

A standards-based Common Operational Environment

*Jaroslav Blaha
NATO ACCS Management Agency, Belgium¹*

About the Author

Jaroslav Blaha, born in the Czech Republic, is Senior Information Systems Engineer in the Project Implementation Division of the NATO ACCS Management Agency in Brussels. He worked as chief instructor at the German Air Force Technical Academy, and as project manager for communications and information systems at the NATO Headquarters in the Netherlands. Before joining NACMA he was Senior Analyst for mission-critical real-time systems at the NATO Programming Centre in Belgium. He holds university degrees in computer science and economics. He is a permanent member of the NATO COE working group since its establishment, and NACMA's lead for open system engineering. Special interests are software engineering, standardisation and human-machine interaction.

Mailing Address: NATO ACCS Management Agency (NACMA), Rue de Genève 8, B-1140 Brussels, Belgium; E-Mail: mail@jaroblaha.com

Descriptors

standard adoption, standard profiles, strategic products, best-of-breed, portability, runtime environment, interoperability, manageability, architecture, life-cycle management.

Special thanks for their support and contributions to

El Wells (Mitre), Dr. Fred Moxley (DISA), Wouter Konings (NC3A), Wolfgang Bauer (NACMA), and the members of the NATO COE Working Group.

¹ *This paper reflects solely the opinion of the author. It does not necessarily reflect the opinion, policy or intentions of the author's employer.*

A standards-based Common Operational Environment

ABSTRACT

For many years government and especially military organisations were known for their well engineered and meticulously specified, but proprietary systems. While in the past government development has driven many important standards (e.g. COBOL, TCP/IP), nowadays very few of the standards developed specifically for government purposes are being adopted by the civilian world. As government organisations have to rely increasingly on commercial solutions based on widely available and accepted standards, a policy for standards adoption and implementation, as opposed to the creation of additional standards, has become imperative.

Several nations and organisations have started initiatives to establish Common Operating Environments (COEs). A COE is a well-defined set of standards, standard profiles and strategic products, which implement those standards. The specified products are carefully selected and tested to be able to co-operate and interoperate in a certain environment. With this approach it is possible to centrally define the acceptable building blocks for application systems, which can be procured and implemented de-centrally with minimal risk. Within a COE special provisions are made to reduce the risk of the replacement of products which contribute to building blocks by better or newer products (best of breed). Other important aspects of a COE are people-portability, which mandates common human-machine interfaces and style-guides, interoperability and manageability, comprising run-time systems, installation-, configuration- and update-mechanisms.

Abstracting from NATO's COE, the paper describes the architecture, components and development steps of a COE. Together with some examples of COE components, this shall allow other organisations to assess the usability and feasibility of a COE for their environment. As the NATO acquisition and management procedures are not necessarily compatible with the commercial world, some organisation-specific aspects related to the COE have been deliberately omitted.

INTRODUCTION - THE NEED FOR A COE

NATO as an international organisation consists of representatives from 19 nations and has to satisfy the political, operational and technical requirements of all members. In addition to the development of specific NATO information systems, it has to consider and support means for interoperability with national systems. Taking into account the life-cycle-cost aspects, it becomes clear that a standardisation policy must consider the cost-effective maintenance, upgrade and replacement of system components and their interfaces.

Standardisation of NATO specific interfaces is a well-established process, which results in 'Standardisation Agreements' (STANAGs). Those are specifications of proprietary standards or of adaptations of international (e.g. ITU, ISO) standards. STANAGs have two major disadvantages: First, the process to develop a new or to modify an existing standard is lengthy; the process to get the specification ratified by the relevant nations is even longer. This can result in STANAGs, which do not reflect the state-of-the-art of standards. Second, as STANAGs often specify standards that are different from international or commercial standards, there is very little market and product support. This leads naturally to increased

development and procurement costs.

In military terms a Command, Control and Information System (CCIS) is the equivalent to a Management Support and Information System (MSS/MIS) in the commercial domain.

The COE efforts originated with the simple observation that in command, control and information systems certain functions (e.g. message exchange, tasking, communication interfaces) are so fundamental that they are required for virtually every CCIS. Yet these functions are built over and over again, in often-incompatible ways, even for systems with almost identical requirements. If such common functions could be extracted, implemented as a set of extensible building blocks, and made readily available to system designers, development schedules could be accelerated and substantial savings could be achieved (although the quantitative demonstration of cost effectiveness is a complex problem). Moreover, interoperability would be significantly improved because common software is used across systems and the functional capability only needs to be built correctly once rather than for each project (DII-COE, 1999).

The COE currently focuses on CCISs and mostly ignores special-to-purpose systems outside this domain (e.g. satellite control, onboard navigation systems). CCISs are currently the most common and therefore preferred domain. However, it is expected that the COE concept will over time be extended into non-CCIS areas. One example is the Domain Specific Software Architectures (DSSA) project of ARPA, which attempts to standardise real-time avionics architectures through an Avionics Domain Application Generation Environment (ADAGE) (ADAGE, 1999).

Throughout this paper the following terms will be used:

- Common applications, modules or sub-systems provide functionality, which is available and needed by effectively all users, independent of their specific task in the organisation. Examples are e-mail or word-processing.
- Functional Area Sub-Systems (FASS) and associated applications provide functionality, which is required only by a limited group of users specifically for their tasks. Examples are logistics or human-resource management.

An integrative approach is needed, which could provide guidance to project managers, engineers and budget authorities on how to

- derive generic technical requirements from existing operational requirements, i.e. by mapping of existing building blocks of technical solutions (e.g. an e-mail system) against a common class of operational needs (e.g. communicate tasks effectively with remote subordinates),
- select state-of-the-art standards, which support the desired functionality, and are implemented in commercially available and market-proven products to avoid bespoke developments,
- aggregate standards and the associated products into building blocks with well-known and tested characteristics, which can be assembled to operational systems with minimal development effort, and
- to reduce system development risks, and to ensure the system's ability to evolve in order to benefit from information technology advances.

In addition this approach had to be harmonised with ongoing standardisation activities, both on policy level (interoperability policies) and technical level (STANAG development). To allow smooth co-operation with the member nation's systems, the resulting NATO COE should also be harmonised and compatible with existing national COEs.

BACKGROUND - WHAT IS A COE?

The COE concept is an approach that is much broader in scope than just software reuse or standard compliance. The COE concept encompasses both, but its principles are far more reaching, as it comprises

- from a structural viewpoint
 - an architecture and approach for building interoperable and open systems,
 - an approach and methodology for software and data reuse,
 - an infrastructure comprised of a set of common standards and strategic products, to facilitate the required degrees of interoperability, portability and commonality in functional-area applications,
 - an integration process for achievement of a target operational system,
- from an implementation viewpoint
 - a collection of reusable building blocks, which can “plug and play” to provide services that are common to all applications (e.g. messaging, directory services),
 - a definition of the runtime execution environment,
 - a set of Application Program Interfaces (APIs) for accessing COE components,
 - an environment for sharing data between applications and systems,
- from an integrity assurance viewpoint
 - a reference implementation on which systems can be built, and
 - a toolset for enforcing COE principles and for evaluating compliance against a set of COE requirements.

The COE must be understood as a multi-faceted concept. The four major facets (DII-COE, 1999) are

- the COE as a system foundation,
- the COE as an architecture,
- the COE as a reference implementation, and
- the COE as an implementation strategy.

It is apparent that this multi-faceted view is complicated to comprehend in its entirety. However, each of those facets is important only for a specific group of personnel. E.g. the system foundation and architecture views are mostly important to IT policy makers and budgetary authorities, whereas a reference implementation and the implementation strategy are focal for the technical project management.

The COE as a system foundation

The COE is not a system; it is a foundation for building systems and there is only one COE regardless of the target system(s). Building a target system includes combining COE components with function-specific software. System designers select from the COE's set of building blocks (e.g., products) only those which are required for their application. As a result, for different COE compliant systems the basic functions (e.g., communications interfaces, operating system) are either identical (e.g. all systems use an Oracle database) or are selections from a restricted choice of components with well-defined and acceptable interoperability characteristics (e.g. Windows NT or Unix operating system, both providing POSIX compliant services). This implies that target systems utilise as far as possible common components for common functions as the architectural backbone. Additional function-specific software (either developed or purchased) accesses this foundation through well-defined and preferably standardised APIs or interfaces. Implementing proprietary algorithms or substituting components of the foundation by bespoke products would violate COE compliance and potentially jeopardise interoperability, and is therefore prohibited. COE compliant system design and component selection can be enforced through budgetary

mechanisms (i.e. no money for projects, which propose components that violate specific COE preconditions without proper rationale) and through process mechanisms (i.e. before project completion and acceptance, compliance has to be proven by interoperation with a reference system).

The COE as an architecture

An architecture is defined as the organisational structure of a system or component, their relationships, and the principles and guidelines governing their design and evolution over time (IEEE 610.12). Concerning the COE, one can distinguish three architectural views (JTA, 1999):

- The Operational Architecture is a description of the operational elements, assigned tasks, and information flows required for accomplishing or supporting a function. It defines the type of information, the frequency of exchange, and what tasks are supported by these information exchanges. The goal of a COE is to identify 'packages' of operational functions, which can be used as building blocks for a system's specification.
- The Technical Architecture defines a minimal set of rules for the characteristics (services, interfaces, standards) of, and the relationships between, the technical COE components. It provides the technical guidelines for implementation of systems upon which engineering specifications are based, common building blocks are built, and bespoke products are developed.
- The Systems Architecture is a description of the systems and interconnections providing for or supporting operational functions. It shows how those systems link and interoperate, describes their internal construction, defines the physical characteristics (e.g. connections, location, identification) of nodes and networks, and specifies system and component performance parameters. The implemented systems shall satisfy the Operational Architecture requirements through standards defined in the Technical Architecture.

The relation between those views is depicted in figure 1.

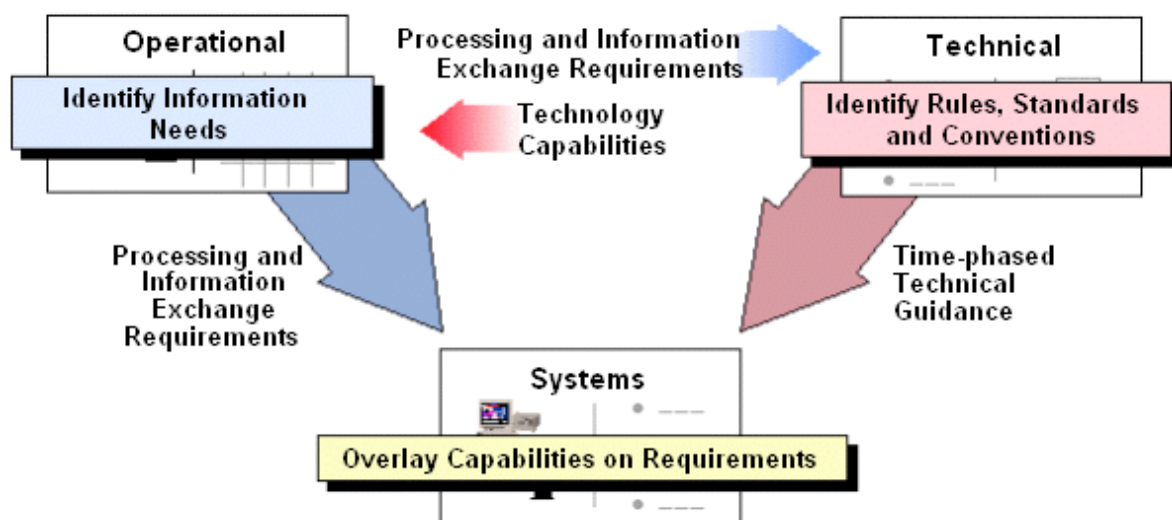


Figure 1 - Architecture relationships (JTA, 1999)

The COE as a reference implementation

The COE shall include an implementation of the components defined to be in the COE. A reference implementation is the key to reusability and interoperability. Use of the reference implementation provided as a testbed is required to assure interoperability and is therefore a

fundamental requirement for compliance. The reference implementation shall be used as an environment to test the integration and functionality of new products or technologies, while preserving backward compatibility.

The COE as an implementation strategy

The COE also is a basis for an evolutionary acquisition and implementation strategy by emphasising incremental development and fielding to reduce the time to 'market' for the user. This approach also referred to as “build a little - test a little - field a lot”, is a process of continually evolving a stable baseline to take advantage of new technologies as they mature and to introduce new capabilities. Especially in projects, which heavily rely on Commercial-Off-The-Shelf (COTS) products, evolutionary development has become the only practical means, as the traditional development cycle normally exceeds the technical obsolescence cycle. Specification and implementation of large-scale NATO projects with this evolutionary approach, based on some COE core concepts (i.e. mandatory open standards), shows promising results.

The NATO COE definition

A complete specification and implementation of the above COE requirements and characteristics is obviously an enormous and long-term task. Especially the US have a pioneer role through the already existing, and very successful, implementation of a COE compliant with the above (DII-COE, 1999). NATO started its COE efforts in 1997 and quickly had to face the large complexity of the undertaking. To be able, in the spirit of evolutionary development, to provide useful results early, it was decided to first concentrate on the definition and specification of the standards framework with the major aim to improve interoperability between NATO and national systems. A specification of the runtime execution environment is planned for a later phase. Although the NATO definition already incorporates the runtime execution environment, for the time being the focus is on the standardisation perspective.

Definition:

"The NATO COE is the standards-based computing and communications infrastructure, comprised of selected Off-The-Shelf (OTS) products and supporting services, that provides the structural foundation necessary to build interoperable and open systems. The NATO COE will facilitate a common understanding of the concepts, constructs and methods ... required for targeted systems. The NATO COE provides the set of building blocks and guidance necessary for effective open system design, development, implementation and integration, based, as much as possible, on market proven solutions. It supports an evolutionary systems development approach and provides the executable runtime environment necessary to facilitate the migration and implementation of integrated applications across NATO ... systems."

This paper will refrain from presenting or discussing specific standards or products, which have been selected within the NATO COE for two reasons. First, the goal of this paper is to present the concept of a COE and not a snapshot on selected items for a specific environment. Whereas the concept of the COE is independent of an organisation's structure and context, the components are selected specifically for an organisation's tasks and functional requirements. Second, the selection of components and standards is an ongoing process, which would make any detailed market-oriented list of selected items obsolete in short time.

Naturally, a COE in itself can be considered a standard for the organisation.

COE BUILDING BLOCKS

From a specification point of view, the COE consists of various building blocks, which provide dedicated information for the developer of a CCIS. These building blocks can provide helpful guidance even outside a COE compliant implementation project, as they are self-standing and independent. When combined through a process, which links them together they can be used directly as a roadmap from the initial operational requirements to the final selection and integration of products.

Operational requirements

Operational requirements are the starting point for any development of a system. They shall, from a user's point of view, define what functionalities the final system will provide. Two types of requirements can be distinguished:

- **Functionality requirements** specify the functions needed within a component or system by the user. They describe, on an abstract and implementation independent level, the types of data, the operations performed on these data, the formats for input and output, the performance desired etc.
- **Interoperability requirements** specify the behaviour and characteristics of (sub-)systems in relation to other (sub-)systems. They define the ability of systems to provide and accept services from other systems, and to use those services to enable them to operate effectively together. Interoperability naturally comprises external interfaces for data interchange, but also issues like people-portability and data-reusability.

From a COE perspective, interoperability requirements are an important aspect. In order to structure and classify interoperability requirements, several taxonomies were proposed within NATO for the structuring and automation of exchange and interpretation of data. The currently accepted approach proposes four degrees of interoperability as follows:

- **Degree 1: Unstructured Data Exchange** comprises the exchange of human-interpretable unstructured data such as the free text found in memos, reports and papers.
- **Degree 2: Structured Data Exchange** comprises the exchange of human-interpretable structured data intended for manual and/or automated handling, that requires manual compilation, receipt and/or message dispatch.
- **Degree 3: Seamless Sharing of Data** comprises the automated sharing of data amongst systems based on a common data exchange model.
- **Degree 4: Seamless Sharing of Information** is an extension of degree 3 to the universal interpretation of information through data processing based on co-operating applications.

Obviously these degrees are too coarse to enable or even support standards selection, therefore they have been further refined into functionally oriented sub-degrees, that identify specific interoperability services.

Operational components

Sub-degrees of interoperability shall not only enable a classification of requirements with finer granularity, but they shall also allow the association of those sub-degrees to operational components (e.g. E-Mail) or functional domains (e.g. logistics).

Table 1 shows a subset of the NATO interoperability sub-degrees. Naturally such a list is subject to enhancement and revision based on new or modified user requirements.

Interoperability Degree	Sub-degree	Description
1. Unstructured		

Data Exchange		
	1A. Network Connectivity	Networking services (according to ISO/OSI layers 1 to 4) and file transfer.
2. Structured Data Exchange		
	2A. Enhanced Informal Messaging	Services for informal multimedia electronic mail and associated directory services.
	2B. Network Management	Services for network monitoring and management.
	2C. Graphics and GIS	Services for geo-data, overlay formats, and symbology.
3. Seamless Sharing of Data		
	3A. Formal Messaging	Includes services for formal automated messaging (e.g. EDI).
	3B. Security Management	Multi-lateral security services (e.g. PKI).
4. Seamless Sharing of Information		
	4A. Distributed Applications	Services for distributed computing, object interfaces and object middleware.

Table 1 - Examples of interoperability sub-degrees

Typical or common operational interoperability requirements can be easily described as profiles over the sub-degrees.

As a simple example, a network based system for automatic exchange of formal (EDI-like) messages, with some associated management functionality can be expressed as profile 1A-2AB-3A. This then subsequently would lead to the appropriate standards and associated products as described below.

Standards

A further dimension for the establishment of a COE is a broad overview, analysis and eventually selection of base standards, which provide guidance for a system's composition and product selection. The chosen approach is to select all standards, which promise to be relevant for the achievement of interoperability or associated functionality (e.g. portability). This 'shopping basket' of standards then has to be structured and further assessed and refined, in order to become usable. Several aspects are taken into consideration:

- Structuring by domains or areas of applicability (e.g. networking) to allow quick identification of those standards that might support requirements in a specific domain.
- Structuring by standard priority. To allow for employment of COTS products and to enable broadest possible interoperability, the emphasis should be to prefer standards with widest market acceptance. The proposed categories, in decreasing order of priority, are:
 - Organisation specific standards (e.g. NATO STANAGs), although normally not widely supported by the market, have to be considered first, as they have been specifically developed and agreed on for an organisation-specific purpose,
 - International standards (e.g. ISO, ITU),
 - Standards from important independent standards bodies (e.g. W3C, The Open Group),
 - National standards (e.g. ANSI, DIN),

- Standards defined by industry consortia, and finally,
- Proprietary standards (e.g. of individual vendors).
- Structuring by applicability, maturity and relevance for the desired COE. Proposed categories are:
 - Mandatory, where the specified standard has to be used in any case.
 - Restricted or Conditional, comprising a set of standards, where conditional selection of at least one of those is mandatory.
 - Emerging standards, which are foreseen to become relevant in the near future, but that have not gained sufficient market support yet.

This list of base standards shall also serve as a reference book and standards dictionary for project managers and integrators, to provide them an overview, basic explanations and detailed references for potential employment of standards. This document could also be of benefit to organisations that are building non-COE based systems, but seek to improve interoperability by employment of open standards and to lower life-cycle costs.

For NATO the 'NATO Open System Environment (NOSE)' (NOSE, 1998) provides the above standards information. For the purpose of structuring the standards and to classify them by following a reference model, the NOSE has been divided into 11 domains or service areas:

- Software Engineering Services provide standards for programming languages, methods and tools (e.g. for CASE) appropriate to the development and maintenance of applications.
- User Interface Services define how users may interact with an application. Standards comprise graphical user interfaces, associated look-and-feel and the necessary toolkits.
- Data Management Services are concerned with the definition, storage and availability of data, and cover three elements: Data Administration, Repository Control and Database Administration.
- Data Interchange Services provide support for the interchange of data (e.g. Graphics, text files, audio/video, GIS data) between applications on the same, or on heterogeneous, platforms.
- Graphical Services provide interfaces for manipulating and programming applications concerning images and graphics in a device independent manner.
- Communication Services define communications standards, their profiles and protocol stacks structured along the ISO/OSI seven-layer reference model.
- Operating System Services provide support for applications, users, devices and other service areas by means of interfaces (APIs). They also describe system behaviour and internal communication mechanisms.
- System Management Services cover standards for coherent configuration and monitoring of communication systems and networks.
- Internationalisation Services allow a user to define, select, and change between culturally related application environments, and include character sets and data representation, cultural convention services, and natural language support.
- Security Services cover standards for secure communications (on network and operating system level) and associated security management functions.
- Distributed Computing Services provide for the extension of local procedure calls to a distributed environment; e.g. for distributed file and print services, distributed transaction processing, or distributed objects.

Standard profiles

Based on a list of standards, as described above, more detailed profiles are required to

establish a system's architecture. Profiles refine each service area into one or more functional classes, with each class mapping to one or more mandatory or emerging standards. Profiles describe those standards in terms of options, parameters, and guidance for product selection and integration.

Service Area	Functional Class
Operating System	Kernel Operations
	Non Real-time
	Real-time
System & Network Management	TCP management
	OSI management

Table 2 - Example of service areas and functional classes

The Common Standard Profile (CSP) specifies the minimum set of communication and information technology standards within the above listed service areas, to be mandated for the acquisition or major upgrade of all CCIS systems. It focuses on mandating only those standards critical to interoperability, and is based primarily on open system technology, which has strong support in the commercial marketplace.

Some rules apply for the usage of the standards in the CSP:

- If a system has to implement services covered by one of the service areas, it has to adopt the relevant standards mandated in the CSP.
- Specification and usage of other standards than those mandated in the CSP must be additive, complementary, and non-conflicting with CSP mandated standards.
- Legacy standards can be implemented as necessary on a case-by-case basis, on the same premises as in the previous rule.
- New technology can be exploited by the adoption of emerging standards, which may be expected to be elevated to mandatory status when implementations of the standards mature and availability on the market is assured.
- Mandatory standards have priority over emerging standards.

Starting with an initial set of standards, technical discussions and potential consensus on an analysis and presentation of perceived benefits (e.g. of additional standards) by the group which defines the organisation's COE, result in further refinement of the CSP. For NATO, the standards in the CSP have been selected on the following criteria:

- Maturity of a standard is assumed, when it is technically implemented and when the underlying technology is well understood, robust and tested. A standard is considered not mature when it is implemented with relatively new technology, is not well defined, or when restrictions or problems are known for the standard.
- Availability of a standard depends on the level of adoption of the standard by vendors, and on the implementation of the standard within different products. A standard is considered available if two or more products exist that implement the full standard and if those products are available from different vendors.
- Stability depends on the advancement or changes expected or planned for a standard. A standard is considered stable if no significant changes are expected or planned within the next two years. A standard is considered less stable if incompatibility exists between the current version and expected or planned releases.

Mandatory standards, in the sense of the CSP, shall provide the required interoperability and shall also be mature, available and stable.

The CSP is both a forward-looking and living document, i.e. it guides acquisition and

development of new and emerging information systems, thus providing a baseline towards which existing systems should evolve. It represents those standards that shall be used now and in the future, and will be updated periodically. It is therefore a document, which will evolve as a result of changes in requirements, technology and the commercial market. As a side effect, NATO's CSP shall also communicate NATO's intent to use open systems products and implementations, and help in influencing the direction of IT industry's standards and standards-based product development.

Strategic products

The next step in providing building blocks for the COE is to identify strategic products, which fulfil certain operational requirements and at the same time are compliant with the associated mandatory standards of the CSP.

Exclusivity is not required, i.e. it is possible to have multiple different products for the same purpose on the list. Firstly this gives the user or system integrator a choice to utilise products, which he for some reason prefers; and secondly this allows a real-world evaluation and selection of best-of-breed products. For reasons of configuration control and reduction of interoperability risk, the choice should be restricted to a maximum of two or three similar products.

Strategic products support one or more of the three service domains within a COE:

- Kernel Services (also referred to as the Minimal Baseline) are that subset of the COE, which is required for all workstations and servers. As a minimum, this subset would consist of the operating system, windowing software, security services, installation software and an executive manager for the runtime environment.
- Infrastructure Services directly support the flow of information across various systems. They provide an implementation independent set of integrated capabilities that the applications will access to evoke COE services to move data through the network. Infrastructure services offer capabilities from the following major areas: Management Services, Communications Services, Distribution & Object Management, Data Management Services, and Presentation Services.
- Common Support Application Services provide software services that are necessary to view or share data in a common way across the (networked) system. Those services are the glue, which allows and promotes interoperability between various Functional Area Sub-systems (FASS). Examples of Common Support Applications include widely used end-user applications such as Office Automation, Online Help, Geographic Services, and Messaging. Common Support Application Services shall be seamlessly integrated through APIs, or at least common file-formats, with the other COE Components.

The selected products shall, in conjunction with their underlying standards, ensure or at least improve (JSP450, 1998) the following characteristics:

- Application-portability is the ability of software to be run on differing hardware and/or operating systems, and to use differing network protocols for external connections.
- People-portability is achieved by providing a common look and feel Graphical User Interface (GUI), and by focusing on particular market-leading applications. This enables skills to be transferred from post to post rather than having to retrain on new interfaces and applications to perform, essentially, the same tasks.
- Scalability is achieved when a software application is designed to enable it to operate over a range of platform sizes and to run in small, medium and large multi-user environments.
- Value-For-Money can be achieved through Economies of Scale and by using market-leading products. Significant cost reductions can be realised through the adoption of a market-led approach to standards; it allows to procure COTS products from many

different suppliers, and at different times, with a reasonable degree of confidence that the systems will interoperate and make information available in a consistent and reliable manner. However, it has to be recognised that even established COTS packages are not necessarily free from defects and that adopting a market-led approach can mean commitment to an upgrade cycle that is beyond the control of the organisation.

The linkage of the building blocks

The various components of a COE, as described above, imply a need to link or associate these components into a coherent model for the development of a COE compliant system. The necessary decision flow, as depicted in figure 2, is:

- a) Operational requirements of the future system have to be analysed to determine, which interoperability sub-degrees are providing the necessary functionalities to cover most of the requirements. The result is a functional interoperability sub-degree profile, which identifies the building blocks in a short form (e.g. as described above 1A-2AB-3A). A mapping back to the interoperability degrees is not necessary. The highest sub-degree in the profile (i.e. 3 in the above example) shows the overall interoperability degree of the system.
- b) To identify the standards of the CSP, which support the desired interoperability profile, a mapping has to be established. The basic mechanism is a table showing, which CSP service classes provide support to which sub-degrees. Determining the necessary list of CSP classes and associated standards is a straightforward process.

CSP Service	Class	1.A	1.B	2.A	2.B	2.C	3.A	3.B	4.A	4.B
User Interface	GUI			X	X	X	X	X	X	X
	Look & Feel				X	X	X	X	X	X
	Toolkit							X		X
Data Management	Dictionary/Directory				X		X		X	
	DBMS (relational)				X				X	X
	DBMS (object-oriented)								X	
	Distributed Data					X			X	X

Table 3 - Example mapping of CSP services and classes to interoperability sub-degrees

- c) From the list of strategic products those have to be identified, which provide the required functionality, according to the interoperability sub-degree, and which comply with the mandatory standards determined above. If multiple products are proposed, the integrator has to make a choice based on some project-related rationale. Care has to be taken to select products, which are compatible with the necessary Common Support, Infrastructure, and Kernel Services applications. Typically the composition of a COE has to be performed bottom-up, e.g. by first selecting the Kernel Services, then the Infrastructure Services etc. This shall ensure a strong and harmonic foundation, on which interoperability with Common Support and FASS applications can be achieved easily and with minimal project risk.
- d) Eventually it shall not be forgotten, that the above described selection of products has to be validated by some integration (e.g. in a dedicated testbed) and testing to prove the successful co-existence and functioning of those products.

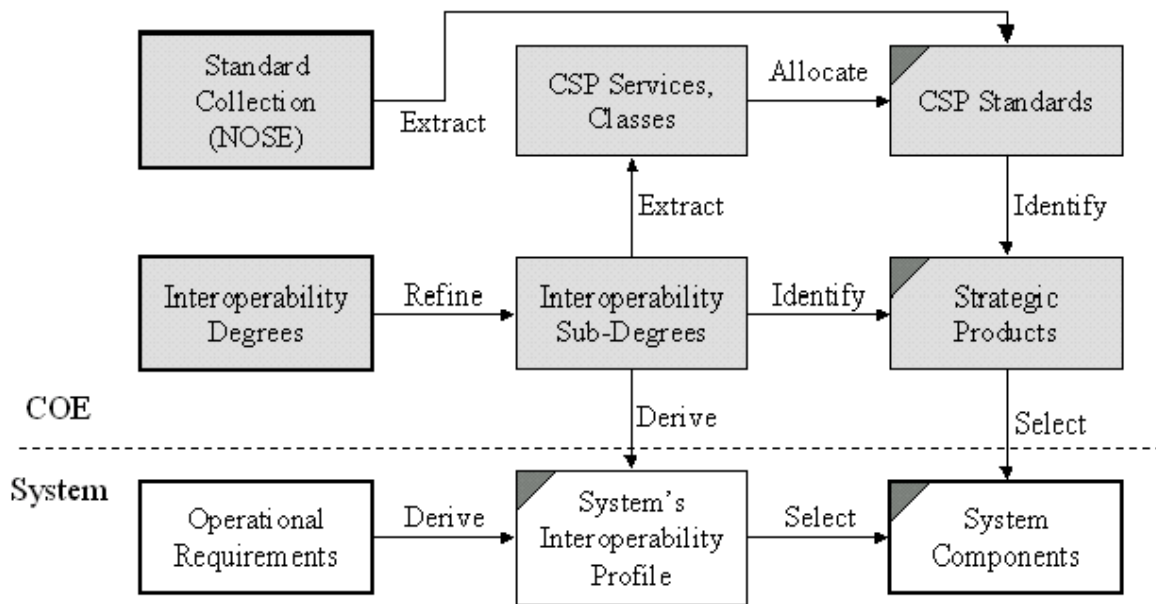


Figure 2 - COE decision flow

MANAGERIAL ASPECTS

Without proper management efforts, the COE will not be able to efficiently guide the development of large-scale systems, nor will it evolve and become accepted from its initial stages of definition. Major COE management aspects are:

- Set-up of a life-cycle model for COE compliant systems, comprising templates for design, implementation, integration/validation, distribution, deployment, upgrade, and, not to forget, disbandment.
- Ongoing review and maintenance of system's compliance with the COE specifications throughout the system's life-cycle:
 - Verification of activities against COE life-cycle model.
 - Incremental implementation in accordance with the COE component model.
 - Compliant integration of new interoperability features (e.g. for interoperability with new external systems).
 - Migration of legacy systems towards compliance.
- Set-up of a management structure devoted to the co-ordination of all COE compliant systems, which is in charge of
 - COE related activities during the system's design, implementation and operational phase,
 - Organisation-wide mid- and long-term policy definition (e.g. for acquisition guidance).

Two major aspects of COE management are highlighted and discussed below.

Life-cycle management

As for every operational system, the management of the life-cycle of a COE comprises certain phases.

- In the Initiation Phase an analysis has to be performed, which shall lead to decisions
 - on how the COE development process will be structured,
 - which major components shall be considered (e.g. only standards, only products, a mixture),

- what the overall philosophy is (e.g. COTS or development based), and
- who the participants, partners and sponsors are.

Especially an early involvement of the budget authorities and the future user community (i.e. project managers) is compulsory. One key decision is the allocation of authority and responsibility to COE staff and project leaders. Typically the engineering and testing of COE specifications will be performed centrally, whereas its application, e.g. through product selection, will be done de-centrally by the project staff.

- The Engineering Phase comprises the efforts of the partners to generate the COE components (e.g. documentation, testbed, runtime environment, tools, and associated configuration control). Here the major decisions on technical level (especially the mandate of standards and products) are made. To keep up-to-date with technological and market developments, and changed operational requirements, the engineering phase shall be foreseen as a repetitive activity with a pre-determined frequency (e.g. forcing a technical COE review once a year).
- The Distribution Phase encompasses the effective rollout of the COE to the project and budget authorities. At that point they receive the COE documentation, get access to the testbed and its test results (e.g. on strategic products), and can utilise training facilities and support from the COE staff. As the COE distribution is a natural and tightly coupled follow-on activity of the COE engineering, it has also to be repeated after every COE review and revision.
- In the Operational Phase, the COE as specified is applied to actual projects. Results and experiences of those projects shall be fed back into the engineering phase to improve the overall COE quality.
- The Disbandment Phase marks the replacement of the current COE by a major upgrade, or even a different approach to system architectures and project support. The critical aspect is not to destroy the COE infrastructure, as there will be legacy COE systems, which can not be disbanded immediately and therefore will still need the available documentation and tools.

Compliance testing

COE compliance testing and validation is essential to assure interoperable systems. Testing and validation has to be performed on multiple levels:

- On the CSP level it has to be ensured that the mandatory standards in each service class are non-conflicting and effectively supporting each other, or that the appropriate provisions and restrictions are properly documented.
- Also on the CSP level, all proposed mandatory standards shall fit into reasonable protocol and application stacks, to allow a coherent integration and interoperation throughout the target system's architecture.
- The proposed strategic products shall be tested for compliance with the standards they are supposed to support, to cater for the vendors' varied implementation and interpretation of the agreed standards.
- The strategic products have to be tested for their co-operation with each other, to avoid having products, which formally comply with certain standards, but do not interoperate due to probable non-critical deviations in standard implementation or parameterisation.
- And finally, it is necessary to perform regression testing of the relevant parts of the above, when changes to standards or upgrades to products shall be incorporated into the COE.

Typically such a test and validation effort has to be performed on a testbed, where all mandated standards are available in a reference implementation, the strategic products are available and can be installed, and a complete COE compliant system can be rebuilt. Also from the testing viewpoint the desire for a restriction of the COE specification to the bare minimum is obvious, thereby avoiding excessive requirements for testing of inter-relations

of all potential components.

A further formalisation of the validation effort for COE compliance of products and systems would be to follow the approach taken in the DII-COE (1999). Here compliance is rated in eight levels from level 1 (e.g. system uses same standards for operating system, windowing, and database queries), up to level 8 (e.g. no duplication of functions with COE or co-hosted applications; no duplication of data with COE or COE-based target system; 100% compliant with runtime specification, 100% style-guide compliant). Such a detailed view requires a very well defined runtime system, with associated automatic tools for integration and validation, and, implicitly, a long experience and huge effort in building such a demanding environment.

CONCLUSION

A COE provides the functionality and support services necessary to effectively develop, distribute, and integrate interoperable systems across an organisation. The COE is scalable, as well as flexible. It embraces a standard-based “Plug-and-Play” open architectural approach that enables functionality to be easily and evolutionary added to a system. The effect of decomposing the system’s software into controllable building blocks reduces the time required to put new functionality into the hands of the end-user. And this without sacrificing quality, or incurring unreasonable program risks or expenditures. With the provision of a decision flow from the formulation of operational requirements, over the identification of applicable standards to the selection of strategic products, a most valuable tool is given to the designer of an IT system. Additionally, when strictly obeyed, this approach gives a justification and a proof of value-for-money to the budget authorities, which eventually have to subscribe to the acquisition or development of products. In reverse, an organisation's management needs to define a process to enforce the COE compliant system implementation. Typically this would be realised through the approval mechanism for new projects (e.g. technical controlling).

REFERENCES

- ADAGE (1999) Avionics Domain Application Generation Environment. [Online]. Available: www.owego.com/dssa
- DII-COE (1999) Defense Information Infrastructure (DII) - Common Operating Environment (COE). Washington, DC: DISA. [Online]. Available: spider.osfl.disa.mil/dii
- JSP450 (1998) Defence Communication and Information Systems (DCIS) Standards Guides (JSP450). London: MOD UK. [Online]. Available: www.mod.uk/dgics/dcsa_web/dseg_web/index.htm
- JTA (1999) DOD Joint Technical Architecture (JTA). Washington, DC: DOD. [Online]. Available: www-jta.itsi.disa.mil
- NOSE (1998) NATO Open Systems Environment. Brussels: NATO. [Online]. Available: www.nc3a.nato.int/ppdiv/nose/nosehead.htm