handle/10945/37270/blais_a558504.pdf)
[11] Gupton, K., et al. "Management of C4I and M&S Data Standards with Modular OWL Ontologies." 2011 (http://calhoun.nps.edu/public/bitstream/handle/10945/30791/11S-SIW-061.pdf).
[12] Bautista, J., S. Metzmaker, T. Rhodes, "Navy Full Motion Video", UCSD Capstone Project, 2013 (http://jacobsschool.ucsd.edu/mas/aese/team_project/SSC-SAIC/NFMV_Final_Report_Outline.pdf; http://jacobsschool.ucsd.edu/mas/aese/team_project/SSC-SAIC/NFMV_Final_Presentation.pdf)
[13] Buyya, Rajkumar, et al., "Software-Defined Cloud Computing: Architectural Elements and Open Challenges", ICACCI, 2014 (http://arxiv.org/pdf/1408.6891)
[14] Li, C., et al. "Software defined environments: An introduction." IBM Journal of Research and Development, 2014 (http://rboutaba.cs.uwaterloo.ca/Courses/CS856-F14/Papers/06798712.pdf)
[15] Quintero, D., et al., "IBM Software Defined Environment (SDE)", Draft, 12 Jan 2015 (http://www.redbooks.ibm.com/redpieces/abstracts/sg248238.html?Open)
[16] OME Committee, "Software-defined networking: The new norm for networks", ONF, 2012 (https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf)
[17] Mininet, acc'd April 2015 (http://mininet.org/overview/)
[18] Smith, P., et al., "Management patterns: SDN-enabled network resilience management", IEEE NOMS, 2014 (https://www.seccrit.eu/upload/PID3058477.pdf)
[19] Cetinkaya, E. K., "Modelling and Design of Resilient Networks under Challenges", 2013 (http://kuscholarworks.ku.edu/dspace/bitstream/1808/12985/1/Cetinkaya_ku_0099D_13157_DATA_1.pdf)
[20] Qadir, Qahhar Muhammad, Alexander Kist, and Zhongwei Zhang. "Mechanisms for QoE optimisation of Video Traffic: A review paper." Australasian Journal of Information, Communication Technology and Applications 1.1 (2015): 1-18 (http://eprints.usq.edu.au/26943/13/Qadir_Kist_Zhang_PV.pdf)
[21] González, L., and R. Ruggia, "Addressing QoS issues in service based systems through an adaptive ESB infrastructure", Proc. 6th Workshop on Middleware for Service Oriented Computing, ACM, 2011 (http://svn.wso2.org/repos/wso2/scratch/papers/mediation/ieee/papers/adaptive-esb.pdf)
[22] Morand, Denis, Issac Garcia, and Philippe Lalanda. "Autonomic enterprise service bus." Emerging Technologies & Factory Automation (ETFA), 2011 IEEE 16th Conference on. IEEE, 2011 (https://hal.archives-ouvertes.fr/hal-00746017/document)
[23] Morand, Denis, Issac Garcia, and Philippe Lalanda. "Towards autonomic enterprise service bus." Workshop on Middleware and Architectures for Autonomic and Sustainable Computing. ACM, 2011 (http://hal.univ-grenoble-alpes.fr/docs/00/74/86/57/PDF/Morand-Garcia-Lalanda_MAASC2011.pdf)
[24] Cilia, home page (http://wikiadele.imag.fr/index.php/Cilia; http://adele.imag.fr/cilia-an-autonomic-component-based-mediation-framework/)

[25] Kephart, Jeffrey O., and David M. Chess. "The vision of autonomic computing." Computer 36.1 (2003): 41-50 (http://www.cs.cmu.edu/~15849g/readings/kephart03.pdf)
[26] Cheng, Shang-Wen, and David Garlan. "Stitch: A language for architecture-based self-adaptation." Journal of Systems and Software 85.12 (2012): 2860-2875 (http://repository.cmu.edu/cgi/viewcontent.cgi?article=2002&context=isr)
[27] Verbancsics, Phillip, and Douglas S. Lange. Using Autonomics to Exercise Command and Control of Networks in Degraded Environments. Space And Naval Warfare Systems Center Pacific San Diego CA, 2013 (http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA587015)
[28] Gerostathopoulos, Ilias, et al. "Meta-Adaptation Strategies for Adaptation in Cyber-Physical Systems." Software Architecture. Springer International Publishing, 2015. 45-52 (http://d3s.mff.cuni.cz/publications/download/D3S-TR-2015-01.pdf)
[29] Carroll, N., M. Helfert, and T. Lynn, "Towards the Development of a Cloud Service Capability Assessment Framework", Continued Rise of the Cloud, Springer London, 2014 (http://www.researchgate.net/profile/Noel_Carroll2/publication/259608593_Towards_the_Development_of_a_Cloud_Service_Capability_Assessment_Framework/links/53e5550d0cf21cc29fcf81df.pdf)
[30] Malik, Aqsa, et al., "QoS in IEEE 802.11-Based Wireless Networks: A Contemporary Survey", arXiv preprint arXiv:1411.2852, 2014 (http://arxiv.org/pdf/1411.2852)
[31] Iqbal, Muhammad, et al. "Wireless Sensor Network Optimization: Multi-Objective Paradigm." Sensors 15.7 (2015): 17572-17620. (http://www.mdpi.com/1424-8220/15/7/17572/htm)
[32] Yang, Zhuoqun, et al. "Review on Requirements Modeling and Analysis for Self-Adaptive Systems: A Ten-Year Perspective." arXiv:1704.00421, 2017 (https://arxiv.org/pdf/1704.00421)
[33] Andersson, Jesper, et al. "Modeling dimensions of self-adaptive software systems." Software engineering for self-adaptive systems. Springer Berlin Heidelberg, 2009. 27-47 (http://www.academia.edu/download/30810809/ebooksclub.org__Software_Engineering_for_Self_Adaptive_Systems__Lecture_Notes_in_Computer_Science___Programming_and_Software_Engineering.pdf#page=36)
[34] Software Engineering for Self-Adaptive Systems: Assurances, Dagstuhl Seminar 13511, 2014 (https://www.dagstuhl.de/en/program/calendar/semhp/?semnr=13511; http://boemund.dagstuhl.de/mat/index.en.phtml?13511).
[35] Cunha, Carlos Augusto da Silva. Self-healing Techniques for Video-streaming Applications. Diss. 2016 (https://estudogeral.sib.uc.pt/jspui/bitstream/10316/29628/3/Self-healing%20Techniques%20for%20Video-streaming%20Applications.pdf)
[36] Klaue, Jirka, Berthold Rathke, and Adam Wolisz. "Evalvid–A framework for video transmission and quality evaluation." International Conference on Modelling Techniques and Tools for Computer Performance Evaluation. Springer Berlin Heidelberg, 2003 (http://www.tkn.tu-berlin.de/fileadmin/fg112/Hard_Software_Components/Software/evalvid.pdf)
[37] Febrero, Felipe, Coral Calero, and M. Ángeles Moraga. "Software reliability modeling based on ISO/IEC SQuaRE." Information and Software Technology 70 (2016): 18-29 (http://shop.tarjomeplus.com/Uploads/site-1/DownloadDoc/903.pdf)
[38] Stoicescu, Miruna, Jean-Charles Fabre, and Matthieu Roy. "Architecting Resilient Computing Systems: A Component-Based Approach for Adaptive Fault Tolerance." Journal of Systems Architecture, 2017 (https://hal.archives-ouvertes.fr/hal-01472877/document)
[39] Trapp, Mario. Assuring Functional Safety in Open Systems of Systems. Diss. Habilitationsschrift, Kaiserslautern, Technische Universität Kaiserslautern, 2016 (https://kluedo.ub.uni-kl.de/files/4422/_Assuring+Functional+Safety+in+Open+Systems+of+Systems.pdf)
[40] Joint Software Systems Safety Engineering Handbook, Naval Ord. Safety & Security Activity, Ver.

1.0, 2010 (http://www.acq.osd.mil/se/docs/Joint-SW-Systems-Safety-Engineering-Handbook.pdf)
[41] MIL-STD-882E, 2012 (http://www.dtic.mil/ndia/2012system/track914863.pdf; http://www.system-safety.org/Documents/MIL-STD-882E.pdf)
[42] Berzins, Valdis. Combining risk analysis and slicing for test reduction in open architecture. No. NPS-AM-14-C11P07R03-038, NPS, 2014 (http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA624722)
[43] Avizienis, Algirdas, et al. "Basic concepts and taxonomy of dependable and secure computing." IEEE transactions on dependable and secure computing 1.1 (2004): 11-33 (http://drum.lib.umd.edu/bitstream/handle/1903/6459/TR_2004-47.pdf?sequence=1)
[44] Berzins, Valdis, et al. Use of Automated Testing to Facilitate Affordable Design of Military Systems. No. SYM-AM-15-097. Naval Postgraduate School Monterey Ca, 2015 (http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA623091)
[45] Bonne, Susan M., and Jason K. Rupert. Use of Unified Modeling Language (UML) in Model-Based Development (MBD) For Safety-Critical Applications. No. RDECOM-TR-RDMR-BA-14-01. Army Aviation And Missile Research Development And Eng Ctr Redstone Arsenal Al Software Engineering Directorate, 2014 (http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA621409)
[46] Favarò, Francesca M., and Joseph H. Saleh. "Toward risk assessment 2.0: Safety supervisory control and model-based hazard monitoring for risk-informed safety interventions." Reliability Engineering & System Safety 152 (2016): 316-330 (https://www.researchgate.net/profile/Joseph_Saleh/publication/301567375_Toward_risk_assessment_20_Safety_supervisory_control_and_model-based_hazard_monitoring_for_risk-informed_safety_interventions/links/571aa23908ae7f552a4734e6.pdf)
[47] Johnson, Kip Edward. Systems-Theoretic Safety Analyses Extended for Coordination. Diss. Wright State University, 2017 (http://sunnyday.mit.edu/johnson-dissertation.pdf)
[48] Šljivo, Irfan, et al. "A method to generate reusable safety case argument-fragments from compositional safety analysis." Journal of Systems and Software, 2016 (http://www.es.mdh.se/pdf_publications/4435.pdf)
[49] Kriaa, Siwar. Joint safety and security modeling for risk assessment in cyber physical systems. Diss. Université Paris-Saclay, 2016 (https://tel.archives-ouvertes.fr/tel-01318118/file/77299_KRIAA_2016_archivage.pdf)
[50] Hissam, Scott A., Sagar Chaki, and Gabriel A. Moreno. "High assurance for distributed cyber physical systems." Proceedings of the 2015 European Conference on Software Architecture Workshops. ACM, 2015 (http://www.contrib.andrew.cmu.edu/~schaki/publications/ASDS-2015.pdf)
[51] Sljivo, Irfan, and Barbara Gallina. "Building multiple-viewpoint assurance cases using assumption/guarantee contracts." Proccedings of the 10th European Conference on Software Architecture Workshops. ACM, 2016 (http://www.es.mdh.se/pdf_publications/4509.pdf)
[52] Calinescu, Radu, et al. "Engineering Trustworthy Self-Adaptive Software with Dynamic Assurance Cases." arXiv preprint arXiv:1703.06350, 2017 (https://arxiv.org/pdf/1703.06350)
[53] Pahl, Claus, P. Jamshidi, and D. Weyns. "Cloud architecture continuity: Change models and change rules for sustainable cloud software architectures." J. of Software: Evolution & Process (2016)
[54] Ramirez, A. J., and B. HC Cheng. "Design patterns for developing dynamically adaptive systems." Proc. of ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems. ACM, 2010 (https://pdfs.semanticscholar.org/fd3a/831104b0bdc7c860b0ebede01fd4e6679bce.pdf)
[55] Moreno, Gabriel A., et al. "Comparing model-based predictive approaches to self-adaptation: CobRA and PLA." Proceedings of the 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. IEEE Press, 2017

(http://acme.able.cs.cmu.edu/pubs/uploads/pdf/PLAvsCobRA.pdf)
[56] Angelopoulos, Konstantinos, et al. "Model predictive control for software systems with CobRA." Proceedings of the 11th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. ACM, 2016 (http://www.inf.ufes.br/~vitorsouza/wp-content/papercite-data/pdf/angelopoulos-et-al-seams16.pdf)
[57] Moreno, Gabriel A., et al. "Efficient decision-making under uncertainty for proactive self-adaptation." Autonomic Computing (ICAC), 2016 IEEE International Conference on. IEEE, 2016 (http://acme.able.cs.cmu.edu/pubs/uploads/pdf/moreno-plasdp.pdf)
[58] Cámara, Javier, et al. "Analyzing latency-aware self-adaptation using stochastic games and simulations." ACM Transactions on Autonomous and Adaptive Systems (TAAS) 10.4 (2016): 23 (http://acme.able.cs.cmu.edu/pubs/show.php?type=both&ord=year)

Glossary of Topics and Technical Concepts

---

1 Information Sharing Environment (ISE; http://en.wikipedia.org/wiki/Information_Sharing_Environment)
ISE home page (http://www.ise.gov/)
2 Information Dominance Corps (IDC; http://en.wikipedia.org/wiki/Information_Dominance_Corps)
Network-centric warfare (NCW; http://en.wikipedia.org/wiki/Network-centric_warfare)
NCOW (Network-Centric Operations and Warfare; http://en.wikipedia.org/wiki/NCOW)
3 Functional requirement (https://en.wikipedia.org/wiki/Functional_requirement)
Non-functional requirement (https://en.wikipedia.org/wiki/Non-functional_requirement)
Architecturally Significant Requirements (https://en.wikipedia.org/wiki/Architecturally_Significant_Requirements)
List of system quality attributes (https://en.wikipedia.org/wiki/List_of_system_quality_attributes)
4 Distributed operations (http://en.wikipedia.org/wiki/Distributed_operations)
Maneuver warfare (http://en.wikipedia.org/wiki/Maneuver_warfare)
5 Network-enabled capability (NEC; https://en.wikipedia.org/wiki/Network-enabled_capability)
Network-centric organization (https://en.wikipedia.org/wiki/Network-centric_organization)
6 Information and communications technology (ICT; https://en.wikipedia.org/wiki/Information_and_communications_technology)
Unified communications (https://en.wikipedia.org/wiki/Unified_communications)
Decision support system (https://en.wikipedia.org/wiki/Decision_support_system)
Augmented cognition (https://en.wikipedia.org/wiki/Augmented_cognition)
7 Cloud computing (https://en.wikipedia.org/wiki/Cloud_computing)
Distributed computing (https://en.wikipedia.org/wiki/Distributed_computing)
Computer network (https://en.wikipedia.org/wiki/Computer_network)
Cyberspace (https://en.wikipedia.org/wiki/Cyberspace)
8 Continual improvement process (https://en.wikipedia.org/wiki/Continual_improvement_process)
Converged infrastructure (https://en.wikipedia.org/wiki/Converged_infrastructure)
Shared services (https://en.wikipedia.org/wiki/Shared_services)
9 Model-driven engineering (MDE; http://en.wikipedia.org/wiki/Model-driven_engineering)
Systems engineering (SE; https://en.wikipedia.org/wiki/Systems_engineering)
Model-based systems engineering (MBSE; https://en.wikipedia.org/wiki/Model-based_systems_engineering)
System of systems engineering (SOSE; http://en.wikipedia.org/wiki/System_of_systems_engineering)
10 Value chain (https://en.wikipedia.org/wiki/Value_chain)
Performance engineering (https://en.wikipedia.org/wiki/Performance_engineering)
Requirement (https://en.wikipedia.org/wiki/Requirement)
11 Application performance engineering (https://en.wikipedia.org/wiki/Application_performance_engineering)
12 Area of responsibility (AOR; https://en.wikipedia.org/wiki/Area_of_responsibility)
Line-of-sight propagation (https://en.wikipedia.org/wiki/Line-of-sight_propagation)
13 Internet Protocol (https://en.wikipedia.org/wiki/Internet_Protocol)
Internet protocol suite (https://en.wikipedia.org/wiki/Internet_protocol_suite)

14 JCSS Home Page (http://www.disa.mil/Mission-Support/Enterprise-Engineering/JCSS/About-Us)
Government off-the-shelf (GOTS; https://en.wikipedia.org/wiki/Government_off-the-shelf)
Commercial off-the-shelf (COTS; https://en.wikipedia.org/wiki/Commercial_off-the-shelf)

15 Network traffic simulation (https://en.wikipedia.org/wiki/Network_traffic_simulation)
Traffic generation model (https://en.wikipedia.org/wiki/Traffic_generation_model)

16 System integration (https://en.wikipedia.org/wiki/System_integration)
Interoperability (https://en.wikipedia.org/wiki/Interoperability)

17 Use case (https://en.wikipedia.org/wiki/Use_case)
Performance measurement (https://en.wikipedia.org/wiki/Performance_measurement)
Performance metric (https://en.wikipedia.org/wiki/Performance_metric)
Effectiveness (https://en.wikipedia.org/wiki/Effectiveness)
Organizational effectiveness (https://en.wikipedia.org/wiki/Organizational_effectiveness)
Organizational performance (https://en.wikipedia.org/wiki/Organizational_performance)
CCRP (https://en.wikipedia.org/wiki/Command_and_Control_Research_Program)
CCRP Code of best practice for experimentation (http://dodccrp.org/files/Alberts_Experimentation.pdf)

18 Virtualization (https://en.wikipedia.org/wiki/Virtualization)
Abstraction (computer science) (https://en.wikipedia.org/wiki/Abstraction_%28computer_science%29)
Encapsulation (https://en.wikipedia.org/wiki/Encapsulation_%28object-oriented_programming%29)
Virtual function (https://en.wikipedia.org/wiki/Virtual_function)
Generic function (https://en.wikipedia.org/wiki/Generic_function)
Function overloading (https://en.wikipedia.org/wiki/Function_overloading)
Late binding (https://en.wikipedia.org/wiki/Late_binding)
Duck typing (https://en.wikipedia.org/wiki/Duck_typing)
Multiple dispatch (https://en.wikipedia.org/wiki/Multiple_dispatch)
Dynamic dispatch (https://en.wikipedia.org/wiki/Dynamic_dispatch)

19 Hypervisor (https://en.wikipedia.org/wiki/Hypervisor)
Virtual machine (VM; https://en.wikipedia.org/wiki/Virtual_machine)
Hardware virtualization (https://en.wikipedia.org/wiki/Hardware_virtualization)

20 Network functions virtualization (NFV; https://en.wikipedia.org/wiki/Network_functions_virtualization)
Network virtualization (https://en.wikipedia.org/wiki/Network_virtualization)
I/O virtualization (https://en.wikipedia.org/wiki/I/O_virtualization)

21 Domain (software engineering) (https://en.wikipedia.org/wiki/Domain_%28software_engineering%29)
Domain of discourse (https://en.wikipedia.org/wiki/Domain_of_discourse)

22 Business activity monitoring (BAM; https://en.wikipedia.org/wiki/Business_activity_monitoring)

23 Universal Joint Task List (UJTL; https://en.wikipedia.org/wiki/Universal_Joint_Task_List)
UJTL, updated quarterly (http://www.dtic.mil/doctrine/training/ujtl_tasks.htm;
http://www.dtic.mil/doctrine/training/ujtl_tasks.pdf)
Universal Joint Task Manual, CJCSM 3500.04F, 1 June 2011
(http://www.dtic.mil/cjcs_directives/cdata/unlimit/m350004.pdf)
Joint Mission Essential Task List (JMETL) Development Handbook, Sept 2002
(http://www.dtic.mil/doctrine/training/trainingsystem/JMETLbook.pdf)
DJTEP, Instruction 310-50-4, 4 April 2016,
(http://www.disa.mil/~/media/Files/DISA/About/Publication/Instruction/di310504.pdf)

24 GQM (https://en.wikipedia.org/wiki/GQM)
Goal-Driven Measurement (http://www.sei.cmu.edu/measurement/tools/goaldriven/index.cfm)

25 GQM+Strategies (https://en.wikipedia.org/wiki/GQM%2BStrategies)

26 Business rule management system (BRMS; https://en.wikipedia.org/wiki/Business_rule_management_system)
Rule-based system (https://en.wikipedia.org/wiki/Rule-based_system)
Drools (https://en.wikipedia.org/wiki/Drools; http://www.drools.org/)
WildFly (https://en.wikipedia.org/wiki/WildFly; http://www.wildfly.org/)

27 Semantics of Business Vocabulary and Business Rules (SBVR;
https://en.wikipedia.org/wiki/Semantics_of_Business_Vocabulary_and_Business_Rules)
Universal Data Element Framework (UDEF; https://en.wikipedia.org/wiki/Universal_Data_Element_Framework)

Semantic interoperability (https://en.wikipedia.org/wiki/Semantic_interoperability)
28 Software design pattern (https://en.wikipedia.org/wiki/Software_design_pattern)
Dependency injection (https://en.wikipedia.org/wiki/Dependency_injection)
Adapter pattern (https://en.wikipedia.org/wiki/Adapter_pattern)
Aspect (computer programming) (https://en.wikipedia.org/wiki/Aspect_%28computer_programming%29)
Aspect weaver (https://en.wikipedia.org/wiki/Aspect_weaver)
29 Aspect-oriented software development (https://en.wikipedia.org/wiki/Aspect-oriented_software_development)
Aspect-oriented programming (AOP; https://en.wikipedia.org/wiki/Aspect-oriented_programming)
30 Abstract methods (https://en.wikipedia.org/wiki/Method_%28computer_programming%29#Abstract_methods)
Virtual function (https://en.wikipedia.org/wiki/Virtual_function)
Information hiding (https://en.wikipedia.org/wiki/Information_hiding)
31 Network topology (https://en.wikipedia.org/wiki/Network_topology)
Logical topology (https://en.wikipedia.org/wiki/Logical_topology)
Latency (engineering) (https://en.wikipedia.org/wiki/Latency_%28engineering%29)
Responsiveness (https://en.wikipedia.org/wiki/Responsiveness)
Throughput (https://en.wikipedia.org/wiki/Throughput)
Network delay (https://en.wikipedia.org/wiki/Network_delay)
Worst-case execution time (WCET; https://en.wikipedia.org/wiki/Worst-case_execution_time)
Bandwidth (computing) (https://en.wikipedia.org/wiki/Bandwidth_%28computing%29)
Bandwidth management (https://en.wikipedia.org/wiki/Bandwidth_management)
Traffic shaping (https://en.wikipedia.org/wiki/Traffic_shaping)
Active queue management (https://en.wikipedia.org/wiki/Active_queue_management)
Network scheduler (https://en.wikipedia.org/wiki/Network_scheduler)
Scheduling (computing) (https://en.wikipedia.org/wiki/Scheduling_%28computing%29)
Scheduling analysis real-time systems (https://en.wikipedia.org/wiki/Scheduling_analysis_real-time_systems)
Modeling and Analysis of Real Time and Embedded systems (MARTE;
https://en.wikipedia.org/wiki/Modeling_and_Analysis_of_Real_Time_and_Embedded_systems)
Model checking (https://en.wikipedia.org/wiki/Model_checking)
32 Reliability (computer networking) (https://en.wikipedia.org/wiki/Reliability_%28computer_networking%29)
Dependability (https://en.wikipedia.org/wiki/Dependability)
Availability (https://en.wikipedia.org/wiki/Availability)
Robustness (computer science) (https://en.wikipedia.org/wiki/Robustness_%28computer_science%29)
Fault tolerance (https://en.wikipedia.org/wiki/Fault_tolerance)
Resilience (network) (https://en.wikipedia.org/wiki/Resilience_%28network%29)
Survivability (https://en.wikipedia.org/wiki/Survivability)
33 Multilayered architecture (https://en.wikipedia.org/wiki/Multilayered_architecture)
Multitier architecture (https://en.wikipedia.org/wiki/Multitier_architecture)
Domain (software engineering) (https://en.wikipedia.org/wiki/Domain_%28software_engineering%29)
Domain engineering (https://en.wikipedia.org/wiki/Domain_engineering)
34 Object-oriented design (OOD; https://en.wikipedia.org/wiki/Object-oriented_design)
RM-ODP (https://en.wikipedia.org/wiki/RM-ODP)
ITU-T Rec. X.906 | ISO/IEC 19793 (http://www.lcc.uma.es/~av/download/UML4ODP_IS_V2.pdf)
Information hiding (https://en.wikipedia.org/wiki/Information_hiding)
Application programming interface (API; https://en.wikipedia.org/wiki/Application_programming_interface)
35 Coupling (computer programming) (https://en.wikipedia.org/wiki/Coupling_%28computer_programming%29)
Cohesion (computer science) (https://en.wikipedia.org/wiki/Cohesion_%28computer_science%29)
36 OSI model (https://en.wikipedia.org/wiki/OSI_model)
Abstraction layer (https://en.wikipedia.org/wiki/Abstraction_layer)
Scope (computer science) (https://en.wikipedia.org/wiki/Scope_%28computer_science%29)
37 Middleware (https://en.wikipedia.org/wiki/Middleware)
Middleware (distributed applications) (https://en.wikipedia.org/wiki/Middleware_%28distributed_applications%29)
38 Telecommunications link (https://en.wikipedia.org/wiki/Telecommunications_link)
Data link (https://en.wikipedia.org/wiki/Data_link)

Common Data Link (CDL; https://en.wikipedia.org/wiki/Common_Data_Link)
Tactical Data Link (https://en.wikipedia.org/wiki/Tactical_Data_Link)

39 OSI model (http://en.wikipedia.org/wiki/OSI_model)
Protocol stack (http://en.wikipedia.org/wiki/Protocol_stack)

40 Software as a service (SaaS; https://en.wikipedia.org/wiki/Software_as_a_service)
Data as a service (DaaS; https://en.wikipedia.org/wiki/Data_as_a_service)
Platform as a service (https://en.wikipedia.org/wiki/Platform_as_a_service)
Infrastructure as a Service (IaaS;
https://en.wikipedia.org/wiki/Cloud_computing#Infrastructure_as_a_service_.28IaaS.29)
Network as a service (NaaS; https://en.wikipedia.org/wiki/Network_as_a_service)

41 As a service (https://en.wikipedia.org/wiki/As_a_service)

42 Sequence diagram (https://en.wikipedia.org/wiki/Sequence_diagram)
System sequence diagram (https://en.wikipedia.org/wiki/System_sequence_diagram)
UML Interaction diagram (https://en.wikipedia.org/wiki/Unified_Modeling_Language#Interaction_diagrams)
Message sequence chart (https://en.wikipedia.org/wiki/Message_sequence_chart)

43 Battle management language (BML; https://en.wikipedia.org/wiki/Battle_management_language)

44 Capacity planning (https://en.wikipedia.org/wiki/Capacity_planning)
Provisioning (https://en.wikipedia.org/wiki/Provisioning)

45 Assertion (software development) (http://en.wikipedia.org/wiki/Assertion_%28software_development%29)
Design by contract (http://en.wikipedia.org/wiki/Design_by_contract)
Precondition (http://en.wikipedia.org/wiki/Precondition)
Postcondition (http://en.wikipedia.org/wiki/Postcondition)
Class invariant (http://en.wikipedia.org/wiki/Class_invariant)

46 Hollywood principle (https://en.wikipedia.org/wiki/Hollywood_principle)
Event-driven architecture (https://en.wikipedia.org/wiki/Event-driven_architecture)
Process control network (https://en.wikipedia.org/wiki/Process_control_network)
Distributed control system (DCS; https://en.wikipedia.org/wiki/Distributed_control_system)
Cyber-physical system (CPS; https://en.wikipedia.org/wiki/Cyber-physical_system)

47 Event-driven process chain (https://en.wikipedia.org/wiki/Event-driven_process_chain)
Workflow patterns (https://en.wikipedia.org/wiki/Workflow_patterns)
Workflow Patterns, home page (http://www.workflowpatterns.com/)
Business Process Model and Notation (BPMN; https://en.wikipedia.org/wiki/Business_Process_Model_and_Notation)
XPDL (https://en.wikipedia.org/wiki/XPDL)

48 Autonomic computing (https://en.wikipedia.org/wiki/Autonomic_computing)
Autonomic networking (https://en.wikipedia.org/wiki/Autonomic_networking)
Self-management (computer science) (https://en.wikipedia.org/wiki/Self-management_%28computer_science%29)

49 Concern (computer science) (https://en.wikipedia.org/wiki/Concern_%28computer_science%29)
Separation of concerns (https://en.wikipedia.org/wiki/Separation_of_concerns)
Abstraction principle (computer programming) (https://en.wikipedia.org/wiki/Abstraction_principle_
%28computer_programming%29)
Core concern (https://en.wikipedia.org/wiki/Core_concern)
Single responsibility principle (https://en.wikipedia.org/wiki/Single_responsibility_principle)

50 OODA loop (https://en.wikipedia.org/wiki/OODA_loop)
PDCA (https://en.wikipedia.org/wiki/PDCA)

51 Service Measurement Index (SMI; https://en.wikipedia.org/wiki/Service_Measurement_Index)

52 Quality of service (QoS; https://en.wikipedia.org/wiki/Quality_of_service)
Mobile QoS (https://en.wikipedia.org/wiki/Mobile_QoS)

53 Software quality (https://en.wikipedia.org/wiki/Software_quality)
Quality assurance (https://en.wikipedia.org/wiki/Quality_assurance)
Software quality assurance (https://en.wikipedia.org/wiki/Software_quality_assurance)
Software quality management (https://en.wikipedia.org/wiki/Software_quality_management)