Carnegie Mellon University
Research Showcase @ CMU

Software Engineering Institute

2-2011

A Framework for Evaluating Common Operating Environments: Piloting, Lessons Learned, and Opportunities

Cecilia Albert
Carnegie Mellon University, cca@sei.cmu.edu

Steven Rosemergy
Carnegie Mellon University, swrosemergy@sei.cmu.edu

Follow this and additional works at: http://repository.cmu.edu/sei

This Technical Report is brought to you for free and open access by Research Showcase @ CMU. It has been accepted for inclusion in Software Engineering Institute by an authorized administrator of Research Showcase @ CMU. For more information, please contact research-showcase@andrew.cmu.edu.

Software Engineering Institute

A Framework for Evaluating Common Operating Environments: Piloting, Lessons Learned, and Opportunities

Cecilia Albert
Steve Rosemergy

February 2011

SPECIAL REPORT
CMU/SEI-2010-SR-025

Acquisition Support Program
Unlimited distribution subject to the copyright.

http://www.sei.cmu.edu

Carnegie Mellon

This report was prepared for the

SEI Administrative Agent
ESC/XPK
5 Eglin Street
Hanscom AFB, MA 01731-2100

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

This work is sponsored by the U.S. Department of Defense. The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

Copyright 2011 Carnegie Mellon University.

NO WARRANTY

THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN “AS-IS” BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

Use of any trademarks in this report is not intended in any way to infringe on the rights of the trademark holder.

Internal use. Permission to reproduce this document and to prepare derivative works from this document for internal use is granted, provided the copyright and “No Warranty” statements are included with all reproductions and derivative works.

External use. This document may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other external and/or commercial use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

This work was created in the performance of Federal Government Contract Number FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.

For information about SEI publications, please visit the library on the SEI website (www.sei.cmu.edu/library).

Table of Contents

List of Figures

List of Tables

Acknowledgments

The authors would like to acknowledge the time and contributions of Dr. Lawrence E. Grosberg, Steve Ford, Monica Farah-Stapleton, Phillip Minor, and Lt. Colonel Steward Liles with the U.S. Army. Their participation in this study provided invaluable insight and guidance regarding the business objectives of the technical solutions under review. We would also like to acknowledge Pete Dugan and Evan Jalbert, with MITRE, for their contributions and feedback associated with the technical aspects of this study.

Finally, we would like to acknowledge John Klein, Edwin Morris, and Bryce Meyer of the SEI. Each was instrumental to the successful development of the evaluation framework, leveraging past work, experience, and practical insight.

Abstract

This report explores the interdependencies among common language, business goals, and software architecture as the basis for a common framework for conducting evaluations of software technical solutions. It also describes the SEI's experience piloting this framework, which integrated commercial technologies, customized open-source systems, and legacy systems, and the insights gained from the project. As described in the report, those insights have enabled the SEI to further refine the framework to make it reusable and applicable for a variety of technical solutions.

1 Introduction

1.1 Background

On May 24, 2010, the Pentagon published the execution order *Army Enterprise Common Operating Environment Convergence Plan.* This order directs the merging of two network modernization strategies into one plan. In response, the Assistant Secretary of the Army for Acquisition, Logistics, and Technology (ASA(ALT)) directed assessments of alternative technical solutions to both support Army requirements and to develop insight into possible solution alternatives.

To promote both objectivity and timely closure of assessment activities, ASA(ALT) assembled a team of internal and external technical experts, including U.S. Army Test & Evaluation Command (ATEC), the Program Executive Office for Integration (PEO-I), MITRE, and the SEI.

With the goal of completing technical assessments by the end of July 2010, the SEI (under the direction of ASA(ALT) was assigned the role of third-party subject matter expert and was tasked with

1. creating a foundation of common language for all technical studies
2. developing a software evaluation framework, which can be used to independently validate internal and external technical studies
3. facilitating the collection of data to populate the framework for use by the U.S. Army.

These three tasks are the basis for the evaluation framework described in this report.

2 Software Evaluation Framework

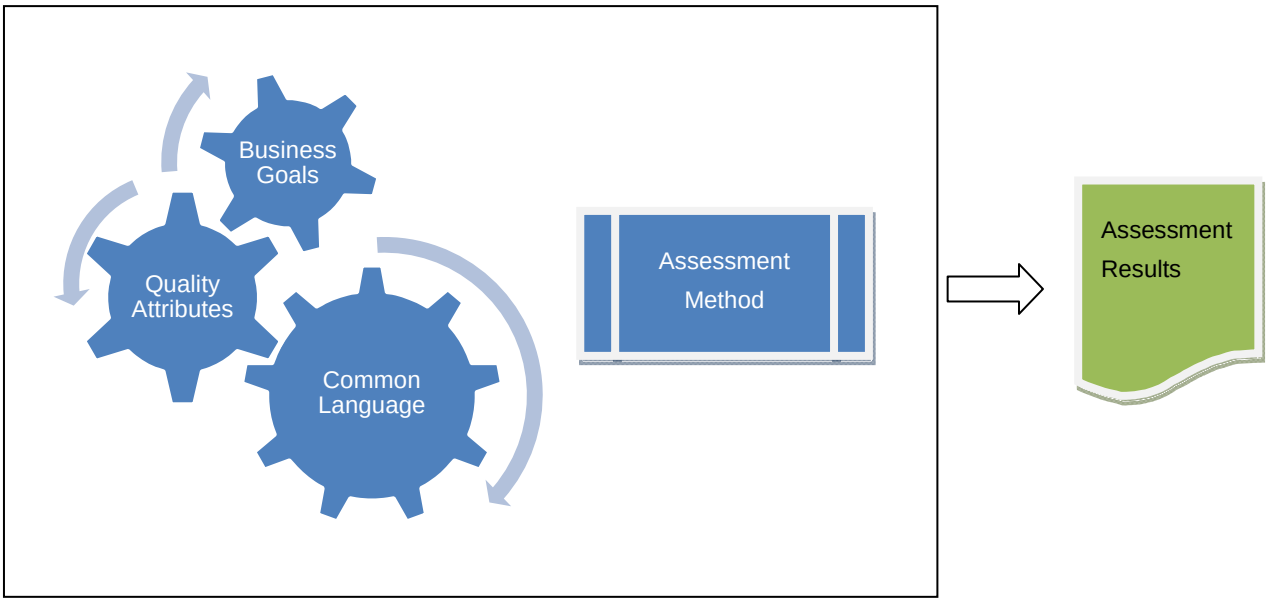


Figure 1: Software Evaluation Framework

As described in Section 1, the Army approached the SEI with an immediate need to develop objective insight into prospective technical solutions, using an approach based on broad goals. What resulted was a reusable framework presented in this document, assessment results, and the basis for developing informed decisioning on technical programs. Shown in Figure 1 are the key aspects of the software evaluation framework, which are *common language*, to facilitate alignment of evaluation activities amongst the key stakeholders, *quality attributes*, which provide insight into the underlying technical solution capabilities and constraints, and *business goals*. Taken together, these three aspects are used to develop objective and informative insight.

2.1 Common Language: Why it is Important

Common language is the main cog or gear, shown in Figure 1, of this interrelated set of activities. Common language serves at least two purposes:

- to baseline common understanding of the scope of the overall assessment activities and provide declarative understanding of which quality attributes are important
- to facilitate and frame the most important business goals in the commonly understood context, coupled with an assessment method, business goals, quality attributes, and common language

Technical experts and business leaders in a multi-faceted team have differing perspectives and source knowledge; this often creates gaps in understanding among involved stakeholders. Because of this, establishing a common and objective understanding of key technical terms from authoritative sources sets the stage for a common language. For this particular study, source terminology, referenced from U.S. Army Chief Information Officer/G-6, the U.S. Army Training and Doctrine Command (TRADOC), G-3/5/7 (Operations, Plans, and Training), and Boeing, created confusion

and competing discussions due to inconsistencies between source definitions, institutionalized understanding, and commercial definitions and norms. Within the context and authority of each organization, these technical terms were defined correctly—but taken out of their native context, each was incomplete or insufficient for use in completing evaluation activities for the technical solutions intended for common use across the Army enterprise.

In order to address this issue and establish common understanding and a foundation for proceeding with evaluation activities, the SEI and ASA(ALT) collected references to each of the relevant technical terms defined by the Department of Defense (DoD) and commercial industry. Each term was then normalized and elaborated upon to incorporate illustrative descriptions and commonly recognized commercial examples. Stakeholder reviews of the normalized definitions were then conducted, followed by updates and enhancements to the terms. At the conclusion of this activity, ASA(ALT) established a common set of terms to be used as part of the subsequent evaluation activities and plans. The critical accomplishments associated with this activity were

- Centralized terminology: One defined set of terminology definitions, which include illustrative elaborations, architectural descriptions, and commonly understood commercial examples.
- Reduced confusion and ambiguity: With one set of established terminology, debates and discussions surrounding which terms to use as the basis for subsequent studies were set aside. While some disagreement remained with some aspects of the definitions, all stakeholders agreed that the established terminology was sufficiently declarative for use for follow-on evaluation activities.
- Stakeholder collaboration: Stakeholders contributed to the development of terms by sourcing the authoritative descriptions, debating the merits of each, and providing feedback to the finalized terms. This early involvement in the development and evolution of the terms served to establish rapport, understanding, and cooperation between the team members—important elements needed for conducting evaluation activities.

2.2 Software Evaluation Framework for Common Operating Environments

With a foundation of common understanding of terms, quality attributes and business goals could then be elaborated upon to develop our assessment framework. Because the technical solution under review used many aspects of service-oriented architectures (SOAs), the SEI team leveraged insights gleaned from development work associated with the SEI report titled *Quality Attributes and Service-Oriented Architectures*, which was used as a starting point [O'Brien 2005]. The rationale for leveraging this body of knowledge was based in part on the type of technical solutions under review. Composable stateless, location-transparent, and network-addressable services were foundational attributes of the technical solutions—key hallmarks of service-oriented architectures.

The quality attribute areas for this framework are described in Table 1.

Table 1: Quality Attribute Areas

Quality Attribute Area	Explanation
Adaptability	Adaptability enables the common operating environment (COE) to respond to changes in internal configuration and external environment.
Architecture	Architecture of a software-reliant system is the structure of the system, which comprises software components, the externally visible properties of those components, and the relationships among them.
Functional	This area concerns a focus on functional capabilities provided by the COE.
Support for Governance	Architecture governance is the practice and orientation by which architectures are managed and controlled at an enterprise-wide level.
Interoperability	This area concerns the ability of the COE to support interoperation among different systems, versions of systems, and development environments.
Development and Test	These concerns focus on the development and test environments for both the candidate COE and the applications developed to execute within the COE.
Hardware and Software Platform	These concerns focus on the computing environments upon which the COE will execute.
Quality of Service	This area concerns a focus on the availability, performance, and other characteristics of the services delivered by the COE.
Information Assurance	This area concerns a focus on secure operation of the COE.

The quality attribute areas served as "containers" for individual quality attributes, which were then transformed into a series of questions, designed to elicit the extent to which each alternative addressed the attribute. Sample questions by quality attribute are shown in Table 2 (see Appendix A for the full list of quality attribute questions). Note the cross referencing between quality attributes and business goals.

Table 2: Sample Questions by Quality Attribute Area

ID	Question
Q1	To what extent do development tools enforce established standards of the candidate COE system? What aspects of the standards are not programmatically enforced? How might this affect governance over application development?
Q2	How is compliance with candidate COE standards ensured for developed applications and services?
Q3	How do installation and deployment tools support governance through mechanisms such as environment compatibility validation, version checking, etc.?
Q4	What mechanisms are used to handle evolution of the infrastructure, tools, and deployed applications in the current development of the candidate COE?

2.3 Linking Business Goals and Quality Attributes

In the absence of organizational goals, any evaluation of a technical solution against quality attributes is of limited value. Quality attributes by themselves assume broadly associated business goals (e.g., agility, streamlining, ease, and flexibility). This step in the process was designed to incorporate declarative guidance to link the specific business goals of the Army, their relative priority, and a declarative relationship to the quality attributes. Once this step was completed, each goal was mapped to relevant quality attribute questions, providing a transparent prioritized set of quality attribute questions with direct linkages to business goals.

Table 3: Quality Attribute to Business Mapping

	Priority	Q1	Q2	Q3	Q4	Q5	…	Qn
Goal #1	1	x						
Goal #2	2		x			x		
…	…							x
Goal #m	m			x	x			

2.4 Software Evaluation Process

To ensure that results of an evaluation provide objective, informative, and inclusive results, the SEI felt that a declarative and transparent process for conducting the evaluation was needed. Based on this need, the essential characteristics the SEI team deemed the most relevant to conducting evaluation activities are outlined in Table 4.

Table 4: Essential Characteristics of the Evaluation Method Applied to Technical Solutions

Characteristic	Explanation
Accuracy	Appraisal characterizations reflect the technical solution's capability against the stated business goals. The approach could be used for comparison across technical solutions. Appraisal results reflect the strengths and weaknesses of the evaluated technical solution.
Repeatability	The characterizations and findings of a characterization will likely be consistent with those of another independent appraisal conducted under comparable conditions.
Cost/Resource and Effectiveness	The appraisal method is efficient in terms of person-hours spent planning, preparing, and executing an appraisal. The method takes into account the organizational investment in obtaining the appraisal results, including the resources of the host organization, the impact on the appraised organization, and the appraisal team.
Meaningfulness of Results	Appraisal results are useful to the appraisal sponsor in supporting decision-making. This support of decision-making may include application of the appraisal results in the context of technical solution selection, or continued engineering investment.

Rather than invent standards and rules for conducting the evaluation, the team borrowed from proven approaches developed by the SEI, the Standard CMMI®[1] Appraisal Method for Process Improvement (SCAMPI℠). As in the SCAMPI method, the team identified roles and responsibilities for establishing the plan for the appraisal. However, because the appraisal was conducted against a set of business goals and quality attributes instead of against a capability model, the business goals served as the basis for appraisal results. The high-level process is outlined in Table 5.

Table 5: The High-Level Process

Step	Activity	Roles and Responsibilities
1	Quality attribute questions are answered with supporting reference data to demonstrate performance capabilities.	A. The assessment team provides quality attribute questions to the technical solution owner. B. The technical solution owner answers questions and provides supporting data. C. The assessment team provides quality attribute question guidance.
2	Establish business goals associated for the prospective technical solution under consideration and review. Prioritize goals and correlate goals to quality attribute questions. Identify critical quality attributes and goals that must be achieved.	A. The sponsor coordinates definition, refinement, and prioritization of mission and business goals. B. The assessment team provides process support, as requested. C. The sponsor and assessment team conducts quality attribute to goal mapping.
3	For each candidate COE, characterize answers to questions to identify key strengths and weaknesses based on the established and prioritized business goals.	A. The assessment team performs the assessment. B. The assessment team validates assessment characterizations with the technical solution owner. C. The technical solution owner responds to findings with supporting data where findings disagree. The technical solution owner acknowledges key strengths and weaknesses.
4	For each candidate COE, conduct a roll-up characterization of identified strengths and weaknesses by business goal. Aspects that definitively achieve the identified business goal are identified as key strengths, weaknesses which place high or medium priority goals at risk are characterized as risk areas.	A. The assessment team performs the assessment. B. The assessment team presents results to the sponsor. C. The sponsor validates the assessment team's findings.

Table 6 shows a sample characterization. The quality attributes are linked to business goals, with indicators of strength and weakness. Key strengths indicate goal achievement of medium- or high-priority goals. Risk area indicates that a medium- or high-priority goal may not be achievable based on the indicated weaknesses. The weaknesses column indicates that one or more weaknesses have been identified. The business goal (L, M, H) can be used to weigh or normalize characterizations.

[1] CMMI is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University. SCAMPI is a service mark of Carnegie Mellon University.

Table 6: Quality Attribute Characterization

Business Goal	Strengths	Weaknesses/ Observations	Importance	Characterization
Goal ID(s)	Attributes of the technical solution that demonstrate achievement of business goal(s).	Indicators that a goal may not be met	[L, M, H] based on business priority	[*Key strengths*, *weaknesses, and/or risk areas identified*] Characterize based on whether technical solution can achieve the business goal. See step 4.

3 Post-Mortem Notes

After completing the software evaluation pilot/case study, the SEI team conducted a post-mortem discussion to identify both improvement opportunities and key strengths of the assessment framework. Table 7 shows a summary of positive aspects and opportunities for improvement. The software assessment framework had a positive impact on all assessment activities through the development of foundational understanding among assessment participants, leading to more accurate assessment findings. This was confirmed by complementary results from independent studies.

Table 7: Post-Mortem Strengths and Opportunities

Key Strengths	Opportunities
The identified strengths and weaknesses reinforced findings of other technical studies.	There were too many quality attribute questions.
The framework uniquely uncovered areas of risk associated with the achievement of business goals.	There was insufficient time to conduct the assessment.
There was transparency of findings and characterizations.	Business goals were not very quantifiable.
The overall assessment process aligned all assessment activities and foundational understanding among teams.	Prepping the technical team on quality attributes by employing the mission thread workshop would likely have helped.
A "perfect" assessment framework is not necessary to develop representative findings.	The technical team had difficulty with questions. The assessment team should complete the QA answers through a series of interviews and data collection with the technical team.
	Information assurance and licensing questions are relatively thin.

For use of the framework in future studies, the following is recommended:

1. Reduce the size and refine the questions. Some of the quality attribute questions are duplicative or unclear. Improvement could streamline the assessment process.
2. Improve information assurance questions. Leverage the CERT® Resilience Management Model (CERT-RMM)[2] to develop quality attribute questions relating to security, secure coding, and resilience. This quality attribute will become more important over time.
3. Develop formalized mechanisms for eliciting measurable business goals from business sponsors. This activity became a critical path item in completing assessment activities.
4. Refine the framework documentation to facilitate consistent use and reuse.

[2] CERT is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

Appendix A: Assessment Framework Questions

Adaptability

Adaptability enables the COE to respond to changes in internal configuration and the external environment.

1. What protocols and/or communication standards are adapted or translated for use in the candidate COE? What are the performance impacts? How does this affect interoperability with other systems?
2. What (if any) off-the-shelf tools may be used or configured to work with the candidate COE without modification? How?
3. Describe the system-tailoring options to adapt to internal configuration variations. What mechanisms and tools are used to tailor the candidate COE for different platforms? What is the process or workflow required to create and deploy a tailored system?
4. How can applications interface with other systems within the candidate COE? What are the interoperability and interfacing constraints?
5. Which modules, components, or subsystems of this operating environment can be utilized and/or ported to other systems or OEs? What are the constraints and/or limitations of doing so (e.g., licensing, technical)?
6. A new task organization is created. Describe the workflow required to configure the COE to reflect this.

Architecture

The architecture of a software-reliant system is the structure of the system, which comprises software components, the externally visible properties of those components, and the relationships among them.

1. What is the architecture of the candidate COE (including rationale for key architectural decisions)? How will the candidate COE architecture support both new and legacy applications?
2. How does the architecture support the strategic plans for the Army enterprise?
3. What mechanisms are available for synchronizing component and application interactions? What is the rationale for these decisions?
4. How can data be shared across applications and domains?
5. What restrictions or constraints does the candidate COE place on application design?
6. What architectural patterns and technologies are necessary to bring legacy systems into alignment with the candidate COE? Are the necessary technologies available today?
7. How does the candidate COE manage shared Computing Environment (CE) and network resources used by multiple applications? Shared resources include CE throughput, CE memory, user interface (UI) display real estate and network bandwidth.

8. How does the candidate COE support applications operating in environments with compromised operating conditions, such as intermittent network connectivity or disconnected operation?
9. If all data communications in and out of a single Tactical Operations Center (TOC) were lost for one hour, how would the operation of the candidate COE be affected? When data communication to the impacted TOC is restored, what is the sequence of events that the candidate COE would perform to restore full capability?
10. Can an application discover available services within the candidate COE at runtime (i.e., runtime binding between the application and other applications and middleware)? How is this achieved?
11. How are errors handled and reported by the candidate COE? How do applications detect and respond to errors in the candidate COE?
12. Does the candidate COE manage application processes and memory? How are application faults handled?

Functional

These concerns focus on functional capabilities provided by the COE.

1. What functional capabilities of the underlying computing environment are explicitly made unavailable when operating within the candidate COE? What is the rationale for removal?
2. What functional capabilities provided by the underlying computing environment are replaced or overridden? What is the rationale for replacement?
3. What capabilities are provided for managing operations in the deployed COE (including performance, capabilities, configurations, data, and infrastructure)?

Support for Governance

1. Architecture governance is the practice and orientation by which architectures are managed and controlled at an enterprise-wide level.
2. To what extent do development tools enforce established standards of the candidate COE system? What aspects of the standards are not programmatically enforced? How might this affect governance over application development?
3. How is compliance with candidate COE standards ensured for developed applications and services?
4. How do installation and deployment tools support governance through mechanisms such as environment compatibility validation and version checking?
5. What mechanisms are used to handle the evolution of the infrastructure, tools, and deployed applications in the current development of the candidate COE?

Interoperability

Interoperability concerns the ability of the COE to support interoperation among different systems, different versions of systems, and different development environments.

1. What software interfaces are exposed for use by external applications or services, including new and legacy applications? What tools and technologies are these interfaces compatible with? What standard technologies are used to realize these interfaces? What data models, types, and formats are exposed on external interfaces to the candidate COE?
2. Can multiple versions of the candidate COE interoperate? How is this achieved? Describe any constraints.
3. Can multiple versions of an application execute within the candidate COE? How is this achieved? Describe any constraints.
4. Can multiple versions of a CE operate within the candidate COE? How is this achieved? Describe any constraints.
5. Can non-certified applications operate within the candidate COE? How is this achieved? Describe any constraints.
6. How are external capabilities invoked from the candidate COE and how do they invoke internal capabilities within the candidate COE?
7. What constraints are placed on systems outside to interoperate with this system?
8. How many distinct code bases are used to support CE and interface variability in the candidate COE?

Development and Test

These concerns focus on the development and test environments for both the candidate COE and applications developed to execute within the COE.

1. Describe the development, testing, staging, and deployment environments for application and services development.
2. What capabilities and support of the infrastructure are available for off-the-shelf development tools?
3. What testing tools are included? Are there black box/white box, performance and profiling tools? Is there a test executive included?
4. What installation and deployment tools are used for the candidate COE itself and for new applications deployed to the candidate COE?
5. What logging, tracing, and debugging capabilities are available to support development and maintenance of the candidate COE itself?
6. What logging, tracing, and debugging capabilities are available to support development and maintenance of applications running within the candidate COE?

Hardware and Software Platform

These concerns focus on the computing environments upon which the COE will execute.

1. What computing environments are supported by the candidate COE?
2. What assumptions does the candidate COE make about its environment (e.g., platform, infrastructure, and operational characteristics)?
3. What dependencies does the candidate COE have with single-source off-the-shelf software components?

Quality of Service

These concerns focus on the availability, performance, and other characteristics of the services delivered by the COE.

1. What is the estimated availability of the candidate COE? What is the basis for this estimate?
2. What is the estimated capacity of the system (such as the number of nodes and simultaneous transactions)? What mechanisms enable the management and monitoring of capacity performance of the system?

Information Assurance

These concerns focus on secure operation of the COE.

1. What are the security monitoring capabilities?
2. How does the candidate COE implement information assurance (IA) policies? What aspects are realized using COTS technology?
3. How does the candidate COE implement user management and identity management? Does the COE provide an infrastructure for policy-based security?
4. How does the chosen IA approach impact other system quality attributes such as performance, usability, and modularity.

Appendix B: Case Study—ASA(ALT) Case Study Assessment

Technical Memo: Developing Common Understanding

To:	Monica Farah-Stapleton	**Date:**	June 10, 2010
From:	Ceci Albert, Dennis Smith, Ed Morris, Bryce Meyer, Sholom Cohen, Steve Rosemergy	**Memo ID:**	ASA(ALT) 10-1-1 v.3
Subject:	Refined OE, COE, and Middleware Definitions	**Keywords:**	computing environment, operating environment, common operating environment, middleware,
Project:	Army Strategic Software Improvement Program (ASSIP)	**No. Pages:**	7

Summary

The SEI reviewed public definitions of the terms "computing environment," "common operating environment," and "middleware" and elaborated on these definitions and their attributes to aid common understanding for the purposes of evaluating systems solutions. Examples included in this paper are intended only to provide illustrative insight into the definitions independent of the problem space.

References

a) Vice Chief of Staff of the Army (VCSA) Execution Order: (U) Army Enterprise Common Operating Environment (COE) Convergence Plan
b) U.S. Army CIO/G-6 Common Operating Environment Technical Architecture, Appendix F to the Strategy for 'End State' Army Network Architecture—Tactical

Army Definitions

Common Operating Environment

The common operating environment (COE) illustrated in Figure 2 is an approved set of computing technologies and standards that enable secure and interoperable applications to be rapidly developed and executed across a variety of computing environments (i.e., server(s), client, mobile, sensors, and platform).

Figure 2: Common Operating Environment

Operating Environment

Each *Operating*+ Environment has a minimum standard configuration that supports the Army's ability to rapidly produce and deploy high-quality applications and to reduce the complexities of configuration, support, and training associated with the computing environment. Reference document *U.S. Army CIO/G-6 Common Operating Environment Technical Architecture, Appendix F to the Strategy for 'End State' Army Network Architecture—Tactical*

elaborates further to specify specific functional requirements of an Army *operating* environment.

[+]**NOTE**: U.S. Army CIO/G-6 Common Operating Environment Technical Architecture, Appendix F to the *Strategy for 'End State' Army Network Architecture—Tactical*: label in draft document mistakenly labels an Operating Environment as Computing Environment.

Computing Environment

Computing environments are: server(s), client, mobile, sensors, and platform.

Discussion: Concept and Aspects of a Common Operating Environment

In its simplest form, a common operating environment is simply an infrastructure for enabling distributed computing. Such an infrastructure incorporates reference standards (both commercial and problem domain specific), and includes oversight mechanisms to provide governance over both the evolution of the infrastructure and compliance over the application intended for deployment.

For software and software/hardware systems, a common operating environment will ordinarily start with a set of reference standards, software interfaces, data formats, protocols, and systems used to allow distributed applications and systems to communicate, coordinate and execute tasks, and respond to events in an integrated or predictable manner.

Governance

Governance is the largest and most complex aspect of a given operating environment and is realized in several complementary and integrated forms, including:

Standards Committee provides oversight over the standards and their evolution and develops the requirements and constraints relating to the other aspects of governance. The standards committee designs and evolves the common operating environment based on both the business and technology needs of the consumer and organization over the life of the common operating environment.

Application Governance is one of the more critical elements of a common operating environment. Application governance oversees the deployment and usage of applications. Application governance involves coordination between both automated processes implemented by a supporting infrastructure (or the operating environment itself), and people processes. Like the standards committee, all aspects of application governance are tied directly to the business objectives of the governing organization.

Governance Tools reinforce the technical aspects relating to application governance and technologies accessible and approved for use within the common operating environment. Generally, there are several types of governance tools available:

- Application development tools are designed to reinforce the rules associated with data formats, protocols, and software interfacing.
- Automated testing tools are generally used to uncover software defects and nonconformance with system or application requirements.
- Certification tools are used to aid reviewers and administrators of the system in gaining insight into critical elements of an application before its deployment. Critical elements may include safety and security, privacy, look and feel, application resiliency, resource usage, and design and compliance rules.
- Deployment tools are used to ensure that applications are installed correctly in the target computing environments without disrupting or compromising either a target system or common operating environment.
- Technical/peer review is a people process executed by technical experts to find and resolve software defects.
- Configuration management is used to manage versioning of integrated components for applications and their deployment targets.
- Operations support provides maintenance support and monitors the health of the system.

Examples

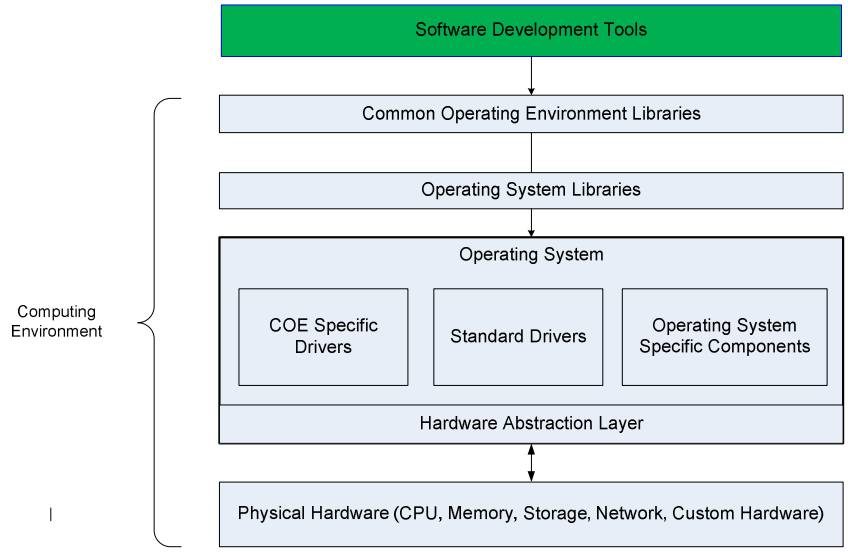
Commercial examples of software system COEs include Cisco WebEx, Microsoft NetMeet-

ing, and Apple iTunes.

Example	Problem Space	Reference
Description		
Cisco WebEx	Distributed conferencing	www.cisco.com/en/US/prod/ps10352/collaboration_cloud.html
WebEx both provides applications and enables application development using Cisco standardized APIs and standard OS and hardware platforms to enable secure collaboration among a diverse set of connected devices/OS/protocols as long as they use commercial protocols.		
Microsoft NetMeeting	Desktop collaboration	http://technet.microsoft.com/en-us/library/cc507850.aspx#E5
NetMeeting uses Microsoft methods for implementing internet security and communications protocols and standards, allowing developers to field applications that use provided services via reuse to collaborate over a variety of platforms that run a Microsoft OS.		
Apple iTunes	Online sales and distribution of streaming and stored entertainment content	www.itunes.com
Apple provides the development environment and registers applications to allow application development by integrating common components. The download software allows diverse hardware and OSs to use the iTunes structure. SDK and Apple resources over internet protocols secure communications.		

Discussion: Concept of an Operating Environment

At its core, an operating environment is an integral component of a common operating environment where applications are developed. Generally, it includes a set of hardware and software (but it could be purely software) configured to interface with both the common operating environment and its underlying hardware or software components. Operating environments vary in complexity and size (depending on their end-use application), but all encapsulate the critical elements needed to both interoperate AND develop applications within the common operating environment. In most cases, one or more sets of governance tools are deployed onto the operating environment to provide a framework for developing applications compatible with the common operating environment.



Commercial OE examples of systems and software systems include VMWare, Microsoft Visual Studio.NET, Google Android Development Environment, Mobile Phones, VMware Workstation/Server, and Sun Java Platform.

Examples

Example	Governance	OE Tools
Description		
Cisco WebEx	Distributed conferencing	WebEx Connect Integration Platform
WebEx Connect Integration Platform provides a development environment that enables integration between distributed conferencing and custom client applications. Cisco provides interfaces between applications and conference hosting for applications using web services and platform-neutral mechanisms. Cisco's governance infrastructure allows a high degree of flexibility in how WebEx is used, while retaining common functional support for both internal and co-branded/third-party applications.		
Microsoft NetMeeting	Desktop collaboration	Microsoft Visual Studio.NET
Microsoft VS.NET includes all necessary libraries and tools needed to integrate custom user-defined applications with terminal services supporting Microsoft Windows (including Windows Mobile), Linux, Unix, Mac OS X, and other current operating systems. Microsoft provides governance support for VS.NET and shares governance responsibility for the underlying protocols with the ITU. Governance allows a high degree of flexibility and only governs to underlying protocols and platform support.		
Apple iTunes	Online sales and distribution of streaming and stored entertainment content	PodCast Creator
Podcast Creator provides integrated support for creating iTunes applications, submission to governing authority, and deployment to the iTunes network for sharing or purchase. Podcast creator provides the necessary tools to create content and compile its required metadata for publishing, discovery, cataloging, and extending the content to a series. Tools dictate the constraints on the applications and content, constraining options for developers, thereby providing safeguards against non-compliant content types. Additional governance mechanisms have been institutionalized by Apple to review content for appropriateness and security before it is published and to review the underlying protocol for communication (itms).		

Discussion: Concept of Computing Environment

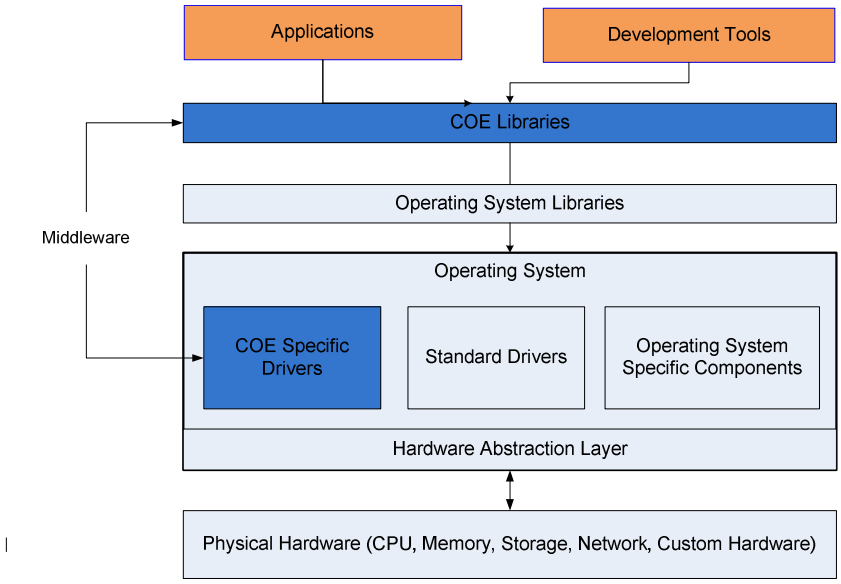
A *computing environment* (CE) comprises the necessary hardware and software required to run applications within the common operating environment*. Operating environments* and *computing environment*s are essentially the same, the key difference being what each runs. Operating environments are used to develop applications, and use development tools to design and create applications to run in a computing environment. The computing environment by itself includes the necessary hardware, operating system, and library support in order for it to run applications within the common operating environment.

Example	Computing Environments	Common Operating Environment
Description		
Cisco WebEx	MS Windows PC, MAC-0S4, iPhone, Blackberry	WebEx Meeting Services Platform
Microsoft NetMeeting	MS Windows, Linux, MAC-OS4	Open or closed network, custom user defined
Apple iTunes	MS Windows, MAC OSX, Apple TV, iOS	Mac OS X 10.6 Snow Leopard, iTunes Protocol (itms)

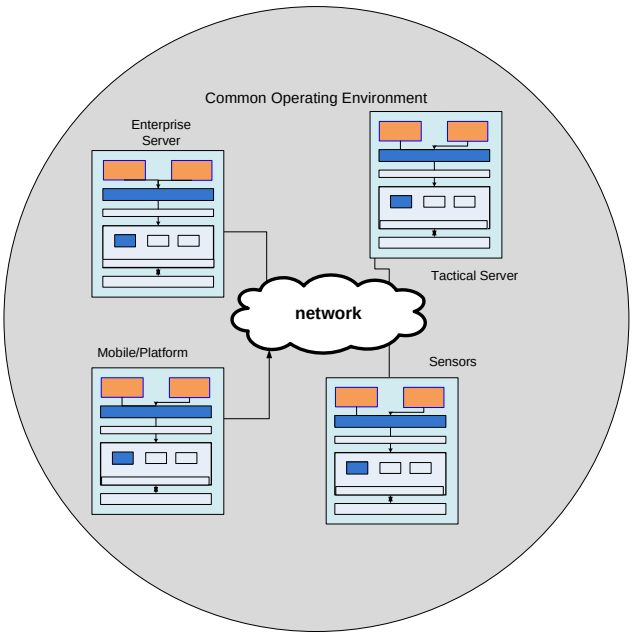
Discussion: Concept of Middleware

Middleware is custom software developed to support specific applications or COEs installed and configured in an OE. Middleware can be thought of as drivers and libraries used to develop and run specific applications in a given COE and OE, targeting specific CEs. Middleware is customized software that modifies, overrides, or enhances the standard offerings of a commercial off-the-shelf COE and is tightly coupled to applications and the OE it supports. It is developed and packaged as a set of software components that are reusable from application to application to provide a consistent look and feel or functionality within a given application domain.

Commercial examples of middleware include Embedded Windows XP/CE (customized libraries in Windows for embedded applications) and the Adobe Reader plug-ins for Internet browsers.



Overall System View



Appendix C: Case Study—Software Assessment Results

Business Context

As described in Section 2.3 of this report, to fully characterize the key strengths, weaknesses, and risks associated with a given technical solution, a prioritized set of business goals provide the necessary guidance in conducting an evaluation. As part of the case study, the ASA(ALT) team provided the prioritized list of goals for evaluating the capability and suitability of a common operating environment, shown in Table 8. (A fifth goal "reduced overall cost" was also provided. However, because this technical solution was evaluated in isolation, this goal was not included.)

Table 8: Business Goals and Priorities

Business Goal	Description	Importance
Operational relevance	The COE supports the needs of the warfighter, not the application development organization.	1
Interoperability	The COE enables interoperation and data sharing of systems, applications, and data sources.	2
Reduced development time	The COE enables rapid application development, integration, and deployment.	3
Reduced certification cost	The COE reduces certification costs through the use of common technology that can be certified once and used in multiple computing environments.	4

Table 9: Quality Attribute to Business Goal Mapping

Business Goal	Questions								
	Adaptability	Architecture	Functional	Governance	Interoperability	Development/Test	HW/SW Platform	Quality of Service	Information Assurance
Operational relevance	Q6	Q3,8, 10	Q1,3	Q4	Q2-7		Q1,2	Q1,2	Q1-4
Interoperability	Q1,4	Q5,7, 9			Q1,5,6, 7			Q1-3	Q3,4
Reduced development time	Q2,3,5	Q5,6, 11,12, 13	Q2,3	Q1-4	Q2-8	Q1-9		Q1-3	Q2-4
Reduced certification cost	Q2,3,5	Q11	Q2,3	Q1-4	Q2-8	Q1,2	Q3	Q1-4	Q2-4

Summary Characterization

Based on the answers and supporting documentation provided by the technical team, a characterization was developed using the business goals described in the Business Context section on page 20 of this document. For details relating to the characterizations, see Table 10, which summarizes the significant strengths and weaknesses across all areas.

Table 10: Summary Technical Solution Strengths and Weaknesses

Business Goal	Strengths	Weaknesses	Assessment
Operational Relevance	Because technical solution is middleware, much of the operational relevance comes from the applications that the technical solution enables, most notably fault-tolerant communications capabilities. Quality Attribute Areas: • Architecture Q3,9, 10	1) Applications and Infrastructure must be in lockstep, creating tight coupling between the COE and deployed applications. Already deployed applications are not guaranteed to function with COE changes. 2) Licensing constraints may preclude support for platforms beyond the current solution set. Quality Attribute Areas: • Interoperability Q2, 3 • Development & Test Q7 • Support for Governance Q4 • Hardware and Software Platform Q1,2	Risk Area
Interoperability	Loose coupling between infrastructure and task capabilities with adapters to provide interoperation support.	Adapter extensibility support could lead to fragmentation and undermine the Army's ability to migrate toward unified approaches.	Key Strength
Reduced Development Time	1) Dynamic services discovery (AR11) 2) COTS development tools can be used (DT2) Quality Attribute Areas: • Architecture Q11 • Development & Test Q2 • Interoperability Q6, 7	1) Development tools can bypass system infrastructure. 2) Task integrated network (TIN) is a custom-developed workflow solution that competes with commercial alternatives. 3) Technical solution may be tightly coupled with licensing constraints. 4) Performance relating to coordination across application boundaries has not been established. 5) The complexity, fragility, and complexity of both configuration files and their proliferation is high. Quality Attribute Areas: • Architecture Q4, 6, 13 • Functional Q1, 2 • Support for Governance Q1, 2 • Adaptability Q5	Risk Area
Reduced Certification Cost	COTS testing and certification tools could be used in place of custom in-house solutions. Quality Attribute Areas: • Development & Test: Q2	Provided development tools only govern source syntax and do not govern usage. This may result in extensive process/policy governance and testing. This, in turn, may offset any savings afforded by use of a common framework Quality Attribute Areas: • Architecture Q6 • Support for Governance Q1, 2	Weaknesses Identified

Operational Relevance: Risk Area

This area was characterized as an area of risk because of weaknesses in how versioning is handled within this framework. Because only a single version of the framework is supported on a given node or set of connected nodes, applications must be developed, tested, certified, and deployed

with the framework each time changes occur in the common libraries. While it is possible to assure functional backward compatibility with this architectural approach, behavioral and performance-related compatibility cannot be guaranteed nor expected whenever common code is changed and deployed onto one or more connected nodes. Because this weakness may have a direct impact on warfighter access and availability to applications in mission-critical environments, this goal is deemed a risk area with this technical solution.

Interoperability: Key Strength

This area was characterized as an area of strength for this technical solution. That is, it provides a complete technical solution for interoperation with existing applications, services, and yet-to-be-developed applications and services using other technical solutions. This area of strength could become an area of weakness and liability because it can be used to undermine migration to unified approaches, increasing the number of interoperable interfaces and adapters.

Reduced Development Time: Risk Area

This area was characterized as an area of risk for this technical solution. No one weakness stands out as a critical risk element, but the number of weaknesses, when taken collectively, serve to put this goal at risk. Some of the more notable weaknesses include:

- Complexity and fragility of configuration files

  Configuration files drive the behaviors and security of applications developed under this framework. Because applications are wholly dependent on these configuration files to operate, they require developer expertise to develop, integrate, test, and deploy. Additionally, while they are field modifiable, this further increases the complexity of their use and increases the opportunities for errors and omissions.
- TINS workflow

  This is a custom-developed workflow solution. With the ever-increasing set of technologies and programming paradigms, the commercial sector will likely migrate toward one dominant solution for workflow: Business Process Execution Language (BPEL). Developing competing alternatives may not make financial or technical sense in the long run.
- Tools can bypass COE infrastructure

  Because the technical solution is a set of custom-developed libraries integrated on a commercial platform base, COTS development tools can circumvent these libraries in a manner that may violate the intent of their use. Put simply, the technical solution does not constrain the developer to use (or not use) its libraries as intended. Additionally, this technical solution provides no means or mechanisms to detect when and how it has been bypassed, increasing the complexity associated with development, testing, evolution of the framework, and governance of programming and application-development practices.

Reduced Certification Cost: Weaknesses Identified

This area was characterized as an area of weakness. While it should be recognized that COTS development and testing tools can be used to develop, test, and certify applications, the acquiring

organization must develop all necessary infrastructure, tools, and standards for certifying applications. Additionally, taking into consideration the ability of a developer to circumvent the technical solution's functional libraries, this is an added complexity with which the acquiring organization must contend.

References

URLs are valid as of the publication date of this document.

[O'Brien 2005]
O'Brien, L.; Bass, L.; & Merson, P. *Quality Attributes and Service-Oriented Architectures* (CMU/SEI-2005-TN-014). Software Engineering Institute, Carnegie Mellon University, 2005. www.sei.cmu.edu/library/abstracts/reports/05tn014.cfm

[U.S. ARMY CIO/G-6 2010]
U.S. Army CIO/G-6. *Common Operating Environment Technical Architecture, Appendix F to the Strategy for 'End State' Army Network Architecture—Tactical*, August 2010. www.us.army.mil/suite/files/22614179

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	February 2011	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
A Framework for Evaluating Common Operating Environments: Piloting, Lessons Learned, and Opportunities	FA8721-05-C-0003

6. AUTHOR(S)

Cecilia Albert & Steve Rosemergy

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2010-SR-025

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116	

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT	12B DISTRIBUTION CODE
Unclassified/Unlimited, DTIC, NTIS	

13. ABSTRACT (MAXIMUM 200 WORDS)

This report explores the interdependencies among common language, business goals, and software architecture as the basis for a common framework for conducting evaluations of software technical solutions. It also describes the SEI's experience piloting this framework, which integrated commercial technologies, customized open-source systems, and legacy systems, and the insights gained from the project. As described in the report, those insights have enabled the SEI to further refine the framework to make it reusable and applicable for a variety of technical solutions.

14. SUBJECT TERMS	15. NUMBER OF PAGES
system of systems engineering, common operating environment, computing environment, operating environment, software systems quality attributes, business goals and software quality attributes, business goals and system quality attributes, Army enterprise common operating environment convergence, software evaluation framework	38

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL