MTR100307

MITRE TECHNICAL REPORT

MITRE

Governance of Composable Capability on Demand (CCOD)

Robert L. Pancotti, Scott Ankrum, Denise C. R. Benel, Scott R. Bennett, Joseph A. Burley, Robert A. Martin

September 2010

This page intentionally left blank.

MTR100307

MITRE TECHNICAL REPORT

MITRE

Governance of Composable Capability on Demand (CCOD)

Robert L. Pancotti, Scott Ankrum,
Denise C. R. Benel, Scott R. Bennett,
Joseph A. Burley, Howard Kong,
Robert A. Martin

September 2010

Sponsor: US Army
Dept. No.: E133
Contract No.: W15P7T-10-C-F600
Project No.: 0710M640-AA

Approved for Public Release: 10-3924.
Distribution Unlimited.

Bedford, MA

Approved By:

Name and Title of Approval Signature	Date

Abstract

This paper documents the results of the Mission Oriented Investigation and Experimentation project called "Governance of Composable Capability on Demand (CCOD)." In this project we modify and extend the definition of IT Governance - "a framework that addresses strategic alignment, performance measurement, risk management, value delivery and resource management." (IT Governance Institute) - to cover the unique aspects of CCOD, particularly, the fact that users are the primary contributors to the information technology capability. "Composable Capability" implies that users compose pre-existing low level functional software components into compositions that themselves can be further composed into higher level capabilities without the need to resort to programming. The concept of "on Demand" refers to the desire to create software capability when and where it is needed, preferably, in the time frame of minutes to days.

The most compelling form of CCOD is that in which the users are empowered to compose their own capabilities and those capabilities are made available for others to reuse, modify and extend. In an environment where sharing and reuse are encouraged, governance becomes important. Processes that assure performance and reduce risk are clearly needed. In a CCOD enabled community there is a need for a level of control that assures a sense of trust and, therefore, reduced risk. Users must know, to some extent, that the capability they are about to use will perform with some level of accuracy, timeliness, security, reliability, etc.

Beyond trust there are other needs for governance. For example, there will be a need for a process to respond to problems encountered during use. Original composers will not likely be able to respond to problems in their compositions.

In order to create this governance process we have defined the following:

- The environment being governed,
- The stakeholders in this process,
- The life cycle model to be used, and, most importantly,
- A way to measure the trustworthiness of the CCOD user products, the trust taxonomy.

Each of these products is described in the body of the paper and is provided in their entirety in the Appendices that follow.

The governance processes we define in this paper are intended to provide a starting point for a community planning to adopt a CCOD environment. Governance of CCOD, like any other form of governance, must be reassessed periodically to assure that the processes are providing the level of risk reduction desired while not becoming onerous to the extent where the community rebels. For CCOD, the balance will be between assuring trustworthy capability while not significantly affecting the desired timeliness and agility that CCOD can provide.

Table of Contents

List of Figures

List of Tables

This page intentionally left blank.

1 Introduction

This paper documents the results of the Mission Oriented Investigation and Experimentation project called "Governance of Composable Capability on Demand (CCOD)." There are three concepts present in the title of this project that require definition: "Governance," "Composable Capability" and "On Demand." These definitions form the foundation upon which this work is based.

1.1 Governance

CCOD is in the information technology domain; therefore, the concepts of CCOD Governance and information technology governance are closely related. The IT Governance Institute defines information technology governance as "a framework that addresses strategic alignment, performance measurement, risk management, value delivery and resource management" (IT Governance Institute).

In this project we modify and extend this definition to cover the unique aspects of CCOD, particularly, the fact that users are the primary contributors to the information technology capability.

1.2 Composable Capability

Traditional software implementation is performed by teams of programmers that create functional software (capability) by developing software line-by-line using various formal processes. These processes include requirements definition, software design, software development, formal test and evaluation and, finally, deployment.

"Composable capability" refers to the concept of creating IT capability by composing pre-existing low level functional software components into compositions that themselves can be further composed into higher level capabilities without the need to resort to programming.

1.3 On Demand

Within Department of Defense (DoD) software capability is obtained through a set of formal acquisition processes called the DoD Decision Support Systems. There are three parts to the DoD Decision Support Systems (Defense Acquisition University):

- Planning, Programming, Budgeting and Execution (PPBE) Process - The Department's strategic planning, program development, and resource determination process. The PPBE process is used to craft plans and programs that satisfy the demands of the National Security Strategy within resource constraints.
- Joint Capabilities Integration and Development System (JCIDS) - The systematic method established by the Chairman of the Joint Chiefs of Staff (CJCS) for identifying, assessing, and prioritizing gaps in joint war fighting capabilities and recommending potential solution approaches to resolve these gaps. CJCS Instruction 3170.01 and the Joint Capabilities Integration and Development System (JCIDS) Manual describe the policies and procedures for the requirements process.
- Defense Acquisition System - The management process by which the Department acquires weapon systems and automated information systems. Although the system is based on centralized policies and principles, it allows for decentralized and streamlined execution of acquisition activities. This approach provides flexibility and encourages innovation, while

maintaining strict emphasis on discipline and accountability. The Defense Acquisition System is described in a set of documents known as the DoD 5000 series of manuals.

These processes traditionally take months to years to complete. Often, by the time these processes are completed, the software delivered does not the meet the current need of the customer.

"On demand" refers to the desire to create software capability when and where it is needed, preferably, in the time frame of minutes to days.

1.4 The Need for Governance of CCOD

The most compelling form of CCOD is that in which the users are empowered to compose their own capabilities and those capabilities are made available for others to reuse, modify and extend. In an environment where sharing and reuse are not allowed, the need for governance is minimized. In a non-sharing environment the risk of using a composed capability falls only on the composer/user of the capability. In an environment where sharing and reuse are encouraged, governance becomes more important. Processes that assure performance and reduce risk (as defined in IT Governance) are clearly needed. In a sharing environment there is no easy way for the person reusing or extending a capability to know its trustworthiness (defined here broadly as: to rely upon or have confidence in). The formal DoD Decision Support Systems attempt to assure trustworthiness and manage risk through the strict processes of requirements definition and, later, testing and evaluating that the resulting capability properly addresses those requirements. In an environment that allows users to compose capability it will be unlikely that there will be formally defined requirements. Instead, users will compose capabilities as needs arise. In addition, the demand for capabilities and the ability to modify and extend them as needed reduces the need for formal testing.

So, why not let the CCOD community govern itself? Let the chips fall where they may? In an environment where the risks are minimal, like an on-line gaming community, this is viable option. If a composed game doesn't work properly or isn't easy to use, it will be shunned by the community resulting in its disuse. In a military community (or any other more demanding community, for that matter) there is a need for a level of control that assures a sense of trust and, therefore, reduced risk. Users must know, to some extent, that the capability they are about to use will perform with some level of accuracy, timeliness, security, reliability, etc.

Beyond trust there are other needs for governance. For example, there will be a need for a process to respond to problems encountered during use. Original composers will not likely be able to respond to problems in their compositions. So, in these environments governance processes are needed.

The governance processes we define in this paper are intended to provide a starting point for a community planning to adopt a CCOD environment. Governance of CCOD, like any other form of governance, must be reassessed periodically to assure that the processes are providing the level of risk reduction desired while not becoming onerous to the extent where the community rebels. For CCOD, the balance will be between assuring trustworthy capability while not significantly affecting the desired timeliness and agility that CCOD can provide.

2 The Governance Concept

The basic concept we propose for the governance of CCOD deals with defining a state transition process or life cycle process for the primary products of a CCOD environment that allows users

to compose capability. This life cycle process defines how these products move from the "private" state (where they are only available for use by their originator and the originator takes on the risk of use) to the "public" state (where everyone in the community that has access to the CCOD environment can reuse and modify them and the risk of general use is mitigated by the evaluation of the product followed by potential improvement prior to the state change). State transitions are governed using community developed policies that assess the trustworthiness of the products being transitioned. Once the product is deemed sufficiently trustworthy, it transitions to the public state and its ownership is changed from the originator to an organization whose job it is to maintain all public composition products. In order to create this governance process we have to define the following:

- The environment being governed,
- The stakeholders in this process,
- The life cycle model to be used, and, most importantly,
- A way to measure the trustworthiness of the CCOD user products, the trust taxonomy.

These products are defined in the following subsections.

2.1 The environment being governed

A prototypical CCOD environment primarily consists of three parts:

- A software environment in which composition of capability is made possible. We call this the CCOD User Environment.
- Components which are the lowest level of building blocks for compositions are made up of software and should provide fairly low level cohesive functionality. An example of a component is the ability to filter data based on some criteria.
- Within the CCOD User Environment users are able create compositions that form end user capability. Compositions are not software, per se; they are representations of linked together compositions and/or components.

We describe each of these concepts in further detail below.

2.1.1 CCOD User Environment

A CCOD User Environment provides the tools and processes needed to develop components, compositions and capabilities in an integrated fashion. Not all CCOD User Environments will provide the same range of function as described below.

Component developers and composers in most cases will be the primary users of the CCOD User Environment. In some cases, the end user may take advantage of some of the functionality of the CCOD User Environment to gain access to the capabilities that are ultimately provided.

The CCOD User Environment should maintain a collection of meta-data for components, compositions and capabilities. These meta-data will describe the trustworthiness of each of the products (components and compositions) within the CCOD User Environment. Goal functionality for a CCOD User Environment should be to compose trust as the composer composes. In this way the composer immediately sees the effect of selecting particular components and compositions for inclusion in his/her composition.

The CCOD User Environment itself will have some effect on the trustworthiness of capability developed within it, therefore, a collection of trust elements should be maintained for the CCOD User Environment.

A full-functioned CCOD User Environment will likely provide a programming environment for development of components. Less functional CCOD User Environments will rely on external software development environments for component implementation. An integrated component development environment can provide methods to help gather metrics associated with components. More capable CCOD User Environments will provide the ability to create test cases for a component and execute those test cases as new versions of a component are developed. The CCOD User Environment should maintain versions of components in some form as a configuration management facility. The CCOD User Environment should maintain a catalog or directory of components and compositions for use by the composer. We propose that this catalog provide information about the trustworthiness of the components/compositions to assist the composer in creating the most trustworthy compositions and capabilities. CCOD User Environments should provide a method for compositions developed in one environment to be used in other environments. A software architecture that is particularly suited for this type of integration is Representational State Transfer (REST)[1]. This should be done by exposing compositions as a RESTful web services[2]. CCOD User Environments should provide a method of reporting what web services are available, along with the trust meta-data associated with the composition for potential use in a receiving CCOD User Environment.

2.1.2 CCOD Component

A component is the lowest level of functionality within a composition. It generally consists of software that provides a loosely coupled, highly cohesive function. A component should have clearly defined input and output interfaces and functional assumptions. Input and output data should be provided in widely accepted standard formats, such as XML, JSON, etc.

Components are used by composers to create compositions.

2.1.3 CCOD Composition

A composition is a collection of components “stitched” together by a composer to form a higher level function. The method of “stitching” components is dependent on the CCOD User Environment. For example, several commercial composition environments, such as MashableLogic, JackBe, and Yahoo Pipes, use a “pipe” or “wire” metaphor to connect one component’s outputs to another’s inputs.

Some commercial composition environments, i.e., Yahoo Pipes, only allow compositions composed of components. This essentially results in “flat” or single-level compositions where the addition of functional complexity results in larger and larger compositions. We feel compositions should be hierarchical. This allows complexity to be hidden within sub-compositions and, as a result, provides higher level abstractions for end users. This implies that

[1] REST-style architectures consist of clients and servers. Clients initiate requests to servers; servers process requests and return appropriate responses. Requests and responses are built around the transfer of "representations" of "resources". A resource can be essentially any coherent and meaningful concept that may be addressed. A representation of a resource is typically a document that captures the current or intended state of a resource.

[2] A RESTful web service (also called a RESTful web API) is a simple web service implemented using HTTP and the principles of Representational State Transfer (REST). Such a web service can be thought about as a collection of resources. The definition of such a web service can be thought of as comprising three aspects:

The base URI for the web service, such as http://example.com/resources/

The MIME type of the data supported by the web service. This is often JSON, XML or YAML but can be any other valid MIME type.

The set of operations supported by the web service using HTTP methods (e.g., POST, GET, PUT or DELETE).

compositions can be made up of other compositions. This also implies that a composition, like a component, should have clearly defined input and output interfaces and functional assumptions.

Compositions may execute outside the CCOD User Environment as Restful web services. These web services may be exposed from some other CCOD User Environment, in which case they should provide the needed meta-data to help understand the trustworthiness of the web service.

A composition's trustworthiness is derived from the trustworthiness of its composed components and compositions, and from the CCOD User Environment in which it was composed.

Compositions are used by composers to create other compositions and, ultimately, capabilities through the addition of user interface components.

All compositions that have been publicly released (see Section 2.3 - Composable Capability Life Cycle, below) are available for other composers to reuse in new compositions. Publicly released capabilities are also usable by end users.

2.2 Stakeholders

Early in this effort we performed a stakeholder analysis for a CCOD community. We used a process defined by Ian Alexander called the "Onion Model of Stakeholder Roles" (Alexander, 2005). This model consists of four default concentric circles forming an "onion" and, hence, a taxonomy of stakeholders. The four circles represent viewpoints and interactions. A stakeholder in one layer interacts directly with other stakeholders in the same layer and those in adjacent layers. Interactions across multiple layers must pass through stakeholders in the intermediate layers. The four default circles described by Alexander, from outer most to inner are:

- 'The Wider Environment': 'The Containing System' plus any other Stakeholders. For our model this includes all other stakeholders including those that can affect or are affected by the CCOD User Environment.
- 'The Containing System': 'Our System' plus any human Beneficiaries of Our System (whether they are involved in operations or not). In our model this includes stakeholders such as those that benefit from composed capability and the owners of the physical environment in which the CCOD User Environment lives.
- 'Our System': 'The Product' plus its human Operators and the standard operating procedures or rules governing its operation. In our model this is the CCOD User Environment Layer. In this layer are the stakeholders that interact to form and manage compositions.
- 'The Product' or 'The Kit': the item under development, e.g., a software program, a consumer electronics device, an aircraft, a communications network. For our model we use the concept of CCOD as the center. This represents the collection of composed capabilities that can be used.

The diagram below represents the onion model for CCOD. Subsections following the diagram will describe each layer and its associated stakeholders.

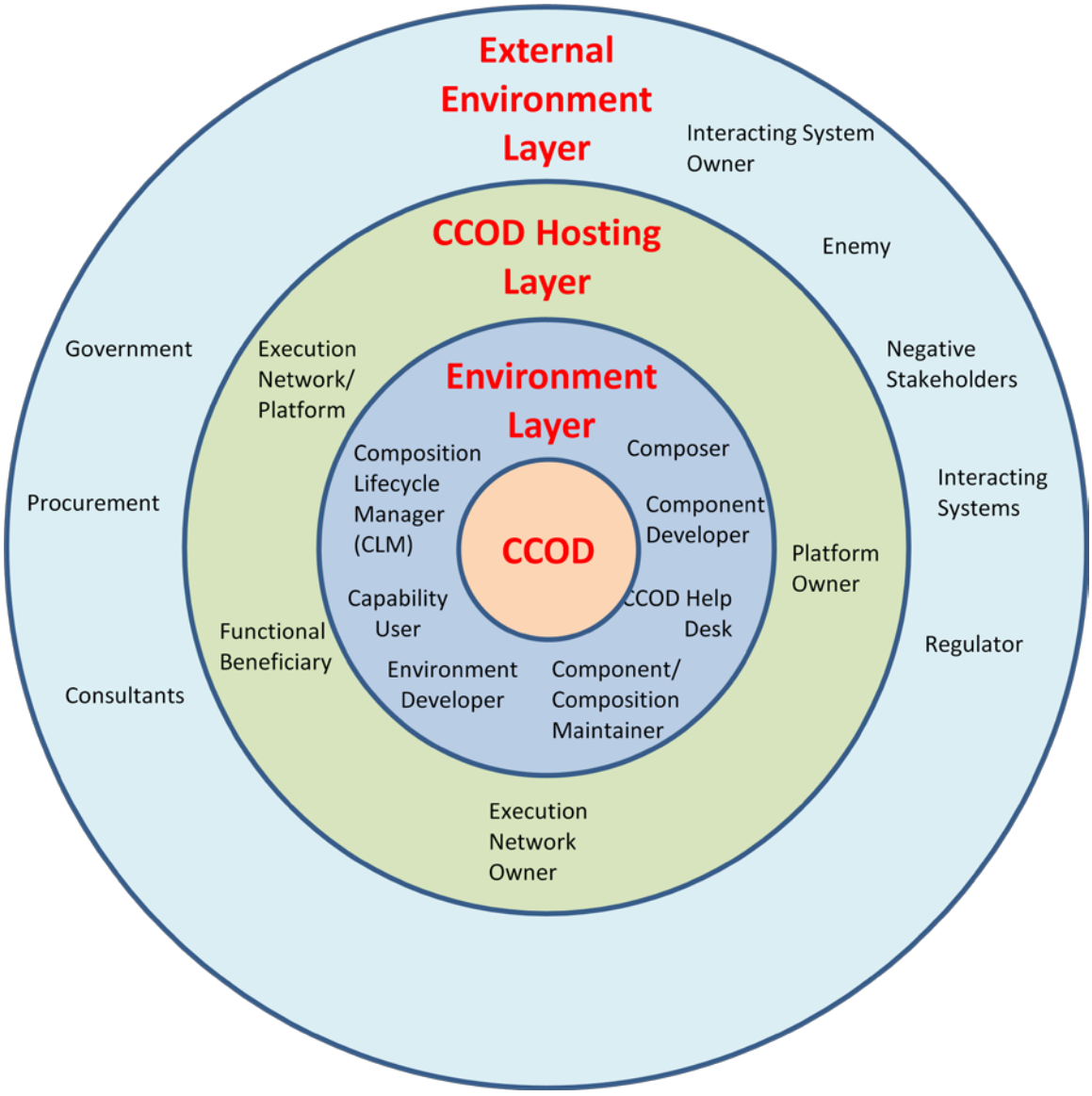


Figure 2-1 Stakeholder Onion Diagram

2.2.1 External Environment Layer

Stakeholders in this layer:

- Interacting System Owner – Owner of the system with which the CCOD composition interacts.
- Interacting Systems – Any system that is affected by the execution of a composition in the CCOD User Environment. Of primary concern here are the resources that are utilized by the composition.
- Negative stakeholder – Any role that could be harmed by CCOD physically, financially or in any other way that might be found justifiable.
- Regulator – Any role responsible for regulating the quality, safety, security, cost or other aspects CCOD. For example, aviation authorities, health and safety authorities, security authorities. Regulators may act as surrogates for the Public.

- Consultants - Any one of many roles (expert in topic, software expert, security, analyst, etc., involved in supporting some aspect of CCOD.
- Procurement – Any role responsible for having parts of a CCOD User Environment developed. A purchaser, a surrogate for the consumer/user of CCOD product.
- Government – Political beneficiary. A political beneficiary is any role in public office or private business that can benefit in terms of power, influence and prestige through the success of a CCOD product.
- Enemy – Anyone who has hostile intent on harming the system or accessing the system without proper authorization. Without this group of stakeholders there would be little need for security measures.

2.2.2 CCOD Hosting Layer

- Execution Network/Platform – The network/platform where the CCOD User Environment and by extension, its compositions, run.
- Execution Network Owner – The owners of the network upon which the CCOD User Environment runs.
- Platform Owner – Owner of the platform which hosts the CCOD component or composition.
- Functional Beneficiary – Any role that benefits from the results or outputs created by the CCOD. Interacts with Capability Users in the CCOD User Environment layer. In the military world this would be the commander of a person that composed a capability, where that capability was used to support a decision made by the commander.

2.2.3 The CCOD User Environment Layer

- Capability User – Users of composed capabilities. One of the three classes of "users" that participate in the CCOD life cycle: component developers, composers and capability users.
- Component Developer – One of the three classes of "users" that participate in the CCOD life cycle. Develops the CCOD component. They interact directly with the CCOD User Environment; with Composers (may request a new component) and the Composition Life Cycle Manager (during the transition of developed components from public to private state). Component developers create components using a programming environment. The trustworthiness of the component developer contributes to overall trustworthiness of compositions that contain his/her components.
- Composer – One of the three classes of "users" that participate in the CCOD life cycle. Persons in this role compose compositions from CCOD components. They interact directly with the CCOD User Environment; with Component Developers (to request a component) and the Composition Life Cycle Manager (during the transition of compositions from public to private state). The composer does not need programming skills to compose a capability. The trustworthiness of the composer contributes to overall trustworthiness of the compositions that he/she composes.
- CCOD Help Desk – Provides operational support to the Capability User and other stakeholders when using CCOD compositions.
- CCOD User Environment Developer – Is responsible for developing the CCOD User Environment and supports the CCOD Help Desk when problems cannot be resolved locally.
- Composition Life Cycle Manager (CLM) – Manages the life cycle of the CCOD components, compositions and capabilities. This stakeholder is instrumental in executing the processes needed to transition components, compositions and capabilities from private state to public

state. The CLM also manages the configuration of the CCOD User Environment and all of its parts.

- Component/Composition Maintainer – Maintains components and compositions after they change ownership during the private to public transition in the CCOD Life Cycle.

2.3 Composable Capability Life Cycle

We propose that the products of a CCOD User Environment, components and compositions, move through a well defined life cycle that we have chosen to model using a state transition model. The highest level of this model is shown below, more detailed models are provided in Appendix A.

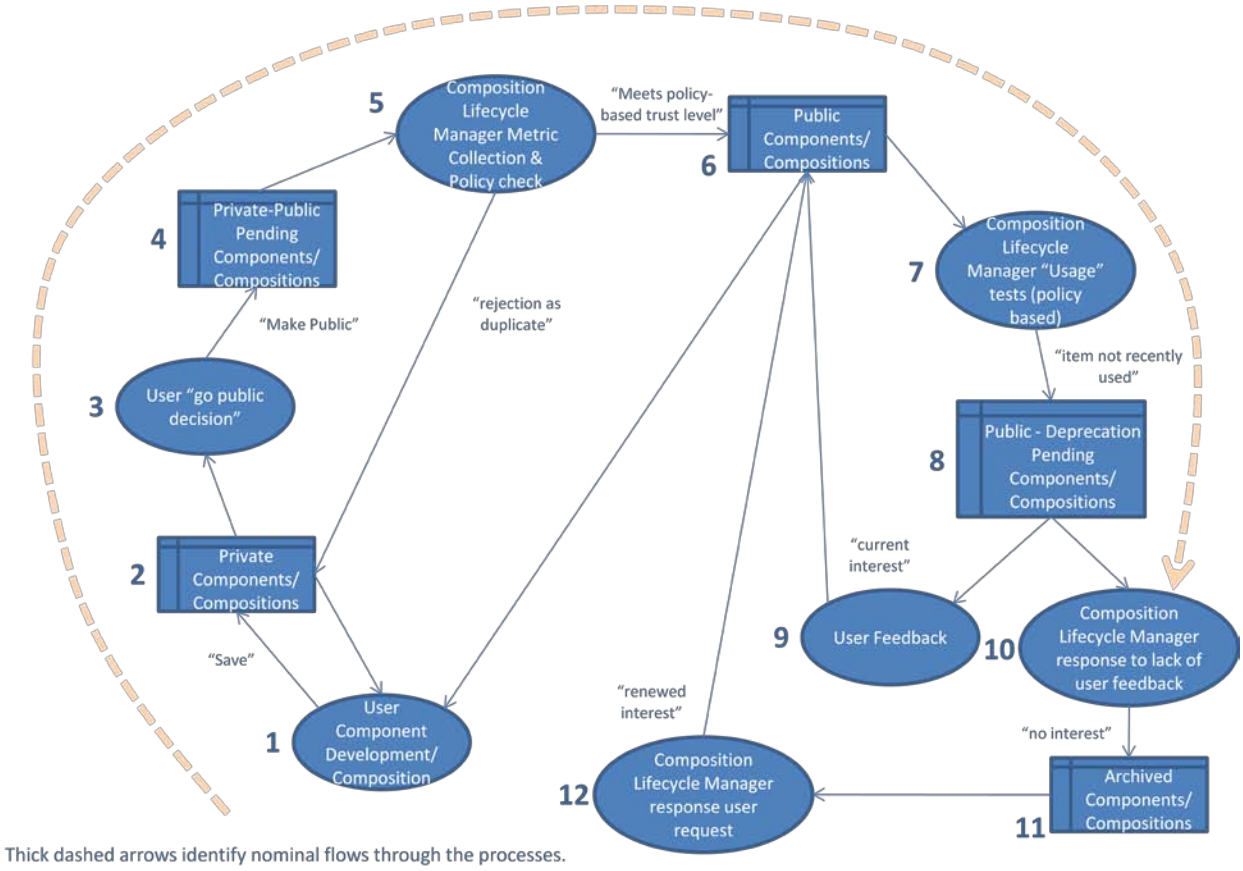


Figure 2-2 Composable Capability Life Cycle

The following text describes each of the elements of the high level state model.

1. We believe that there are classes of users (as defined in the stakeholder section) that have the capability to develop components and create compositions and, ultimately, capabilities. In this process composers have at their disposal their own private components and compositions, as well as all the public components and compositions to create new compositions.
2. When a user saves either a component or composition it is assigned the state of "private."
3. This process suggests that the originator of a private component or composition initiates the transition of a component or composition to the private state. This process could be changed to automated process that scans for private components and compositions that meet some level of trustworthiness or provide some desired functionality.

4. Once the decision is made to make a component or composition public it is moved the state of "Private – public pending. In this state it is still considered to be in the private state from the perspective of availability for use in other compositions. The CCOD product will remain in this state until it is deemed trustworthy enough (a policy based decision comparing the product's trust metrics against the desired level of trust given the current operational situation) to move to the public state.
5. The stakeholder called the Composition Life Cycle Manager evaluates the Private – Public Pending products to determine if they are trustworthy enough to be made public. Policies should be developed that determine the desired level of trustworthiness. This policy can be changed over time based on the community's need. For example, during crises, the level of acceptable trustworthiness may be lowered to allow products to move more freely to the public state, making more capability available sooner but with higher risk. Conversely, during less stressful time, the level of acceptable trustworthiness may be raised to assure products that move to the public state are more trusted.
6. Public components and compositions are available for all to use in new compositions or to modify. If a public component is modified, it becomes a new version in the private state and moves through the process as though it was a new product.
7. CCOD products transition out of the public state when they are no longer used. This implies that there is a need to maintain usage statistics for products within the CCOD User Environment. Policy must be defined that determines when a product is considered no longer used. When a product is determined to be no longer used it is moved the Public – Deprecation Pending state.
8. Products in the Public – Deprecation Pending state are advertised to the community as pending deprecation.
9. If a user uses a product in this state or provides feedback that the product is needed it will be moved back to the public state where its usage is reset.
10. If no user provides feedback after a policy defined time period the product is moved to the Archived state and removed from the catalog of available public products.
11. The product, however, is retained in the Archived state and users are given access to the list of archived products.
12. Users can trigger a state change from Archived back to the Public state whenever the product is needed again.

2.4 Trust Taxonomy

The Trust Taxonomy describes elements of trust for successful development and use of user composed capability. Trust elements are broken down into categories. For each trust element a definition is provided, as well as metrics defined to allow one to measure that trust element. Main categories for trust elements include: CCOD User Environment, Component, Composition (a capability is a special case of a composition that includes user-interface components), Component Developer, Composer and Component/Composition Maintainer.

Component developer, Composer and Component/Composition Maintainer are user roles within the CCOD User Environment. The end user (of capabilities) is also a user role of the CCOD User Environment. A single person can perform all roles if properly trained or skills are present,

but it is assumed that training requirements or skills needed diminish as we move from component developer to composer to end user.

The overall strategy for how this taxonomy was developed is as follows: We believe that in order for CCOD to be successful two things must be in place. First, composed capabilities, compositions and components must be reusable beyond their original creators. Second, there must be some way for the person reusing these products to trust their execution. The dictionary definition of trust is used here: "To have or place confidence in; depend on." In standard government software acquisition this type of trust is earned through a formal requirements definition process, followed by a formal test and evaluation process. If the product successfully passes its test and evaluation, it is trusted to perform the functions specified in its requirements.

User composed capability is not likely to use formal requirement definition or test and evaluation processes, because these formal trust process would be significantly more burdensome than the composition process. Trust must be gained through some other method. We propose that trust in a composed capability be gained through a governance process that requires components, compositions and CCOD User Environments undergo gathering of rigorous quality metrics that focus on the following areas: functionality, security, reliability, usability, efficiency, maintainability, portability, and quality in use. To avoid burdening the operational user, these metrics will be gathered during the transition of components and compositions from the private to public state prior to their availability for reuse. This state change also results in a change of ownership from the composer or component developer to a community organization (this could be a contractor) whose role is to gather these metrics and determine, based on community policy, if the item is "trustworthy" enough to be publicly released. These trust metrics when combined will represent the trustworthiness of the composed capability.

The ISO 9126 - Software Engineering - Product Quality standard (ISO01, 2001) (ISO02, 2001) (ISO03, 2001) (ISO04, 2001) defines a taxonomy of metrics to measure software system quality. Alvaro, et al (A. Alvaro, 2005) (A. Alvaro, 2005) (A. Alvaro, 2006), tailored the 9126 quality taxonomy specifically for software developed as reusable components, resulting in the Component Quality Model (CQM). Reusable software components, either within a CCOD User Environment or external to a CCOD User Environment, as web services, are required to make CCOD viable. The Component Quality Model (CQM), however, does not adequately support the CCOD concept. Reusable software components within the CQM are assumed to be developed such that other software developers can use them to rapidly create larger software functionalities. In the case of CCOD, the intended user is not a software developer, so some changes to Component Quality Model (CQM) are needed. Our approach began with the CQM model. We then tailored CQM to deal with CCOD concepts. As the CQM model only defined the structure of the taxonomy, we referred to ISO 9126 to define the appropriate metrics for many of the categories and sub-categories.

The most significant tailoring to CQM involved:

- Creation of a major branch of the taxonomy dealing with the quality of the CCOD User Environment, within which compositions are created. The CCOD User Environment can be considered a "system," so the taxonomy for this branch closely follows that defined in ISO 9126.
- Creation of a major branch in the taxonomy for the "composition." A composition is a CCOD concept in which components are composed into greater functional products. A composition is a collection of components held together by the CCOD User Environment during composition time and execution time.

- The major branch called "Component" closely follows CQM as these are the lowest level elements of a composition made up of software and can be thought of as software components.
- The security section for the CCOD User Environment was defined based on NIST SP 800-53 (NIST, 2009) (defining a collection of security functions that should be present in a system) and the CWE/SANS Top 25 Most Dangerous Software Errors (MITRE, 2010) (defining a collection of software defects that contribute to insecure software). We desire to examine the software for the lack of these defects.
- The security section for the Component was defined based on the CWE/SANS Top 25 Software Defects.
- Additional functional capabilities of the CCOD User Environment were derived from the "Reference Architecture Foundation for Service Oriented Architecture" (OASIS, 2009). This document describes functions that should be present in a service oriented architecture environment, such as: Life cycle management, configuration management, policy management, event monitoring, accounting and quality of service management.
- Additional major branches have been added to gather metrics about the human composer, component developer and component/composition maintainer. These metrics will help determine the quality of new compositions/components based on past history of the users' products.

Throughout our taxonomy we retain the reference to the document where the item was originally defined. Changes, additions and deletions of elements defined in the original references are noted, with supporting rationale, in the section "Changes/Additions to Taxonomy from Reference Documents." The full Trust taxonomy can be found in Appendix B.

2.4.1 Major sections of the taxonomy

Appendix B is organized into three major sections:

1. Section one, The Taxonomy, consists of six subsections in two groups of three. The first three subsections describe trust metrics associated with the three functional elements of CCOD: CCOD User Environment, CCOD Component and CCOD Composition. In each of these subsections the concepts from the CQM are addressed (some are tailored out in particular subsections): functionality, security, reliability, usability, efficiency, maintainability, portability, and quality in use. All concepts but "quality in use" have a one-to-one relationship with the element being measured. Quality in use, which is intended to provide a mechanism for users to rate the element during use, can have a many-to-one relationship with CCOD elements. Many users can use and rate components, compositions and the CCOD User Environment.

   The second group of three subsections deal with trust metrics associated with the three defined stakeholders roles that contribute directly to the trustworthiness of components and compositions: Component Developer, Composer, and Component/Composition Maintainer. The meta-data for these stakeholders deals with the proficiency of the person performing the role. Proficiency is measured through two methods: by assessing the types and levels of formal training the person has taken; and by measuring and recording the average complexity of the components or compositions the person has maintained or developed.

   The rationale for training contributing to proficiency is that a trained individual will likely produce better results when developing or maintaining CCOD products. The rationale for collecting meta-data associated with complexity is that a person who successfully creates

CCOD products of a particular complexity will likely be able to successfully produce the next product of similar complexity. If, however, the user has consistently produced products of lower complexity and now has produced a product of much higher complexity, the likelihood of success is lower. Over time, other measures can be added to these human elements of the trust taxonomy.

- CCOD Component/Composition Maintainer – Each component or composition can have multiple maintainers. As a component or composition is maintained, the trust meta-data for that maintainer is associated with the modified item. In addition, the maintainer's meta-data is updated to reflect the complexity of the item maintained.
- CCOD Component Developer – Each component has a single developer. Whenever the component is saved the component developer's meta-data is associated with the component. In addition, the component developer's meta-data is updated to reflect the complexity of the component developed.
- CCOD Composer – Each composition has a single composer. When the composition is saved the composer's meta-data is associated with the composition. In addition, the composer's meta-data is updated to reflect the complexity of the composition.

2. Section two of Appendix B, Change Rationale, provides descriptions of and rationale for changes we made to the models defined in our reference models. This section consists of two subsections.
   - Changes/Additions to Taxonomy from Reference Documents – This subsection describes changes made to the reference models with rationale for each change.
   - Elements not used from Reference Documents – This subsection describes those elements of the reference models we chose not to use in the trust taxonomy. For each element not used a rationale is provided.
3. Section three of Appendix B, References, provides a list of reference documents used to derive this taxonomy.

2.4.2 Rolled-up Trust Value

The Trust Taxonomy is hierarchical. We feel that trust measures at any level in the hierarchy can be mathematically combined to form a trust value for the parent element in the taxonomy. A first cut algorithm for this role-up is to take simple averages of the metric values at each level in the tree. In order for this process to work properly, all metrics have been defined with ranges from zero to five. These averages can be rolled-up to ultimately give a composition or component a single trustworthiness value.

It is clear, however, that some metrics in the taxonomy should be considered more important than others. We believe the community should determine the relative weights of the trust metrics and those weights should be used to create weighted averages for each branch of the tree. To this effect we developed and administered a survey to determine how a small test community might weigh each of the metrics relative to one another. The questions and results from 13 respondents of this survey can be found in Appendix C. The version of the taxonomy used in the survey is earlier than the one presented here, so some inconsistencies with Appendix B will be noted. The most significant change was to the Security section of the taxonomy that was completely redefined, partially as a result of the survey.

3 Tools to Support Governance

The Trust Taxonomy is fairly extensive. Managing the development of and the run-time use of the taxonomy requires the assistance of automated tools. One of the project team members had experience in the Adelard[3] Assurance and Safety Case Environment (ASCE). This tool helps to develop what are called "Assurance Models" for the community concerned with system reliability. It was obvious this tool could help us develop the taxonomy and could, potentially, perform the roll-up of trust through the hierarchy. The standard roll-up capability with ASCE functions on Boolean values. For the governance of CCOD we need to roll-up numeric values. This can be done using the advanced plug-in capability of ASCE. Unfortunately, training is required in this advanced technique and we planned to extend the tool using this capability in FY11. ASCE was used extensively to validate the trust taxonomy, see Section 3.1.

For testing purposes we evaluated several data feeds present in the Simple C2 Marketplace. Rather than store the trust data in the Simple C2 Marketplace, we chose to store them separately and federate the data back to the Marketplace. In this way the Marketplace is able to display additional information about resources registered without the need to know the full details about that information. For more information about this capability, see Section 3.2.

With the absence of roll-up capability in ASCE, we developed a prototype stand-alone software capability to perform the roll-up. See Section 3.3 for more information on this prototype.

3.1 The Assurance Model

We used ASCE to structure and validate the trust taxonomy. The taxonomy is a text file with a hierarchical structure that can be modeled within ASCE. For each named CCOD element (CCOD User Environment, Component and Composition) and each role (Component Developer, Composer, and CCOD Component/Composition Maintainer), there is a hierarchical list in the trust taxonomy. We have used ASCE to construct a graphical tree for each of those hierarchies. These trees are not only visual hierarchies, but they contain all of the text and references to appropriate sections of the reference materials. Construction of the trees serves two purposes:

- The trees make each hierarchy easy to understand and traverse. It is much easier to see the relationships of the attributes and sub-attributes in a graphical tree than in text. The justifications or arguments of how sub-attributes support or fulfill the parent are more apparent. The logic of how the measures at the leaf nodes answer the goals they support and even the higher level attributes becomes apparent. Graphical representations of the six elements of the taxonomy are shown in Appendix D, The Assurance Model. The fully expandable version with documentation can be found in the Governance of CCOD Project Page.
- As each tree was built, the ASCE tool and the formal nature of the tree structure helped to identify mistakes and inconsistencies in the taxonomy. The tool has a built in syntax checker that flags invalid structures. For example, all leaf nodes (the bottom of the tree) must be measures and, therefore, define metrics. If any other node type exists at a leaf node, ASCE will flag it as an error. More importantly, the mere process of building each of the trees invokes a formality that reveals problems. For example, when an attribute (a goal in ASCE terminology) is supported by some number of sub-goals, there needs to be a strategy (justification or argument) that explains how those sub-goals support the parent goal. There have been several occasions when structuring the taxonomy in ASCE revealed the absence of

[3] More information about Adelard and ASCE can be found here: http://www.adelard.com/web/index.html.

proper justifications. The result in a higher quality taxonomy by assuring that each section of the tree is both necessary and sufficient.

ASCE has the capability to roll up Boolean measures to provide an overall trust measure for the entire tree. Our metrics are numeric values with a range of zero to five. Within the ASCE tool it is possible to develop plug-ins that can roll up numeric values. We intended to develop these plug-ins if follow-on funding was provided.

3.2 Federated Governance Data Store

During this project we collaborated with the project called SimpleC2. One outcome of that effort is the SimpleC2 Marketplace which provides a registry and discovery mechanism for finding resources (data sources, widget/gadgets, applications). The discovery component of the Marketplace extends to other participating searchable services by utilizing a "Federated Search" pattern whereby a user may use the SimpleC2 Federated Search web page to enter search criteria, and the SimpleC2 Marketplace server will relay and collate results from each search service presenting the results back to the user in a consistent manner. This enables the user community to have a "one-stop-shop" for all resources, whether they are registered with the Marketplace or elsewhere, and provides a model for an agile operating environment where disparate systems are brought together in an ad-hoc way.

In a similar way, the SimpleC2 Marketplace recognized that different partners in the federation will want to store different data about resources. For example, web services provided by a non-CCOD environment may be registered in the Marketplace while the trust metrics associated with those web services are stored elsewhere. SimpleC2 Marketplace uses an "Augmented Data" pattern for each result item from the Federated Search to augment the results with additional information from each partner. In this way, the user community truly benefits from the richness of the information in the federation from a single "one-stop-shop".

3.3 Trust roll-up prototype

We developed a software prototype roll-up capability for the trust taxonomy. This application takes as input an XML document containing the metric values for a composition and a similar document containing relative weights and produces the rolled-up values for trust at each level of the taxonomy as a new XML document.

The metric values are static and absolute in nature, providing the basis upon which all compositions and components can be evaluated consistently. The roll-up values are generated using a weighted average based on the metric and weight values, providing a means to evaluate and compare compositions and components dynamically. Through this technique weights can be changed to reflect the current policy set of the community. For example, during high levels of cyber attack threat, the weight for security can be raised to give it more importance.

The application also allows users to enter the trust metric values and/or the associated weight values. The resulting XML document, containing the rolled-up trust values, could then be stored on the federated server to be referenced by any application with federated search capability. Software developed for this effort can be found in the MITRE Internal Source Forge repository.

The XML schema and and an example document used to store a composition's trust values, rollups and weight values are provided in Appendix E, The Trust Taxonomy XML Structures. Note that the XML structures are slightly different than the trust taxonomy. Additional nodes have been added for identification purposes. In addition, the XML documents are structured for the representation of compositions that consist of components and other compositions. The

example documents provided are samples that show a composition made up of one component (a composition is made up of one or more components) and one composition (a composition can be made up of zero or more compositions). In an actual composition these nodes would be repeated for all components and compositions contained in the parent composition. The XML document also contains a node for the composer and zero or more nodes for maintenance events that each includes a node for the maintainer. Component nodes contain a sub-node for the component developer and zero or more nodes for maintenance events that include a node for the maintainer.

4 Application to Other Domains

The governance processes defined in this paper can be applied to domains other than CCOD. The metrics defined in the Component section of the taxonomy and a modified version of the life cycle model could be used as part of a larger governance process for Service Oriented Architecture developments. Services can be thought of as components and evaluated using the same metrics as those defined for components. Although this would not replace the formal acquisition processes inherent in non-CCOD environments, it could be used as an adjunct by providing additional information about the quality of software products. The trust taxonomy could be used to evaluate contractor's software deliveries where a desired trust value could be put on contract and become part of an overall evaluation for software acceptance.

5 Potential Follow-on Work

There are several opportunities for follow-on work:

- There are many metrics defined in the trust taxonomy. In order to be useful, automated processes need to be developed for the evaluation of the CCOD products. Fully manual processes for evaluation of components and compositions will likely take much longer than will be acceptable. We feel that an average complexity component or composition should be able to be evaluated for trustworthiness in just a few hours from the time they reach the Private – Public Pending state. For this to be possible we need to automate as much of the evaluation process as possible.
- Further work is needed to determine the best approach for rolling up trust values. The desire is to provide a single metric that represents the trustworthiness of a component or composition. The notion of weighed averages helps, but it may be necessary to allow the weights to be defined by individual users as they work to compose capability. In other words, the weights may need to change as the users' needs change. Some weights may be defined by the CCOD community as policies. More effort is needed in this area.
- Adelard's ASCE tool can be extended to provide numerical roll-up values for the trust taxonomy. It could be possible to use the tool through an existing programming interface to provide rollups to independently developed composition user environments. This would require working with Adelard to investigate this new way of using ASCE.
- Adelard's ASCE tool could be replaced with an updated version that conforms to the new Object Management Group (OMG) Structured Assurance Case Metamodel (SACM), making this experiment into an international standard's compliant demostration versus its current proprietary-tool implementation.
- It would be useful to incorporate trust metrics into an existing CCOD User Environment so that users, as they compose, can see how trust is affected by the choice of components and compositions used during composing. We envision a dashboard being provided that shows the current trustworthiness of a composition which would change values as components and

compositions are added. This concept is dependent upon the development of automated processes for the role up of trust values, possibly the use of ASCE, as described above.

- A follow-on activity includes completion of the roll-up prototype along with the ability to store metric and weight values on the federated data store. A RESTful web service that dynamically generates roll-up values could be provided to allow any application to send metric and weight values copied from the federated data store, and to receive the weighted roll-up in return.

Appendix A The Composable Capability Life Cycle

A.1 CCOD Component Developer View - 1.0

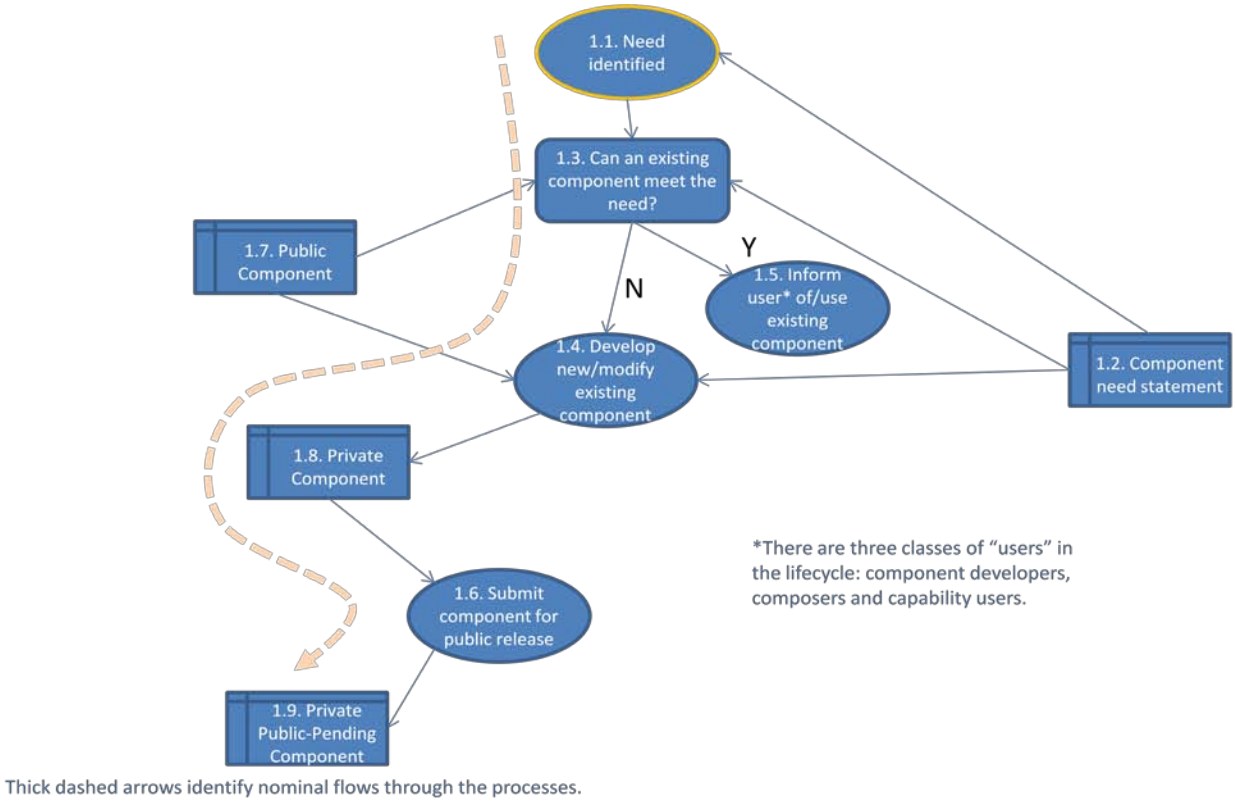


Figure A-1. CCOD Component Developer View - 1.0

All of the high level diagrams start at the orange outlined bubble. This diagram describes the processes by which the component developer within the CCOD User Environment develops new or modifies existing components and submits them for public release.

1. A need is identified. The need can come from a composer or can come from the component developer directly. In essence there is a need for a component to be developed.
2. User description of component needed along with user contact information. This may be informal information from a phone call or an email or may be in the head of the component developer.
3. A check is made against existing public components for an existing component that meets the need or can be modified to meet the need.
4. If there is no component that can meet the need either develop a new component or modify an existing component.
5. There already is a public component that meets the need, so inform the requestor that the component already exists.
6. When a new or modified component is completed, submit it for public release.
7. The directory of public components.
8. The directory of private components for this component.
9. Those components waiting for public release.

A.2 Develop new/modify existing component - 1.4

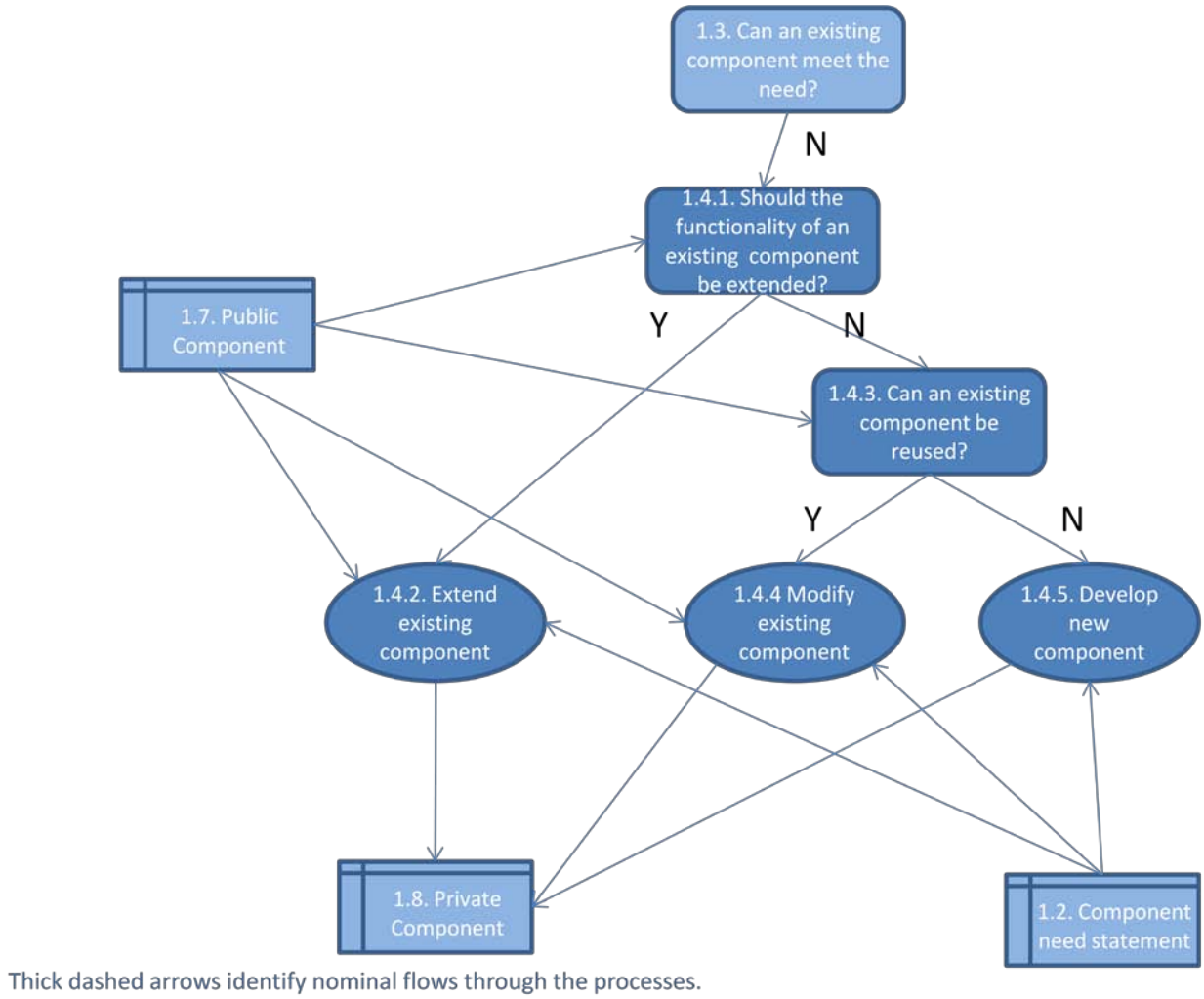


Figure A-2. Develop new/modify existing component - 1.4

This diagram describes how a component developer should fulfill the need for a new component.

The new component could be a new version of an existing component (with added functionality), a new component derived from an existing component (reuse the functionality) or a brand new component developed from scratch.

1. Check to see if there exists a component that could be extended to perform the desired function.
2. Extend an existing component to meet the user need by creating a new version of the original component.
3. Check if an existing component can be reused to simplify development.
4. Modify the existing component to meet the user need, creating a new component.
5. Develop a new component.

A.3 CCOD Composer View - 2.0

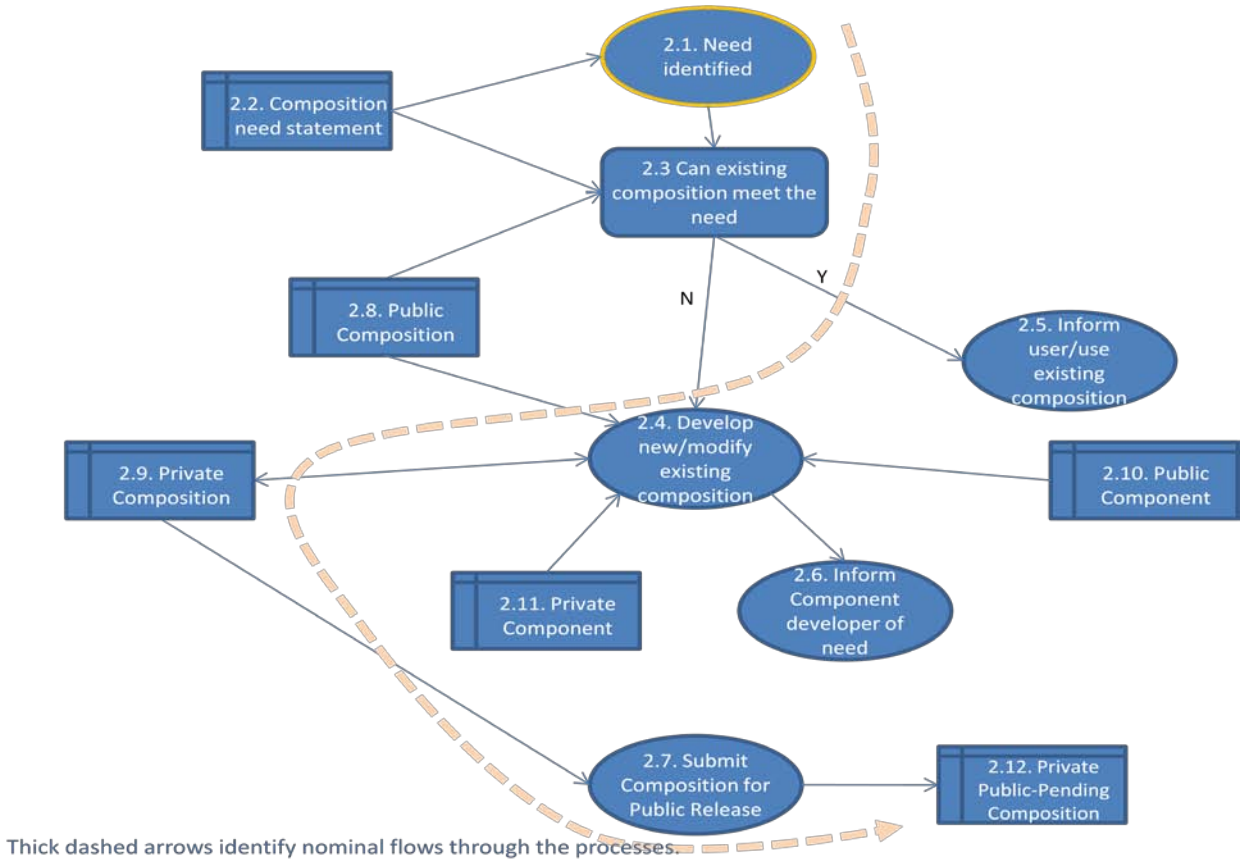


Figure A-3. CCOD Composer View - 2.0

This diagram describes the processes by which the composer within the CCOD User Environment creates new or modifies existing compositions and submits them for public release.

1. A need is identified. The need can come from someone other than the composer or can come from the composer directly. In essence there is a need for a composition to be created.
2. User description of composition needed.
3. A check is made against existing public compositions for an existing composition that meets the need or can be modified to meet the need.
4. If there is no composition that can meet the need either create a new composition or modify an existing composition.
5. There already is a public composition that meets the user's need so inform the requestor that the composition already exists.
6. As the composer creates a composition the need for a new component arises.
7. When desired, submit compositions to be promoted to a public composition.
8. The directory of public compositions.
9. The directory of private compositions for the composer.
10. The directory of public components.
11. The directory of private components for this composer.
12. Those compositions waiting for public release.

A.4 Composition Life Cycle Manager View (Component) - 3.0

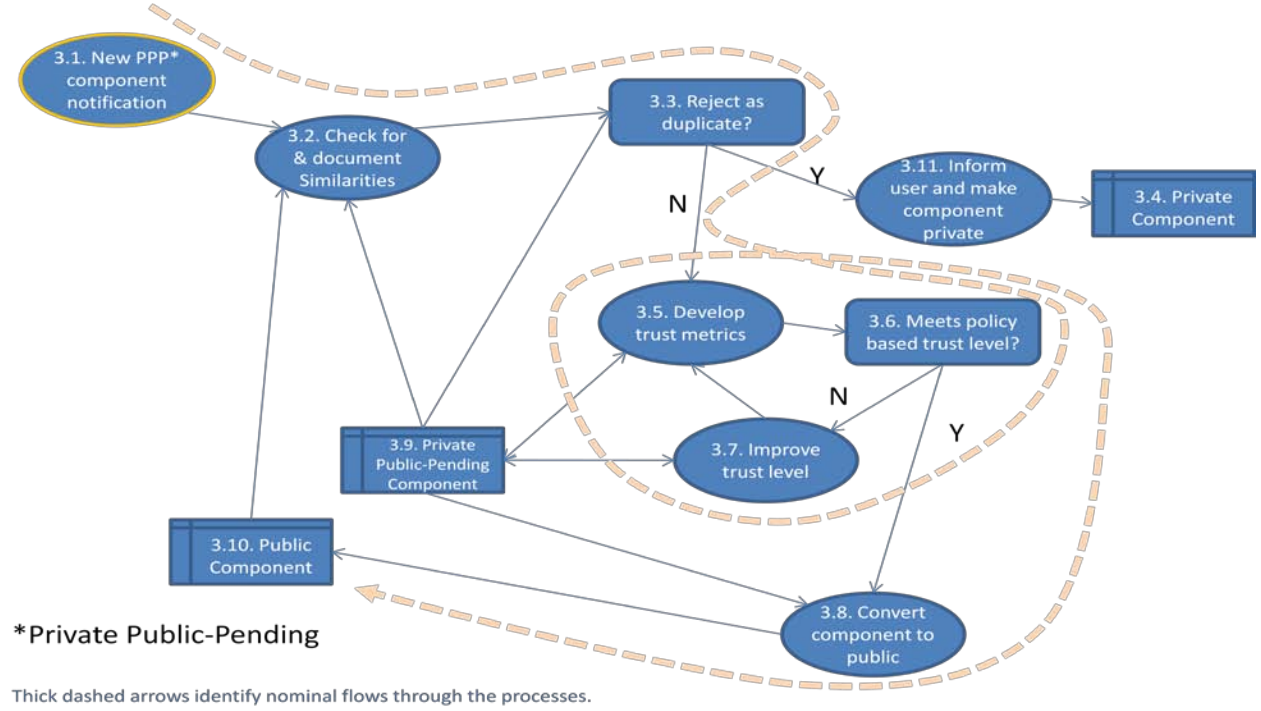


Figure A-4. Composition Life Cycle Manager View (Component) - 3.0

This diagram describes the processes by which the Composition Life Cycle Management organization for the CCOD User Environment converts a private component to a public component.

1. A notification is made that a new public pending component is available.
2. Examine the existing documentation for this component and determine if there are other components that perform similar functions. Document, as part of the component those links to similar components.
3. Determine whether duplicate.
4. Component is returned to private state if it was found to be a duplicate of an existing public component.
5. Develop the set of trust metrics for the component.
6. Verify that the trust metrics for this component meet or exceed the policy based trust level. This trust level can be changed and is likely to change over time. During peace time the trust level required may be higher than during a crisis. During a crisis the user community is willing to take on more risk, therefore the component may be deemed acceptable at a lower trust level.
7. If the component does not meet the acceptable level of trust, improve it and loop until the component is considered “trustworthy.”
8. Once the component is considered trustworthy, convert it to a public component and inform user population of new component.
9. Those components waiting for public release.
10. The directory of public components.
11. Inform component developer of rejection and return component to private state.

A.5 Composition Life Cycle Manager View (Composition) - 4.0

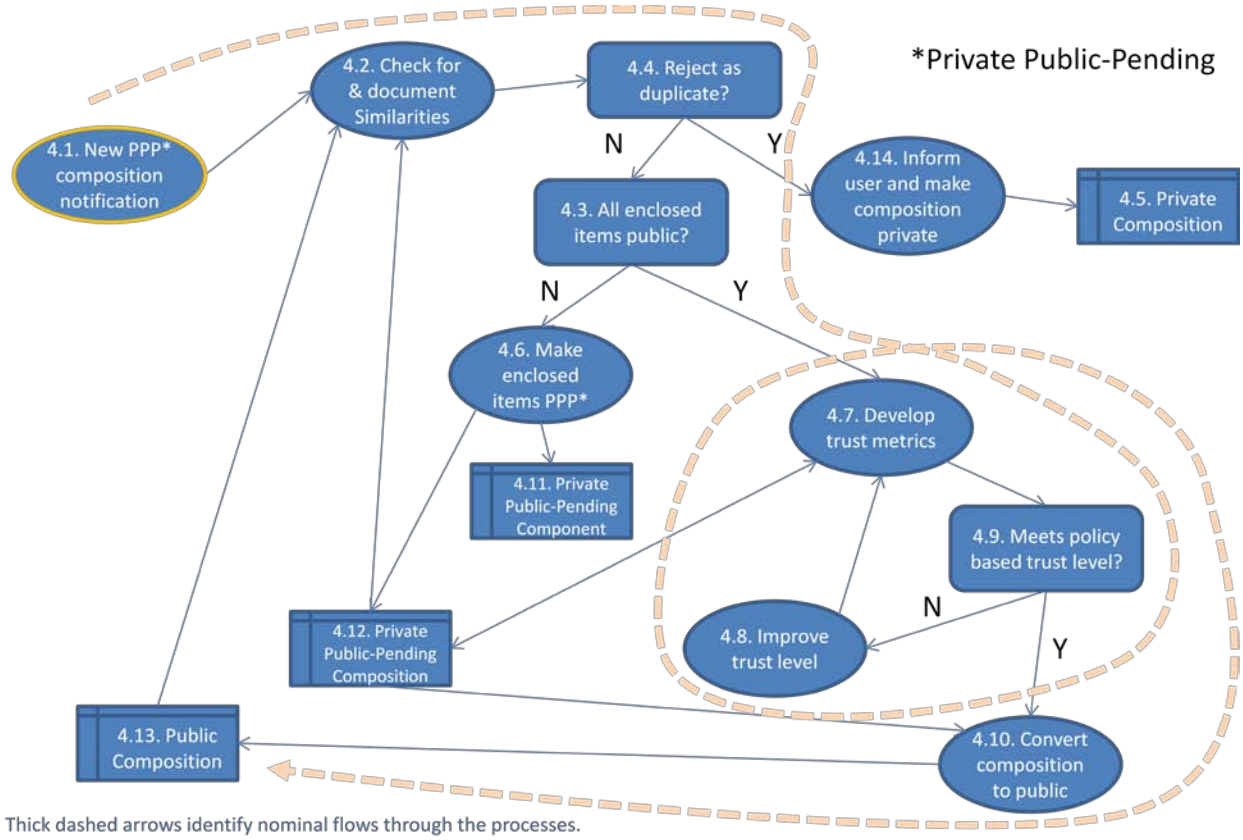


Figure A-5. Composition Life Cycle Manager View (Composition) - 4.0

This diagram describes the processes by which the Composition Life Cycle Management organization for the CCOD User Environment converts a private composition to a public composition.

1. A notification is made that a new public pending composition is available.
2. Examine the existing documentation for this composition and determine if there are other compositions that perform similar functions. Document, as part of the composition those links to similar components.
3. Determine if all subordinate components and compositions within this composition are public.
4. Determine if duplicate.
5. Composition is returned to private state if it was found to be a duplicate of an existing public composition.
6. Submit each private component or composition for public release.
7. Develop the set of trust metrics for the composition.
8. If the composition does not meet the acceptable level of trust, improve it and loop until the composition is considered “trustworthy.” “Improving” a composition could include adding test cases, replacing (or “improving”) existing subordinate components and compositions to make the larger composition more trustworthy.
9. Verify that the trust metrics for this composition meet or exceed the policy based trust level. This trust level can be changed and is likely to change over time. During peace time the trust level required may be higher than during a crisis. During a crisis the user community is

willing to take on more risk, therefore the composition may be deemed acceptable at a lower trust level.

10. Once the composition is considered trustworthy, convert it to a public composition and inform user population of new composition.
11. Those components waiting for public release.
12. Those compositions waiting for public release.
13. The directory of public compositions.
14. Inform composer of rejection and return composition to private state.

Appendix B The Trust Taxonomy

B.1 The Taxonomy

B.1.1 CCOD User Environment

CCOD User Environment trust metrics could affect the trustworthiness of components and compositions developed in that environment, lower quality CCOD User Environment could breed lower quality CCOD products. These values are used to support or modify metric values of those products developed in the CCOD User Environment.

The ability to gather many of these metrics will depend upon who controls the CCOD User Environment software. Commercial products may not provide access to the software for proper analysis.

B.1.1.1 Functionality

Express the ability of the CCOD User Environment to provide the required services when used under specified conditions.

(CQM-1 Page 2)

B.1.1.1.1 Accuracy

The capability of the CCOD User Environment to provide the right or agreed results or effects with the needed degree of precision.

(CQM-1 Pages 2, 4)

B.1.1.1.1.1 Correctness

Based on available documentation determine if the CCOD User Environment does what the documentation indicates. If there is no documentation then the correctness cannot be determined.

Scale 0-5.

- 0 – no documentation available
- 1 – less than 25% documented functions appear to operate correctly
- 2 – more than 25 % but less than 50% documented functions appear to operate correctly
- 3 – more than 50% but less than 75% documented functions appear to operate correctly
- 4 – more than 75% but less than 95% documented functions appear to operate correctly
- 5 – more than 95% documented functions appear to operate correctly.

(CQM-1 Page 4 and CQM-2 Page 5)

B.1.1.1.2 Suitability

The capability of the CCOD User Environment to provide an appropriate set of functions for specified tasks and user objectives. In a CCOD User Environment this includes the collection of fundamental components provided by the CCOD User Environment developer as a starting point for composers.

(CQM-1 Pages 2, 4)

B.1.1.1.2.1 Coverage

Physical inspection of existing documentation vs. actual functionality. Does the CCOD User Environment attempt to do everything the documentation says it is supposed to do?

Scale 0-5.

- 0 – no documentation available
- 1 – less than 25% documented functions appear to be implemented
- 2 – more than 25 % but less than 50% documented functions appear to be implemented
- 3 – more than 50% but less than 75% documented functions appear to be implemented
- 4 – more than 75% but less than 95% documented functions appear to be implemented
- 5 – more than 95% documented functions appear to be implemented.

(CQM-1 Page 4 and CQM-2 Page 6)

B.1.1.1.2.2 Completeness

Implemented functionalities/total of specified functionalities

Physical inspection of existing documentation vs. actual functionality. For every function that the CCOD User Environment attempts to perform, does it actually do the function?

Scale 0-5.

- 0 – no documentation available
- 1 – less than 25% documented functions actually implemented
- 2 – more than 25 % but less than 50% documented functions actually implemented
- 3 – more than 50% but less than 75% documented functions actually implemented
- 4 – more than 75% but less than 95% documented functions actually implemented
- 5 – more than 95% documented functions actually implemented.

(CQM-1 Page 4 and CQM-2 Page 6)

B.1.1.1.3 Functionality Compliance

The capability of the CCOD User Environment to adhere to standards, conventions or regulations in laws and similar prescriptions relating to functionality.

(CQM-1 Pages 2, 4)

B.1.1.1.3.1 Standardization

Implementation and documentation analysis

Claim to be conformant in documentation and perform simple code check or execution to verify.

Scale 0&5.

- 0 – non-conformant
- 5 – conformant

(CQM-1 Page 4 and CQM-2 Pages 6, 7)

B.1.1.1.3.2 Certification

Verify documentation

Inspect documentation for certifications.

Scale 0&5.

- 0 – not certified
- 5 – certified

(CQM-1 Page 4 and CQM-2 Pages 6, 7)

B.1.1.1.4 Internal Interoperability

The capability of the CCOD User Environment to assure the interoperability between compositions and components.

In general, the CCOD User Environment enables and manages the interoperability between compositions and components by assuring that their interfaces have been developed properly and execute properly at run-time. If the CCOD User Environment does not provide such an assurance, the onus is on the composer to assure interoperability. A lower value for this metric will result in more attention on the related metrics for components and compositions. If this metric is highly rated, then the corresponding metrics for components and compositions are less important.

(CQM-1 Pages 2, 4).

B.1.1.1.4.1 Data compatibility using widely accepted standards (XML, JSON, etc.)

Analysis of the data standard

By inspection determine if data standards are used.

Scale 0, 3, 5.

- 0 – indicates no data standard used and a specific data structure is defined (e.g., a proprietary messaging system)
- 3 – a well known data standard is used throughout with a restriction on structure or schema (e.g., XML with Cursor on Target the only schema allowed).
- 5 – a well known data standard is used throughout with no restriction on structure or schema (e.g., XML or JSON).

(CQM-1 Page 4 and CQM-2 Page 6, 7)

B.1.1.1.4.2 Interoperation complexity

Complexity level

A measure of the difficulty of making components and compositions interoperate. It is not sufficient to assure that the data is compatible.

Scale 0-5.

- 0 – it is nearly impossible to make components and composition interoperate
- 1 – making components and compositions interoperate can be done with extreme effort and time
- 2 – making components and compositions interoperate can be done with some effort and time
- 3 – making components and compositions interoperate can be done with reasonable effort and time
- 4 – making components and compositions interoperate can be done with little effort and time
- 5 – making components and compositions interoperate is trivial and takes very little time

B.1.1.1.5 External Interoperability

The ability for capabilities defined within the CCOD User Environment to function in another CCOD User Environment.

(CQM-1 Pages 2, 4)

B.1.1.1.5.1 Data compatibility using widely accepted standards (XML, JSON, etc.)

Analysis of the data standard

Scale 0, 3, 5.

- 0 – indicates no data standard used and a specific data structure is defined (e.g., a proprietary messaging system)
- 3 – a well known data standard is used throughout with a restriction on structure or schema (e.g., XML with Cursor on Target the only schema allowed).
- 5 – a well known data standard is used throughout with no restriction on structure or schema (e.g., XML or JSON).

(CQM-1 Page 4 and CQM-2 Page 6, 7)

B.1.1.1.5.2 Interoperation complexity

Complexity level

A measure of the difficulty of making the component interoperate. It is not sufficient to assure that the data is compatible.

Scale 0-5.

- 0 – it is nearly impossible to make external CCOD User Environments interoperate, special technical skills are required to succeed
- 1 – making external CCOD User Environments interoperate can be done with extreme effort and time, the external interoperability process is not consistent with that used for internal interoperability, some special technical skills are required
- 2 – making external CCOD User Environments interoperate can be done with some effort and time, the external interoperability process is not consistent with that used for internal interoperability, some special technical skills may be required
- 3 – making external CCOD User Environments interoperate can be done with reasonable effort and time, the external interoperability process is consistent with that used for internal interoperability, some special technical skills may be required
- 4 – making external CCOD User Environments interoperate can be done with little effort and time, the external interoperability process is consistent with that used for internal interoperability, no special technical skills are required
- 5 – making external CCOD User Environments interoperate is trivial and takes very little time, the external interoperability process is consistent with that used for internal interoperability, no special technical skills are required

B.1.1.1.6 Life-cycle management

Permits the life cycle of the components and composition to be managed. In this case provides necessary meta-data to manage transitions from public to private to deprecated states.

Mechanism Implemented

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

(SOA-RA-02 Section 5.3 Management Model)

B.1.1.1.7 Configuration management

A capability that permits the configuration of components and compositions to be managed. Also deals with dependencies between these elements. For stability rating, this includes noting the types of modifications being made to components/compositions, when and by whom.

Mechanism Implemented

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

(SOA-RA-02 Section 5.3 Management Model)

B.1.1.1.8 Policy management

The ability to manage those policies associated with the component and composition life cycle as well as any other management areas.

Mechanism Implemented

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

(SOA-RA-02 Section 5.3 Management Model)

B.1.1.2 Security

There are two major concepts in this security taxonomy. The first is derived from NIST 800-53 –"Recommended Security Controls for Federal Information Systems and Organizations". This document defines a collection of security functions (called "controls" in the document) that should be present in all government software applications. This NIST 800-53 groups the security functions into "families" and categorizes the families into classes. The families and classes are shown in the table below. Not all families are relevant to the CCOD Trust taxonomy. We have chosen to include 5 families and exclude all others. The justification for the exclusion of a family is also shown in the table. The five families included in the taxonomy are described in detail in the next five subsections.

Table B-1. Security Function Families & Classes

FAMILY	CLASS	Justification for exclusion
Access Control	Technical	
Awareness and Training	Operational	this is an organizational responsibility that will not be assessed
Audit and Accountability	Technical	
Security Assessment and Authorization	Management	this is an organizational responsibility that will not be assessed
Configuration Management	Operational	CM is included in the Functionality section of the CCOD User Environment
Contingency Planning	Operational	this is an organizational responsibility that will not be assessed
Identification and Authentication	Technical	
Incident Response	Operational	this is an organizational responsibility that will not be assessed
Maintenance	Operational	this is an organizational responsibility that will not be assessed
Media Protection	Operational	this is an organizational responsibility that will not be assessed
Physical and Environmental Protection	Operational	this is an organizational responsibility that will not be assessed
Planning	Management	this is an organizational responsibility that will not be assessed
Personnel Security	Operational	this is an organizational responsibility that will not be assessed
Risk Assessment	Management	this is an organizational responsibility that will not be assessed
System and Services Acquisition	Management	this is an organizational responsibility that will not be assessed
System and Communications Protection	Technical	
System and Information Integrity	Operational	
Program Management	Management	this is an organizational responsibility that will not be assessed

NIST 800-53 further breaks down families into the individual "controls" which are then prioritized. Our taxonomy converts a family into a sub characteristic of security. Controls are then converted into attributes. Measures are derived from the controls' descriptions. The description for each sub characteristic below will present a similar table to that above indicating which controls have been included and a justification for exclusion. For those controls that have been included the definitions have been slightly changed to focus on the assessable quality that the control represents.

The next major concept deals with "secureness." Secureness is about the structure and implementation of software that make up the CCOD User Environment and those components available within the CCOD User Environment. The measures we use for secureness are derived from the CWE/SANS (Common Weakness Enumeration/SysAdmin, Audit, Network, Security) top 25 Most Dangerous Software Errors. The 2010 CWE/SANS Top 25 Most Dangerous Software Errors is a community developed list of the most widespread and critical programming errors that can lead to serious software vulnerabilities. The absence of these weaknesses results in software possessing the attribute of "secureness."

Subsection 6, below, describes the Top 25 Software Errors.

User developed components can be evaluated for the presence or absence of these weaknesses. The CCOD User Environment may also be evaluated provided the source code is made available for inspection.

B.1.1.2.1 Access Control

The table below lists the controls included in the taxonomy and justifications for exclusion. Details for each control follow.

Table B-2. Access Control Controls

Number	Control Name	Priority	Justification for exclusion
AC-1	Access Control Policy and Procedures	P1	this is an organizational responsibility that will not be assessed
AC-2	Account Management	P1	
AC-3	Access Enforcement	P1	
AC-4	Information Flow Enforcement	P1	
AC-5	Separation of Duties	P1	this is an organizational responsibility that will not be assessed
AC-6	Least Privilege	P1	this is an organizational responsibility that will not be assessed
AC-7	Unsuccessful Login Attempts Limitation	P2	
AC-8	System Use Notification	P1	
AC-9	Previous Logon Notification	P0	
AC-10	Concurrent Session Control	P2	
AC-11	Session Lock Mechanism	P3	
AC-12	Session Termination		Withdrawn in original document
AC-13	Supervision and Review—Access Control		Withdrawn in original document
AC-14	Permitted Actions without Identification or Authentication	P1	this is an organizational responsibility that will not be assessed
AC-15	Automated Marking		Withdrawn in original document
AC-16	Security Attribute Management	P0	
AC-17	Remote Access	P1	Not applicable – this function is left to the computer environment in which users operate
AC-18	Wireless Access	P1	Not applicable – this function is left to the computer environment in which users operate
AC-19	Access Control for Mobile Devices	P1	Not applicable – this function is left to the computer environment in which users operate
AC-20	Use of External Information Systems	P1	this is an organizational responsibility that will not be assessed
AC-21	User-Based Collaboration and Information Sharing	P0	Not applicable – this function is left to the computer environment in which users operate
AC-22	Publicly Accessible Content	P2	this is an organizational responsibility that will not be assessed

B.1.1.2.1.1 AC-2 Account Management

The CCOD User Environment manages accounts, to include:

B.1.1.2.1.1.1 AC-2a Account type identification

A mechanism is in place that identifies account types

i.e., individual, group, system, application, guest/anonymous, and temporary

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.1.1.2 AC-2b Authorized user identification

A mechanism is in place that identifies authorized users of the CCOD User Environment and specifies access privileges

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.1.1.3 AC-2c Account management mechanism

A mechanism is in place that establishes, activates, modifies, disables, and removes accounts;

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.1.1.4 AC-2d Guest and temporary accounts

A mechanism is in place that specifically authorizes and monitors the use of guest/anonymous and temporary accounts

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.1.1.5 AC-2e Access control mechanism

A mechanism is in place that grants access to the system based on: (i) a valid access authorization; (ii) intended system usage; and (iii) other attributes as required by the organization or associated missions/business functions

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.1.1.6 AC-2f Inactive accounts

A mechanism is in place that automatically disables inactive accounts after an assigned period of time.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.1.1.7 AC-2g Account auditing

A mechanism is in place that automatically audits account creation, modification, disabling, and termination actions and notifies, as required, appropriate individuals.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.1.2 AC-3 Access Enforcement

The CCOD User Environment enforces approved authorizations for logical access to the system in accordance with applicable policy.

B.1.1.2.1.2.1 AC-3a Role Based Access Control

A mechanism is in place provides a Role Based Access Control mechanism that ensures access rights are grouped by role name, and access to resources is restricted to users who have been authorized to assume the associated role

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.1.3 AC-4 Information Flow Enforcement

The CCOD User Environment enforces approved authorizations for controlling the flow of information within the CCOD User Environment and between interconnected systems in accordance with applicable policy. All measures associated with transferring information between different security domains are outside the scope of the CCOD User Environment.

B.1.1.2.1.3.1 AC-4a Information flow enforcement mechanism

A mechanism is in place that enforces information flow control using explicit security attributes on information, source, and destination objects as a basis for flow control decisions.

Information flow enforcement mechanisms compare security attributes on all information (data content and data structure), source and destination objects, and respond appropriately (e.g., block, quarantine, alert administrator) when the mechanisms encounter information flows not explicitly allowed by the information flow policy. Information flow enforcement using explicit security attributes can be used, for example, to control the release of certain types of information.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.1.3.2 AC-4b Protected processing domains

A mechanism is in place that enforces information flow control using protected processing domains (e.g., domain type-enforcement) as a basis for flow control decisions.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.1.3.3 AC-4c Dynamic flow control

A mechanism is in place that enforces dynamic information flow control based on policy that allows or disallows information flows based on changing conditions or operational considerations.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.1.3.4 AC-4d Encrypted data bypass

A mechanism is in place that prevents encrypted data from bypassing content-checking mechanisms.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.1.3.5 AC-4e Embedded data types

A mechanism is in place that enforces limitations on the embedding of data types within other data types.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.1.3.6 AC-4f Metadata flow control

A mechanism is in place that enforces information flow control on metadata.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.1.3.7 AC-4g Security Policy filters

A mechanism is in place that enforces information flow control using organization-defined security policy filters as a basis for flow control decisions.

Organization-defined security policy filters include, for example, dirty word filters, file type checking filters, structured data filters, unstructured data filters, metadata content filters, and hidden content filters. Structured data permits the interpretation of its content by virtue of atomic elements that are understandable by an application and indivisible. Unstructured data refers to masses of (usually) digital information that does not have a data structure or has a data structure that is not easily readable by a machine. Unstructured data consists of two basic categories: (i) bitmap objects that are inherently non language-based (i.e., image, video, or audio files); and (ii)

textual objects that are based on a written or printed language (i.e., commercial off-the-shelf word processing documents, spreadsheets, or emails).

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.1.3.8 AC-4h Human review mechanism

A mechanism is in place that enforces the use of human review for organization-defined security policy filters when the system is not capable of making an information flow control decision.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.1.3.9 AC-4i Policy filter control

A mechanism is in place that provides the capability for a privileged administrator to enable/disable organization-defined security policy filters.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.1.3.10 AC-4j Interconnected systems

A mechanism is in place enforces security policies regarding information on interconnected systems.

Transferring information between interconnected information systems of differing security policies introduces risk that such transfers violate one or more policies. While security policy violations may not be absolutely prohibited, policy guidance from information owners/stewards is implemented at the policy enforcement point between the interconnected systems. Specific architectural solutions are mandated, when required, to reduce the potential for undiscovered vulnerabilities. Architectural solutions include, for example: (i) prohibiting information transfers between interconnected systems (i.e. implementing access only, one way transfer mechanisms); (ii) employing hardware mechanisms to enforce unitary information flow directions; and (iii) implementing fully tested, re-grading mechanisms to reassign security attributes and associated security labels.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.1.3.11 AC-4k Security attributes

A mechanism is in place that uniquely identifies and authenticates source and destination domains for information transfer; binds security attributes to information to facilitate information flow policy enforcement; and tracks problems associated with the security attribute binding and information transfer.

Attribution is a critical component of a security concept of operations. The ability to identify source and destination points for information flowing in an information system, allows forensic reconstruction of events when required, and increases policy compliance by attributing policy violations to specific organizations/individuals. Means to enforce this enhancement include ensuring that the information system resolution labels distinguish between information systems and organizations, and between specific system components or individuals involved in preparing, sending, receiving, or disseminating information.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.1.4 AC-7 Unsuccessful Login Attempts Limitation

A mechanism is in place that enforces a limit of a defined number of consecutive invalid login attempts by a user during a defined time period and automatically locks the account for a defined time period or locks the account until released by an administrator or delays next login prompt according to a defined delay algorithm when the maximum number of unsuccessful attempts is exceeded. The control applies regardless of whether the login occurs via a local or network connection.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.1.5 AC-8 System Use Notification

System use notification messages can be implemented in the form of warning banners displayed when individuals log in to the information system. System use notification is intended only for information system access that includes an interactive login interface with a human user and is not intended to require notification when an interactive interface does not exist.

B.1.1.2.1.5.1 AC-8a Notification message mechanism

A mechanism is in place that displays an approved system use notification message or banner before granting access to the system that provides privacy and security notices consistent with applicable federal laws, Executive Orders, directives, policies, regulations, standards, and guidance.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.1.5.2 AC-8b Retention mechanism

A mechanism is in place that retains the notification message or banner on the screen until users take explicit actions to log on to or further access the information system.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.1.5.3 AC-8c Public access mechanism

A mechanism is in place that for public access: (i) displays the system use information when appropriate, before granting further access; (ii) displays references, if any, to monitoring, recording, or auditing that are consistent with privacy accommodations for such systems that generally prohibit those activities; and (iii) includes in the notice given to public users of the information system, a description of the authorized uses of the system.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.1.6 AC-9 Previous Logon Notification

This control is intended to cover both traditional logons to information systems and general accesses to information systems that occur in other types of architectural configurations (e.g., service oriented architectures).

B.1.1.2.1.6.1 AC-9a Last logon notification

A mechanism is in place that notifies the user, upon successful logon (access), of the date and time of the last logon (access).

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.1.6.2 AC-9b Unsuccessful logon Notification

A mechanism is in place that notifies the user, upon successful logon/access, of the number of unsuccessful logon/access attempts since the last successful logon/access.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.1.6.3 AC-9c Attempts over time

A mechanism is in place that notifies the user of the number of logon events (successes and failures) during a defined time period.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.1.6.4 AC-9d Account change notification

A mechanism is in place that notifies the user of security related changes to the user's account during a defined time period.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.1.7 AC-10 Concurrent Session Control

A mechanism is in place that limits the number of concurrent sessions for each system account to a defined number.

The organization may define the maximum number of concurrent sessions for an information system account globally, by account type, by account, or a combination. This control addresses concurrent sessions for a given account and does not address concurrent sessions by a single user via multiple accounts.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.1.8 AC-11 Session Lock Mechanism

A mechanism is in place that prevents further access to the CCOD User Environment by initiating a session lock after a defined period of inactivity or upon receiving a request from a user; and retains the session lock until the user reestablishes access using established identification and authentication procedures.

A session lock is a temporary action taken when a user stops work and moves away from the immediate physical vicinity of the information system but does not want to log out because of the temporary nature of the absence. The session lock is implemented at the point where session activity can be determined. This is typically at the operating system-level, but may be at the application-level. A session lock is not a substitute for logging out of the information system, for example, if the organization requires users to log out at the end of the workday.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.1.9 AC-16 Security Attribute Management

B.1.1.2.1.9.1 AC-16a Binding security attributes

A mechanism is in place that supports and maintains the binding of organization defined security attributes to information in storage, in process, and in transmission.

Security attributes are abstractions representing the basic properties or characteristics of an entity (e.g., subjects and objects) with respect to safeguarding information. These attributes are typically associated with internal data structures (e.g., records, buffers, files) within the information system and are used to enable the implementation of access control and flow control policies, reflect special dissemination, handling or distribution instructions, or support other

aspects of the information security policy. The term security label is often used to associate a set of security attributes with a specific information object as part of the data structure or that object (e.g., user access privileges, nationality, and affiliation as contractor). .

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.1.9.2 AC-16b Dynamic reconfiguration security attributes

A mechanism is in place that dynamically reconfigures security attributes in accordance with an identified security policy as information is created and combined.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.1.9.3 AC-16c Changes to security attributes

A mechanism is in place that allows authorized entities to change security attributes.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.1.9.4 AC-16d Assurance for security attributes

A mechanism is in place that maintains the binding of security attributes to information with sufficient assurance that the information--attribute association can be used as the basis for automated policy actions.

Examples of automated policy actions include automated access control decisions (e.g., Mandatory Access Control decisions), or decisions to release (or not release) information (e.g., information flows via cross domain systems).

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.1.9.5 AC-16e Association of security attributes

A mechanism is in place that allows authorized users to associate security attributes with information.

The support provided by the information system can vary from prompting users to select security attributes to be associated with specific information objects, to ensuring that the combination of attributes selected is valid.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.1.9.6 AC-16f Display security attributes

A mechanism is in place that displays security attributes in human-readable form on each object output from the system to system output devices to identify the organization specified set of special dissemination, handling, or distribution instructions using organization specified human readable, standard naming conventions.

Objects output from the information system include, for example, pages, screens, or equivalent. Output devices include, for example, printers and video displays on computer terminals, monitors, screens on notebook/laptop computers and personal digital assistants.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.2 Audit and Accountability

The table below lists the controls included in the taxonomy and justifications for exclusion. Details for each control follow.

Table B-3. Audit and Accountability Controls

Number	Control Name	Priority	Justification for exclusion
AU-1	Audit and Accountability Policy and Procedures	P1	this is an organizational responsibility that will not be assessed
AU-2	Auditable Events	P1	this is an organizational responsibility that will not be assessed
AU-3	Content of Audit Records	P1	
AU-4	Audit Storage Capacity	P1	this is an organizational responsibility that will not be assessed
AU-5	Response to Audit Processing Failures	P1	
AU-6	Audit Review, Analysis, and Reporting	P1	
AU-7	Audit Reduction and Report Generation	P2	
AU-8	Audit Record Time Stamp	P1	
AU-9	Protection of Audit Information	P1	
AU-10	Non-repudiation	P1	
AU-11	Audit Record Retention	P3	this is an organizational responsibility that will not be assessed
AU-12	Audit Generation	P1	
AU-13	Monitoring for Information Disclosure	P0	this is an organizational responsibility that will not be assessed
AU-14	Session Audit	P0	

B.1.1.2.2.1 AU-3 Content of Audit Records

The CCOD User Environment produces audit records that contain sufficient information to, at a minimum, establish what type of event occurred, when (date and time) the event occurred, where the event occurred, the source of the event, the outcome (success or failure) of the event, and the identity of any user/subject associated with the event.

Audit record content that may be necessary to satisfy the requirement of this control, includes, for example, time stamps, source and destination addresses, user/process identifiers, event descriptions, success/fail indications, filenames involved, and access control or flow control rules invoked.

A mechanism is in place that includes organization defined information in the audit records for audit events identified by type, location, or subject.

An example of information that the organization may require in audit records is full-text recording of privileged commands or the individual identities of group account users.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.2.2 AU-5 Response to Audit Processing Failures

The CCOD User Environment alerts designated organizational officials in the event of an audit processing failure and takes one of the following actions: shut down the CCOD User Environment, overwrite oldest audit records, or stop generating audit records.

Audit processing failures include, for example, software/hardware errors, failures in the audit capturing mechanisms, and audit storage capacity being reached or exceeded.

B.1.1.2.2.2.1 AU-5a Audit Storage Failure

A mechanism is in place that provides a warning when allocated audit record storage volume reaches a defined percentage of maximum audit record storage capacity.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.2.2.2 AU-5b Audit Failure Alert

A mechanism is in place that provides a real-time alert when audit failure events occur.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.2.2.3 AU-5c Audit traffic control

A mechanism is in place that enforces configurable traffic volume thresholds representing auditing capacity for network traffic and rejects or delays network traffic above those thresholds.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.2.2.4 AU-5d Audit failure response

A mechanism is in place that responds to audit failure by either restricting access to the CCOD User Environment or providing an alternate auditing mechanism.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.2.3 AU-6 Audit Review, Analysis, and Reporting

The CCOD User Environment supports audit review, analysis and reporting

B.1.1.2.2.3.1 AU-6a Audit integration

A mechanism is in place that integrates audit review, analysis, and reporting processes to support organizational processes for investigation and response to suspicious activities.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.2.3.2 AU-6b Audit record centralization

A mechanism is in place that centralizes the review and analysis of audit records from multiple components within the system.

An example of an automated mechanism for centralized review and analysis is a Security Information Management (SIM) product.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.2.4 AU-7 Audit Reduction and Report Generation

The CCOD User Environment provides an audit reduction and report generation capability.

An audit reduction and report generation capability provides support for near real-time audit review, analysis, and reporting requirements and after-the fact investigations of security incidents. Audit reduction and reporting tools do not alter original audit records.

A mechanism is in place that provides the capability to automatically process audit records for events of interest based on selectable event criteria.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.2.5 AU-8 Audit Record Time Stamp

A mechanism is in place that uses internal system clocks to generate time stamps for audit records.

Time stamps generated by the CCOD User Environment include both date and time. The time may be expressed in Coordinated Universal Time (UTC), a modern continuation of Greenwich Mean Time (GMT), or local time with an offset from UTC.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.2.6 AU-9 Protection of Audit Information

Audit information includes all information (e.g., audit records, audit settings, and audit reports) needed to successfully audit information system activity.

B.1.1.2.2.6.1 AU-9a Protection mechanism

A mechanism is in place that protects audit information and audit tools from unauthorized access, modification, and deletion.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.2.6.2 AU-9b Write once media

A mechanism is in place that produces audit records on hardware-enforced, write-once media.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.2.6.3 AU-9c Backup mechanism

A mechanism is in place that backs up audit records at a defined frequency onto a different system or media than the system being audited.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.2.6.4 AU-9d Encryption mechanism

A mechanism is in place that uses cryptographic mechanisms to protect the integrity of audit information and audit tools.

An example of a cryptographic mechanism for the protection of integrity is the computation and application of a cryptographic-signed hash using asymmetric cryptography, protecting the confidentiality of the key used to generate the hash, and using the public key to verify the hash information.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.2.7 AU-10 Non-repudiation

The CCOD User Environment protects against an individual falsely denying having performed a particular action.

Examples of particular actions taken by individuals include creating information, sending a message, approving information (e.g., indicating concurrence or signing a contract), and receiving a message. Non-repudiation protects individuals against later claims by an author of not having authored a particular document, a sender of not having transmitted a message, a receiver of not having received a message, or a signatory of not having signed a document. Non-repudiation services can be used to determine if information originated from an individual, or if an individual took specific actions (e.g., sending an email, signing a contract, approving a

procurement request) or received specific information. Non-repudiation services are obtained by employing various techniques or mechanisms (e.g., digital signatures, digital message receipts).

B.1.1.2.2.7.1 AU-10a Producer identity mechanism

A mechanism is in place that associates the identity of the information producer with the information.

This control enhancement supports audit requirements that provide appropriate organizational officials the means to identify who produced specific information in the event of an information transfer. The nature and strength of the binding between the information producer and the information are determined and approved by the appropriate organizational officials based on the security categorization of the information and relevant risk factors.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.2.7.2 AU-10b Producer identity binding validation

A mechanism is in place that validates the binding of the information producer's identity to the information.

This control enhancement is intended to mitigate the risk that information is modified between production and review. The validation of bindings can be achieved, for example, by the use of cryptographic checksums.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.2.7.3 AU-10c Reviewer identity mechanism

A mechanism is in place that maintains reviewer/releaser identity and credentials within the established chain of custody for all information reviewed or released.

If the reviewer is a human or if the review function is automated but separate from the release/transfer function, the information system associates the identity of the reviewer of the information to be released with the information and the information label. In the case of human reviews, this control enhancement provides appropriate organizational officials the means to identify who reviewed and released the information. In the case of automated reviews, this control enhancement helps ensure that only approved review functions are employed.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.2.7.4 AU-10d Reviewer identity binding validation

A mechanism is in place that validates the binding of the reviewer's identity to the information at the transfer/release point prior to release/transfer from one security domain to another security domain.

This control enhancement is intended to mitigate the risk that information is modified between review and transfer/release.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.2.8 AU-12 Audit Generation

B.1.1.2.2.8.1 AU-12a Content control mechanism

A mechanism is in place that provides audit record generation capability for the list of auditable events, allows designated organizational personnel to select which auditable events are to be audited and generates audit records for the list of audited events defined in

Audits records can be generated from various components within the CCOD User Environment. The list of audited events is the set of events for which audits are to be generated. This set of events is typically a subset of the list of all events for which the system is capable of generating audit records (i.e., auditable events).

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.2.8.2 AU-12b Time correlation mechanism

A mechanism is in place that compiles audit records into an CCOD User Environment-wide (logical or physical) audit trail that is time correlated.

The audit trail is time-correlated if the time stamp in the individual audit records can be reliably related to the time stamp in other audit records to achieve a time ordering of the records within the organization-defined tolerance.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.2.8.3 AU-12c Standardized format mechanism

A mechanism is in place that produces a system-wide (logical or physical) audit trail composed of audit records in a standardized format.

Audit information normalized to a common standard promotes interoperability and exchange of such information between dissimilar devices and information systems. This facilitates an audit system that produces event information that can be more readily analyzed and correlated. System log records and audit records compliant with the Common Event Expression (CEE) are examples of standard formats for audit records. If individual logging mechanisms within the CCOD User Environment do not conform to a standardized format, the CCOD User Environment may convert individual audit records into a standardized format when compiling the CCOD User Environment-wide audit trail.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.2.9 AU-14 Session Audit

A mechanism is in place that provides the capability to capture/record and log all content related to a user session; and remotely view all content related to an established user session in real time.

Session auditing activities are developed, integrated, and used in consultation with legal counsel in accordance with applicable federal laws, Executive Orders, directives, policies, or regulations.

A mechanism is in place that initiates session audits at system start-up.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.3 Identification and Authentication

The table below lists the controls included in the taxonomy and justifications for exclusion. Details for each control follow.

Table B-4. Identification and Authentication Controls

Number	Control Name	Priority	Justification for exclusion
IA-1	Identification and Authentication Policy and Procedures	P1	this is an organizational responsibility that will not be assessed
IA-2	Identification and Authentication Mechanisms	P1	
IA-3	Device Identification and Authentication	P1	This control is outside the scope of the CCOD User Environment
IA-4	Identifier Management	P1	this is an organizational responsibility that will not be assessed
IA-5	Authenticator Management	P1	
IA-6	Authenticator Feedback Mechanism	P1	
IA-7	Cryptographic Module Authentication	P1	This control is outside the scope of the CCOD User Environment
IA-8	Identification and Authentication (Non-Organizational Users)	P1	This control is outside the scope of the CCOD User Environment

B.1.1.2.3.1 IA-2 Identification and Authentication Mechanisms

The CCOD User Environment uniquely identifies and authenticates organizational users (or processes acting on behalf of organizational users).

Organizational users include organizational employees or individuals the organization deems to have equivalent status of employees (e.g., contractors, guest researchers, individuals from allied nations). Users are uniquely identified and authenticated for all accesses other than those accesses explicitly identified and documented by the organization in AC-14. Unique identification of individuals in group accounts (e.g., shared privilege accounts) may need to be considered for detailed accountability of activity. Authentication of user identities is accomplished through the use of passwords, tokens, biometrics, or in the case of multifactor authentication, some combination thereof. Access to organizational information systems is defined as either local or network. Local access is any access to an organizational information

system by a user (or process acting on behalf of a user) where such access is obtained by direct connection without the use of a network. Network access is any access to an organizational information system by a user (or process acting on behalf of a user) where such access is obtained through a network connection. Remote access is a type of network access which involves communication through an external network (e.g., the Internet). Internal networks include local area networks, wide area networks, and virtual private networks that are under the control of the organization. For a virtual private network (VPN), the VPN is considered an internal network if the organization establishes the VPN connection between organization-controlled endpoints in a manner that does not require the organization to depend on any external networks across which the VPN transits to protect the confidentiality and integrity of information transmitted. Identification and authentication requirements for information system access by other than organizational users are described in IA-8. The identification and authentication requirements in this control are satisfied by complying with Homeland Security Presidential Directive 12 consistent with organization-specific implementation plans provided to OMB.

B.1.1.2.3.1.1 IA-2a Privileged network access

A mechanism is in place that authenticates network access to privileged accounts.

Scale 0,3,4,5.

- 0 – the capability is not present
- 3 – single factor authentication is present
- 4 – multifactor authentication is present
- 5 – multifactor authentication is present where one of the factors is provided by a device separate from the information system being accessed

B.1.1.2.3.1.2 IA-2b Non-privileged network access

A mechanism is in place that authenticates network access to non-privileged accounts.

Scale 0,3,4,5.

- 0 – the capability is not present
- 3 – single factor authentication is present
- 4 – multifactor authentication is present
- 5 – multifactor authentication is present where one of the factors is provided by a device separate from the information system being accessed

B.1.1.2.3.1.3 IA-2c Privileged local access

A mechanism is in place that uses multifactor authentication for local access to privileged accounts.

Scale 0,3,4,5.

- 0 – the capability is not present
- 3 – single factor authentication is present
- 4 – multifactor authentication is present
- 5 – multifactor authentication is present where one of the factors is provided by a device separate from the information system being accessed

B.1.1.2.3.1.4 IA-2d Non-privileged local access

A mechanism is in place that uses multifactor authentication for local access to non-privileged accounts.

Scale 0,3,4,5.

- 0 – the capability is not present
- 3 – single factor authentication is present
- 4 – multifactor authentication is present
- 5 – multifactor authentication is present where one of the factors is provided by a device separate from the information system being accessed

B.1.1.2.3.1.5 IA-2e Privileged network access replay-resistance

A mechanism is in place that uses replay-resistant authentication mechanisms for network access to privileged accounts.

An authentication process resists replay attacks if it is impractical to achieve a successful authentication by recording and replaying a previous authentication message. Techniques used to address this include protocols that use nonces or challenges (e.g., TLS), and time synchronous or challenge-response one-time authenticators.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.3.1.6 IA-2f Non-privileged network access replay-resistance

A mechanism is in place that uses replay-resistant authentication mechanisms for network access to non-privileged accounts.

An authentication process resists replay attacks if it is impractical to achieve a successful authentication by recording and replaying a previous authentication message. Techniques used to address this include protocols that use nonces or challenges (e.g., TLS), and time synchronous or challenge-response one-time authenticators.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.3.2 IA-5 Authenticator Management

The CCOD User Environment follows organizational rules with regard to the structure of authenticators.

A mechanism is in place that, for password-based authentication: (a) Enforces minimum password complexity of case sensitivity, number of characters, mix of upper-case letters, lower-case letters, numbers, and special characters, including minimum requirements for each type; (b) Enforces at least a defined number of changed characters when new passwords are created; (c) Encrypts passwords in storage and in transmission; (d) Enforces password minimum and maximum lifetime restrictions; and (e) Prohibits password reuse for a defined number of generations.

This control enhancement is intended primarily for CCOD User Environments where passwords are used as a single factor to authenticate users, or in a similar manner along with one or more additional authenticators.

A mechanism is in place that, for PKI-based authentication: (a) Validates certificates by constructing a certification path with status information to an accepted trust anchor; (b) Enforces authorized access to the corresponding private key; and (c) Maps the authenticated identity to the user account.

Status information for certification paths includes, for example, certificate revocation lists or online certificate status protocol responses.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.3.3 IA-6 Authenticator Feedback Mechanism

A mechanism is in place that obscures feedback of authentication information during the authentication process to protect the information from possible exploitation/use by unauthorized individuals.

The feedback from the CCOD User Environment does not provide information that would allow an unauthorized user to compromise the authentication mechanism. Displaying asterisks when a user types in a password, is an example of obscuring feedback of authentication information.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.4 System and Communications Protection

The table below lists the controls included in the taxonomy and justifications for exclusion. Details for each control follow.

Table B-5. System and Communications Controls

Number	Control Name	Priority	Justification for exclusion
SC-1	System and Communications Protection Policy and Procedures	P1	this is an organizational responsibility that will not be assessed
SC-2	Application Partitioning	P1	
SC-3	Security Function Isolation	P1	Outside scope of CCOD User Environment
SC-4	Information in Shared Resources	P1	
SC-5	Denial of Service Protection	P1	
SC-6	Resource Priority	P0	
SC-7	Boundary Protection	P1	
SC-8	Transmission Integrity	P1	Outside scope of CCOD User Environment
SC-9	Transmission Confidentiality	P1	Outside scope of CCOD User Environment
SC-10	Network Disconnect	P2	
SC-11	Trusted Path	P0	Outside scope of CCOD User Environment
SC-12	Cryptographic Key Establishment and Management	P1	this is an organizational responsibility that will not be assessed
SC-13	Use of Cryptography	P1	
SC-14	Public Access Protections	P1	Outside scope of CCOD User Environment
SC-15	Collaborative Computing Devices	P1	Outside scope of CCOD User Environment
SC-16	Transmission of Security Attributes	P0	Outside scope of CCOD User Environment
SC-17	Public Key Infrastructure Certificates	P1	this is an organizational responsibility that will not be assessed
SC-18	Mobile Code	P1	Outside scope of CCOD User Environment
SC-19	Voice Over Internet Protocol	P1	Outside scope of CCOD User Environment
SC-20	Secure Name /Address Resolution Service (Authoritative Source)	P1	Outside scope of CCOD User Environment
SC-21	Secure Name /Address Resolution Service (Recursive or Caching Resolver)	P1	Outside scope of CCOD User Environment
SC-22	Architecture and Provisioning for Name/Address Resolution Service	P1	Outside scope of CCOD User Environment
SC-23	Session Authenticity	P1	Outside scope of CCOD User Environment
SC-24	Fail in Known State	P1	
SC-25	Thin Nodes	P0	Outside scope of CCOD User Environment
SC-26	Honeypots	P0	Outside scope of CCOD User Environment
SC-27	Operating System-Independence	P0	
SC-28	Protection of Information at Rest	P1	Outside scope of CCOD User Environment
SC-29	Heterogeneity	P0	Outside scope of CCOD User Environment
SC-30	Virtualization Techniques	P0	Outside scope of CCOD User Environment
SC-31	Covert Channel Analysis	P0	this is an organizational responsibility that will not be assessed
SC-32	Information System Partitioning	P1	this is an organizational responsibility that will not be assessed
SC-33	Transmission Preparation Integrity	P0	Outside scope of CCOD User Environment
SC-34	Non-Modifiable Executable Programs	P0	Outside scope of CCOD User Environment

B.1.1.2.4.1 SC-2 Application Partitioning

The CCOD User Environment separates user functionality (including user interface services) from environment management functionality.

CCOD User Environment management functionality includes, for example, functions necessary to administer databases, network components, or servers, and typically requires privileged user access.

An example of this type of separation is observed in web administrative interfaces that use separate authentication methods for users of any other information system resources. This may include isolating the administrative interface on a different domain and with additional access controls.

A mechanism is in place that prevents the presentation of CCOD User Environment management-related functionality at an interface for general (i.e., non-privileged) users.

The intent of this control enhancement is to ensure that administration options are not available to general users (including prohibiting the use of the grey-out option commonly used to eliminate accessibility to such information). For example, administration options are not presented until the user has appropriately established a session with administrator privileges.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.4.2 SC-4 Information in Shared Resources

A mechanism is in place that prevents unauthorized and unintended information transfer via shared system resources.

The purpose of this control is to prevent information, including encrypted representations of information, produced by the actions of a prior user/role (or the actions of a process acting on behalf of a prior user/role) from being available to any current user/role (or current process) that obtains access to a shared system resource (e.g., registers, main memory, secondary storage) after that resource has been released back to the information system. Control of information in shared resources is also referred to as object reuse. This control does not address: (i) information remanence which refers to residual representation of data that has been in some way nominally erased or removed; (ii) covert channels where shared resources are manipulated to achieve a violation of information flow restrictions; or (iii) components in the information system for which there is only a single user/role.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.4.3 SC-5 Denial of Service Protection

The CCOD User Environment protects against or limits the effects of denial of service attacks

A variety of technologies exist to limit, or in some cases, eliminate the effects of denial of service attacks.

B.1.1.2.4.3.1 SC-5a Restriction mechanism

A mechanism is in place that restricts the ability of users to launch denial of service attacks against the CCOD User Environment.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.4.3.2 SC-5b Resource management mechanism

A mechanism is in place that manages excess capacity, bandwidth, or other redundancy to limit the effects of information flooding types of denial of service attacks.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.4.4 SC-6 Resource Priority

A mechanism is in place that limits the use of resources by priority.

Priority protection helps prevent a lower-priority process from delaying or interfering with the information system servicing any higher-priority process. This control does not apply to components in the information system for which there is only a single user/role.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.4.5 SC-7 Boundary Protection

The CCOD User Environment connects to external networks or information systems only through managed interfaces consisting of boundary protection devices arranged in accordance with an organizational security architecture.

Restricting external web traffic only to organizational web servers within managed interfaces and prohibiting external traffic that appears to be spoofing an internal address as the source are examples of restricting and prohibiting communications. Managed interfaces employing boundary protection devices include, for example, proxies, gateways, routers, firewalls, guards, or encrypted tunnels arranged in an effective security architecture (e.g., routers protecting firewalls and application gateways residing on a protected subnetwork commonly referred to as a demilitarized zone or DMZ). The organization considers the intrinsically shared nature of commercial telecommunications services in the implementation of security controls associated with the use of such services. Commercial telecommunications services are commonly based on network components and consolidated management systems shared by all attached commercial customers, and may include third-party provided access lines and other service elements. Consequently, such interconnecting transmission services may represent sources of increased risk despite contract security provisions. Therefore, when this situation occurs, the organization either implements appropriate compensating security controls or explicitly accepts the additional risk.

B.1.1.2.4.5.1 SC-7a Incoming flow protection

A mechanism is in place that prevents public access into the CCOD User Environment except as appropriately mediated by managed interfaces employing boundary protection devices.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.4.5.2 SC-7b Outgoing flow protection

A mechanism is in place that routes internal communications traffic to external networks through authenticated proxy servers within the managed interfaces of boundary protection devices.

External networks are networks outside the control of the organization. Proxy servers support logging individual Transmission Control Protocol (TCP) sessions and blocking specific Uniform Resource Locators (URLs), domain names, and Internet Protocol (IP) addresses. Proxy servers are also configurable with organization defined lists of authorized and unauthorized websites.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.4.5.3 SC-7c Incoming validation mechanism

A mechanism is in place that checks incoming communications to ensure that the communications are coming from an authorized source and routed to an authorized destination.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.4.5.4 SC-7d Privileged flow mechanism

A mechanism is in place that routes all networked, privileged accesses through a dedicated, managed interface for purposes of access control and auditing.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.4.6 SC-10 Network Disconnect

A mechanism is in place that terminates the network connection associated with a communications session at the end of the session or after an assigned period of inactivity.

This control applies to both internal and external networks. Terminating network connections associated with communications sessions include, for example, de-allocating associated TCP/IP address/port pairs at the operating-system level, or de-allocating networking assignments at the application level if multiple application sessions are using a single, operating system-level network connection. The time period of inactivity may, as the organization deems necessary, be a set of time periods by type of network access or for specific accesses.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.4.7 SC-13 Use of Cryptography

A mechanism is in place that implements required cryptographic protections using cryptographic modules that comply with applicable federal laws, Executive Orders, directives, policies, regulations, standards, and guidance.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.4.8 SC-24 Fail in Known State

A mechanism is in place that fails to a known-state for a defined set of failures preserving CCOD User Environment state information in failure.

Failure in a known state can address safety or security in accordance with the mission/business needs of the organization. Failure in a known secure state helps prevent a loss of confidentiality, integrity, or availability in the event of a failure of the information system or a component of the system. Failure in a known safe state helps prevent systems from failing to a state that may cause injury to individuals or destruction to property. Preserving information system state information facilitates system restart and return to the operational mode of the organization with less disruption of mission/business processes.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.4.9 SC-27 Operating System-Independence

The CCOD User Environment can run under multiple operating systems.

CCOD User Environments that run under multiple operating systems promote portability and reconstitution on different platform architectures, increasing the availability for critical functionality within an organization.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.5 System and Information Integrity

The table below lists the controls included in the taxonomy and justifications for exclusion. Details for each control follow.

Table B-6. System and Information Integrity Controls

Number	Control Name	Priority	Justification for exclusion
SI-1	System and Information Integrity Policy and Procedures	P1	this is an organizational responsibility that will not be assessed
SI-2	Flaw Remediation	P1	this is an organizational responsibility that will not be assessed
SI-3	Malicious Code Protection	P1	This control is outside the scope of the CCOD User Environment
SI-4	Information System Monitoring	P1	
SI-5	Security Alerts, Advisories, and Directives	P1	this is an organizational responsibility that will not be assessed
SI-6	Security Functionality Verification	P1	
SI-7	Software and Information Integrity	P1	
SI-8	Spam Protection	P1	This control is outside the scope of the CCOD User Environment
SI-9	Information Input Restrictions	P2	This control is outside the scope of the CCOD User Environment
SI-10	Information Input Validation	P1	This control is outside the scope of the CCOD User Environment
SI-11	Error Handling	P2	
SI-12	Information Output Handling and Retention	P2	this is an organizational responsibility that will not be assessed
SI-13	Predictable Failure Prevention	P0	this is an organizational responsibility that will not be assessed

B.1.1.2.5.1 SI-4 Information System Monitoring

B.1.1.2.5.1.1 SI-4a Communication monitoring

A mechanism is in place that monitors inbound and outbound communications for unusual or unauthorized activities or conditions.

Unusual/unauthorized activities or conditions include, for example, internal traffic that indicates the presence of malicious code within an information system or propagating among system components, the unauthorized export of information, or signaling to an external information system. Evidence of malicious code is used to identify potentially compromised information systems or information system components.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.5.1.2 SI-4b Real-time alert mechanism

A mechanism is in place that provides near real-time alerts when there are indications of compromise or potential compromise.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.5.1.3 SI-4c Circumvention mechanism

A mechanism is in place that prevents non-privileged users from circumventing intrusion detection and prevention capabilities.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.5.1.4 SI-4d Suspicious events mechanism

A mechanism is in place that notifies appropriate personnel of suspicious events and takes actions to terminate suspicious events.

The least-disruptive actions may include initiating a request for human response.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.5.2 SI-6 Security Functionality Verification

A mechanism is in place that verifies the correct operation of security functions and takes action when anomalies are found.

Actions can include notifying appropriate personnel, restricting access or shutting down the CCOD User Environment.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.5.3 SI-7 Software and Information Integrity

The CCOD User Environment detects unauthorized changes to software and information.

The CCOD User Environment employs integrity verification mechanisms to look for evidence of information tampering, errors, and omissions.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.5.4 SI-11 Error Handling

A mechanism is in place that identifies potential security-relevant error conditions; generates error messages that provide information necessary for corrective actions without revealing sensitive or potentially harmful information in error logs and administrative messages that could be exploited by adversaries; and reveals error messages only to authorized personnel.

The extent to which the information system is able to identify and handle error should be guided by organizational policy and operational requirements. Sensitive information includes, for example, account numbers, social security numbers, and credit card numbers.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

B.1.1.2.6 Run-time Secureness

Run-time secureness is based on the 2010 CWE/SANS Top 25 Most Dangerous Software Errors. This is a list of the most widespread and critical programming errors that can lead to serious software vulnerabilities. They are often easy to find, and easy to exploit. They are dangerous because they will frequently allow attackers to completely take over the software, steal data, or prevent the software from working at all.

B.1.1.2.6.1 Insecure Component Interaction

These weaknesses are related to insecure ways in which data is sent and received between separate components, modules, programs, processes, threads, or systems.

B.1.1.2.6.1.1 Cross-site Scripting

Cross-site scripting (XSS) vulnerabilities occur when:

Untrusted data enters a web application, typically from a web request.

- The web application dynamically generates a web page that contains this untrusted data.
- During page generation, the application does not prevent the data from containing content that is executable by a web browser, such as JavaScript, HTML tags, HTML attributes, mouse events, Flash, ActiveX, etc.
- A victim visits the generated web page through a web browser, which contains malicious script that was injected using the untrusted data.
- Since the script comes from a web page that was sent by the web server, the victim's web browser executes the malicious script in the context of the web server's domain.
- This effectively violates the intention of the web browser's same-origin policy, which states that scripts in one domain should not be able to access resources or run code in a different domain.

There are three main kinds of XSS:

Type 1: Reflected XSS (or Non-Persistent)

> The server reads data directly from the HTTP request and reflects it back in the HTTP response. Reflected XSS exploits occur when an attacker causes a victim to supply dangerous content to a vulnerable web application, which is then reflected back to the victim and executed by the web browser. The most common mechanism for delivering malicious content is to include it as a parameter in a URL that is posted publicly or e-mailed directly to the victim. URLs constructed in this manner constitute the core of many phishing schemes, whereby an attacker convinces a victim to visit a URL that refers to a vulnerable site. After the site reflects the attacker's content back to the victim, the content is executed by the victim's browser.

Type 2: Stored XSS (or Persistent)

> The application stores dangerous data in a database, message forum, visitor log, or other trusted data store. At a later time, the dangerous data is subsequently read back into the application and included in dynamic content. From an attacker's perspective, the optimal

place to inject malicious content is in an area that is displayed to either many users or particularly interesting users. Interesting users typically have elevated privileges in the application or interact with sensitive data that is valuable to the attacker. If one of these users executes malicious content, the attacker may be able to perform privileged operations on behalf of the user or gain access to sensitive data belonging to the user. For example, the attacker might inject XSS into a log message, which might not be handled properly when an administrator views the logs.

Type 0: DOM-Based XSS

In DOM-based XSS, the client performs the injection of XSS into the page; in the other types, the server performs the injection. DOM-based XSS generally involves server-controlled, trusted script that is sent to the client, such as Javascript that performs sanity checks on a form before the user submits it. If the server-supplied script processes user-supplied data and then injects it back into the web page (such as with dynamic HTML), then DOM-based XSS is possible.
Once the malicious script is injected, the attacker can perform a variety of malicious activities. The attacker could transfer private information, such as cookies that may include session information, from the victim's machine to the attacker. The attacker could send malicious requests to a web site on behalf of the victim, which could be especially dangerous to the site if the victim has administrator privileges to manage that site. Phishing attacks could be used to emulate trusted web sites and trick the victim into entering a password, allowing the attacker to compromise the victim's account on that web site. Finally, the script could exploit vulnerability in the web browser itself possibly taking over the victim's machine, sometimes referred to as "drive-by hacking."

In many cases, the attack can be launched without the victim even being aware of it. Even with careful users, attackers frequently use a variety of methods to encode the malicious portion of the attack, such as URL encoding or Unicode, so the request looks less suspicious.

CWE-79 rank [1]

Scale 0&5.

- 0 – the weakness is present
- 5 – the weakness is not present

B.1.1.2.6.1.2 SQL Injection

Without sufficient removal or quoting of SQL syntax in user-controllable inputs, the generated SQL query can cause those inputs to be interpreted as SQL instead of ordinary user data. This can be used to alter query logic to bypass security checks, or to insert additional statements that modify the back-end database, possibly including execution of system commands.

SQL injection has become a common issue with database-driven web sites. The flaw is easily detected, and easily exploited, and as such, any site or software package with even a minimal user base is likely to be subject to an attempted attack of this kind. This flaw depends on the fact that SQL makes no real distinction between the control and data planes.

CWE-89 rank [2]

Scale 0&5.

- 0 – the weakness is present
- 5 – the weakness is not present

B.1.1.2.6.1.3 Cross-Site Request Forgery (CSRF)

When a web server is designed to receive a request from a client without any mechanism for verifying that it was intentionally sent, then it might be possible for an attacker to trick a client into making an unintentional request to the web server which will be treated as an authentic request. This can be done via a URL, image load, XMLHttpRequest, etc. and can result in data disclosure or unintended code execution.

CWE-352 rank [4]

Scale 0&5.

- 0 – the weakness is present
- 5 – the weakness is not present

B.1.1.2.6.1.4 Unrestricted Upload of Dangerous Files Type

The software allows the attacker to upload or transfer files of dangerous types that can be automatically processed within the CCOD User Environment.

Arbitrary code execution is possible if an uploaded file is interpreted and executed as code by the recipient. This is especially true for .asp and .php extensions uploaded to web servers because these file types are often treated as automatically executable, even when file system permissions do not specify execution. For example, in Unix environments, programs typically cannot run unless the execute bit is set, but PHP programs may be executed by the web server without directly invoking them on the operating system.

CWE-434 rank [8]

Scale 0&5.

- 0 – the weakness is present
- 5 – the weakness is not present

B.1.1.2.6.1.5 OS Command Injection

This could allow attackers to execute unexpected, dangerous commands directly on the operating system. This weakness can lead to vulnerability in environments in which the attacker does not have direct access to the operating system, such as in web applications. Alternately, if the weakness occurs in a privileged program, it could allow the attacker to specify commands that normally would not be accessible, or to call alternate commands with privileges that the attacker does not have. The problem is exacerbated if the compromised process fails to follow the principle of least privilege, because the attacker-controlled commands may run with special system privileges that increase the amount of damage.

There are at least two subtypes of OS command injection:

- The application intends to execute a single, fixed program that is under its own control. It intends to use externally-supplied inputs as arguments to that program. For example, the program might use system("nslookup [HOSTNAME]") to run nslookup and allow the user to supply a HOSTNAME, which is used as an argument. Attackers cannot prevent nslookup from executing. However, if the program does not remove command separators from the HOSTNAME argument, attackers could place the separators into the arguments, which allows them to execute their own program after nslookup has finished executing.

- The application accepts an input that it uses to fully select which program to run, as well as which commands to use. The application simply redirects this entire command to the operating system. For example, the program might use "exec([COMMAND])" to execute the [COMMAND] that was supplied by the user. If the COMMAND is under attacker control, then the attacker can execute arbitrary commands or programs. If the command is being executed using functions like exec() and CreateProcess(), the attacker might not be able to combine multiple commands together in the same line.

From a weakness standpoint, these variants represent distinct programmer errors. In the first variant, the programmer clearly intends that input from untrusted parties will be part of the arguments in the command to be executed. In the second variant, the programmer does not intend for the command to be accessible to any untrusted party, but the programmer probably has not accounted for alternate ways in which malicious attackers can provide input.

CWE-78 rank [9]

Scale 0&5.

- 0 – the weakness is present
- 5 – the weakness is not present

B.1.1.2.6.1.6 Error Message Information Exposure through an

The sensitive information may be valuable information on its own (such as a password), or it may be useful for launching other, more deadly attacks. If an attack fails, an attacker may use error information provided by the server to launch another more focused attack. For example, an attempt to exploit a path traversal weakness (CWE-22) might yield the full pathname of the installed application. In turn, this could be used to select the proper number of ".." sequences to navigate to the targeted file. An attack using SQL injection (CWE-89) might not initially succeed, but an error message could reveal the malformed query, which would expose query logic and possibly even passwords or other sensitive information used within the query.

CWE-209 rank [17]

Scale 0&5.

- 0 – the weakness is present
- 5 – the weakness is not present

B.1.1.2.6.1.7 URL Redirection to Untrusted Site

An http parameter may contain a URL value and could cause the web application to redirect the request to the specified URL. By modifying the URL value to a malicious site, an attacker may successfully launch a phishing scam and steal user credentials. Because the server name in the modified link is identical to the original site, phishing attempts have a more trustworthy appearance.

CWE-601 rank [23]

Scale 0&5.

- 0 – the weakness is present
- 5 – the weakness is not present

B.1.1.2.6.1.8 Race Condition

This can have security implications when the expected synchronization is in security-critical code, such as recording whether a user is authenticated, or modifying important state information that should not be influenced by an outsider.

CWE-362 rank [25]

Scale 0&5.

- 0 – the weakness is present
- 5 – the weakness is not present

B.1.1.2.6.2 Risky Resource Management

The weaknesses in this category are related to ways in which software does not properly manage the creation, usage, transfer, or destruction of important system resources.

B.1.1.2.6.2.1 Classic Buffer Overflow

A buffer overflow condition exists when a program attempts to put more data in a buffer than it can hold, or when a program attempts to put data in a memory area outside of the boundaries of a buffer. The simplest type of error, and the most common cause of buffer overflows, is the "classic" case in which the program copies the buffer without checking its length at all. Other variants exist, but the existence of a classic overflow strongly suggests that the programmer is not considering even the most basic of security protections.

CWE-120 rank [3]

Scale 0&5.

- 0 – the weakness is present
- 5 – the weakness is not present

B.1.1.2.6.2.2 IPath Traversal

Many file operations are intended to take place within a restricted directory. By using special elements such as ".." and "/" separators, attackers can escape outside of the restricted location to access files or directories that are elsewhere on the system. One of the most common special elements is the "../" sequence, which in most modern operating systems is interpreted as the parent directory of the current location. This is referred to as relative path traversal. Path traversal also covers the use of absolute pathnames such as "/usr/local/bin", which may also be useful in accessing unexpected files. This is referred to as absolute path traversal.

In many programming languages, the injection of a null byte (the 0 or NUL) may allow an attacker to truncate a generated filename to widen the scope of attack. For example, the software may add ".txt" to any pathname, thus limiting the attacker to text files, but a null injection may effectively remove this restriction.

CWE-22 rank [7]

Scale 0&5.

- 0 – the weakness is present
- 5 – the weakness is not present

B.1.1.2.6.2.3 Buffer Access with Incorrect Length Value

The software uses a sequential operation to read or write a buffer, but it uses an incorrect length value that causes it to access memory that is outside of the bounds of the buffer.

CWE-805 rank [12]

Scale 0&5.

- 0 – the weakness is present
- 5 – the weakness is not present

B.1.1.2.6.2.4 Improper Check for Exceptional Conditions

The programmer may assume that certain events or conditions will never occur or do not need to be worried about, such as low memory conditions, lack of access to resources due to restrictive permissions, or misbehaving clients or components. However, attackers may intentionally trigger these unusual conditions which thus violating the programmer's assumptions, possibly introducing instability, incorrect behavior, or a vulnerability.

Note that this entry is not exclusively about the use of exceptions and exception handling, which are mechanisms for both checking and handling unusual or unexpected conditions.

CWE-754 rank [13]

Scale 0&5.

- 0 – the weakness is present
- 5 – the weakness is not present

B.1.1.2.6.2.5 PHP File Inclusion

In certain versions and configurations of PHP, this can allow an attacker to specify a URL to a remote location from which the software will obtain the code to execute. In other cases in association with path traversal, the attacker can specify a local file that may contain executable statements that can be parsed by PHP.

CWE-98 rank [14]

Scale 0&5.

- 0 – the weakness is present
- 5 – the weakness is not present

B.1.1.2.6.2.6 Improper Validation of Array Index

The product uses untrusted input when calculating or using an array index, but the product does not validate or incorrectly validates the index to ensure the index references a valid position within the array.

CWE-129 rank [15]

Scale 0&5.

- 0 – the weakness is present
- 5 – the weakness is not present

B.1.1.2.6.2.7 Integer Overflow or Wraparound

An integer overflow or wraparound occurs when an integer value is incremented to a value that is too large to store in the associated representation. When this occurs, the value may wrap to become a very small or negative number. While this may be intended behavior in circumstances that rely on wrapping, it can have security consequences if the wrap is unexpected. This is especially the case if the integer overflow can be triggered using user-supplied inputs. This becomes security-critical when the result is used to control looping, make a security decision, or determine the offset or size in behaviors such as memory allocation, copying, concatenation, etc.

CWE-190 rank [16]

Scale 0&5.

- 0 – the weakness is present
- 5 – the weakness is not present

B.1.1.2.6.2.8 Incorrect Calculation of Buffer Size

The software does not correctly calculate the size to be used when allocating a buffer, which could lead to a buffer overflow.

CWE-131 rank [18]

Scale 0&5.

- 0 – the weakness is present
- 5 – the weakness is not present

B.1.1.2.6.2.9 Download of Code without Integrity Check

The product downloads source code or an executable from a remote location and executes the code without sufficiently verifying the origin and integrity of the code.

CWE-494 rank [20]

Scale 0&5.

- 0 – the weakness is present
- 5 – the weakness is not present

B.1.1.2.6.2.10 Allocation of Resources without Limits or Throttling

The software allocates a reusable resource or group of resources on behalf of an actor without imposing any restrictions on how many resources can be allocated, in violation of the intended security policy for that actor.

CWE-770 rank [22]

Scale 0&5.

- 0 – the weakness is present
- 5 – the weakness is not present

B.1.1.2.6.3 Prevention of Porous Defenses

The weaknesses in this category are related to defensive techniques that are often misused, abused, or just plain ignored.

B.1.1.2.6.3.1 Improper Access Control (Authorization)

When access control checks are not applied consistently - or not at all - users are able to access data or perform actions that they should not be allowed to perform. This can lead to a wide range of problems, including information leaks, denial of service, and arbitrary code execution.

CWE-285 rank [5]

Scale 0&5.

- 0 – the weakness is present
- 5 – the weakness is not present

B.1.1.2.6.3.2 Reliance on Untrusted Inputs in a Security Decision

Developers may assume that inputs such as cookies, environment variables, and hidden form fields cannot be modified. However, an attacker could change these inputs using customized clients or other attacks. This change might not be detected. When security decisions such as authentication and authorization are made based on the values of these inputs, attackers can bypass the security of the software.

Without sufficient encryption, integrity checking, or other mechanism, any input that originates from an outsider cannot be trusted.

CWE-807 rank [6]

Scale 0&5.

- 0 – the weakness is present
- 5 – the weakness is not present

B.1.1.2.6.3.3 Missing Encryption of Sensitive Data

The lack of proper data encryption passes up the guarantees of confidentiality, integrity, and accountability that properly implemented encryption conveys

CWE-311 rank [10]

Scale 0&5.

- 0 – the weakness is present
- 5 – the weakness is not present

B.1.1.2.6.3.4 Use of Hard-coded Credentials

A hard-coded password typically leads to a significant authentication failure that can be difficult for the system administrator to detect. Once detected, it can be difficult to fix, so the administrator may be forced into disabling the product entirely. There are two main variations:

Inbound: the software contains an authentication mechanism that checks for a hard-coded password.

Outbound: the software connects to another system or component, and it contains hard-coded password for connecting to that component.

In the Inbound variant, a default administration account is created, and a simple password is hard-coded into the product and associated with that account. This hard-coded password is the same for each installation of the product, and it usually cannot be changed or disabled by system administrators without manually modifying the program, or otherwise patching the software. If

the password is ever discovered or published (a common occurrence on the Internet), then anybody with knowledge of this password can access the product. Finally, since all installations of the software will have the same password, even across different organizations, this enables massive attacks such as worms to take place.

The Outbound variant applies to front-end systems that authenticate with a back-end service. The back-end service may require a fixed password which can be easily discovered. The programmer may simply hard-code those back-end credentials into the front-end software. Any user of that program may be able to extract the password. Client-side systems with hard-coded passwords pose even more of a threat, since the extraction of a password from a binary is usually very simple.

CWE-798 rank [11]

Scale 0&5.

- 0 – the weakness is present
- 5 – the weakness is not present

B.1.1.2.6.3.5 Missing Authentication for Critical Function

The software does not perform any authentication for functionality that requires a provable user identity or consumes a significant amount of resources.

CWE-306 rank [19]

Scale 0&5.

- 0 – the weakness is present
- 5 – the weakness is not present

B.1.1.2.6.3.6 Incorrect Permission Assignment for Critical Resource

When a resource is given a permissions setting that provides access to a wider range of actors than required, it could lead to the disclosure of sensitive information or the modification of that resource by unintended parties. This is especially dangerous when the resource is related to program configuration, execution or sensitive user data.

CWE-732 rank [21]

Scale 0&5.

- 0 – the weakness is present
- 5 – the weakness is not present

B.1.1.2.6.3.7 Use of a Insecure Cryptographic Algorithm

The use of a broken or risky cryptographic algorithm is an unnecessary risk that may result in the disclosure of sensitive information.

CWE-327 rank [24]

Scale 0&5.

- 0 – the weakness is present
- 5 – the weakness is not present

B.1.1.3 Reliability

Express the ability of the CCOD User Environment to maintain a specified level of performance, when used under specified conditions.

(CQM-1 Page 2)

B.1.1.3.1 Fault Tolerance

The capability of the CCOD User Environment to maintain a specified level of performance in cases of component faults or of infringement of its specified interface.

(CQM-1 Pages 2, 4)

B.1.1.3.1.1 Mechanism available

Mechanism identification

Graceful failure described in documentation and observed.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

(CQM-1 Page 4 and CQM-2 Page 5)

B.1.1.3.1.2 Mechanism efficiency

Rough ratio of functions with graceful failure / total functionality.

Scale 0-5.

- 0 – no fault tolerance capability observed
- 1 – very few functions appear to be fault tolerant
- 2 – few functions appear to be fault tolerant
- 3 – some functions appear to be fault tolerant
- 4 – most functions appear to be fault tolerant
- 5 – all functions appear to be fault tolerant

(CQM-1 Page 4 and CQM-2 Page 5)

B.1.1.3.2 Recoverability

The capability of the CCOD User Environment to re-establish a specified level of performance and recover the data directly affected in the case of a failure.

(CQM-1 Pages 2, 4)

B.1.1.3.2.1 Mechanism available

Mechanism Implemented

Recovery described in documentation

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

(CQM-1 Page 4 and CQM-2 Page 5)

B.1.1.3.2.2 Mechanism efficiency

Rough ratio of functions with recovery mechanisms/ total functions and level of recovery by inspection (completeness and processes).

Need to understand completeness of recoverability capability.

Scale 0-5.

- 0 – no recovery mechanism observed
- 1 – very few functions appear to have recovery mechanisms
- 2 – few functions appear to have recovery mechanisms
- 3 – some functions appear to have recovery mechanisms
- 4 – most functions appear to have recovery mechanisms
- 5 – all functions appear to have recovery mechanisms

B.1.1.3.3 Maturity

The capability of the CCOD User Environment to avoid failure as a result of faults in the software.

Does not have same weight as in composition and component.

(CQM-1 Pages 2, 4)

B.1.1.3.3.1 Volatility

Analysis of the time between versions

Scale 0-5.

- 0 – critical bugs (Cat 1 & 2) are on the books for longer than 2 years
- 1 – critical bugs (Cat 1 & 2) are on the books for longer than 1 year but less than 2 years
- 2 – critical bugs (Cat 1 & 2) are on the books for longer than 6 months but less than 1 year. Use this rating if no version/bug information is available
- 3 – critical bugs (Cat 1 & 2) are on the books for longer than 3 months but less than 6 months
- 4 – critical bugs (Cat 1 & 2) are on the books for longer than 1 month but less than 3 months
- 5 – critical bugs (Cat 1 & 2) are on the books for less than 1 month

(CQM-1 Pages 4, 5 and CQM-2 Pages 6, 7)

B.1.1.3.3.2 Failure removal

of bugs fixed in a version

Scale 0-5.

- 0 – an unlikely score, all outstanding CCOD User Environment license holder submitted bugs are ignored and never fixed.
- 1 – the number of outstanding CCOD User Environment license holder submitted bugs is increasing exponentially with each version
- 2 – the number of outstanding CCOD User Environment license holder submitted bugs is increasing linearly with each version
- 3 – the number of outstanding CCOD User Environment license holder submitted bugs is remaining constant with each version
- 4 – the number of outstanding CCOD User Environment license holder submitted bugs is decreasing with each version.
- 5 – an unlikely score, all outstanding CCOD User Environment license holder submitted bugs fixed with each version.

(CQM-1 Page 5 and CQM-2 Pages 6, 7)

B.1.1.3.4 Reliability Compliance

The capability of the CCOD User Environment to adhere to standards, conventions or regulations relating to reliability.

Verify claims of compliance. If claimed and verified then CCOD User Environment rates better than one without claims.

Scale 0&5.

- 0 – non-compliant
- 5 – compliant

(ISO/IEC 9126-1 Section 6.2.4)

B.1.1.3.5 Event monitoring

Reporting of events and faults during the execution of compositions and capabilities. This should include identifying the sources of any errant execution.

Mechanism Implemented

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

(SOA-RA-02 Section 5.3 Management Model)

B.1.1.4 Usability

The ability to be used by someone other than the originator when constructing a component, composing a composition or using capability.

These characteristics can be measured for up to three user interfaces provided by the CCOD User Environment (end user, composer, and component developer).

(CQM-1 Page 2)

B.1.1.4.1 Configurability

The ability to specify parameters that aid in the use of the CCOD User Environment within a particular context. The ability of the CCOD User Environment be configurable (e.g. through a XML file or a text file, the number of parameters, etc.)

(CQM-1 Pages 2, 4)

B.1.1.4.1.1 Effort to Configure

By inspection metric (not end user feedback)

Time spent to configure correctly

Scale 0-5.

- 0 – the CCOD User Environment is non-configurable
- 1 – the CCOD User Environment is very hard to configure, the configuration parameters are difficult to understand, or there are insufficient configuration parameters.
- 2 – the CCOD User Environment is hard to configure, many of the configuration parameters are difficult to understand, or many configuration parameters are missing.
- 3 – the CCOD User Environment is somewhat hard to configure, some configuration parameters are difficult to understand, or several configuration parameters are missing.
- 4 – the CCOD User Environment is fairly easy to configure, most configuration parameters are easy to understand, and most configuration parameters are present.
- 5 – the CCOD User Environment is very easily configured, all configuration parameters are easy to understand, the scope of configuration is complete

(CQM-1 Page 4 and CQM-2 Page-5)

B.1.1.4.2 Understandability

The capability of the CCOD User Environment to enable the user to understand how it can be used for developing components or composing compositions and capabilities.

(CQM-1 Pages 2, 5)

B.1.1.4.2.1 Documentation quality

Documentation analysis.

Scale 0-5.

- 0 – the CCOD User Environment is not documented
- 1 – the documentation is minimal
- 2 – the documentation is mostly complete
- 3 – the documentation is complete but complex
- 4 – the documentation is complete but somewhat complex
- 5 – documentation is complete and easy to understand

(CQM-1 Page 5 and CQM-2 Pages 6, 7)

B.1.1.4.3 Learnability

The capability of the CCOD User Environment to enable the user to learn its application.

(CQM-1 Pages 3, 5)

B.1.1.4.3.1 Quality of training materials

Breadth, depth and understandability of training materials

Scale 0-5.

- 0 – there are no training materials
- 1 – the training material is minimal
- 2 – the training material is mostly complete
- 3 – the training material is complete but complex
- 4 – the training material is complete but somewhat complex
- 5 – training material is complete and easy to understand

B.1.1.4.3.2 Common development language used

Components are developed using a common, readily available, programming language.

Scale 0-5.

- 0 – Does not provide a component development environment
- 1 – Uses its own language
- 2 – Uses a little know programming language
- 3 – Uses a recognizable programming language
- 4 – Uses a common language available on several hardware platforms
- 5 – Uses a very common language available on most hardware platforms

B.1.1.4.4 Operability

The capability of the CCOD User Environment to enable the user to operate and control it.

(CQM-1 Pages 3, 5)

B.1.1.4.4.1 Complexity level

Rate the difficulty to execute capabilities

Rate the difficulty to compose

Rate the component develop environment

Steps needed

Scale 0-5.

- 0 – extreme complexity, even with extensive trained and extreme technical skills, few users can operate the environment
- 1 – moderate complexity, users must have extensive training and have further technical skills to operate the environment
- 2 – medium complexity, users must be trained and have further technical skills to operate the environment
- 3 – medium complexity, users must be trained, no additional technical skills are required, most can operate the environment
- 4 – medium complexity, no training is required, no additional technical skills are required, users can learn by doing
- 5 – low complexity, easy to operate, anyone can operate the environment without training or other technical skills

(CQM-1 Page 5 and CQM-2 Pages 6, 7)

B.1.1.4.5 Discoverability of components and compositions

The CCOD User Environment must provide an easy to use mechanism by the composer for discovering the components and compositions that are available for use when developing new compositions.

Mechanism available

Sufficient descriptive data maintained

Ease and robustness of search (reduce false positives and avoid misses)

Scale 0-5.

- 0 – components and compositions are not cataloged and not discoverable by any mechanism
- 1 – components and compositions can be discovered via manual processes (e.g., looking in computer directories)
- 2 – A catalog of components and compositions is present, no search mechanism, categorization or tagging is provided
- 3 – A catalog of components and compositions is present, categorization is provided but no search mechanism, or tagging is provided
- 4 – A catalog of components and compositions is present, a search mechanism and categorization is provided but no tagging is provided
- 5 – A catalog of components and compositions is present, a search mechanism, categorization and tagging is provided

(ISO/IEC 9126-1 Section 6.3.4) with the notion of discoverability.

B.1.1.4.6 Discoverability of capabilities

The CCOD User Environment must provide an easy to use mechanism by the end user for discovering available capabilities.

Mechanism available

Sufficient descriptive data maintained

Ease and robustness of search (reduce false positives and avoid misses)

Scale 0-5.

- 0 – capabilities are not cataloged and not discoverable by any mechanism
- 1 – capabilities can be discovered via manual processes (e.g., looking in computer directories)
- 2 – A catalog of capabilities is present, no search mechanism, categorization or tagging is provided
- 3 – A catalog of capabilities is present, categorization is provided but no search mechanism, or tagging is provided
- 4 – A catalog of capabilities is present, a search mechanism and categorization is provided but no tagging is provided
- 5 – A catalog of capabilities is present, a search mechanism, categorization and tagging is provided

(ISO/IEC 9126-1 Section 6.3.4) with the notion of discoverability.

B.1.1.4.7 Usability Compliance

The capability of the component to adhere to standards, conventions, style guides or regulations relating to usability.

Verify claims of compliance. If claimed and verified then CCOD User Environment rates better than one without claims.

Scale 0&5.

- 0 – non-compliant
- 5 – compliant

(ISO/IEC 9126-1 Section 6.3.5)

B.1.1.5 Efficiency

Express the ability of a CCOD User Environment to provide appropriate performance, relative to the amount of resources used.

Measure the environmental overhead for CCOD User Environment

(CQM-1 Page 2)

B.1.1.5.1 Time Behavior

The capability of the CCOD User Environment to provide appropriate response and processing times and throughput rates when performing its function, under stated conditions.

(CQM-1 Pages 3, 4)

B.1.1.5.1.1 Response time

Time taken between a set of invocations. A measure of the overhead caused by the CCOD User Environment to invoke a capability.

Scale 0, 3, 5.

- 0 – unacceptable overhead observed
- 3 – reasonable overhead observed
- 5 – no perceptible overhead observed

(CQM-1 Page 4 and CQM-2 Page-5)

B.1.1.5.1.2 Throughput

Amount of outputs produced with success / period of time

Scale 0, 3, 5.

- 0 – unacceptable throughput observed
- 3 – reasonable throughput observed
- 5 – high throughput observed

(CQM-1 Page 4 and CQM-2 Page-5)

B.1.1.5.2 Resource Utilization

The capability of the CCOD User Environment to use appropriate amounts and types of resources when performing its function under stated conditions.

(CQM-1 Pages 3, 4)

B.1.1.5.2.1 Memory usage

Memory used

Scale 0, 3, 5.

- 0 – unacceptable amount of memory utilized
- 3 – reasonable amount of memory utilized
- 5 – less than expected amount of memory utilized

(CQM-1 Page 4 and CQM-2 Pages 5, 6)

B.1.1.5.2.2 Disk usage

Disk used

Scale 0, 3, 5.

- 0 – unacceptable amount of disk space utilized
- 3 – reasonable amount of disk space utilized
- 5 – less than expected amount of disk space utilized

(CQM-1 Page 4 and CQM-2 Pages 5, 6)

B.1.1.5.2.3 Network usage

Bandwidth required

Scale 0, 3, 5.

- 0 – unacceptable network bandwidth required
- 3 – reasonable network bandwidth required
- 5 – less than expected network bandwidth required

B.1.1.5.3 Efficiency Compliance

The capability of the component to adhere to standards and conventions relating to efficiency.

Verify claims of compliance. If claimed and verified then CCOD User Environment rates better than one without claims.

Scale 0&5.

- 0 – non-compliant
- 5 – compliant

(ISO/IEC 9126-1 Section 6.4.3)

B.1.1.5.4 Accounting

A capability associated with components and compositions that allows for the use of those resources to be measured and accounted for. This implies that not only can the use of resources be properly measured, but also that those using those resources also be properly identified. This

function is useful in determining the need to deprecate unused components and compositions as well as the potential to implement “fee-for-service” payment models.

This measure will have low weight when considering end-user trust.

Mechanism Implemented

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

(SOA-RA-02 Section 5.3 Management Model)

B.1.1.5.5 Quality of service management

A manageability capability associating QoS information with a component and the CCOD User Environment’s ability to attempt to meet a particular QoS. For example a component may have a QoS defined for execution duration and the CCOD User Environment would attempt to execute that component on a particular CPU that is capable of meeting that QoS, e.g. a more powerful or less busy CPU.

This measure will have low weight when considering end-user trust.

Mechanism Implemented

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

(SOA-RA-02 Section 5.3 Management Model)

B.1.1.6 Maintainability

Express the ability of a CCOD User Environment to be modified without adverse affects on existing components and compositions.

(CQM-1 Page 2)

B.1.1.6.1 Stability

The capability of the CCOD User Environment to avoid unexpected effects upon components and compositions from modifications of the CCOD User Environment software.

B.1.1.6.1.1 Modifications over time

Types of modifications (fixes or enhancements) and their dates would have to be maintained by the CCOD User Environment. Stability would be measured by weighing fixes heaver than enhancements and more recent changes heavier than older ones.

Scale 0-5.

- 0 – The CCOD User Environment is frequently being modified mostly in response to software problems, new versions seems to present new problems
- 1 – Problems fixed are mostly software problems, new versions seem to present new problems.
- 2 – Problems fixed are mostly software problems, new versions seem to present few new problems.
- 3 – Problems fixed are an equal mix of software problems and enhancements, new versions seem to present few new problems. Use this value if maintenance data is not available.
- 4 – Problems fixed are mostly enhancements; new versions seem to present few new problems.
- 5 – The CCOD User Environment is very stable, versions are created on a fixed schedule, most changes are enhancements, software problems are few

(CQM-1 Pages 3, 4)

(CQM-1 Page 4 and CQM-2 Pages 5, 6)

B.1.1.6.2 Changeability

The capability of the composition CCOD User Environment to enable a specified modification to be implemented.

(CQM-1 Pages 3, 5)

B.1.1.6.2.1 Level of Customizability

A rating of the level of customization that can be done by the end user.

Scale 0-5.

- 0 – the CCOD User Environment has no customization features
- 1 – the CCOD User Environment is very hard to customize, the customization parameters are difficult to understand, or there are insufficient customization parameters.
- 2 – the CCOD User Environment is hard to customize, many of the customization parameters are difficult to understand, or many customization parameters are missing.
- 3 – the CCOD User Environment is somewhat hard to customize, some customization parameters are difficult to understand, or several customization parameters are missing.
- 4 – the CCOD User Environment is fairly easy to customize, most customization parameters are easy to understand, and most customization parameters are present.
- 5 – the CCOD User Environment is very easily customized, all customization parameters are easy to understand, the scope of customization is complete

(CQM-1 Page 5 and CQM-2 Pages 6, 7)

B.1.1.6.3 Testability of components & compositions (Life Cycle)

The capability of the CCOD User Environment to enable modifications of components and compositions to be validated.

(CQM-1 Pages 3, 5)

B.1.1.6.3.1 Test suite capability

Mechanism available

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

(CQM-1 Page 5 and CQM-2 Pages 6, 7)

B.1.1.6.4 Maintainability Compliance

The capability of the component to adhere to standards and conventions relating to maintainability.

Verify claims of compliance. If claimed and verified then CCOD User Environment rates better than one without claims.

Scale 0&5.

- 0 – non-compliant
- 5 – compliant

(ISO/IEC 9126-1 Section 6.5.5)

B.1.1.7 Portability

The capability of the CCOD User Environment to be transferred from one run-time environment to another.

Low weight from a trust perspective

Documentation inspection only

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

(CQM-1 Page 2)

B.1.1.8 Quality in Use

These measures are provided by users and provide an assessment of the CCOD User Environment during use.

B.1.1.8.1 Effectiveness

The capability of the CCOD User Environment to enable users to achieve specified goals with accuracy and completeness in a specified context of use. Based on user feedback.

Scale 0-5.

- 0 – not effective, mostly useless and inaccurate
- 1 – mostly ineffective, meets few needs accurately
- 2 – somewhat ineffective, meets some needs accurately
- 3 – effective, meets only critical needs accurately
- 4 – mostly effective, meets most user needs accurately
- 5 – completely effective, meets all user needs accurately

(ISO/IEC 9126-1 Section 7.1.1)

B.1.1.8.2 Productivity

The capability of the CCOD User Environment to enable users to expend appropriate amounts of resources in relation to the effectiveness achieved in a specified context of use. Based on user feedback.

Scale: 0-5.

- 0 – the effort to use this CCOD User Environment is so great that the user would rather not use it
- 1 – the effort to use the CCOD User Environment is much more than the return provided, the user solves only the most valuable problems using the CCOD User Environment
- 2 – the effort to use the CCOD User Environment is more than the return provided, the user solves only valuable problems using the CCOD User Environment
- 3 – the effort to use the CCOD User Environment is consistent with value returned, the user comes back regularly to solve most problems
- 4 – the effort to use the CCOD User Environment is less than the return provided, the user solves all day-to-day problems
- 5 – the effort to use the CCOD User Environment is so low that the user contemplates new problems to solve using the CCOD User Environment.

(ISO/IEC 9126-1 Section 7.1.2)

B.1.1.8.3 Satisfaction

The capability of the CCOD User Environment to satisfy uses in a specified context of use.

User feedback

Scale 0-5.

- 0 – totally unsatisfied
- 1 – mostly unsatisfied
- 2 – somewhat unsatisfied
- 3 – somewhat satisfied
- 4 – mostly satisfied
- 5 – completely satisfied

(ISO/IEC 9126-1 Section 7.1.4)

B.1.1.8.4 Attractiveness

The capability of the composition CCOD User Environment to be attractive to the user.

This refers to attributes of the software intended to make the software more attractive to the user, such as the use of color and the nature of the graphical design.

User Feedback

Scale 0-5.

- 0 – The CCOD User Environment user interface is very difficult to use and unattractive
- 1 – the CCOD User Environment user interface is mostly difficult to use and unattractive
- 2 – the CCOD User Environment user interface is somewhat difficult to use and unattractive
- 3 – the CCOD User Environment user interface is somewhat easy to use and attractive
- 4 – the CCOD User Environment user interface is mostly easy to use and attractive
- 5 – the CCOD User Environment user interface is very easy to use and attractive

(ISO/IEC 9126-1 Section 6.3.4)

B.1.2 CCOD Component

The CCOD Component represents the lowest level of functionality available to the composer. Components are constructed from software and, as such, are much like the components in a component development environment. The significant difference is that users of components in component development environment are, themselves, software developers, whereas the users of components in a CCOD User Environment are not software developers. As a result some of Component Quality Model has been tailored.

B.1.2.1 Functionality

Express the ability of a component to provide the required services when used under specified conditions.

(CQM-1 Page 2)

B.1.2.1.1 Accuracy

The capability of the component to provide the right or agreed results or effects with the needed degree of precision.

(CQM-1 Pages 2, 4)

B.1.2.1.1.1 Correctness

Based on available documentation determine if the component does what the documentation indicates. If there is no documentation then the correctness cannot be determined.

Scale 0-5.

- 0 – no documentation available
- 1 – less than 25% documented functions appear to operate correctly
- 2 – more than 25 % but less than 50% documented functions appear to operate correctly
- 3 – more than 50% but less than 75% documented functions appear to operate correctly
- 4 – more than 75% but less than 95% documented functions appear to operate correctly
- 5 – more than 95% documented functions appear to operate correctly.

(CQM-1 Page 4 and CQM-2 Page 5)

B.1.2.1.2 Suitability

The capability of the component to provide an appropriate set of functions for specified tasks and user objectives.

(CQM-1 Pages 2, 4)

B.1.2.1.2.1 Coverage

Specified functionality/number implemented

Physical inspection of existing documentation vs. actual functionality. Does the component attempt to do everything the documentation says it is supposed to do?

Scale 0-5.

- 0 – no documentation available
- 1 – less than 25% documented functions appear to be implemented
- 2 – more than 25 % but less than 50% documented functions appear to be implemented
- 3 – more than 50% but less than 75% documented functions appear to be implemented
- 4 – more than 75% but less than 95% documented functions appear to be implemented
- 5 – more than 95% documented functions appear to be implemented.

(CQM-1 Page 4 and CQM-2 Page 6)

B.1.2.1.2.2 Completeness

Implemented functionalities/total of specified functionalities

Physical inspection of existing documentation vs. actual functionality. For every function that the component attempts to perform, does it actually do the function?

Scale 0-5.

- 0 – no documentation available
- 1 – less than 25% documented functions actually implemented
- 2 – more than 25 % but less than 50% documented functions actually implemented
- 3 – more than 50% but less than 75% documented functions actually implemented
- 4 – more than 75% but less than 95% documented functions actually implemented
- 5 – more than 95% documented functions actually implemented.

(CQM-1 Page 4 and CQM-2 Page 6)

B.1.2.1.2.3 Pre and Post Conditioned

Verifications of pre and post conditions

For inputs and outputs assure that pre & post conditions are documented. For inputs, outputs and internal pre & post conditions all have been implemented. Determined by inspection of documentation and code.

Scale 0, 3, 5.

- 0 – no pre and post conditions are documented
- 3 – some pre and post conditions are documented and appear to be implemented in the component implementation
- 5 – all pre and post conditions are documented and appear to be implemented in the component implementation

(CQM-1 Page 4 and CQM-2 Page 6)

B.1.2.1.2.4 Proofs of Pre and Post Conditions

Proofs verification

Proofs are not likely to be provided when end users produce components, however, if a component is acquired through formal processes, proofs may be provided as part of the meta data delivered with the component. Metric should indicate presence & verify correctness.

Scale 0&5.

- 0 – no proofs present
- 5 – proofs present

(CQM-1 Page 4 and CQM-2 Page 6)

B.1.2.1.3 Functionality Compliance

The capability of the component to adhere to standards, conventions or regulations in laws and similar prescriptions relating to functionality.

(CQM-1 Pages 2, 4)

B.1.2.1.3.1 Standardization

Implementation and documentation analysis

Claim to be conformant in documentation and perform simple code check or execution to verify.

Scale 0&5.

- 0 – non-conformant
- 5 – conformant

(CQM-1 Page 4 and CQM-2 Pages 6, 7)

B.1.2.1.3.2 Certification

Verify documentation

Inspect documentation. This is much like proofs in that most components will not be certified. Do we need to validate the certification? If this component was formally acquired perhaps just the presence of the certification is sufficient.

Scale 0&5.

- 0 – not certified
- 5 – certified

(CQM-1 Page 4 and CQM-2 Pages 6, 7)

B.1.2.1.4 Self Contained

The functionalities that the component performs must be fully performed within itself.

(CQM-1 Pages 2, 4)

B.1.2.1.4.1 Dependant

Implementation analysis

Determine if the code is self contained; if not, the component is considered less trustworthy. Functionality internal to component / total functionality (by simple inspection)

Scale 0-5.

- 0 – less than 5 % of the functions of the component are internal
- 1 – more than 5 % but less than 25% of the functions of the component are internal
- 2 – more than 25 % but less than 50% of the functions of the component are internal
- 3 – more than 50% but less than 75% of the functions of the component are internal
- 4 – more than 75% but less than 95% of the functions of the component are internal
- 5 – more than 95% of the functions of the component are internal.

(CQM-1 Page 4 and CQM-2 Pages 6, 7)

B.1.2.1.5 Internal Interoperability

The capability of the component to interact with compositions and other components within the CCOD User Environment. In general, the CCOD User Environment enables and manages the interoperability between compositions and components by assuring that their interfaces have been developed properly and execute properly at run-time. If the CCOD User Environment does not provide such an assurance, the onus is on the component developer to assure interoperability. This measure, therefore, becomes more valuable when the CCOD User Environment is less capable and of lesser value when the CCOD User Environment is more capable.

(CQM-1 Pages 2, 4).

B.1.2.1.5.1 Data compatibility

Analysis of the data standard.

By inspection determine if data standards are used.

Scale 0, 3, 5.

- 0 – indicates no data standard used and a specific data structure is defined (e.g., a proprietary messaging system)
- 3 – a well known data standard is used throughout with a restriction on structure or schema (e.g., XML with Cursor on Target the only schema allowed).
- 5 – a well known data standard is used throughout with no restriction on structure or schema (e.g., XML or JSON).

(CQM-1 Page 4 and CQM-2 Page 6, 7)

B.1.2.1.5.2 Interface complexity

Complexity level

A measure of the difficulty of making the component interoperate. It is not sufficient to assure that the data is compatible. You also need to know how complex the interface is to implement.

Scale 0, 3, 5.

- 0 – The component interface is not documented, the number of parameters seems excessive for the function provided and they are difficult to understand
- 3 – some component interface documentation is provided, there are a reasonable number of parameters whose purpose can be discerned from their context
- 5 – The component interface is well documented, there are a reasonable number of understandable parameters

B.1.2.2 Security

Components are low level software capabilities that are not expected to possess the security functions that should be present in the CCOD User Environment. Therefore, none of the security functions described in NIST 800-53 –"Recommended Security Controls for Federal Information Systems and Organizations" are listed in Component section of the taxonomy.

Software secureness is, however, a desirable attribute of components, so this section is duplicated from the CCOD User Environment section.

B.1.2.2.1 Run-time Secureness

Run-time secureness is based on the 2010 CWE/SANS Top 25 Most Dangerous Software Errors. This is a list of the most widespread and critical programming errors that can lead to serious software vulnerabilities. They are often easy to find, and easy to exploit. They are dangerous because they will frequently allow attackers to completely take over the software, steal data, or prevent the software from working at all.

B.1.2.2.1.1 Insecure Component Interaction

These weaknesses are related to insecure ways in which data is sent and received between separate components, modules, programs, processes, threads, or systems.

####### B.1.2.2.1.1.1 Cross-site Scripting

Cross-site scripting (XSS) vulnerabilities occur when:

Untrusted data enters a web application, typically from a web request.

- The web application dynamically generates a web page that contains this untrusted data.
- During page generation, the application does not prevent the data from containing content that is executable by a web browser, such as JavaScript, HTML tags, HTML attributes, mouse events, Flash, ActiveX, etc.
- A victim visits the generated web page through a web browser, which contains malicious script that was injected using the untrusted data.
- Since the script comes from a web page that was sent by the web server, the victim's web browser executes the malicious script in the context of the web server's domain.
- This effectively violates the intention of the web browser's same-origin policy, which states that scripts in one domain should not be able to access resources or run code in a different domain.

There are three main kinds of XSS:

Type 1: Reflected XSS (or Non-Persistent)

> The server reads data directly from the HTTP request and reflects it back in the HTTP response. Reflected XSS exploits occur when an attacker causes a victim to supply dangerous content to a vulnerable web application, which is then reflected back to the victim and executed by the web browser. The most common mechanism for delivering malicious content is to include it as a parameter in a URL that is posted publicly or e-mailed directly to the victim. URLs constructed in this manner constitute the core of many phishing schemes, whereby an attacker convinces a victim to visit a URL that refers to a vulnerable site. After the site reflects the attacker's content back to the victim, the content is executed by the victim's browser.

Type 2: Stored XSS (or Persistent)

The application stores dangerous data in a database, message forum, visitor log, or other trusted data store. At a later time, the dangerous data is subsequently read back into the application and included in dynamic content. From an attacker's perspective, the optimal place to inject malicious content is in an area that is displayed to either many users or particularly interesting users. Interesting users typically have elevated privileges in the application or interact with sensitive data that is valuable to the attacker. If one of these users executes malicious content, the attacker may be able to perform privileged operations on behalf of the user or gain access to sensitive data belonging to the user. For example, the attacker might inject XSS into a log message, which might not be handled properly when an administrator views the logs.

Type 0: DOM-Based XSS

In DOM-based XSS, the client performs the injection of XSS into the page; in the other types, the server performs the injection. DOM-based XSS generally involves server-controlled, trusted script that is sent to the client, such as Javascript that performs sanity checks on a form before the user submits it. If the server-supplied script processes user-supplied data and then injects it back into the web page (such as with dynamic HTML), then DOM-based XSS is possible.
Once the malicious script is injected, the attacker can perform a variety of malicious activities. The attacker could transfer private information, such as cookies that may include session information, from the victim's machine to the attacker. The attacker could send malicious requests to a web site on behalf of the victim, which could be especially dangerous to the site if the victim has administrator privileges to manage that site. Phishing attacks could be used to emulate trusted web sites and trick the victim into entering a password, allowing the attacker to compromise the victim's account on that web site. Finally, the script could exploit vulnerability in the web browser itself possibly taking over the victim's machine, sometimes referred to as "drive-by hacking."

In many cases, the attack can be launched without the victim even being aware of it. Even with careful users, attackers frequently use a variety of methods to encode the malicious portion of the attack, such as URL encoding or Unicode, so the request looks less suspicious.

CWE-79 rank [1]

Scale 0&5.

- 0 – the weakness is present
- 5 – the weakness is not present

B.1.2.2.1.1.2 SQL Injection

Without sufficient removal or quoting of SQL syntax in user-controllable inputs, the generated SQL query can cause those inputs to be interpreted as SQL instead of ordinary user data. This can be used to alter query logic to bypass security checks, or to insert additional statements that modify the back-end database, possibly including execution of system commands.

SQL injection has become a common issue with database-driven web sites. The flaw is easily detected, and easily exploited, and as such, any site or software package with even a minimal user base is likely to be subject to an attempted attack of this kind. This flaw depends on the fact that SQL makes no real distinction between the control and data planes.

CWE-89 rank [2]

Scale 0&5.

- 0 – the weakness is present
- 5 – the weakness is not present

B.1.2.2.1.1.3 Cross-Site Request Forgery (CSRF)

When a web server is designed to receive a request from a client without any mechanism for verifying that it was intentionally sent, then it might be possible for an attacker to trick a client into making an unintentional request to the web server which will be treated as an authentic request. This can be done via a URL, image load, XMLHttpRequest, etc. and can result in data disclosure or unintended code execution.

CWE-352 rank [4]

Scale 0&5.

- 0 – the weakness is present
- 5 – the weakness is not present

B.1.2.2.1.1.4 Unrestricted Upload of Dangerous Files Type

The component allows the attacker to upload or transfer files of dangerous types that can be automatically processed within the CCOD User Environment.

Arbitrary code execution is possible if an uploaded file is interpreted and executed as code by the recipient. This is especially true for .asp and .php extensions uploaded to web servers because these file types are often treated as automatically executable, even when file system permissions do not specify execution. For example, in Unix environments, programs typically cannot run unless the execute bit is set, but PHP programs may be executed by the web server without directly invoking them on the operating system.

CWE-434 rank [8]

Scale 0&5.

- 0 – the weakness is present
- 5 – the weakness is not present

B.1.2.2.1.1.5 OS Command Injection

This could allow attackers to execute unexpected, dangerous commands directly on the operating system. This weakness can lead to vulnerability in environments in which the attacker does not have direct access to the operating system, such as in web applications. Alternately, if the weakness occurs in a privileged program, it could allow the attacker to specify commands that normally would not be accessible, or to call alternate commands with privileges that the attacker does not have. The problem is exacerbated if the compromised process fails to follow the principle of least privilege, because the attacker-controlled commands may run with special system privileges that increase the amount of damage.

There are at least two subtypes of OS command injection:

- The application intends to execute a single, fixed program that is under its own control. It intends to use externally-supplied inputs as arguments to that program. For example, the program might use system("nslookup [HOSTNAME]") to run nslookup and allow the user to

supply a HOSTNAME, which is used as an argument. Attackers cannot prevent nslookup from executing. However, if the program does not remove command separators from the HOSTNAME argument, attackers could place the separators into the arguments, which allows them to execute their own program after nslookup has finished executing.

- The application accepts an input that it uses to fully select which program to run, as well as which commands to use. The application simply redirects this entire command to the operating system. For example, the program might use "exec([COMMAND])" to execute the [COMMAND] that was supplied by the user. If the COMMAND is under attacker control, then the attacker can execute arbitrary commands or programs. If the command is being executed using functions like exec() and CreateProcess(), the attacker might not be able to combine multiple commands together in the same line.

From a weakness standpoint, these variants represent distinct programmer errors. In the first variant, the programmer clearly intends that input from untrusted parties will be part of the arguments in the command to be executed. In the second variant, the programmer does not intend for the command to be accessible to any untrusted party, but the programmer probably has not accounted for alternate ways in which malicious attackers can provide input.

CWE-78 rank [9]

Scale 0&5.

- 0 – the weakness is present
- 5 – the weakness is not present

B.1.2.2.1.1.6 Error Message Information Exposure through an

The sensitive information may be valuable information on its own (such as a password), or it may be useful for launching other, more deadly attacks. If an attack fails, an attacker may use error information provided by the server to launch another more focused attack. For example, an attempt to exploit a path traversal weakness (CWE-22) might yield the full pathname of the installed application. In turn, this could be used to select the proper number of ".." sequences to navigate to the targeted file. An attack using SQL injection (CWE-89) might not initially succeed, but an error message could reveal the malformed query, which would expose query logic and possibly even passwords or other sensitive information used within the query.

CWE-209 rank [17]

Scale 0&5.

- 0 – the weakness is present
- 5 – the weakness is not present

B.1.2.2.1.1.7 URL Redirection to Untrusted Site

An http parameter may contain a URL value and could cause the web application to redirect the request to the specified URL. By modifying the URL value to a malicious site, an attacker may successfully launch a phishing scam and steal user credentials. Because the server name in the modified link is identical to the original site, phishing attempts have a more trustworthy appearance.

CWE-601 rank [23]

Scale 0&5.

- 0 – the weakness is present
- 5 – the weakness is not present

B.1.2.2.1.1.8 Race Condition

This can have security implications when the expected synchronization is in security-critical code, such as recording whether a user is authenticated, or modifying important state information that should not be influenced by an outsider.

CWE-362 rank [25]

Scale 0&5.

- 0 – the weakness is present
- 5 – the weakness is not present

B.1.2.2.1.2 Risky Resource Management

The weaknesses in this category are related to ways in which software does not properly manage the creation, usage, transfer, or destruction of important system resources.

B.1.2.2.1.2.1 Classic Buffer Overflow

A buffer overflow condition exists when a program attempts to put more data in a buffer than it can hold, or when a program attempts to put data in a memory area outside of the boundaries of a buffer. The simplest type of error, and the most common cause of buffer overflows, is the "classic" case in which the program copies the buffer without checking its length at all. Other variants exist, but the existence of a classic overflow strongly suggests that the programmer is not considering even the most basic of security protections.

CWE-120 rank [3]

Scale 0&5.

- 0 – the weakness is present
- 5 – the weakness is not present

B.1.2.2.1.2.2 IPath Traversal

Many file operations are intended to take place within a restricted directory. By using special elements such as ".." and "/" separators, attackers can escape outside of the restricted location to access files or directories that are elsewhere on the system. One of the most common special elements is the "../" sequence, which in most modern operating systems is interpreted as the parent directory of the current location. This is referred to as relative path traversal. Path traversal also covers the use of absolute pathnames such as "/usr/local/bin", which may also be useful in accessing unexpected files. This is referred to as absolute path traversal.

In many programming languages, the injection of a null byte (the 0 or NUL) may allow an attacker to truncate a generated filename to widen the scope of attack. For example, the software may add ".txt" to any pathname, thus limiting the attacker to text files, but a null injection may effectively remove this restriction.

CWE-22 rank [7]

Scale 0&5.

- 0 – the weakness is present
- 5 – the weakness is not present

B.1.2.2.1.2.3 Buffer Access with Incorrect Length Value

The software uses a sequential operation to read or write a buffer, but it uses an incorrect length value that causes it to access memory that is outside of the bounds of the buffer.

CWE-805 rank [12]

Scale 0&5.

- 0 – the weakness is present
- 5 – the weakness is not present

B.1.2.2.1.2.4 Improper Check for Exceptional Conditions

The programmer may assume that certain events or conditions will never occur or do not need to be worried about, such as low memory conditions, lack of access to resources due to restrictive permissions, or misbehaving clients or components. However, attackers may intentionally trigger these unusual conditions which thus violating the programmer's assumptions, possibly introducing instability, incorrect behavior, or a vulnerability.

Note that this entry is not exclusively about the use of exceptions and exception handling, which are mechanisms for both checking and handling unusual or unexpected conditions.

CWE-754 rank [13]

Scale 0&5.

- 0 – the weakness is present
- 5 – the weakness is not present

B.1.2.2.1.2.5 PHP File Inclusion

In certain versions and configurations of PHP, this can allow an attacker to specify a URL to a remote location from which the software will obtain the code to execute. In other cases in association with path traversal, the attacker can specify a local file that may contain executable statements that can be parsed by PHP.

CWE-98 rank [14]

Scale 0&5.

- 0 – the weakness is present
- 5 – the weakness is not present

B.1.2.2.1.2.6 Improper Validation of Array Index

The product uses untrusted input when calculating or using an array index, but the product does not validate or incorrectly validates the index to ensure the index references a valid position within the array.

CWE-129 rank [15]

Scale 0&5.

- 0 – the weakness is present
- 5 – the weakness is not present

B.1.2.2.1.2.7 Integer Overflow or Wraparound

An integer overflow or wraparound occurs when an integer value is incremented to a value that is too large to store in the associated representation. When this occurs, the value may wrap to become a very small or negative number. While this may be intended behavior in circumstances that rely on wrapping, it can have security consequences if the wrap is unexpected. This is especially the case if the integer overflow can be triggered using user-supplied inputs. This becomes security-critical when the result is used to control looping, make a security decision, or determine the offset or size in behaviors such as memory allocation, copying, concatenation, etc.

CWE-190 rank [16]

Scale 0&5.

- 0 – the weakness is present
- 5 – the weakness is not present

B.1.2.2.1.2.8 Incorrect Calculation of Buffer Size

The software does not correctly calculate the size to be used when allocating a buffer, which could lead to a buffer overflow.

CWE-131 rank [18]

Scale 0&5.

- 0 – the weakness is present
- 5 – the weakness is not present

B.1.2.2.1.2.9 Download of Code without Integrity Check

The product downloads source code or an executable from a remote location and executes the code without sufficiently verifying the origin and integrity of the code.

CWE-494 rank [20]

Scale 0&5.

- 0 – the weakness is present
- 5 – the weakness is not present

B.1.2.2.1.2.10 Allocation of Resources without Limits or Throttling

The software allocates a reusable resource or group of resources on behalf of an actor without imposing any restrictions on how many resources can be allocated, in violation of the intended security policy for that actor.

CWE-770 rank [22]

Scale 0&5.

- 0 – the weakness is present
- 5 – the weakness is not present

B.1.2.2.1.3 Prevention of Porous Defenses

The weaknesses in this category are related to defensive techniques that are often misused, abused, or just plain ignored.

B.1.2.2.1.3.1 Improper Access Control (Authorization)

When access control checks are not applied consistently - or not at all - users are able to access data or perform actions that they should not be allowed to perform. This can lead to a wide range of problems, including information leaks, denial of service, and arbitrary code execution.

CWE-285 rank [5]

Scale 0&5.

- 0 – the weakness is present
- 5 – the weakness is not present

B.1.2.2.1.3.2 Reliance on Untrusted Inputs in a Security Decision

Developers may assume that inputs such as cookies, environment variables, and hidden form fields cannot be modified. However, an attacker could change these inputs using customized clients or other attacks. This change might not be detected. When security decisions such as authentication and authorization are made based on the values of these inputs, attackers can bypass the security of the software.

Without sufficient encryption, integrity checking, or other mechanism, any input that originates from an outsider cannot be trusted.

CWE-807 rank [6]

Scale 0&5.

- 0 – the weakness is present
- 5 – the weakness is not present

B.1.2.2.1.3.3 Missing Encryption of Sensitive Data

The lack of proper data encryption passes up the guarantees of confidentiality, integrity, and accountability that properly implemented encryption conveys

CWE-311 rank [10]

Scale 0&5.

- 0 – the weakness is present
- 5 – the weakness is not present

B.1.2.2.1.3.4 Use of Hard-coded Credentials

A hard-coded password typically leads to a significant authentication failure that can be difficult for the system administrator to detect. Once detected, it can be difficult to fix, so the administrator may be forced into disabling the product entirely. There are two main variations:

Inbound: the software contains an authentication mechanism that checks for a hard-coded password.

Outbound: the software connects to another system or component, and it contains hard-coded password for connecting to that component.

In the Inbound variant, a default administration account is created, and a simple password is hard-coded into the product and associated with that account. This hard-coded password is the same for each installation of the product, and it usually cannot be changed or disabled by system administrators without manually modifying the program, or otherwise patching the software. If the password is ever discovered or published (a common occurrence on the Internet), then anybody with knowledge of this password can access the product. Finally, since all installations of the software will have the same password, even across different organizations, this enables massive attacks such as worms to take place.

The Outbound variant applies to front-end systems that authenticate with a back-end service. The back-end service may require a fixed password which can be easily discovered. The programmer may simply hard-code those back-end credentials into the front-end software. Any user of that program may be able to extract the password. Client-side systems with hard-coded passwords pose even more of a threat, since the extraction of a password from a binary is usually very simple.

CWE-798 rank [11]

Scale 0&5.

- 0 – the weakness is present
- 5 – the weakness is not present

B.1.2.2.1.3.5 Missing Authentication for Critical Function

The software does not perform any authentication for functionality that requires a provable user identity or consumes a significant amount of resources.

CWE-306 rank [19]

Scale 0&5.

- 0 – the weakness is present
- 5 – the weakness is not present

B.1.2.2.1.3.6 Incorrect Permission Assignment for Critical Resource

When a resource is given a permissions setting that provides access to a wider range of actors than required, it could lead to the disclosure of sensitive information or the modification of that resource by unintended parties. This is especially dangerous when the resource is related to program configuration, execution or sensitive user data.

CWE-732 rank [21]

Scale 0&5.

- 0 – the weakness is present
- 5 – the weakness is not present

B.1.2.2.1.3.7 Use of a Insecure Cryptographic Algorithm

The use of a broken or risky cryptographic algorithm is an unnecessary risk that may result in the disclosure of sensitive information.

CWE-327 rank [24]

Scale 0&5.

- 0 – the weakness is present
- 5 – the weakness is not present

B.1.2.3 Reliability

Express the ability of the component to maintain a specified level of performance, when used under specified conditions.

(CQM-1 Page 2)

B.1.2.3.1 Fault Tolerance

The capability of the component to maintain a specified level of performance in cases of component faults or of infringement of its specified interface.

(CQM-1 Pages 2, 4)

B.1.2.3.1.1 Mechanism available

Mechanism identification

Graceful failure described in documentation

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

(CQM-1 Page 4 and CQM-2 Page 5)

B.1.2.3.1.2 Mechanism efficiency

Rough ratio of functions with graceful failure / total functionality.

Scale 0-5.

- 0 – no fault tolerance capability observed
- 1 – very few functions appear to be fault tolerant
- 2 – few functions appear to be fault tolerant
- 3 – some functions appear to be fault tolerant
- 4 – most functions appear to be fault tolerant
- 5 – all functions appear to be fault tolerant

(CQM-1 Page 4 and CQM-2 Page 5)

B.1.2.3.2 Recoverability

The capability of the component to re-establish a specified level of performance and recover the data directly affected in the case of a failure.

(CQM-1 Pages 2, 4)

B.1.2.3.2.1 Mechanism available

Mechanism Implemented

Recovery described in documentation

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

(CQM-1 Page 4 and CQM-2 Page 5)

B.1.2.3.2.2 Mechanism efficiency

Rough ratio of functions with recovery mechanisms/ total functions and level of recovery by inspection (completeness and processes).

Need to understand completeness of recoverability capability.

Scale 0-5.

- 0 – no recovery mechanism observed
- 1 – very few functions appear to have recovery mechanisms
- 2 – few functions appear to have recovery mechanisms
- 3 – some functions appear to have recovery mechanisms
- 4 – most functions appear to have recovery mechanisms
- 5 – all functions appear to have recovery mechanisms

B.1.2.3.3 Maturity

The capability of the component to avoid failure as a result of faults in the software.

(CQM-1 Pages 2, 4)

Metrics based on maintenance history. Need to have access to source code control and problem reporting information. An absence of such information implies we cannot determine maturity.

B.1.2.3.3.1 Volatility

Analysis of the time between versions

Scale 0-5.

- 0 – critical bugs (Cat 1 & 2) are on the books for longer than 2 years
- 1 – critical bugs (Cat 1 & 2) are on the books for longer than 1 year but less than 2 years
- 2 – critical bugs (Cat 1 & 2) are on the books for longer than 6 months but less than 1 year. Use this rating if no version/bug information is available
- 3 – critical bugs (Cat 1 & 2) are on the books for longer than 3 months but less than 6 months
- 4 – critical bugs (Cat 1 & 2) are on the books for longer than 1 month but less than 3 months
- 5 – critical bugs (Cat 1 & 2) are on the books for less than 1 month

(CQM-1 Pages 4, 5 and CQM-2 Pages 6, 7)

B.1.2.3.3.2 Failure removal

of bugs fixed in a version

Scale 0-5.

- 0 – an unlikely score, all outstanding submitted bugs are ignored and never fixed.
- 1 – the number of outstanding submitted bugs is increasing exponentially with each version
- 2 – the number of outstanding submitted bugs is increasing linearly with each version
- 3 – the number of outstanding submitted bugs is remaining constant with each version
- 4 – the number of outstanding submitted bugs is decreasing with each version.
- 5 – an unlikely score, all outstanding submitted bugs fixed with each version.

(CQM-1 Page 5 and CQM-2 Pages 6, 7)

B.1.2.4 Usability

The ability to be used by someone other than the originator when constructing a composition.

(CQM-1 Page 2)

B.1.2.4.1 Configurability

The ability to specify parameters that aid in the use of the component within a particular context. The ability of the component be configurable (e.g. through a XML file or a text file, the number of parameters, etc.)

(CQM-1 Page 2, 4)

B.1.2.4.1.1 Effort to Configure

Time spent to configure correctly

By inspection metric (not end user feedback)

Scale 0-5.

- 0 – the component is non-configurable
- 1 – the component is very hard to configure, the configuration parameters are difficult to understand, or there are insufficient configuration parameters.
- 2 – the component is hard to configure, many of the configuration parameters are difficult to understand, or many configuration parameters are missing.
- 3 – the component is somewhat hard to configure, some configuration parameters are difficult to understand, or several configuration parameters are missing.
- 4 – the component is fairly easy to configure, most configuration parameters are easy to understand, and most configuration parameters are present.
- 5 – the component is very easily configured, all configuration parameters are easy to understand, the scope of configuration is complete

(CQM-1 Page 4 and CQM-2 Page-5)

B.1.2.4.2 Understandability

The capability of the component to enable the composer to understand whether the component is suitable, and how it can be used for particular tasks and conditions of use.

(CQM-1 Pages 2, 5)

B.1.2.4.2.1 Documentation quality

Documentation analysis.

Scale 0-5.

- 0 – the component is not documented
- 1 – the documentation is minimal
- 2 – the documentation is mostly complete
- 3 – the documentation is complete but complex
- 4 – the documentation is complete but somewhat complex
- 5 – Documentation is complete and easy to understand

(CQM-1 Page 5 and CQM-2 Pages 6, 7)

B.1.2.4.3 Operability

The capability of the component to enable the user to operate and control it.

(CQM-1 Pages 3, 5)

B.1.2.4.3.1 Parameters

of parameters

Scale 0, 3, 5.

- 0 – more than 10
- 3 – 6 to 9
- 5 – Fewer than 5

(CQM-1 Page 5 and CQM-2 Pages 6, 7)

B.1.2.4.3.2 Required parameters

of required interfaces

Scale 0-5.

- 0 – all parameters are optional
- 1 – greater than 75% but less than 100% of parameters are optional
- 2 – greater than 50% but less than 75 % of parameters are optional
- 3 – greater than 20% but less than 50% of parameters are optional
- 4 – greater than 0% but less than 20% of parameters are optional
- 5 – No optional parameters

(CQM-1 Page 5 and CQM-2 Pages 6, 7)

B.1.2.4.3.3 Effort for operating

Composer rating

Scale 0-5.

- 0 – very difficult to operate
- 1 – difficult to operate
- 2 – somewhat difficult to operate
- 3 – somewhat easy to operate
- 4 – easy to operate
- 5 – very easy to operate

(CQM-1 Page 5 and CQM-2 Pages 6, 7)

B.1.2.5 Efficiency

Express the ability of a component to provide appropriate performance, relative to the amount of resources used.

(CQM-1 Page 2)

B.1.2.5.1 Time Behavior

The capability of the component to provide appropriate response and processing times and throughput rates when performing its function, under stated conditions.

(CQM-1 Pages 3, 4)

B.1.2.5.1.1 Response time

Time taken between a set of invocations

Scale 0, 3, 5.

- 0 – given the amount of data processed and processing performed the response time is less than expected
- 3 – given the amount of data processed and processing performed the response time is acceptable
- 5 – given the amount of data processed and processing performed the response time is better than expected

(CQM-1 Page 4 and CQM-2 Page-5)

B.1.2.5.1.2 Throughput

Amount of outputs produced with success / period of time

Scale 0, 3, 5.

- 0 – unacceptable throughput observed
- 3 – reasonable throughput observed
- 5 – high throughput observed

(CQM-1 Page 4 and CQM-2 Page-5)

B.1.2.5.2 Resource Utilization

The capability of the component to use appropriate amounts and types of resources when performing its function under stated conditions.

(CQM-1 Pages 3, 4)

B.1.2.5.2.1 Memory usage

Memory used

Scale 0, 3, 5.

- 0 – unacceptable amount of memory utilized
- 3 – reasonable amount of memory utilized
- 5 – less than expected amount of memory utilized

(CQM-1 Page 4 and CQM-2 Pages 5, 6)

B.1.2.5.2.2 Disk usage

Disk used

Scale 0, 3, 5.

- 0 – unacceptable amount of disk space utilized
- 3 – reasonable amount of disk space utilized
- 5 – less than expected amount of disk space utilized

(CQM-1 Page 4 and CQM-2 Pages 5, 6)

B.1.2.5.2.3 Network usage

Bandwidth required

Scale 0, 3, 5.

- 0 – unacceptable network bandwidth required
- 3 – reasonable network bandwidth required
- 5 – less than expected network bandwidth required

B.1.2.5.3 Scalability

The ability of the component to accommodate major data volumes without changing its implementation.

(CQM-1 Page 3, 4)

B.1.2.5.3.1 Processing capacity

Determine linearity of response time over a range of data set sizes.

Scale 0, 2, 5.

- 0 – component is unable to handle larger data volumes
- 2 – component appears to handle larger data volumes with exponential response time
- 5 – component appears to handle larger data volumes with linear response time

(CQM-1 Page 4 and CQM-2 Pages 5, 6)

B.1.2.6 Maintainability

Express the ability of a component to be modified.

(CQM-1 Page 2)

B.1.2.6.1 Stability

The capability of the component to avoid unexpected effects from modifications of the software.

B.1.2.6.1.1 Modifications over time

Types of modifications (fixes or enhancements) and their dates would have to be maintained by the CCOD User Environment. Stability would be measured by weighing fixes heaver than enhancements and more recent changes heavier than older ones.

Scale 0-5.

- 0 – The component is frequently being modified mostly in response to software problems, new versions seems to present new problems
- 1 – Problems fixed are mostly software problems, new versions seem to present new problems.
- 2 – Problems fixed are mostly software problems, new versions seem to present few new problems.
- 3 – Problems fixed are an equal mix of software problems and enhancements, new versions seem to present few new problems. Use this value if maintenance data is not available.
- 4 – Problems fixed are mostly enhancements; new versions seem to present few new problems.
- 5 – The component is very stable, versions are created on a fixed schedule, most changes are enhancements, software problems are few

(CQM-1 Pages 3, 4)

(CQM-1 Page 4 and CQM-2 Pages 5, 6)

B.1.2.6.2 Changeability

The capability of the component to enable a specified modification to be implemented.

(CQM-1 Pages 3, 5)

B.1.2.6.2.1 Complexity level

Use complexity analysis tool. All functionalities usage / time to operate

Scale 0-5.

- 0 – Complexity greater than 50
- 1 – Complexity range of 25-49
- 2 – Complexity range of 15-24
- 3 – Complexity range of 10-14
- 4 – Complexity range of 5-9
- 5 – Complexity range of 1-4

(CQM-1 Page 5 and CQM-2 Pages 6, 7)

B.1.2.6.2.2 Cohesion

Cohesiveness is defined as a measure of the internals of module. A module is considered to be highly cohesive if the elements within it are closely related functionally. Six levels of cohesiveness are defined, from lowest to highest:

- Coincidental (random collections of elements),
- Logical (elements are related to one another logically, like all input and output operations),
- Temporal (all elements are related in time or executed in the same time period),
- Communicational (elements related to the same set of input or output data),
- Sequential (elements are related because they pass data from one element to the next),
- Functional (all elements are related to the performance of a single function.

Scale 0-5.

- 0 – Coincidental - the component performs multiple functions that are not related in any way
- 1 – Logical - the component performs multiple functions that are related logically (e.g., all input functions in a single component)
- 2 – Temporal - the component performs multiple functions which are related in time
- 3 – Communicational - the component performs multiple functions on a single data structure or type
- 4 – Sequential - the component performs multiple functions by passing the data from one function to the next
- 5 – Functional - the component performs a single function (e.g. sort or filter).

(CQM-1 Page 5)

B.1.2.6.2.3 Coupling

Coupling is defined as "the measure of the strength of association established by a connection from one module to another." Further, "strong coupling complicates a system since a module is harder to understand, change or correct by itself if it is highly interrelated with other modules. Complexity can be reduced by designing systems with the weakest possible coupling between modules. Coupling depends (1) on how complicated the connection is, (2) on whether the connection refers to the module itself or something inside it, and (3) on what is being sent or received. Coupling increases with increasing complexity or obscurity of the interface. Coupling is lower when the connection is to the normal module interface than when the connection is to an internal component. Coupling is lower with data connections than with control connections, which are in turn lower than hybrid connections (modification of one module's code by another module). The complexity of an interface is a matter of how much information is needed to state or understand the connection. Thus, obvious relationships result in lower coupling than obscure or inferred ones. The more syntactic units (such as parameters) in the statement of a connection, the higher the coupling. Thus, extraneous elements irrelevant to the programmer's and the modules immediate task increase coupling unnecessarily."

Scale 0-5.

- 0 – this component changes the implementation of another component
- 1 – this component uses at least one global variable
- 2 – this component has at least one control parameter
- 3 – this component utilizes parameters where the structure is important
- 4 – this component only parameters where the structure is not important
- 5 – this component is independent of all others

(CQM-1 Page 5)

B.1.2.6.3 Testability

The capability of the component to enable modifications to be validated.

(CQM-1 Pages 3, 5)

B.1.2.6.3.1 Test cases and/or proofs provided

Evaluate the existence of test cases and proofs.

Scale 0-5.

- 0 – no test cases provided
- 1 – Less than 50% of the functions have test cases provided
- 2 – Less than 100% but more than 50% of the functions have test cases provided
- 3 – test cases are provided that provide coverage of the component
- 4 – some proofs provided and the rest covered by test cases
- 5 – proofs provided for all functions

(CQM-1 Page 5 and CQM-2 Pages 6, 7)

B.1.2.6.4 Quality in Use

These measures are provided by users and provide an assessment of the component during use.

B.1.2.6.4.1 Effectiveness

The capability of the component to enable users to achieve specified goals with accuracy and completeness in a specified context of use.

User Feedback

Scale 0-5.

- 0 – not effective, mostly useless and inaccurate
- 1 – mostly ineffective, meets few needs accurately
- 2 – somewhat ineffective, meets some needs accurately
- 3 – effective, meets only critical needs accurately
- 4 – mostly effective, meets most user needs accurately
- 5 – completely effective, meets all user needs accurately

(ISO/IEC 9126-1 Section 7.1.1)

B.1.3 CCOD Composition

CCOD compositions are made up of components and or other compositions to for higher level functional elements. Compositions are unlike components in that they are not developed software. Composition are representations of the components that make them up and are "stitched" together at composition time and run time to create executable functionality. Compositions that present data through a user interface or save in a file represent "capabilities" that can be used by end users. Primary differences between the taxonomy in this section and that in the component section reflect the differences in software and a representation of a collection of components wired together to form a greater functionality.

B.1.3.1 Functionality

Express the ability of a composition to provide the required services when used under specified conditions.

(CQM-1 Page 2)

B.1.3.1.1 Accuracy

The capability of the composition to provide the right or agreed results or effects with the needed degree of precision.

(CQM-1 Pages 2, 4)

B.1.3.1.1.1 Correctness

Based on available documentation determine if the component does what the documentation indicates. If there is no documentation then the correctness cannot be determined.

Scale 0-5.

- 0 – no documentation available
- 1 – less than 25% documented functions appear to operate correctly
- 2 – more than 25 % but less than 50% documented functions appear to operate correctly
- 3 – more than 50% but less than 75% documented functions appear to operate correctly
- 4 – more than 75% but less than 95% documented functions appear to operate correctly
- 5 – more than 95% documented functions appear to operate correctly.

(CQM-1 Page 4 and CQM-2 Page 5)

B.1.3.1.2 Suitability

The capability of the composition to provide an appropriate set of functions for specified tasks and user objectives.

(CQM-1 Pages 2, 4)

B.1.3.1.2.1 Coverage

Specified functionality/number implemented

Physical inspection of existing documentation vs. actual functionality. Does the composition attempt to do everything the documentation says it is supposed to do?

Scale 0-5.

- 0 – no documentation available
- 1 – less than 25% documented functions appear to be implemented
- 2 – more than 25 % but less than 50% documented functions appear to be implemented
- 3 – more than 50% but less than 75% documented functions appear to be implemented
- 4 – more than 75% but less than 95% documented functions appear to be implemented
- 5 – more than 95% documented functions appear to be implemented.

(CQM-1 Page 4 and CQM-2 Page 6)

B.1.3.1.2.2 Completeness

Implemented functionalities/total of specified functionalities

Physical inspection of existing documentation vs. actual functionality. For every function that the composition attempts to perform, does it actually do the function?

Scale 0-5.

- 0 – no documentation available
- 1 – less than 25% documented functions actually implemented
- 2 – more than 25 % but less than 50% documented functions actually implemented
- 3 – more than 50% but less than 75% documented functions actually implemented
- 4 – more than 75% but less than 95% documented functions actually implemented
- 5 – more than 95% documented functions actually implemented.

(CQM-1 Page 4 and CQM-2 Page 6)

B.1.3.1.2.3 Pre and post conditioned

Verifications of pre and post conditions

For inputs and outputs assure that pre & post conditions are documented. For inputs, outputs and internal pre & post conditions all have been implemented. Determined by inspection of documentation and composition.

Scale 0-5.

- 0 – no pre and post conditions are documented
- 1 – some pre and post conditions are documented and appear to be implemented in the composition
- 2 – most pre and post conditions are documented and appear to be implemented in the composition
- 3 – all pre and post conditions are documented and appear to be implemented in the composition
- 4 – pre and post conditions appear to be unnecessary

5 – proofs of pre and post conditions are documented

(CQM-1 Page 4 and CQM-2 Page 6)

B.1.3.1.3 Internal Interoperability

The capability of the composition to interact with other compositions and components within the CCOD User Environment.

In general, for compositions without user interfaces the CCOD User Environment enables and manages the interoperability between compositions and components by assuring that their interfaces have been developed properly and execute properly at run-time. If the CCOD User Environment does not provide such an assurance, the onus is on the composer to assure interoperability. This measure, therefore, becomes more valuable when the CCOD User Environment is less capable and of lesser value with the CCOD User Environment is more capable.

(CQM-1 Pages 2, 4).

B.1.3.1.3.1 Data compatibility

Analysis of the data standard

By inspection determine if data standards are used.

Scale 0, 3, 5.

- 0 – indicates no data standard used and a specific data structure is defined (e.g., a proprietary messaging system)
- 3 – a well known data standard is used throughout with a restriction on structure or schema (e.g., XML with Cursor on Target the only schema allowed).
- 5 – a well known data standard is used throughout with no restriction on structure or schema (e.g., XML or JSON).

(CQM-1 Page 4 and CQM-2 Page 6, 7)

B.1.3.1.3.2 Interface complexity

Complexity level

A measure of the difficulty of making the composition interoperate. It is not sufficient to assure that the data is compatible. You also need to know how complex the interface is to implement.

Scale 0, 3, 5.

- 0 – The composition interface is not documented, the number of parameters seems excessive for the function provided and they are difficult to understand
- 3 – some composition interface documentation is provided, there are a reasonable number of parameters whose purpose can be discerned from their context
- 5 – The composition interface is well documented, there are a reasonable number of understandable parameters

B.1.3.1.4 Capability Interoperability

The capability of an end user composition to interact with other end user compositions within the CCOD User Environment.

This category exists to measure the ability of capability to interact with another. For example, if one capability displays airport icons on map and another lists information about airports, if a user selects an airport on the map the corresponding airport in the list is highlighted and vice versa.

(CQM-1 Pages 2, 4).

B.1.3.1.4.1 Data compatibility

Analysis of the data standard

By inspection determine if data standards are used.

Scale 0, 3, 5.

- 0 – indicates no data standard used and a specific data structure is defined (e.g., a proprietary messaging system)
- 3 – a well known data standard is used throughout with a restriction on structure or schema (e.g., XML with Cursor on Target the only schema allowed).
- 5 – a well known data standard is used throughout with no restriction on structure or schema (e.g., XML or JSON).

(CQM-1 Page 4 and CQM-2 Page 6, 7)

B.1.3.1.4.2 Interface complexity

Complexity level

A measure of the difficulty of making the capability interoperate. It is not sufficient to assure that the data is compatible. You also need to know how complex the interface is to implement.

Scale 0, 3, 5.

- 0 – It is very difficult to pass information from one capability to another within the CCOD User Environment
- 3 – It is somewhat difficult to pass information from one capability to another
- 5 – It is very easy to pass information from one capability to another

B.1.3.1.5 Functionality Compliance

The capability of the composition to adhere to standards, conventions or regulations in laws and similar prescriptions relating to functionality.

(CQM-1 Pages 2, 4)

B.1.3.1.5.1 Standardization

Implementation and documentation analysis

Claim to be conformant in documentation and perform simple code check or execution to verify.

Scale 0, 3, 5.

- 0 – non-conformant with standards
- 3 – conformant with standards by observation
- 5 – certified conformant with standards

(CQM-1 Page 4 and CQM-2 Pages 6, 7)

B.1.3.1.6 Self Contained

The functionalities that the composition performs must be fully performed within itself.

(CQM-1 Pages 2, 4)

B.1.3.1.6.1 Dependant

Implementation analysis

Determine if the code is self contained, if not, the component is considered less trustworthy. Functionality internal to component / total functionality (by simple inspection)

Scale 0-5.

- 0 – less than 5 % of the composition are internal to the CCOD User Environment
- 1 – more than 5 % but less than 25% of the functions of the composition are internal to the CCOD User Environment
- 2 – more than 25 % but less than 50% of the functions of the composition are internal to the CCOD User Environment
- 3 – more than 50% but less than 75% of the functions of the composition are internal to the CCOD User Environment
- 4 – more than 75% but less than 95% of the functions of the composition are internal to the CCOD User Environment
- 5 – more than 95% of the functions of the composition are internal to the CCOD User Environment

(CQM-1 Page 4 and CQM-2 Pages 6, 7)

B.1.3.2 Reliability

Express the ability of the composition to maintain a specified level of performance, when used under specified conditions. This is of interest to the composer and the end-user.

(CQM-1 Page 2)

B.1.3.2.1 Fault Tolerance

The capability of the composition to maintain a specified level of performance in cases of composition faults or of infringement of its specified interface.

(CQM-1 Pages 2, 4)

B.1.3.2.1.1 Mechanism available

Mechanism identification

Graceful failure described in documentation

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

(CQM-1 Page 4 and CQM-2 Page 5)

B.1.3.2.1.2 Mechanism efficiency

Rough ratio of functions with graceful failure / total functionality.

Scale 0-5.

- 0 – no fault tolerance capability observed
- 1 – very few functions appear to be fault tolerant
- 2 – few functions appear to be fault tolerant
- 3 – some functions appear to be fault tolerant
- 4 – most functions appear to be fault tolerant
- 5 – all functions appear to be fault tolerant

(CQM-1 Page 4 and CQM-2 Page 5)

B.1.3.2.2 Recoverability

The capability of the composition to re-establish a specified level of performance and recover the data directly affected in the case of a failure.

(CQM-1 Pages 2, 4)

B.1.3.2.2.1 Mechanism available

Mechanism Implemented

Recovery described in documentation

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

(CQM-1 Page 4 and CQM-2 Page 5)

B.1.3.2.2.2 Mechanism efficiency

Rough ratio of functions with recovery mechanisms/ total functions and level of recovery by inspection (completeness and processes).

Scale 0-5.

- 0 – no recovery mechanism observed
- 1 – very few functions appear to have recovery mechanisms
- 2 – few functions appear to have recovery mechanisms
- 3 – some functions appear to have recovery mechanisms
- 4 – most functions appear to have recovery mechanisms
- 5 – all functions appear to have recovery mechanisms

B.1.3.2.3 Maturity

The capability of the composition to avoid failure as a result of faults in the software.

(CQM-1 Pages 2, 4)

Metrics based on maintenance history. Need to have access to source code control and problem reporting information. An absence of such information implies we cannot determine maturity.

B.1.3.2.3.1 Volatility

Analysis of the time between versions

Scale 0-5.

- 0 – critical bugs (Cat 1 & 2) are on the books for longer than 2 years
- 1 – critical bugs (Cat 1 & 2) are on the books for longer than 1 year but less than 2 years
- 2 – critical bugs (Cat 1 & 2) are on the books for longer than 6 months but less than 1 year. Use this rating if no version/bug information is available
- 3 – critical bugs (Cat 1 & 2) are on the books for longer than 3 months but less than 6 months
- 4 – critical bugs (Cat 1 & 2) are on the books for longer than 1 month but less than 3 months
- 5 – critical bugs (Cat 1 & 2) are on the books for less than 1 month

(CQM-1 Pages 4, 5 and CQM-2 Pages 6, 7)

B.1.3.2.3.2 Failure removal

of bugs fixed in a version

Scale 0-5.

- 0 – an unlikely score, all outstanding submitted bugs are ignored and never fixed.
- 1 – the number of outstanding submitted bugs is increasing exponentially with each version
- 2 – the number of outstanding submitted bugs is increasing linearly with each version
- 3 – the number of outstanding submitted bugs is remaining constant with each version
- 4 – the number of outstanding submitted bugs is decreasing with each version.
- 5 – an unlikely score, all outstanding submitted bugs fixed with each version.

(CQM-1 Page 5 and CQM-2 Pages 6, 7)

B.1.3.3 Composition Usability

The ability to be used by someone other than the originator when constructing a composition. This category is specifically measuring the usability of a composition from the composer's perspective.

(CQM-1 Page 2)

B.1.3.3.1 Configurability

The ability to specify parameters that aid in the use of the composition within a particular context. The ability of the composition be configurable (e.g. through a XML file or a text file, the number of parameters, etc.)

(CQM-1 Page 2, 4)

B.1.3.3.1.1 Effort to Configure

Time spent to configure correctly

By inspection metric (not end user feedback)

Scale 0-5.

- 0 – the composition is non-configurable
- 1 – the composition is very hard to configure, the configuration parameters are difficult to understand, or there are insufficient configuration parameters.
- 2 – the composition is hard to configure, many of the configuration parameters are difficult to understand, or many configuration parameters are missing.
- 3 – the composition is somewhat hard to configure, some configuration parameters are difficult to understand, or several configuration parameters are missing.
- 4 – the composition is fairly easy to configure, most configuration parameters are easy to understand, and most configuration parameters are present.
- 5 – the composition is very easily configured, all configuration parameters are easy to understand, the scope of configuration is complete

(CQM-1 Page 4 and CQM-2 Page-5)

B.1.3.3.2 Understandability

The capability of the composition to enable the composer to understand whether the composition is suitable, and how it can be used for particular tasks and conditions of use.

(CQM-1 Pages 2, 5)

B.1.3.3.2.1 Documentation quality

Documentation analysis Documentation analysis.

Scale 0-5.

- 0 – the composition is not documented
- 1 – the documentation is minimal
- 2 – the documentation is mostly complete
- 3 – the documentation is complete but complex
- 4 – the documentation is complete but somewhat complex
- 5 – documentation is complete and easy to understand

(CQM-1 Page 5 and CQM-2 Pages 6, 7)

B.1.3.3.3 Operability

The capability of the composition to enable the user to operate and control it.

(CQM-1 Pages 3, 5)

B.1.3.3.3.1 Parameters

of parameters

Scale 0, 3, 5.

- 0 – more than 10
- 3 – 6 to 9
- 5 – Fewer than 5

(CQM-1 Page 5 and CQM-2 Pages 6, 7)

B.1.3.3.3.2 Required parameters

of required parameters

Scale 0-5.

- 0 – all parameters are optional
- 1 – greater than 75% but less than 100% of parameters are optional
- 2 – greater than 50% but less than 75 % of parameters are optional
- 3 – greater than 20% but less than 50% of parameters are optional
- 4 – greater than 0% but less than 20% of parameters are optional
- 5 – No optional parameters

(CQM-1 Page 5 and CQM-2 Pages 6, 7)

B.1.3.3.3.3 Effort for operating

Composer rating

Scale 0-5.

- 0 – very difficult to operate
- 1 – difficult to operate
- 2 – somewhat difficult to operate
- 3 – somewhat easy to operate
- 4 – easy to operate
- 5 – very easy to operate

(CQM-1 Page 5 and CQM-2 Pages 6, 7)

B.1.3.4 Capability Usability

The ability of the capability to be used by someone other than the originator. This category specifically measures the usability of a composition with a user interface from the end user's perspective.

(CQM-1 Page 2)

B.1.3.4.1 Configurability

The ability to specify parameters that aid in the use of the capability within a particular context. The ability of the composition be configurable (e.g. through a XML file or a text file, the number of parameters, etc.)

(CQM-1 Page 2, 4)

B.1.3.4.1.1 Effort to Configure

Time spent to configure correctly

By inspection metric (not end user feedback)

Scale 0-5.

- 0 – the capability is non-configurable
- 1 – the capability is very hard to configure, the configuration parameters are difficult to understand, or there are insufficient configuration parameters.
- 2 – the capability is hard to configure, many of the configuration parameters are difficult to understand, or many configuration parameters are missing.
- 3 – the capability is somewhat hard to configure, some configuration parameters are difficult to understand, or several configuration parameters are missing.
- 4 – the capability is fairly easy to configure, most configuration parameters are easy to understand, and most configuration parameters are present.
- 5 – the capability is very easily configured, all configuration parameters are easy to understand, the scope of configuration is complete

(CQM-1 Page 4 and CQM-2 Page-5)

B.1.3.4.2 Understandability

The capability of the composition to enable the user to understand whether the composition is suitable, and how it can be used for particular tasks and conditions of use.

(CQM-1 Pages 2, 5)

B.1.3.4.2.1 Documentation quality

Documentation analysis

Scale 0-5.

- 0 – the capability is not documented
- 1 – the documentation is minimal
- 2 – the documentation is mostly complete
- 3 – the documentation is complete but complex
- 4 – the documentation is complete but somewhat complex
- 5 – documentation is complete and easy to understand

(CQM-1 Page 5 and CQM-2 Pages 6, 7)

B.1.3.4.3 Operability

The capability of the composition to enable the user to operate and control it.

(CQM-1 Pages 3, 5)

B.1.3.4.3.1 Parameters

of parameters

Scale 0, 3, 5.

0 – more than 10

3 – 6 to 9

5 – Fewer than 5

(CQM-1 Page 5 and CQM-2 Pages 6, 7)

B.1.3.4.3.2 Required parameters

of required interfaces

Scale 0-5.

- 0 – all parameters are optional
- 1 – greater than 75% but less than 100% of parameters are optional
- 2 – greater than 50% but less than 75 % of parameters are optional
- 3 – greater than 20% but less than 50% of parameters are optional
- 4 – greater than 0% but less than 20% of parameters are optional
- 5 – No optional parameters

(CQM-1 Page 5 and CQM-2 Pages 6, 7)

B.1.3.5 Efficiency

Express the ability of a composition to provide appropriate performance, relative to the amount of resources used. This is of interest to the composer and the end-user.

(CQM-1 Page 2)

B.1.3.5.1 Time Behavior

The capability of the composition to provide appropriate response and processing times and throughput rates when performing its function, under stated conditions.

(CQM-1 Pages 3, 4)

B.1.3.5.1.1 Response time

Time taken between a set of invocations

Scale 0, 3, 5.

- 0 – given the amount of data processed and processing performed the response time is less than expected
- 3 – given the amount of data processed and processing performed the response time is acceptable
- 5 – given the amount of data processed and processing performed the response time is better than expected

(CQM-1 Page 4 and CQM-2 Page-5)

B.1.3.5.1.2 Throughput

Amount of outputs produced with success / period of time

Scale 0, 3, 5.

- 0 – unacceptable throughput observed
- 3 – reasonable throughput observed
- 5 – high throughput observed

(CQM-1 Page 4 and CQM-2 Page-5)

B.1.3.5.2 Resource Utilization

The capability of the composition to use appropriate amounts and types of resources when performing its function under stated conditions.

(CQM-1 Pages 3, 4)

B.1.3.5.2.1 Memory usage

Memory used

Scale 0, 3, 5.

- 0 – unacceptable amount of memory utilized
- 3 – reasonable amount of memory utilized
- 5 – less than expected amount of memory utilized

(CQM-1 Page 4 and CQM-2 Pages 5, 6)

B.1.3.5.2.2 Disk usage

Disk used

Scale 0, 3, 5.

- 0 – unacceptable amount of disk space utilized
- 3 – reasonable amount of disk space utilized
- 5 – less than expected amount of disk space utilized

(CQM-1 Page 4 and CQM-2 Pages 5, 6)

B.1.3.5.2.3 Network usage

Bandwidth required

Scale 0, 3, 5.

- 0 – unacceptable network bandwidth required
- 3 – reasonable network bandwidth required
- 5 – less than expected network bandwidth required

B.1.3.5.3 Scalability

The ability of the composition to accommodate major data volumes without changing its implementation.

(CQM-1 Page 3, 4)

B.1.3.5.3.1 Processing capacity

Determine linearity of response time over a range of data set sizes.

Scale 0, 2, 5.

- 0 – composition is unable to handle larger data volumes
- 2 – composition appears to handle larger data volumes with exponential response time
- 5 – composition appears to handle larger data volumes with linear response time

(CQM-1 Page 4 and CQM-2 Pages 5, 6)

B.1.3.6 Maintainability

Express the ability of a composition to be modified. This is from the composer's perspective.

(CQM-1 Page 2)

B.1.3.6.1 Stability

The capability of the composition to avoid unexpected effects from modifications of the software.

B.1.3.6.1.1 Modifications over time

Types of modifications (fixes or enhancements) and their dates would have to be maintained by the CCOD User Environment. Stability would be measured by weighing fixes heaver than enhancements and more recent changes heavier than older ones.

Scale 0-5.

- 0 – The composition is frequently being modified mostly in response to software problems, new versions seems to present new problems
- 1 – Problems fixed are mostly software problems, new versions seem to present new problems.
- 2 – Problems fixed are mostly software problems, new versions seem to present few new problems.
- 3 – Problems fixed are an equal mix of software problems and enhancements, new versions seem to present few new problems. Use this value if maintenance data is not available.
- 4 – Problems fixed are mostly enhancements; new versions seem to present few new problems.
- 5 – The composition is very stable, versions are created on a fixed schedule, most changes are enhancements, software problems are few

(CQM-1 Pages 3, 4)

(CQM-1 Page 4 and CQM-2 Pages 5, 6)

B.1.3.6.2 Changeability

The capability of the composition to enable a specified modification to be implemented.

(CQM-1 Pages 3, 5)

B.1.3.6.2.1 Complexity level

Examine complexity.

Scale 0-5.

- 0 – Complexity greater than 50
- 1 – Complexity range of 25-49
- 2 – Complexity range of 15-24
- 3 – Complexity range of 10-14
- 4 – Complexity range of 5-9
- 5 – Complexity range of 1-4

(CQM-1 Page 5 and CQM-2 Pages 6, 7)

B.1.3.6.2.2 Cohesion

Cohesiveness is defined as a measure of the internals of module. A module is considered to be highly cohesive if the elements within it are closely related functionally. Six levels of cohesiveness are defined, from lowest to highest:

- Coincidental (random collections of elements),
- Logical (elements are related to one another logically, like all input and output operations),
- Temporal (all elements are related in time or executed in the same time period),
- Communicational (elements related to the same set of input or output data),
- Sequential (elements are related because they pass data from one element to the next),
- Functional (all elements are related to the performance of a single function.

Scale 0-5.

- 0 – Coincidental - the composition performs multiple functions that are not related in any way
- 1 – Logical - the composition performs multiple functions that are related logically (e.g., all input functions in a single composition)
- 2 – Temporal - the composition performs multiple functions which are related in time
- 3 – Communicational - the composition performs multiple functions on a single data structure or type
- 4 – Sequential - the composition performs multiple functions by passing the data from one function to the next
- 5 – Functional - the composition performs a single function (e.g. sort or filter).

(CQM-1 Page 5)

B.1.3.6.2.3 Coupling

Coupling is defined as "the measure of the strength of association established by a connection from one module to another." Further, "strong coupling complicates a system since a module is harder to understand, change or correct by itself if it is highly interrelated with other modules. Complexity can be reduced by designing systems with the weakest possible coupling between modules. Coupling depends (1) on how complicated the connection is, (2) on whether the connection refers to the module itself or something inside it, and (3) on what is being sent or received. Coupling increases with increasing complexity or obscurity of the interface. Coupling is lower when the connection is to the normal module interface than when the connection is to an internal component. Coupling is lower with data connections than with control connections, which are in turn lower than hybrid connections (modification of one module's code by another module). The complexity of an interface is a matter of how much information is needed to state or understand the connection. Thus, obvious relationships result in lower coupling than obscure or inferred ones. The more syntactic units (such as parameters) in the statement of a connection, the higher the coupling. Thus, extraneous elements irrelevant to the programmer's and the modules immediate task increase coupling unnecessarily."

Scale 0-5.

- 0 – this composition changes the implementation of another composition
- 1 – this composition uses at least one global variable
- 2 – this composition has at least one control parameter
- 3 – this composition utilizes parameters where the structure is important
- 4 – this composition only parameters where the structure is not important
- 5 – this composition is independent of all others

(CQM-1 Page 5)

B.1.3.6.3 Testability

The capability of the composition to enable modifications to be validated.

(CQM-1 Pages 3, 5)

B.1.3.6.3.1 Test cases and/or proofs provided

Evaluate the existence of test cases and proofs.

Scale 0-5.

- 0 – no test cases provided
- 1 – Less than 50% of the functions have test cases provided
- 2 – Less than 100% but more than 50% of the functions have test cases provided
- 3 – test cases are provided that provide coverage of the component
- 4 – some proofs provided and the rest covered by test cases
- 5 – proofs provided for all functions

(CQM-1 Page 5 and CQM-2 Pages 6, 7)

B.1.3.6.4 Analyzability

The capability of the composition to be diagnosed for deficiencies or causes of failures and to identify the components or compositions needing to be modified.

Scale 0&5.

- 0 – the capability is not present
- 5 – the capability is present

(ISO/IEC 9126-1 Section 6.5.1)

B.1.3.7 Portability

Express the ability of a composition to be used in an CCOD User Environment other than that in which it was originally developed.

(CQM-1 Page 2)

B.1.3.7.1 Composition Deployability

The capability of the composition without a user interface to be used outside the CCOD User Environment as a web service.

(CQM-1 Pages 3, 4, Derived from Installability from ISO/IEC 9126-1 Section 6.6.2)

B.1.3.7.1.1 Data compatibility

Analysis of the data standard (such as XML, JSON, etc.),

Scale 0, 3, 5.

- 0 – indicates no data standard used and a specific data structure is defined (e.g., a proprietary messaging system)
- 3 – A well known data standard is used with a restriction on structure or schema (e.g., XML with Cursor on Target the only schema allowed).
- 5 – A well known data standard is used with no restriction on structure or schema (e.g., XML or JSON).

(CQM-1 Page 4 and CQM-2 Page 6, 7)

B.1.3.7.1.2 Composition deployment as a web service complexity

Complexity level

Scale 0-5.

- 0 – compositions cannot be deployed as a web service
- 1 – making the composition available as a web service can be done with extreme effort and time
- 2 – making the composition available as a web service can be done with some effort and time
- 3 – making the composition available as a web service can be done with reasonable effort and time
- 4 – making the composition available as a web service can be done with little effort and time
- 5 – making the composition available as a web service is trivial and takes very little time

B.1.3.7.2 Capability Deployability

The capability of the composition with a user interface to be used outside the CCOD User Environment.

For capabilities, deployment refers to the ability of the end user to gain access to the capability outside the CCOD User Environment. This primarily includes other web pages.

(CQM-1 Pages 3,4c, Derived from Installability from ISO/IEC 9126-1 Section 6.6.2)

B.1.3.7.2.1 Data compatibility

Adherence to browser standards (such as HTML).

Scale 0, 5.

- 0 – indicates no standard used
- 5 – HTML standard is used

(CQM-1 Page 4 and CQM-2 Page 6, 7)

B.1.3.7.2.2 Capability deployment difficulty

Time needed to deploy

Scale 0-5.

- 0 – Capabilities cannot be deployed outside the CCOD User Environment
- 1 – making the capability available outside the CCOD User Environment can be done with extreme effort and time
- 2 – making the capability available outside the CCOD User Environment can be done with some effort and time
- 3 – making the capability available outside the CCOD User Environment can be done with reasonable effort and time
- 4 – making the capability available outside the CCOD User Environment can be done with little effort and time
- 5 – making the capability available outside the CCOD User Environment is trivial and takes very little time

(CQM-1 Page 4 and CQM-2 Pages 5, 6)

B.1.3.8 Quality in Use

These measures are provided by users and provide an assessment of the composition during use.

B.1.3.8.1 Composition Effectiveness

The capability of the composition to enable users to achieve specified goals with accuracy and completeness in another composition.

User Question: Does the composition do what you thought it should?

User feedback

Scale 0-5.

- 0 – not effective, mostly useless and inaccurate
- 1 – mostly ineffective, meets few needs accurately
- 2 – somewhat ineffective, meets some needs accurately
- 3 – effective, meets only critical needs accurately
- 4 – mostly effective, meets most user needs accurately
- 5 – completely effective, meets all user needs accurately

(ISO/IEC 9126-1 Section 7.1.1)

B.1.3.8.2 Capability Effectiveness

The ability of the capability to enable users to achieve specified goals with accuracy and completeness in a specified operational context.

User Question: Does the capability do what you thought it should?

User feedback

Scale 0-5.

- 0 – not effective, mostly useless and inaccurate
- 1 – mostly ineffective, meets few needs accurately
- 2 – somewhat ineffective, meets some needs accurately
- 3 – effective, meets only critical needs accurately
- 4 – mostly effective, meets most user needs accurately
- 5 – completely effective, meets all user needs accurately

(ISO/IEC 9126-1 Section 7.1.1)

B.1.3.8.3 Capability Productivity

The capability of the composition to enable users to expend appropriate amounts of resources in relation to the effectiveness achieved in a specified context of use.

End User: Given the expended resources, was it worth it?

User feedback

Scale: 0-5.

- 0 – the effort to use this composition is so great that the user would rather not use it
- 1 – the effort to use the composition is much more than the return provided, the user solves only the most valuable problems using the composition
- 2 – the effort to use the composition is more than the return provided, the user solves only valuable problems using the composition
- 3 – the effort to use the composition is consistent with value returned, the user comes back regularly to solve most problems
- 4 – the effort to use the composition is less than the return provided, the user solves all day-to-day problems
- 5 – the effort to use the composition is so low that the user contemplates new problems to solve using the composition.

(ISO/IEC 9126-1 Section 7.1.2)

B.1.3.8.4 Capability Satisfaction

The capability of the composition to satisfy user in a specified context of use.

User feedback

Scale 0-5.

- 0 – totally unsatisfied
- 1 – mostly unsatisfied
- 2 – somewhat unsatisfied
- 3 – somewhat satisfied
- 4 – mostly satisfied
- 5 – completely satisfied

(ISO/IEC 9126-1 Section 7.1.4)

B.1.3.8.5 Capability Attractiveness

The level at which the capability is attractive to the user.

This refers to attributes of the software intended to make the capability more attractive to the user, such as the use of color and the nature of the graphical design.

User Feedback

Scale 0-5.

- 0 – The capability user interface is very difficult to use and unattractive
- 1 – the capability user interface is mostly difficult to use and unattractive
- 2 – the capability user interface is somewhat difficult to use and unattractive
- 3 – the capability user interface is somewhat easy to use and attractive
- 4 – the capability user interface is mostly easy to use and attractive
- 5 – the capability user interface is very easy to use and attractive

(ISO/IEC 9126-1 Section 6.3.4)

B.1.4 CCOD Component/Composition Maintainer

The CCOD component/composition maintainer can affect the trustworthiness of the components and compositions that are maintained. More proficient maintainers should produce more trustworthy CCOD products.

B.1.4.1 Proficiency

B.1.4.1.1 Training performed

Scale 0&5.

- 0 – training has not been performed
- 5 – training has been performed

B.1.4.1.2 Average complexity of components maintained

Given that complexity is one of the metrics gathered for the component, the intent here is to determine the average complexity of components modified by this maintainer. If a maintainer modifies a component that is significantly more complex than those modified in the past, then it is assumed that the new component version may not be as trust worthy as those that came before. "Significant" could be two sigma from the norm.

Scale 0-5. Based on the complexity values provided for components

B.1.4.1.3 Average complexity of compositions maintained

Given that complexity is one of the metrics gathered for the composition, the intent here is to determine the average complexity of compositions modified by this maintainer. If a maintainer modifies a composition that is significantly more complex than those modified in the past, then it is assumed that the new composition version may not be as trust worthy as those that came before. "Significant" could be two sigma from the norm.

Scale 0-5. Based on the complexity values provided for compositions

B.1.5 CCOD Component Developer

The CCOD component developer can affect the trustworthiness of the components developed. More proficient developers should produce more trustworthy components.

B.1.5.1 Proficiency

B.1.5.1.1 Training performed

Scale 0&5.

- 0 – training has not been performed
- 5 – training has been performed

B.1.5.1.2 Average complexity of developed components

Given that complexity is one of the metrics gathered for the component, the intent here is to determine the average complexity of components developed by this developer. If the next component developed is significantly more complex than those developed in the past, then it is assumed that the new component may not be as trust worthy as those that came before. "Significant" could be two sigma from the norm.

Scale 0-5. Based on the complexity values provided for components

B.1.6 CCOD Composer

The CCOD composer can affect the trustworthiness of the compositions. More proficient composers should produce more trustworthy compositions.

B.1.6.1 Proficiency

B.1.6.1.1 Training performed

Scale 0&5.

- 0 – training has not been performed
- 5 – training has been performed

B.1.6.1.2 Average complexity of developed compositions

Given that complexity is one of the metrics gathered for the composition, the intent here is to determine the average complexity of compositions created by this composer. If the next composition composed is significantly more complex than those developed in the past, then it is assumed that the new composition may not be as trust worthy as those that came before. "Significant" could be two sigma from the norm.

Scale 0-5. Based on the complexity values provided for compositions

B.2 Change rationale

B.2.1 Changes/Additions to Taxonomy from Reference Documents

This section of the document identifies tailoring performed against the CQM and ISO 9126 Software Quality standard to support the concepts of CCOD. For each change made a rationale is provided. Subsections are provided for changes and additions to each of the following CCOD concepts: Components, Compositions and Capabilities. Throughout this section elements of taxonomy are identified using paths that represent the location in the taxonomy. Paths are written with the following convention Characteristic/Subcharacteristic/Attribute. If more than one element (characteristic, subcharacteristic or attribute) needs to be referenced they will be separated by semi-colons.

B.2.1.1 For CCOD User Environment

B.2.1.1.1 Functionality/Interoperability made into two subcharacteristics (Internal and External Interoperability)

The CCOD User Environment should support two forms of interoperability. Internal interoperability refers to the ability of components and compositions defined within the CCOD User Environment to interoperate. External interoperability refers to the ability of the CCOD User Environment to support capabilities defined within, to function in other environments. We feel it is necessary to measure these concepts separately.

B.2.1.1.2 Functionality/Internal Interoperability/Interoperation complexity added

We added this measure to account for the level of difficulty to make components and compositions interoperate. Data standards help to make the components and compositions interoperate but we feel it is important to measure the level of effort that a non-programmer would expend in making components and compositions interoperate. Note that in most cases the CCOD User Environment should provide an automated mechanism for component/composition interoperability. If that is the case this concept is measured here. If the CCOD User Environment does not provide automatic interoperability the user must make the components and compositions interoperate manually. In that case this concept is measured in that part of the taxonomy.

B.2.1.1.3 Functionality/Security moved up one level and redefined

Security was moved from a sub-characteristic of functionality to become a characteristic of its own (the highest level of the taxonomy). We feel that security is equal in importance to the other characteristic and deserved to be promoted. In addition, we redefined the sub-characteristics of security to contain the concepts of security functions and software secureness. Security functions are derived from NIST 800-53 –"Recommended Security Controls for Federal Information Systems and Organizations" while software secureness is derived from the CWE/SANS (Common Weakness Enumeration/SysAdmin, Audit, Network, Security) top 25 Most Dangerous Software Errors.

B.2.1.1.4 Usability/Learnability/Quality of Training Materials; Common Development Language Used added

CQM only defined the attribute "Time and effort to use" under Learnability. We chose to not use this measure and define two new attributes that are easier to quantify for the CCOD User Environment. We feel that training materials should be provided to help users understand how to

compose capability. Common languages support learnability by not forcing component developers to learn completely new languages.

B.2.1.1.5 Usability/Attractiveness moved to Quality in Use

Quality in Use represents all user feedback for the CCOD User Environment, as such, Attractiveness is moved to Quality in Use.

B.2.1.1.6 Usability/Discoverability of components and compositions added

We feel that the CCOD User Environment must provide a mechanism for composers to discover what components and compositions are available for composition. This concept only applies to the CCOD User Environment.

B.2.1.1.7 Usability/Discoverability of capabilities added

We feel that the CCOD User Environment must provide a mechanism for end users to discover what capabilities are available for use. This concept only applies to the CCOD User Environment.

B.2.1.1.8 Reliability/Recoverability/Error Handling removed and replaced with "Mechanism Available" and "Mechanism Efficiency" attributes

Error handling is not sufficient to measure recoverability. A recovery mechanism must be in place and it should recover from most problems encountered. This concept is similar to that which is used for Fault tolerance. As a result the measure "Error Handling" is removed and replaced with two measures "Mechanism Available" and "Mechanism Efficiency."

B.2.1.1.9 Efficiency/Resource Utilization/Network Usage attribute added

Network usage will be significant to Resource Utilization as Memory usage and Disk usage

B.2.1.1.10 Maintainability/Stability/Modifiability changed to Modifications over time

We decided that it would be very difficult to determine the modifiability of the CCOD User Environment, especially based on the definition of modifiability given in the reference. We chose to measure the stability of the CCOD User Environment based on the number of changes made over a time period. As a result the measure called "Modifiability" is removed and replaced with "Modifications over time."

B.2.1.1.11 Maintainability/Testability of components & compositions/Test Suite Capability added

This attribute measures the ability of the CCOD User Environment to support test suites for components and compositions. We feel that a CCOD User Environment, so equipped, has benefits over those that don't have this ability. This concept only applies to the CCOD User Environment.

B.2.1.1.12 Portability redefined

For the CCOD User Environment we are only interested in the ability of the environment to run in more than one physical environment, further details are not important. All sub-characteristics are replaced by a simple metric.

B.2.1.1.13 Concepts from the SOA Management Model [SOA-RA-02] added to CCOD User Environment Taxonomy

The SOA Management Model describes activities that should be present to properly manage a service oriented environment. A CCOD User Environment is a specific type of service oriented environment; therefore, we feel these concepts should be present in the CCOD User Environment. SOA-RA-02 describes the following management characteristics: Life-Cycle Manageability, Configuration Manageability, Event Monitoring Manageability, Accounting Manageability, Quality of Service Manageability, Business Performance Manageability, and Policy Manageability.

All but "Business Performance Manageability" has been included. This characteristic deals with business-level service level agreements. We feel that this concept does not apply to a CCOD User Environment whose purpose is to provide end users the ability to compose capability. Business-level SLAs would not come into play in such an environment.

We chose to distribute the remaining characteristics as sub-characteristics within the ISO 9126 Software Quality Taxonomy in the follow way: Life-Cycle Manageability, Configuration Manageability and Policy Manageability was placed under the Functionality characteristic. We feel that these represent basic functions of the CCOD User Environment.

Event Monitoring Manageability was placed under the Reliability characteristic. We feel that event monitoring contributes to the reliability of the CCOD User Environment.

Accounting Manageability and Quality of Service Manageability placed under the Efficiency characteristic. We feel that these concepts best fit under the characteristic that deals with the run-time measurement of the CCOD User Environment.

These management concepts apply to the CCOD User Environment only, they do not apply to components or compositions.

B.2.1.1.14 Quality in Use

The CQM references mention and extol the virtues of the ISO 9126 concept of Quality in Use but for some reason exclude it from the CQM model. We explicitly retain Quality in Use to provide a set of user feedback measures that contribute to the trustworthiness of the CCOD User Environment. In addition, to maintain consistency, we have moved the Attractiveness measure originally defined in ISO 9126 under the Usability section to Quality in Use. Attractiveness was the only other user feedback measure identified in ISO 9126 that was, for some reason, not included with the Quality in Use concept.

B.2.1.2 For CCOD Component

B.2.1.2.1 Functionality/Internal Interoperability/Interface complexity added

We added this measure to account for the level of difficulty to make components and compositions interoperate. Data standards help to make the components and compositions interoperate but we feel it is important to measure the level of effort that a non-programmer would expend in making components and compositions interoperate. We choose to measure this by examining the complexity of the component's interface. Note that in most cases the CCOD User Environment should provide an automated mechanism for component/composition interoperability. If that is the case, this concept is measured in that part of the taxonomy. If the CCOD User Environment does not provide automatic interoperability the user must make the components and compositions interoperate manually. In that case this concept is measured here.

B.2.1.2.2 Functionality/Security moved up one level and redefined

Security was moved from a sub-characteristic of functionality to become a characteristic of its own (the highest level of the taxonomy). We felt that security is equal in importance to the other characteristic and deserved to be promoted. In addition, we redefined the sub-characteristics of security to consist of the concept software secureness, derived from the CWE/SANS (Common Weakness Enumeration/SysAdmin, Audit, Network, Security) top 25 Most Dangerous Software Errors. Security functions, as defined in the CCOD User Environment section of the taxonomy, are not needed for the component, they are provided through the CCOD User Environment.

B.2.1.2.3 Reliability/Recoverability/Error Handling removed and replaced with "Mechanism available" and "Mechanism efficiency" attributes

Error handling is not sufficient to measure recoverability. A recovery mechanism must be in place and it should recover from most problems encountered. This concept is similar to that which is used for Fault tolerance. As a result the measure, "Error Handling" is removed and replaced with two measures "Mechanism Available" and "Mechanism Efficiency."

B.2.1.2.4 Usability/Operability/Complexity level moved to Maintainability/Changeability

The complexity of the component's internals measures the ability of a maintainer to change the component more than it is a measure of the operability of the component.

B.2.1.2.5 Usability/Operability/Interfaces; Required Interfaces renamed to Parameters; Required Parameters

"Parameters" are used here rather than "interfaces" as described in the referenced text to support our definition of a component which has only one interface. In order to measure operability we focus on the numbers of parameters defined in the interface. As a result the measures "Interfaces" and "Required Interfaces" are removed and replaced with the measures "Parameters" and "Required Parameters."

B.2.1.2.6 Efficiency/Resource Utilization/Network Usage added

Network usage will be significant to Resource Utilization as Memory usage and Disk usage

B.2.1.2.7 Maintainability/Stability/Modifiability changed to Modifications over time

We decided that it would be very difficult to determine the modifiability of the component, especially based on the definition of modifiability given in the reference. We chose to measure the stability of the component based on the number of changes made over a time period. As a result the measure called "Modifiability" is removed and replaced with "Modifications over time."

B.2.1.2.8 Maintainability/Changeability/Complexity Level added

Complexity is a commonly obtained metric for software. We added this to support the notion that complexity of a component will directly affect its maintainability.

B.2.1.2.9 Maintainability/Testability/Extensive Test Cases; Proofs Provided combined into a single measure "Test cases and/or proofs provided"

The existence of proofs negates the need for test cases. As a result we combined these two attributes so a single measure could be used where the presence of proofs is the best case and coverage tests is very good and no test cases is bad. As a result the measures called "Extensive

Test Cases" and "Proofs Provided" are removed and replaced with "Test cases and/or proofs provided."

B.2.1.2.10 Portability/Reusability/Coupling moved to Maintainability/Changeability

The portability of a component outside the CCOD User Environment in which it runs is not considered a requirement, so the portability characteristic is removed from the taxonomy. Coupling, however, is still an important measure when considering a maintainer's ability to make changes to a component. Looser coupling will result in easier maintenance.

B.2.1.2.11 Portability/Reusability/Cohesion moved to Maintainability/Changeability

The portability of a component outside the CCOD User Environment in which it runs is not considered a requirement, so the portability characteristic is removed from the taxonomy. Cohesion, however, is still an important measure when considering a maintainer's ability to make changes to a component. Higher cohesion will result in easier use of the maintenance.

B.2.1.2.12 Quality in Use

The CQM references mention and extol the virtues of the ISO 9126 concept of Quality in Use but for some reason exclude it from the CQM model. We explicitly retain Quality in Use to provide a set of user feedback measures that contribute to the trustworthiness of components.

B.2.1.3 For CCOD Composition

B.2.1.3.1 Functionality/Interoperability made into Internal Interoperability and Capability Interoperability

Needed to deal with two distinct forms of interoperability one for compositions - of primary interest of the composer and one for capabilities - of primary interest of the end user. These concepts are independent of one another and must be measured separately.

B.2.1.3.2 Functionality/Capability Interoperability/Complexity Level added

We added this measure to account for the level of difficulty to make compositions interoperate. Data standards help to make the compositions interoperate but we feel it is important to measure the level of effort that a non-programmer would expend in making compositions interoperate. Note that in most cases the CCOD User Environment should provide an automated mechanism for composition interoperability. If that is the case this concept is measured in that part of the taxonomy. If the CCOD User Environment does not provide automatic interoperability the user must make the compositions interoperate manually. In that case this concept is measured here.

B.2.1.3.3 Reliability/Recoverability/Error Handling removed and replaced with "Mechanism available" and "Mechanism efficiency" attributes

Error handling is not sufficient to measure recoverability. A recovery mechanism must be in place and it should recover from most problems encountered. This concept is similar to that which is used for Fault tolerance. As a result the measure "Error Handling" is removed and replaced with two measures "Mechanism Available" and "Mechanism Efficiency."

B.2.1.3.4 Usability made into Composition Usability and Capability Usability

Needed to deal with two distinct forms of usability one for compositions of primary interest of the composer and one for capabilities of primary interest of the end user. These concepts are independent of one another and must be measured separately.

B.2.1.3.5 Usability/Operability/Interfaces; Required Interfaces renamed to Parameters; Required Parameters

"Parameters" are used here rather than "interfaces" as described in the referenced text to support our definition of a component which has only one interface. In order to measure operability we focus on the numbers of parameters defined in the interface. As a result the measures "Interfaces" and "Required Interfaces" are removed and replaced with the measures "Parameters" and "Required Parameters."

B.2.1.3.6 Capability Usability/Operability/Complexity level moved to Maintainability/Changeability

The complexity of the composition's internals measures the ability of a maintainer to change the composition more than it is a measure of the operability of the composition.

B.2.1.3.7 Usability/Attractiveness moved to Quality in Use

Quality in Use represents all user feedback for the CCOD User Environment, as such, Attractiveness is move to Quality in Use. Attractiveness only applies to Capabilities.

B.2.1.3.8 Efficiency/Resource Utilization/Network Usage added

Network usage will be significant to Resource Utilization as Memory usage and Disk usage

B.2.1.3.9 Maintainability/Stability/Modifiability changed to Modifications over time

We decided that it would be very difficult to determine the modifiability of the composition, especially based on the definition of modifiability given in the reference. We chose to measure the stability of the composition based on the number of changes made over a time period. As a result the measure called "Modifiability" is removed and replaced with "Modifications over time."

B.2.1.3.10 Maintainability/Changeability/Complexity Level added

Complexity is a commonly obtained metric for software. We added this to support the notion that complexity of a composition will directly affect its maintainability. Composition complexity is a measure of its number of components and branches.

B.2.1.3.11 Maintainability/Testability/Extensive Test Cases; Proofs Provided combined into a single attribute measure "Test cases and/or proofs provided"

The existence of proofs negates the need for test cases. As a result we combined these two attributes so a single measure could be used where the presence of proofs is the best case and coverage tests are very good and no test cases are bad. As a result the measures called "Extensive Test Cases" and "Proofs Provided" are removed and replaced with "Test cases and/or proofs provided."

B.2.1.3.12 Portability redefined

Compositions, as we define them, only exist within the CCOD User Environment in which they were composed. Compositions should, however, be able to be executed within the CCOD User

Environment and their results used outside the environment either as web services (for compositions) or by URL reference (for capabilities). Therefore, our concept of portability for compositions is limited to the ability to deploy **compositions** as web services and **capabilities** via URL. We have defined portability of compositions with two sub-characteristics "Composition Deployability" and "Capability Deployability." In both cases we have defined metrics for the following:

For adherence to common web based data standards to assure that the composition as a web service and the capability used outside the CCOD User Environment can properly interoperate in a web/browser based environment, and

A metric the measures the ease of deployment (as a web service, for composition or via URL, for capabilities) with a zero value indicating the ability to deploy as a web service or via URL is not available.

We feel these two metrics form an adequate basis to measure the portability of compositions as we have it defined. All sub-characteristics of portability found in our reference materials for component based development (**Adaptability, Installability, Co-existence, Replaceability, Portability compliance** and **Reusability**) don't apply to compositions, as they are not meant to be portable in the same sense as software components.

B.2.1.3.13 Portability/Reusability/Coupling moved to Maintainability/Changeability

Coupling is an important measure when considering a maintainer's ability to make changes to a composition. Looser coupling will result in easier maintenance of the composition.

B.2.1.3.14 Portability/Reusability/Cohesion moved to Maintainability/Changeability

Cohesion is an important measure when considering a maintainer's ability to make changes to a composition. Higher cohesion will result in easier maintenance of the composition.

B.2.1.3.15 Quality in Use

The CQM references mention and extol the virtues of the ISO 9126 concept of Quality in Use but for some reason exclude it from the CQM model. We explicitly retain Quality in Use to provide a set of user feedback measures that contribute to the trustworthiness of compositions. In addition, to maintain consistency, we have moved the Attractiveness measure originally defined in ISO 9126 under the Usability section to Quality in Use. Attractiveness was the only other user feedback measure identified in ISO 9126 that was, for some reason, not included with the Quality in Use concept.

Compositions that are not end user capabilities can be evaluated by composers for effectiveness only, therefore all other Quality in Use measures are excluded for compositions. However, compositions that are end user capabilities can be evaluated by end users in for: effectiveness (is the capability useful?); productivity (is the amount of effort needed to use the capability consistent with the gain achieved by using it?); attractiveness (does it have a good user interface); and the overall level of user satisfaction.

B.2.2 Elements not used from Reference Documents

This section of the document identifies tailoring performed against the CQM and ISO 9126 Software Quality standard to support the concepts of CCOD. For each item removed a rationale is provided. Subsections are provided for each of the following CCOD concepts: Components, Compositions and Capabilities. Throughout this section elements of taxonomy are identified

using paths that represent the location in the taxonomy. Paths are written with the following convention Characteristic/Subcharacteristic/Attribute. If more than one element (characteristic, subcharacteristic or attribute) needs to be referenced they will be separated by semi-colons.

B.2.2.1 For CCOD User Environment

B.2.2.1.1 Functionality/Suitability/Pre and Post Conditioned; Proofs of Pre and Post Conditions

These were defined as attributes of components by the reference document. They do not apply to the CCOD User Environment as the environment is not intended to be a component.

B.2.2.1.2 Functionality/Self Contained

This subcharacteristic was defined by the CQM reference document to measure this for components. It does not apply to the CCOD User Environment as the environment is not intended to be a component.

B.2.2.1.3 Usability/Understandability/Documentation Available

The documentation quality metric (in the same category) covers this metric. If the documentation is not present then the documentation quality is considered "zero."

B.2.2.1.4 Usability/Learnability/Time and effort to (use, configure and expertise needed)

The learnability of the CCOD User Environment will be measured by the quality of training materials and the use of a common component development language. Actual measurement of time and effort to use is difficult in the CCOD User Environment.

B.2.2.1.5 Usability/Operability/Provided Interfaces; Required Interfaces

Interfaces are not considered important to the CCOD User Environment as they do not provide a measure of operability (as they would for components and compositions).

B.2.2.1.6 Usability/Operability/Effort for operating

This is a remnant of the component quality model that measures the effort to use an interface. The CCOD User Environment is a system, not a component. The effort to use the CCOD User Environment will be measured in the complexity section of the CCOD User Environment.

B.2.2.1.7 Efficiency/Time Behavior/Latency - Processing Capacity

Amount of inputs produced with success / period of time

There only needs to be one concept of latency, throughput.

(CQM-1 Page 4 and CQM-2 Pages 5, 6)

B.2.2.1.8 Efficiency/Scalability

Scalability is measured within the composition part of the taxonomy. It does not need to be measured twice.

B.2.2.1.9 Maintainability/Analyzability

The capability of the CCOD User Environment to be diagnosed for deficiencies or causes of failures in the software, or for the parts to be modified to be identified. (ISO/IEC 9126-1 Section 6.5.1)

It is assumed the CCOD User Environment will be maintained by a third party (it could be a COTS or GOTS product). It is outside the purview of the users of the CCOD User Environment to require this concept.

B.2.2.1.10 Maintainability/Changeability/Extensibility

Extensions to the CCOD User Environment will not be performed by the users of the environment; therefore, this attribute is not needed.

B.2.2.1.11 Maintainability/Testability/Extensive component test cases; Proofs

Testability for the CCOD User Environment measures the presence of the ability to provide test cases for compositions and components. The need for test cases and proofs for the CCOD User Environment is not needed. These are measured for compositions and components.

B.2.2.1.12 Maintainability/Testability/Environment test cases

We don't expect the CCOD User Environment to expose the test cases used to determine the portability of the CCOD User Environment. Portability will be measure on its own.

B.2.2.1.13 Quality in Use/Safety

The capability of the component to achieve acceptable levels of risk of harm to people, business, software, property or the environment in a specified context of use. (ISO/IEC 9126-1 Section 7.1.3)

Components are not executed stand alone in any way that safety would be an issue. Other quality metrics associated with components are to be combined to define a level of quality for each component. Components will be combined to form compositions and compositions combined to form capabilities. Safety could be a concern for capabilities but we leave it to the end user to determine the level of risk that is acceptable given the current situation. The overall quality metric for the capability is intended to assist the user in making the risk decision.

B.2.2.1.14 Marketability

Express the marketing characteristics of a component, complementing the quality characteristics of the model. (CQM-1 Page 2 and CQM-2 Page 4)

Marketing/business characteristics not considered relevant for trustworthiness of the CCOD User Environment.

B.2.2.1.15 Business performance manageability

The business need for a composition is only known by the composer or the end-user that requested it. It would be exceptional difficult to define a management construct in this area.

(SOA-RA-02, Section 5.3 Management Model)

B.2.2.2 For CCOD Component

B.2.2.2.1 Reliability/Reliability Compliance

The capability of the component to adhere to standards, conventions or regulations relating to reliability. (ISO/IEC 9126-1 Section 6.2.4)

The "reliability compliance" of components will not be measured directly. Instead, using the other reliability characteristics and metrics we will measure the reliability of the component.

User defined policies can then be applied to determine if the component's reliability "complies" with the stated policies. We have less interest in the components' adherence to "standards, conventions or regulations" stated in general terms by non-users of the components than actual user defined policies in the area of reliability.

B.2.2.2.2 Usability/Understandability/Documentation available

The documentation quality metric (in the same category) covers this metric. If the documentation is not present then the documentation quality is considered "zero."

B.2.2.2.3 Usability/Learnability

The capability of the component to enable the composer to learn its application. (CQM-1 Pages 3, 5)

Removed in favor of Understandability. Documentation should include all forms that make the component understandable.

B.2.2.2.4 Usability/Attractiveness

The capability of the component to be attractive to the user. (ISO/IEC 9126-1 Section 6.3.4)

Attractiveness of a component is replaced with the notion of discoverability as a characteristic of the composition development environment.

B.2.2.2.5 Usability/Usability Compliance

The capability of the component to adhere to standards, conventions, style guides or regulations relating to usability. (ISO/IEC 9126-1 Section 6.3.5)

The "usability compliance" of components will not be measured directly. Instead, using the other usability characteristics and metrics we will measure the usability of the component. User defined policies can then be applied to determine if the component's usability "complies" with the stated policies. We have less interest in the components' adherence to "standards, conventions, style guides or regulations" stated in general terms by non-users of the components than actual user defined policies in the area of usability.

B.2.2.2.6 Efficiency/Time Behavior/Latency - Processing capacity

Amount of inputs produced with success / period of time (CQM-1 Page 4 and CQM-2 Pages 5, 6)

There only needs to be one concept of latency, throughput. For components, there is a direct correlation between inputs and outputs, if you measure the output of component you have effectively measured the input.

B.2.2.2.7 Efficiency/Efficiency Compliance

The capability of the component to adhere to standards and conventions relating to efficiency. The "efficiency compliance" of components will not be measured directly. Instead, using the other efficiency characteristics and metrics we will measure the efficiency of the component. User defined policies can then be applied to determine if the component's efficiency "complies" with the stated policies. We have less interest in the components' adherence to "standards and conventions" stated in general terms by non-users of the components than actual user defined policies in the area of efficiency.

B.2.2.2.8 Maintainability/Changeability/Customizability

of parameters to configure the provided interface / # of interfaces

We have chosen to measure the changeability of a component by measuring its software complexity, coupling and cohesion. Other measures are considered redundant and not useful.

B.2.2.2.9 Maintainability/Changeability/Extensibility

% of the functionalities that could be extended

We have chosen to measure the changeability of a component by measuring its software complexity, coupling and cohesion. Other measures are considered redundant and not useful.

B.2.2.2.10 Maintainability/Testability/Component tests

of environments that the component was tested. CQM-1 Page 5 and CQM-2 Pages 6, 7)

Components are expected to run only within the CCOD User Environment in which they were developed.

B.2.2.2.11 Maintainability/Analyzability

The capability of the component to be diagnosed for deficiencies or causes of failures in the software, or for the parts to be modified to be identified. (ISO/IEC 9126-1 Section 6.5.1)

It is assumed that all other quality metrics of components will be obtainable, therefore the ability to analyze the component is a given.

B.2.2.2.12 Maintainability/Maintainability Compliance

The capability of the component to adhere to standards and conventions relating to maintainability. (ISO/IEC 9126-1 Section 6.5.5)

The "maintainability compliance" of components will not be measured directly. Instead, using the other maintainability characteristics and metrics we will measure the maintainability of the component. User defined policies can then be applied to determine if the component's maintainability "complies" with the stated policies. We have less interest in the components' adherence to "standards and conventions" stated in general terms by non-users of the components than actual user defined policies in the area of maintainability.

B.2.2.2.13 Portability

Express the ability of a component to be used in an environment other than that in which it was originally developed. (CQM-1 Page 2)

The portability of a component outside the CCOD User Environment in which it runs is not considered a requirement, so the portability characteristic is removed from the taxonomy.

B.2.2.2.14 Marketability

Express the marketing characteristics of a component, complementing the quality characteristics of the model. (CQM-1 Page 2 and CQM-2 Page 4)

Marketing/business characteristics not considered relevant for trustworthiness for components in CCOD User Environment.

B.2.2.2.15 Quality In Use/Productivity

The capability of the component to enable users to expend appropriate amounts of resources in relation to the effectiveness achieved in a specified context of use.

Users of components (composers) are not concerned with the productivity of the component. Other metrics will measure the resources used. Only end-users are concerned with productivity, so this attribute will be captured in the composition (capability) metrics.

B.2.2.2.16 Quality in Use/Safety

The capability of the component to achieve acceptable levels of risk of harm to people, business, software, property or the environment in a specified context of use. (ISO/IEC 9126-1 Section 7.1.3)

Components are not executed stand alone in any way that safety would be an issue. Other quality metrics associated with components are to be combined to define a level of quality for each component. Components will be combined to form compositions and compositions combined to form capabilities. Safety could be a concern for capabilities but we leave it to the end user to determine the level of risk that is acceptable given the current situation. The overall quality metric for the capability is intended to assist the user in making the risk decision.

B.2.2.2.17 Quality In Use/Satisfaction

The capability of the component to satisfy user in a specified context of use.

Users of components (composers) are not concerned with satisfaction. Other metrics will measure the component's functionality. Only end-users are concerned with satisfaction, so this attribute will be captured in the composition (capability) metrics.

B.2.2.3 For CCOD Composition

B.2.2.3.1 Functionality/Suitability/Proofs of Pre and Post Conditions

Proof of pre and post conditions for compositions are not likely and may not be possible. Compositions are collections of components held together by the CCOD User Environment. The ability to prove pre and post conditions is not likely in such an arrangement without proofs for the entire environment.

B.2.2.3.2 Functionality/Functionality Compliance/Certification

Certifications for compositions are not likely and may not be possible. Compositions are collections of components held together by the CCOD User Environment. The ability to certify a composition is not likely in such an arrangement without certifications for the entire environment.

B.2.2.3.3 Functionality/Security

Compositions rely on the security functions and software secureness of the CCOD User Environment in which they run as well as the software secureness of the components from which they are composed, therefore, security is not an attribute of composition, rather, security attributes are found in the CCOD User Environment and component sections of the taxonomy.

B.2.2.3.4 Reliability/Reliability Compliance

The capability of the composition to adhere to standards, conventions or regulations relating to reliability. (ISO/IEC 9126-1 Section 6.2.4)

The "reliability compliance" of compositions will not be measured directly. Instead, using the other reliability characteristics and metrics we will measure the reliability of the composition. User defined policies can then be applied to determine if the composition's reliability "complies" with the stated policies. We have less interest in the compositions' adherence to "standards, conventions or regulations" stated in general terms by non-users of the compositions than actual user defined policies in the area of reliability.

B.2.2.3.5 Usability/Understandability/Documentation available

The documentation quality metric (in the same category) covers this metric. If the documentation is not present then the documentation quality is considered "zero."

B.2.2.3.6 Composition Usability; Capability Usability/Learnability

The capability of the composition/capability to enable the composer/end user to learn its application.

Removed in favor of Understandability. Documentation should include all forms that make the component understandable.

B.2.2.3.7 Usability/Usability Compliance

The capability of the composition to adhere to standards, conventions, style guides or regulations relating to usability. (ISO/IEC 9126-1 Section 6.3.5)

The "usability compliance" of compositions will not be measured directly. Instead, using the other usability characteristics and metrics we will measure the usability of the composition. User defined policies can then be applied to determine if the composition's usability "complies" with the stated policies. We have less interest in the compositions' adherence to "standards, conventions, style guides or regulations" stated in general terms by non-users of the compositions than actual user defined policies in the area of usability.

B.2.2.3.8 Capability Usability/Operability/Effort for operating

Operations in all provided interfaces / total of the provided interfaces

This measure is redundant with Understandability when you get to the capability level.

B.2.2.3.9 Composition Usability/Attractiveness

The capability of the composition to be attractive to the user. (ISO/IEC 9126-1 Section 6.3.4)

Attractiveness of a composition from the perspective of the composer is not of interest. This measure is concerned with the attractiveness from an end user's perspective and exists only under Capability Usability.

B.2.2.3.10 Efficiency/Time Behavior/Latency - Processing capacity

Amount of inputs produced with success / period of time (CQM-1 Page 4 and CQM-2 Pages 5, 6)

There only needs to be one concept of latency, throughput. For components, there is a direct correlation between inputs and outputs, if you measure the output of component you have effectively measured the input.

B.2.2.3.11 Efficiency/Efficiency Compliance

The capability of the composition to adhere to standards and conventions relating to efficiency. (ISO/IEC 9126-1 Section 6.4.3)

The "efficiency compliance" of compositions will not be measured directly. Instead, using the other efficiency characteristics and metrics we will measure the efficiency of the composition. User defined policies can then be applied to determine if the composition's efficiency "complies" with the stated policies. We have less interest in the compositions' adherence to "standards and conventions" stated in general terms by non-users of the compositions than actual user defined policies in the area of efficiency.

B.2.2.3.12 Maintainability/Changeability/Extensibility

% of the functionalities that could be extended

We have chosen to measure the changeability of a composition by measuring its complexity, coupling and cohesion. Other measures are considered redundant and not useful.

B.2.2.3.13 Maintainability/Changeability/Customizability

of parameters to configure the provided interface / # of interfaces

We have chosen to measure the changeability of a composition by measuring its complexity, coupling and cohesion. Other measures are considered redundant and not useful.

B.2.2.3.14 Maintainability/Maintainability Compliance

The capability of the composition to adhere to standards and conventions relating to maintainability. (ISO/IEC 9126-1 Section 6.5.5)

The "maintainability compliance" of compositions will not be measured directly. Instead, using the other maintainability characteristics and metrics we will measure the maintainability of the composition. User defined policies can then be applied to determine if the composition's maintainability "complies" with the stated policies. We have less interest in the compositions' adherence to "standards and conventions" stated in general terms by non-users of the compositions than actual user defined policies in the area of maintainability.

B.2.2.3.15 Maintainability/Testability/Environment test cases

of environments that the composition was tested. CQM-1 Page 5 and CQM-2 Pages 6, 7)

Compositions are expected to run only within the CCOD User Environment in which they were created. Even as web services or capabilities they will still run within the environment they were developed.

B.2.2.3.16 Marketability

Express the marketing characteristics of a composition, complementing the quality characteristics of the model. (CQM-1 Page 2 and CQM-2 Page 4)

Marketing/business characteristics not considered relevant for trustworthiness for compositions in CCOD User Environment.

B.2.2.3.17 Quality In Use/Productivity of compositions (not capabilities)

The capability of the composition to enable users to expend appropriate amounts of resources in relation to the effectiveness achieved in a specified context of use.

Users of compositions (composers) are not concerned with the productivity of the composition. Other metrics will measure the resources used. Only end-users are concerned with productivity, so this attribute will be captured in the composition (capability) metrics.

B.2.2.3.18 Quality In Use/Safety

The capability of the composition to achieve acceptable levels of risk of harm to people, business, software, property or the environment in a specified context of use. (ISO/IEC 9126-1 Section 7.1.3)

Compositions are not executed stand alone in any way that safety would be an issue. Other quality metrics associated with compositions are to be combined to define a level of quality for each composition. Compositions will be combined to form capabilities. Safety could be a concern for capabilities but we leave it to the end user to determine the level of risk that is acceptable given the current situation. The overall quality metric for the capability is intended to assist the user in making the risk decision.

B.2.2.3.19 Quality In Use/Satisfaction for compositions (not capabilities)

The capability of the composition to satisfy user in a specified context of use.

Users of compositions (composers) are not concerned with satisfaction. Other metrics will measure the component's functionality. Only end-users are concerned with satisfaction, so this attribute will be captured in the composition (capability) metrics.

B.3 Taxonomy References

ISO/IEC 9126-1 - "ISO/IEC 9126-1 Software Engineering - Product Quality - Part 1 Quality Model." 2001. (ISO01, 2001)

ISO/IEC 9126-2 - "ISO/IEC 9126-2 Software Engineering - Product Quality - Part 2 External Metrics." 2001. (ISO02, 2001)

ISO/IEC 9126-3 - "ISO/IEC 9126-3 Software Engineering - Product Quality - Part 3 Internal Metrics." 2001. (ISO03, 2001)

ISO/IEC 9126-4 - "ISO/IEC 9126-4 Software Engineering - Product Quality - Part 4 Quality In Use Metrics." 2001 (ISO04, 2001)

CQM-1 - A. Alvaro, E.S. Almeida, S.R.L. Meira. "A Software Component Quality Model: A Preliminary Evaluation." 2005. (A. Alvaro, 2006)

CQM-2 - A. Alvaro, E.S. Almeida, S.R.L. Meira. "Quality Attributes for a Component Quality Model." 2005. (A. Alvaro, 2005)

CQM-3 - A. Alvaro, E.S. Almeida, A.M.L. Vasconcelos, S.R.L. Meira. "Towards a Software Component Quality Model." 2005. (A. Alvaro, 2005)

SOA-RA-02 - "Reference Architecture Foundation for Service Oriented Architecture Version 1.0, Committee Draft 02" 2009. (OASIS, 2009)

NIST SP 800-53 Revision 3 - "Recommended Security Controls for Federal Information Systems and Organizations" 2009. (NIST, 2009)

CWE/SANS Top 25 - Common Weakness Enumeration/SysAdmin, Audit, Network, Security Top 25 Most Dangerous Software Errors, http://cwe.mitre.org/top25/ downloaded 8/16/2010 (MITRE, 2010)

Appendix C The Trust Taxonomy Survey Results

1. If presented with two or more components or compositions that perform the same function, do you feel that the presence of trust metrics will be useful during composition time to help you select the best one to use?

answered question	**13**
skipped question	**0**

	Response Percent	Response Count
Yes - understanding the trustworthiness of a component or composition would be useful during composition time.	**100.0%**	**13**
No - trustworthiness of components or compositions would serve no use during composition time.	0.0%	0

2. As an end-user, do you feel if trust metrics were associated with a capability that you would be better prepared to make proper decisions based on that capability's output?

answered question	**13**
skipped question	**0**

	Response Percent	Response Count
Yes - trust information about a capability would be useful when making decisions.	**100.0%**	**13**
No - trust information about a capability would be would serve no use when making decisions.	0.0%	0

3. Please add any comments on this section of the survey.

#	Response Date	Response Text
1	Jun 16, 2010 8:02 PM	Reputation of source of trust metrics is important.
2	Jun 17, 2010 3:30 PM	For #1, there's a huge difference between "understanding the trustworthiness" and "presence of trust metrics". I almost answered no but then went with the assumption that you were asking if trust metrics existed so I could make an assessment of "trustworthiness". The second survey was much better represented in
3	Jun 18, 2010 8:42 PM	Trustworthiness would be one factor, but may not be the only factor. For example, if two components or compositions do the same thing, but one is much more efficient than the other and that difference has significant performance impacts on overall system performance, then I might give more consideration to the composition that does not have a pedigree. But in the absence of significant differences, I would definitely select the trustworthy one.
4	Jun 21, 2010 11:53 AM	Trustworthiness is a factor, but it's not everything.

4. Given these categories of analytical trust measures for a capability or composition: Functionality, Security, Reliability, Usability, Efficiency, Maintainability, and Portability. Rate them from least important (1) to most important (7) when considering the broad concept of trust (to rely upon or have confidence in).

							answered question	**13**
							skipped question	**0**
	1	**2**	**3**	**4**	**5**	**6**	**7**	**Response Count**
Functionality - Evaluate the ability to provide the required services when used under specified conditions.	0.0% (0)	0.0% (0)	0.0% (0)	7.7% (1)	15.4% (2)	15.4% (2)	**61.5% (8)**	13
Security - Evaluate the ability to protect information and data so that unauthorized persons or systems cannot read or modify them and authorized persons or systems are not denied access to them.	0.0% (0)	0.0% (0)	0.0% (0)	30.8% (4)	7.7% (1)	**38.5% (5)**	23.1% (3)	13
Reliability - Evaluate the ability to maintain a specified level of performance, when used under specified conditions.	0.0% (0)	7.7% (1)	7.7% (1)	15.4% (2)	7.7% (1)	**30.8% (4)**	**30.8% (4)**	13
Usability – Evaluate the ability to be used by someone other than the originator.	7.7% (1)	0.0% (0)	15.4% (2)	7.7% (1)	15.4% (2)	**38.5% (5)**	15.4% (2)	13
Efficiency - Evaluate the ability to provide appropriate performance, relative to the amount of resources used.	15.4% (2)	15.4% (2)	**46.2% (6)**	0.0% (0)	15.4% (2)	7.7% (1)	0.0% (0)	13
Maintainability - Evaluate the ability to be modified.	0.0% (0)	**38.5% (5)**	15.4% (2)	7.7% (1)	**38.5% (5)**	0.0% (0)	0.0% (0)	13
Portability - Evaluate the ability to be used in an environment other than that in which it was originally developed.	**38.5% (5)**	30.8% (4)	0.0% (0)	15.4% (2)	7.7% (1)	0.0% (0)	7.7% (1)	13

5. Please add any comments on this section of the survey.

#	Response Date	Response Text
1	Jun 17, 2010 2:29 PM	I feel that Portability and Usability are the two most defining characteristics of a composable environment. Thing such as Security and Reliability are of course very important, but they are more situational. I feel that Portability and Usability (as necessities or issues) will encapsulate a greater portion of composable scenarios than Security and Reliability which are required only in specific kinds of scenarios.
2	Jun 17, 2010 3:37 PM	Where are the "author/composer" attributes for trust? All things being equal, my evaluation/perception of the track record of or relevance of the source/author/composer will be critical in modifying how I weight the above criteria.
3	Jun 18, 2010 8:52 PM	I think the answers to this question might depend on whether you were answering them from a developer's or end user's perspective. For example, a developer might not be too concerned about portability, but an end user might. I think they're all important for gaining CCOD acceptance.
4	Jun 21, 2010 11:54 AM	As long as functionality is there, other measures can be worked.

6. Within the category "Functionality" for compositions and capabilities, the following sub-categories exist: Accuracy, Suitability, Internal Interoperability, External Interoperability, Functionality Compliance and Self Contained. Rate them least important (1) to most important (6) when considering the broad concept of trust (to rely upon or have confidence in).

						answered question	**13**
						skipped question	**0**
	1	**2**	**3**	**4**	**5**	**6**	**Response Count**
Accuracy - Evaluate the ability of the composition to provide the right or agreed results or effects with the needed degree of precision. This subcategory consists of one measure: correctness - Based on available documentation determine if the component does what the documentation indicates. If there is no documentation then the correctness cannot be determined.	7.7% (1)	7.7% (1)	15.4% (2)	15.4% (2)	15.4% (2)	**38.5% (5)**	13
Suitability - Evaluate the ability to provide an appropriate set of functions for specified tasks and user objectives.	0.0% (0)	0.0% (0)	15.4% (2)	15.4% (2)	**38.5% (5)**	30.8% (4)	13
Internal Interoperability - Evaluate the ability to interact with other compositions and components within the composition environment.	7.7% (1)	0.0% (0)	23.1% (3)	**53.8% (7)**	0.0% (0)	15.4% (2)	13
External Interoperability - Evaluate the ability for compositions and components within a composition environment to be used within another environment.	7.7% (1)	7.7% (1)	**53.8% (7)**	7.7% (1)	7.7% (1)	15.4% (2)	13
Functionality Compliance - Evaluate the ability to adhere to standards, conventions or regulations in laws and similar prescriptions relating to functionality. This subcategory contains one measure: standardization.	7.7% (1)	23.1% (3)	7.7% (1)	23.1% (3)	**30.8% (4)**	7.7% (1)	13
Self Contained - Evaluate the level to which functionalities that the composition performs are fully performed within itself. This subcategory contains one measure: dependent - Determine if the code is self contained, if not, the component is considered less trustworthy. Functionality internal to component / total functionality (by simple inspection)	30.8% (4)	**38.5% (5)**	7.7% (1)	7.7% (1)	15.4% (2)	0.0% (0)	13

7. Within the subcategory "Suitability" of the category "Functionality" for compositions and capabilities, the following measures exist: Coverage, Completeness and Pre and post conditioned. Rate them least important (1) to most important (3) when considering the broad concept of trust (to rely upon or have confidence in).

	answered question			13
	skipped question			0
	1	**2**	**3**	**Response Count**
Coverage - Does the composition attempt to do everything the documentation says it is supposed to do?	30.8% (4)	**61.5% (8)**	7.7% (1)	13
Completeness - For every function that the composition attempts to perform, does it actually do the function?	7.7% (1)	23.1% (3)	**69.2% (9)**	13
Pre and post conditioned - For inputs and outputs assure that pre & post conditions are documented. For inputs, outputs and internal pre & post conditions all have been implemented.	**53.8% (7)**	23.1% (3)	23.1% (3)	13

8. Within the subcategory "Internal Interoperability" of the category "Functionality" for compositions and capabilities, the following measures exist: Data compatibility and Interface complexity. Rate them least important (1) to most important (2) when considering the broad concept of trust (to rely upon or have confidence in).

	answered question		13
	skipped question		0
	1	**2**	**Response Count**
Data compatibility - Analysis of the data standard(s) used	46.2% (6)	**53.8% (7)**	13
Interface complexity - A measure of the difficulty of making compositions interoperate with other compositions and components while composing.	38.5% (5)	**61.5% (8)**	13

9. Within the subcategory "External Interoperability" of the category "Functionality" for compositions and capabilities, the following measures exist: Data compatibility and Interface complexity. Rate them least important (1) to most important (2) when considering the broad concept of trust (to rely upon or have confidence in).

	answered question	13
	skipped question	0

	1	2	Response Count
Data compatibility - Analysis of the data standard(s) used	38.5% (5)	**61.5% (8)**	13
Interface complexity - A measure of the difficulty of making capabilities interoperate with other capabilities within the end-user environment.	**53.8% (7)**	46.2% (6)	13

10. Please add any comments on this section of the survey.

#	Response Date	Response Text
1	Jun 17, 2010 1:04 PM	Data is the sine non qua of interoperability.
2	Jun 17, 2010 3:40 PM	I'm not sure the subcategories are useful and if they are, the breakdowns don't seem to really relate well to the parent.

11. Within the category "Security" for compositions and capabilities, the following sub-categories exist: Confidentiality, Integrity, Availability and Non-repudiation. Rate them least important (1) to most important (4) when considering the broad concept of trust (to rely upon or have confidence in).

	answered question				13
	skipped question				0
	1	**2**	**3**	**4**	**Response Count**
Confidentiality - To prevent the disclosure of information to unauthorized individuals or systems	7.7% (1)	7.7% (1)	7.7% (1)	**76.9% (10)**	13
Integrity - To assure that data (including components, compositions and capabilities) can only be modified by authorized users or systems	7.7% (1)	15.4% (2)	**46.2% (6)**	30.8% (4)	13
Availability - To assure that component is available when needed. This subcategory consists of one measure: strong authentication.	**38.5% (5)**	15.4% (2)	**38.5% (5)**	7.7% (1)	13
Non-repudiation - Provides proof of integrity or origin of data. This subcategory consists of one measure: auditability - The ability to identify the person or system that performed, or is responsible for, actions affecting information, as well as the specific actions performed and when.	15.4% (2)	**53.8% (7)**	7.7% (1)	23.1% (3)	13

12. Within the subcategory "Confidentiality" of the category "Security" for compositions and capabilities, the following measures exist: Authentication, Authorization and Auditability. Rate them least important (1) to most important (3) when considering the broad concept of trust (to rely upon or have confidence in).

	answered question			13
	skipped question			0
	1	**2**	**3**	**Response Count**
Authentication - The act of verifying a claim of identity	7.7% (1)	38.5% (5)	**53.8% (7)**	13
Authorization - The determination of what information resources an authenticated user or system can access	15.4% (2)	38.5% (5)	**46.2% (6)**	13
Auditability - The ability to identify the person or system that performed, or is responsible for, actions affecting information, as well as the specific actions performed and when.	**38.5% (5)**	30.8% (4)	30.8% (4)	13

13. Within the subcategory "Integrity" of the category "Security" for compositions and capabilities, the following measures exist: Authentication, Authorization and Auditability. Rate them least important (1) to most important (3) when considering the broad concept of trust (to rely upon or have confidence in).

	answered question			13
	skipped question			0
	1	**2**	**3**	**Response Count**
Authentication - The act of verifying a claim of identity	0.0% (0)	38.5% (5)	**61.5% (8)**	13
Authorization - The determination of what information resources an authenticated user or system can access	23.1% (3)	**46.2% (6)**	30.8% (4)	13
Data Encryption - The process of transforming information (referred to as plaintext) using an algorithm (called cipher) to make it unreadable to anyone except those possessing special knowledge, usually referred to as a key.	**46.2% (6)**	23.1% (3)	30.8% (4)	13

14. Please add any comments on this section of the survey.

#	Response Date	Response Text
1	Jun 17, 2010 3:45 PM	Isn't it a fundamental breach of the security model to take the fundamental aspects of security and ask which is most important in a static nature? In general, all should have equal importance and the relative priority of their importance would only change based on the instance requirements.

15. Within the category "Reliability" for compositions and capabilities, the following sub-categories exist: Fault Tolerance, Recoverability and Maturity. Rate them least important (1) to most important (3) when considering the broad concept of trust (to rely upon or have confidence in).

		answered question		**13**
		skipped question		**0**
	1	**2**	**3**	**Response Count**
Fault Tolerance - Evaluate the ability to maintain a specified level of performance in cases of faults or of infringement of its specified interface.	38.5% (5)	**53.8% (7)**	7.7% (1)	13
Recoverability - Evaluate the ability to re-establish a specified level of performance and recover the data directly affected in the case of a failure.	15.4% (2)	**46.2% (6)**	38.5% (5)	13
Maturity - Evaluate the ability to avoid failure as a result of faults in the software.	30.8% (4)	7.7% (1)	**61.5% (8)**	13

16. Within the subcategory "Fault Tolerance" of the category "Reliability" for compositions and capabilities, the following measures exist: Mechanism available and Mechanism efficiency. Rate them least important (1) to most important (2) when considering the broad concept of trust (to rely upon or have confidence in).

	answered question		**12**
	skipped question		**1**
	1	**2**	**Response Count**
Mechanism available - Graceful failure described in documentation	33.3% (4)	**66.7% (8)**	12
Mechanism efficiency - Rough ratio of functions with graceful failure versus total functionality.	**66.7% (8)**	33.3% (4)	12

17. Within the subcategory "Maturity" of the category "Reliability" for compositions and capabilities, the following measures exist: Volatility and Failure removal. Rate them least important (1) to most important (2) when considering the broad concept of trust (to rely upon or have confidence in).

	answered question		13
	skipped question		0
	1	**2**	**Response Count**
Volatility - Analysis of the time between versions	**92.3% (12)**	7.7% (1)	13
Failure removal – Measures the rate that problems are resolved.	8.3% (1)	**91.7% (11)**	12

18. Within the subcategory "Recoverability" of the category "Reliability" for compositions and capabilities, the following measures exist: Mechanism available and Mechanism efficiency. Rate them least important (1) to most important (2) when considering the broad concept of trust (to rely upon or have confidence in).

	answered question		13
	skipped question		0
	1	**2**	**Response Count**
Mechanism available - Recovery described in documentation.	38.5% (5)	**61.5% (8)**	13
Mechanism efficiency - Rough ratio of functions with recovery mechanisms versus total functions and level of recovery by inspection (completeness and processes).	**61.5% (8)**	38.5% (5)	13

19. Please add any comments on this section of the survey.

#	Response Date	Response Text
1	Jun 17, 2010 2:37 PM	Maturity vs. Fault tolerance was difficult for me to choose between, however I did not want to rate them equally and I feel that Maturity is a less reliable metric because the maturity of a component does not always mean anything about how good it is.
2	Jun 17, 2010 3:50 PM	Again, the subcategories for the most part seem to down in the weeds? With reliability, one usually cares about two things: is the thing usually available when I need it (or more important, can it be guaranteed to be available when I need it - so mechanism may matter in that case); how often does it crash/exhibit bugs or anomalous behavior but more importantly, how responsive are the maintainers in addressing issues.
3	Jun 21, 2010 12:01 PM	Not confident about my answers on this page -- didn't understand all the descriptions.

20. Within the category "Usability" for compositions and capabilities, the following sub-categories exist: Configurability, Understandability and Operability. Rate them least important (1) to most important (3) when considering the broad concept of trust (to rely upon or have confidence in).

	answered question			**13**
	skipped question			**0**
	1	**2**	**3**	**Response Count**
Configurability - Evaluate the ability to specify parameters that aid in the use of the composition within a particular context. The ability of the composition be configurable (e.g. through a XML file or a text file, the number of parameters, etc.). This subcategory consists of one measure: Effort to configure - The ability to specify parameters that aid in the use of the capability or composition within a particular context.	38.5% (5)	**46.2% (6)**	15.4% (2)	13
Understandability - Evaluate the ability for the composer or end user to understand whether the composition is suitable, and how it can be used for particular tasks and conditions of use. This subcategory consists of one measure: documentation quality.	23.1% (3)	23.1% (3)	**53.8% (7)**	13
Operability - Evaluate the ability for the user to operate and control the composition.	15.4% (2)	38.5% (5)	**46.2% (6)**	13

21. Within the subcategory "Operability" of the category "Usability" for compositions and capabilities, the following measures exist: Parameters and Required parameters. Rate them least important (1) to most important (2) when considering the broad concept of trust (to rely upon or have confidence in).

	answered question		**13**
	skipped question		**0**
	1	**2**	**Response Count**
Parameters – measures the number of parameters	**84.6% (11)**	15.4% (2)	13
Required parameters – measures the number of required parameters	15.4% (2)	**84.6% (11)**	13

22. Please add any comments on this section of the survey.

#	Response Date	Response Text
1	Jun 17, 2010 3:50 PM	Same comment on subcategory.

23. Within the category "Efficiency" for compositions and capabilities, the following sub-categories exist: Time Behavior, Resource Utilization and Scalability. Rate them least important (1) to most important (3) when considering the broad concept of trust (to rely upon or have confidence in).

	answered question			12
	skipped question			1
	1	**2**	**3**	**Response Count**
Time Behavior - Evaluate the ability to provide appropriate response, processing times and throughput rates when performing its function, under stated conditions.	16.7% (2)	33.3% (4)	**50.0% (6)**	12
Resource Utilization - Evaluate the ability to use appropriate amounts and types of resources when performing its function under stated conditions.	**50.0% (6)**	33.3% (4)	16.7% (2)	12
Scalability - Evaluate the ability to accommodate major data volumes without changing implementation. This subcategory consists of one measure: processing capacity – the ability of the composition to accommodate major data volumes without changing its implementation.	16.7% (2)	**41.7% (5)**	**41.7% (5)**	12

24. Within the subcategory "Time Behavior" of the category "Efficiency" for compositions and capabilities, the following measures exist: Response time and Throughput. Rate them least important (1) to most important (2) when considering the broad concept of trust (to rely upon or have confidence in).

	answered question		12
	skipped question		1
	1	**2**	**Response Count**
Response time – evaluate the time required to perform the designated task of the capability or composition.	25.0% (3)	**75.0% (9)**	12
Throughput - evaluate the time required to perform a number of the designated task of the capability or composition.	**75.0% (9)**	25.0% (3)	12

25. Within the subcategory "Resource Utilization" of the category "Efficiency" for compositions and capabilities, the following measures exist: Memory usage, Disk usage and Network usage. Rate them least important (1) to most important (3) when considering the broad concept of trust (to rely upon or have confidence in).

answered question	**12**
skipped question	**1**

	1	2	3	Response Count
Memory usage – evaluate the amount of memory used	16.7% (2)	**66.7% (8)**	16.7% (2)	12
Disk usage – evaluate the amount of disk space used	41.7% (5)	**58.3% (7)**	0.0% (0)	12
Network usage – evaluate the amount of network resources used	8.3% (1)	8.3% (1)	**83.3% (10)**	12

26. Please add any comments on this section of the survey.

No responses

27. Within the category "Maintainability" for compositions and capabilities, the following sub-categories exist: Stability, Changeability, Testability, Analyzability, Composition Deployability, Capability Deployability and Coexistence. Rate them least important (1) to most important (4) when considering the broad concept of trust (to rely upon or have confidence in).

			answered question	**12**
			skipped question	**1**

	1	**2**	**3**	**4**	**Response Count**
Stability - The capability to avoid unexpected effects from modifications of the software. This subcategory consists of one measure: Modifications over time – Measure the types of modifications (fixes or enhancements) over time.	0.0% (0)	16.7% (2)	33.3% (4)	**50.0% (6)**	12
Changeability - Evaluate the ability to enable a specified modification to be implemented.	8.3% (1)	**41.7% (5)**	25.0% (3)	25.0% (3)	12
Testability - Evaluate the existence of test cases and proofs.	33.3% (4)	16.7% (2)	**41.7% (5)**	8.3% (1)	12
Analyzability - Evaluate the ability of a composition to be diagnosed for deficiencies or causes of failures and to identify the components or compositions needing to be modified.	**33.3% (4)**	25.0% (3)	8.3% (1)	**33.3% (4)**	12

28. Within the subcategory "Changeability" of the category "Maintainability" for compositions and capabilities,the following measures exist: Complexity level, Cohesion and Coupling. Rate them least important (1) to most important (3) when considering the broad concept of trust (to rely upon or have confidence in).

	answered question			12
	skipped question			1
	1	**2**	**3**	**Response Count**
Complexity level – determine the level of complexity	16.7% (2)	16.7% (2)	**66.7% (8)**	12
Cohesion - A composition is considered to be highly cohesive if the elements within it are closely related functionally. Higher cohesion implies easier modifications.	33.3% (4)	**50.0% (6)**	16.7% (2)	12
Coupling - the measure of the strength of association established by a connection from one composition to another. Lower coupling implies easier modifications.	**33.3% (4)**	**33.3% (4)**	**33.3% (4)**	12

29. Within the subcategory "Testability" of the category "Maintainability" for compositions and capabilities, the following measures exist: Test cases and/or proofs provided and Environment test cases. Rate them least important (1) to most important (2) when considering the broad concept of trust (to rely upon or have confidence in).

	answered question		12
	skipped question		1
	1	**2**	**Response Count**
Test cases and/or proofs provided - Assess the quality of the test cases or presence of proofs.	33.3% (4)	**66.7% (8)**	12
Environment test cases – evaluate the number of environments that the composition was tested	**58.3% (7)**	41.7% (5)	12

30. Please add any comments on this section of the survey.

No responses

31. Within the category "Portability" for compositions and capabilities, the following sub-categories exist: Stability, Changeability, Testability, Analyzability, Composition Deployability, Capability Deployability and Coexistence. Rate them least important (1) to most important (3) when considering the broad concept of trust (to rely upon or have confidence in).

	answered question			12
	skipped question			1
	1	**2**	**3**	**Response Count**
Composition Deployability - Evaluate the ability of a composition, without a user interface (i.e., not a capability), to be used within another CCOD environment as a web service.	8.3% (1)	**50.0% (6)**	41.7% (5)	12
Capability Deployability – Evaluate the ability of a composition with a user interface to be used independent of its originating CCOD environment.	8.3% (1)	41.7% (5)	**50.0% (6)**	12
Coexistence - Evaluate the ability of compositions and capabilities to coexist in a common environment sharing common resources.	**50.0% (6)**	8.3% (1)	41.7% (5)	12

32. Within the subcategory "Composition Deployability" of the category "Portability," the following measures exist: Deployable as web service, Data compatibility and Interface complexity. Rate them least important (1) to most important (3) when considering the broad concept of trust (to rely upon or have confidence in).

	answered question			12
	skipped question			1
	1	**2**	**3**	**Response Count**
Deployable as web service – determine if the composition can be deployed as a web service available for use outside the environment.	25.0% (3)	**50.0% (6)**	25.0% (3)	12
Data compatibility – determine if the composition utilizes a common data standard and whether data passed to/from the composition is restricted in schema or structure.	16.7% (2)	8.3% (1)	**75.0% (9)**	12
Interface complexity - evaluate the complexity level of exposing a composition as a web service.	**41.7% (5)**	**41.7% (5)**	16.7% (2)	12

33. Within the subcategory "Capability Deployability" of the category "Portability," the following measures exist: Deployable to another environment, Adherence to browser standards and Capability deployment difficulty. Rate them least important (1) to most important (3) when considering the broad concept of trust (to rely upon or have confidence in).

			answered question	**12**
			skipped question	**1**
	1	**2**	**3**	**Response Count**
Deployable to another environment – determine if it is possible to deploy an end-user capability into an environment other than in which it was composed.	16.7% (2)	33.3% (4)	**50.0% (6)**	12
Adherence to browser standards – determine if the process of deploying a capability into another environment adheres to browser standards.	33.3% (4)	**41.7% (5)**	25.0% (3)	12
Capability deployment difficulty – assess the time and effort to deploy a capability to another environment.	16.7% (2)	**50.0% (6)**	33.3% (4)	12

34. Please add any comments on this section of the survey.

No responses

35. End users will be asked to assess capabilities using the following criteria: Effectiveness, Productivity, Satisfaction and Attractiveness. Rate them least important (1) to most important (4) when considering the broad concept of trust (to rely upon or have confidence in).

			answered question		**12**
			skipped question		**1**
	1	**2**	**3**	**4**	**Response Count**
Effectiveness - Evaluate the ability of the capability to enable users to achieve specified goals with accuracy and completeness.	8.3% (1)	0.0% (0)	25.0% (3)	**66.7% (8)**	12
Productivity - Evaluate the ability of the capability to enable users to expend appropriate amounts of resources in relation to the effectiveness achieved in a specified context of use.	8.3% (1)	33.3% (4)	**41.7% (5)**	16.7% (2)	12
Satisfaction - Evaluate the ability of the capability to satisfy users in a specified context of use.	0.0% (0)	**50.0% (6)**	25.0% (3)	25.0% (3)	12
Attractiveness - Evaluate the user interface of the capability.	**50.0% (6)**	33.3% (4)	16.7% (2)	0.0% (0)	12

36. Composers will be asked to assess compositions using the following criteria: Effectiveness, Productivity, Satisfaction and Effort for operating. Rate them least important (1) to most important (4) when considering the broad concept of trust (to rely upon or have confidence in).

			answered question		**12**
			skipped question		**1**
	1	**2**	**3**	**4**	**Response Count**
Effectiveness - Evaluate the ability of the composition to enable to achieve specified goals with accuracy and completeness.	0.0% (0)	0.0% (0)	8.3% (1)	**91.7% (11)**	12
Productivity - Evaluate the ability of the composition to enable users to expend appropriate amounts of resources in relation to the effectiveness achieved in a specified context of use.	0.0% (0)	**50.0% (6)**	41.7% (5)	8.3% (1)	12
Satisfaction - Evaluate the ability of the composition to satisfy composers in a specified context of use.	**33.3% (4)**	**33.3% (4)**	25.0% (3)	8.3% (1)	12
Effort for operating – Evaluate the amount of effort needed to successfully use a composition within another composition.	**33.3% (4)**	25.0% (3)	**33.3% (4)**	8.3% (1)	12

37. Please add any comments on this section of the survey.

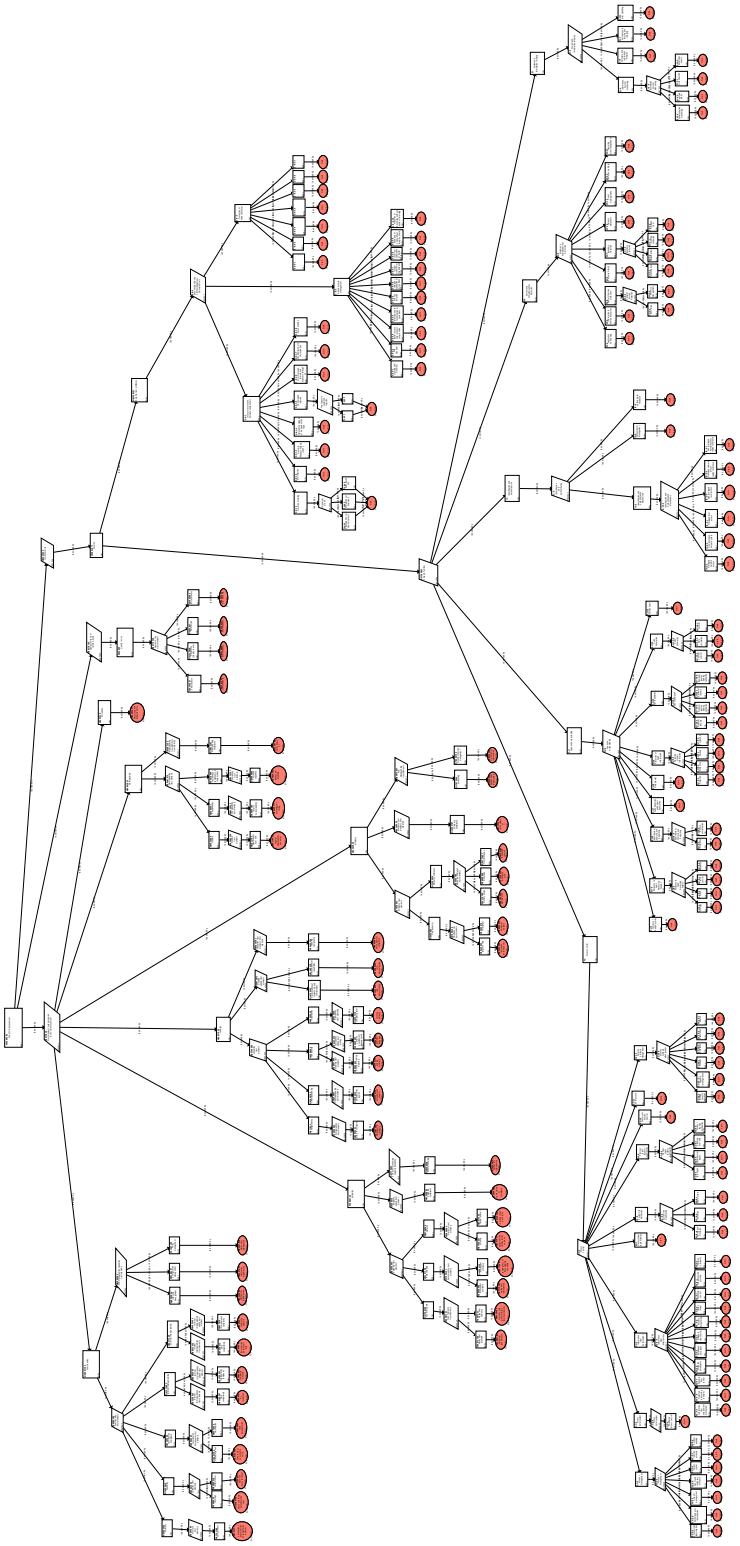
No responses

38. Please add any final comments.

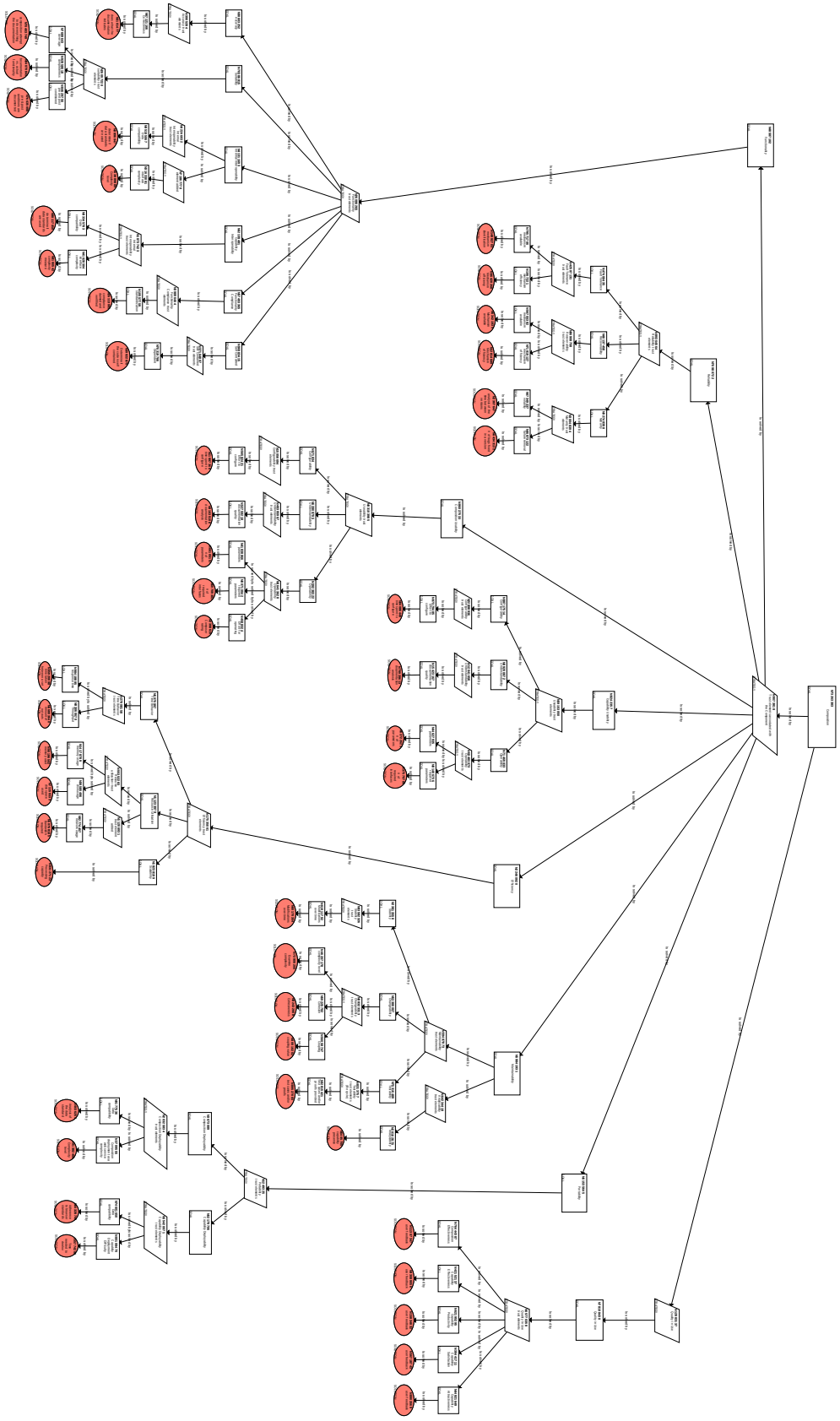
#	Response Date	Response Text
1	Jun 16, 2010 8:31 PM	Next time a more quiet area for experiment. Someone even turn on a TV besides me.
2	Jun 18, 2010 9:25 PM	I answered the questions to the best of my ability, but in so doing I was constantly reevaluating my answers in trying to decide upon the context for the answers, e.g. developer or consumer, safety critical application or best effort, etc. Relative importance may be affected by the environment or specific application.
3	Jun 21, 2010 12:25 PM	It was an interesting survey -- I think I understand what CCOD is about better.

Appendix D The Assurance Model

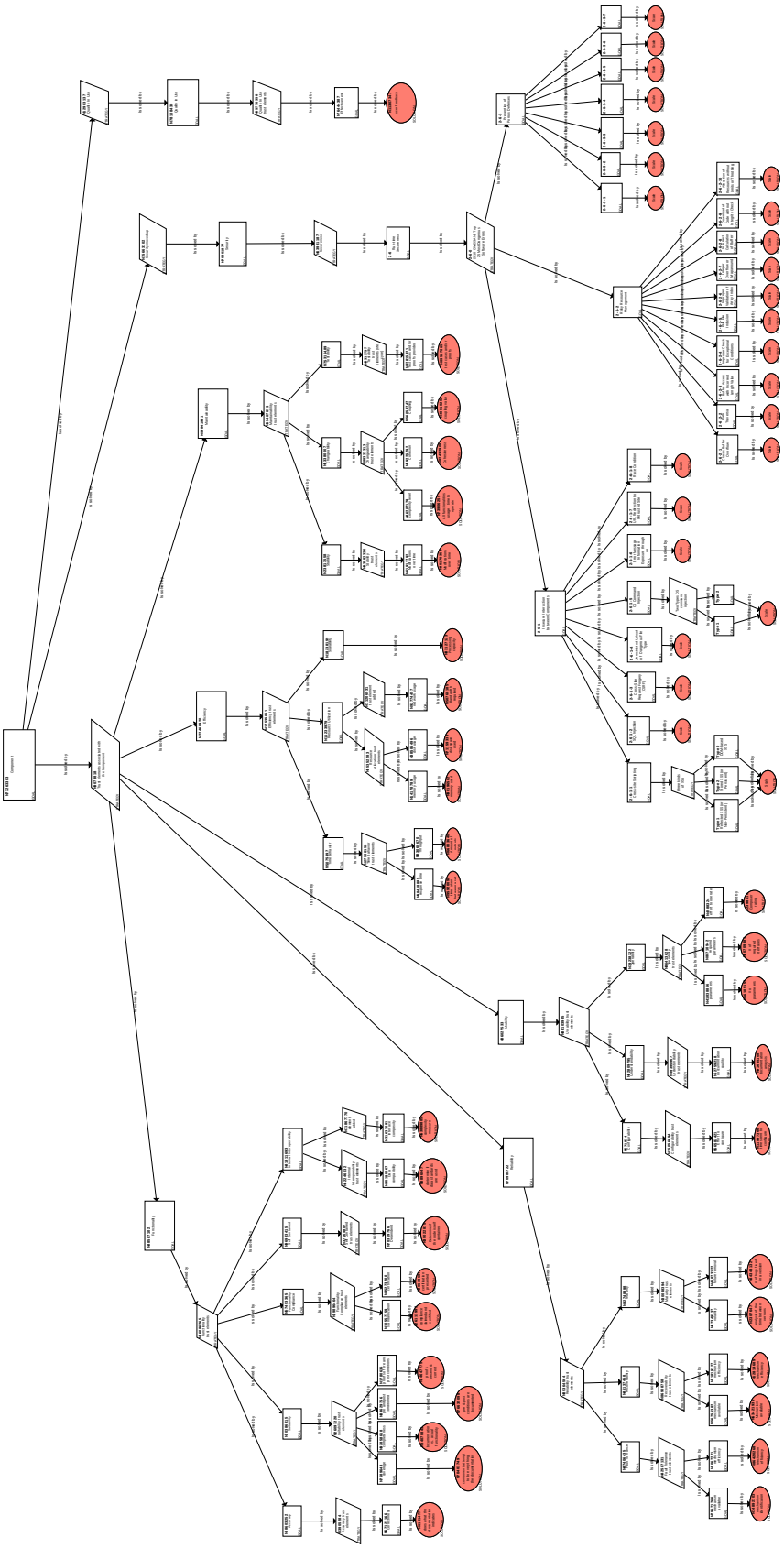
D.1 CCOD User Environment



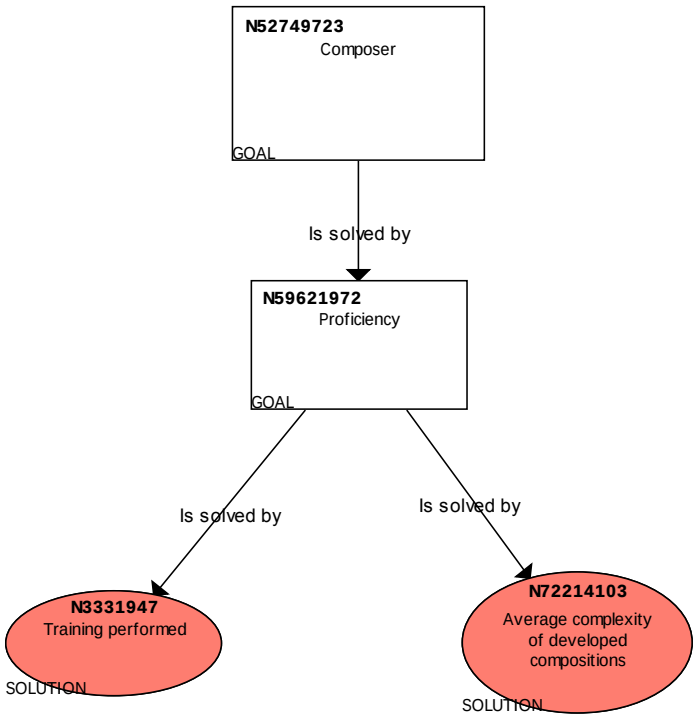
D.2 Composition



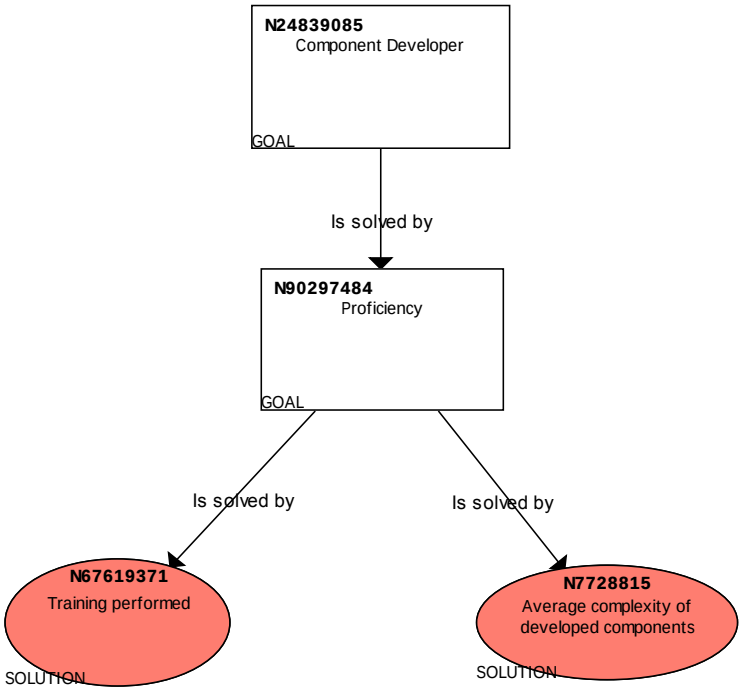
D.3 Component



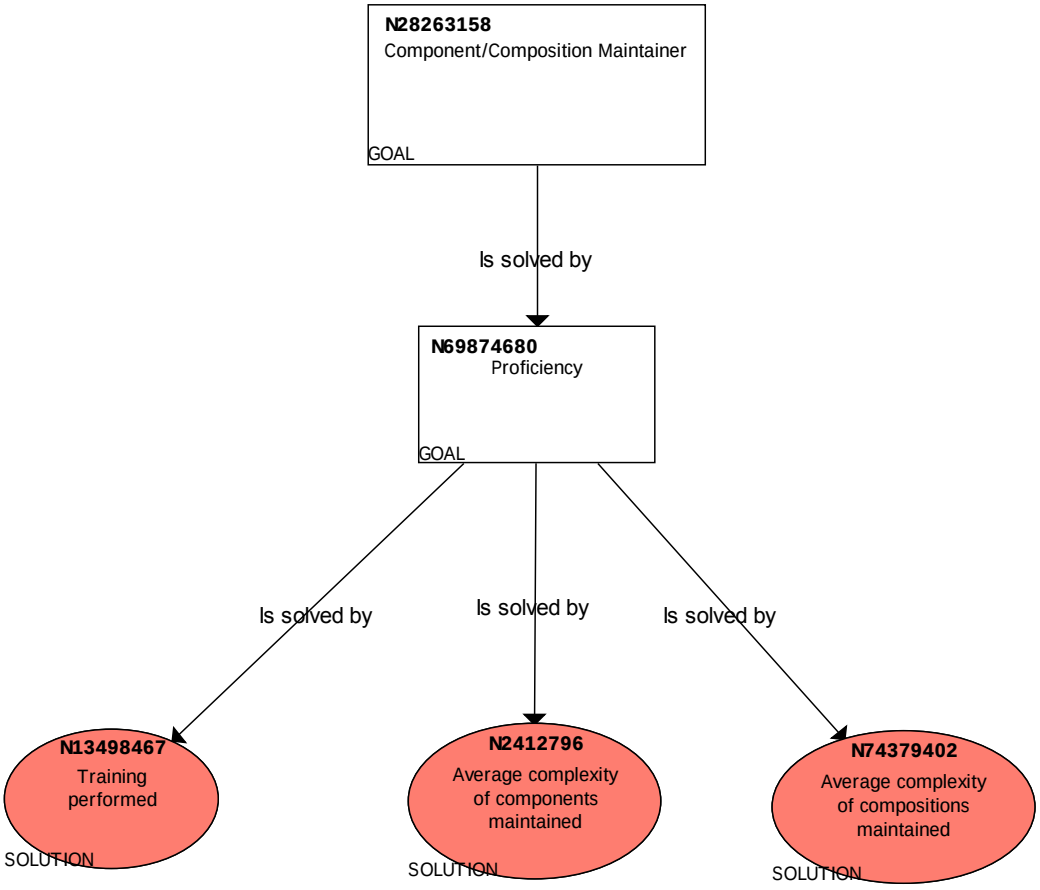
D.4 Composer


N52749723
Composer
GOAL
Is solved by
N59621972
Proficiency
GOAL
Is solved by
Is solved by
N3331947
Training performed
SOLUTION
N72214103
Average complexity of developed compositions
SOLUTION

D.5 Component Developer



D.6 Component/Composition Maintainer



Appendix E The Trust Taxonomy XML Structures

E.1 XML Schema for trust taxonomy metric values, rollups and weights

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:wmh="http://www.wmhelp.com/2003/eGenerator" elementFormDefault="qualified">
  <xs:element name="Composition">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Name"/>
        <xs:element name="Functionality">
          <xs:complexType>
            <xs:sequence>
              <xs:element ref="Accuracy"/>
              <xs:element name="Suitability">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element ref="Coverage"/>
                    <xs:element ref="Completeness"/>
                    <xs:element ref="PrePostConditioned"/>
                  </xs:sequence>
                  <xs:attribute ref="rollup" use="optional"/>
                  <xs:attribute ref="weight" use="optional"/>
                </xs:complexType>
              </xs:element>
              <xs:element ref="InternalInteroperability"/>
              <xs:element ref="CapabilityInteroperability"/>
              <xs:element name="FunctionalityCompliance">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element ref="Standardization"/>
                  </xs:sequence>
                  <xs:attribute ref="rollup" use="optional"/>
                  <xs:attribute ref="weight" use="optional"/>
                </xs:complexType>
              </xs:element>
              <xs:element ref="SelfContained"/>
            </xs:sequence>
            <xs:attribute ref="rollup" use="optional"/>
            <xs:attribute ref="weight" use="optional"/>
          </xs:complexType>
        </xs:element>
        <xs:element ref="Reliability"/>
        <xs:element ref="CompositionUsability"/>
        <xs:element ref="CapabilityUsability"/>
        <xs:element ref="Efficiency"/>
        <xs:element name="Maintainability">
          <xs:complexType>
            <xs:sequence>
              <xs:element ref="Stability"/>
              <xs:element name="Changeability">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element ref="ComplexityLevel"/>
                    <xs:element ref="Cohesion"/>
```

```
            <xs:element ref="Coupling"/>
           </xs:sequence>
           <xs:attribute ref="rollup" use="optional"/>
           <xs:attribute ref="weight" use="optional"/>
          </xs:complexType>
         </xs:element>
         <xs:element name="Testability">
          <xs:complexType>
           <xs:sequence>
            <xs:element ref="TestCasesProofs"/>
           </xs:sequence>
           <xs:attribute ref="rollup" use="optional"/>
           <xs:attribute ref="weight" use="optional"/>
          </xs:complexType>
         </xs:element>
         <xs:element ref="Analyzeability"/>
        </xs:sequence>
        <xs:attribute ref="rollup" use="optional"/>
        <xs:attribute ref="weight" use="optional"/>
       </xs:complexType>
      </xs:element>
      <xs:element ref="Portability"/>
      <xs:element ref="QualityInUse" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Composer" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="MaintenanceEvent" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="Composition" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="Component" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="CCODEnvironment"/>
    </xs:sequence>
    <xs:attribute ref="rollup" use="optional"/>
    <xs:attribute ref="weight" use="optional"/>
  </xs:complexType>
</xs:element>
<xs:element name="Name" type="xs:string"/>
<xs:element name="Accuracy">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Correctness"/>
    </xs:sequence>
    <xs:attribute ref="rollup" use="optional"/>
    <xs:attribute ref="weight" use="optional"/>
  </xs:complexType>
</xs:element>
<xs:element name="Correctness" type="AttributesScoreType" default="0"/>
<xs:element name="Coverage" type="AttributesScoreType" default="0"/>
<xs:element name="Completeness" type="AttributesScoreType" default="0"/>
<xs:element name="PrePostConditioned" type="AttributesScoreType" default="0"/>
<xs:element name="PrePostConditionsProofs" type="AttributesScoreType" default="0"/>
<xs:element name="CompositionInteroperability">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="DataCompatibility"/>
      <xs:element ref="InterfaceComplexity"/>
    </xs:sequence>
    <xs:attribute ref="rollup" use="optional"/>
    <xs:attribute ref="weight" use="optional"/>
  </xs:complexType>
</xs:element>
```

```
<xs:element name="CapabilityInteroperability">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="DataCompatibility"/>
      <xs:element ref="InterfaceComplexity"/>
    </xs:sequence>
    <xs:attribute ref="rollup" use="optional"/>
    <xs:attribute ref="weight" use="optional"/>
  </xs:complexType>
</xs:element>
<xs:element name="DataCompatibility" type="AttributesScoreType" default="0"/>
<xs:element name="InterfaceComplexity" type="AttributesScoreType" default="0"/>
<xs:element name="FunctionalityCompliance">
  <xs:complexType>
      <xs:sequence>
        <xs:element ref="Standardization"/>
        <xs:element ref="Certification"/>
      </xs:sequence>
    <xs:attribute ref="rollup" use="optional"/>
    <xs:attribute ref="weight" use="optional"/>
  </xs:complexType>
</xs:element>
<xs:element name="Standardization" type="AttributesScoreType" default="0"/>
<xs:element name="Certification" type="AttributesScoreType" default="0"/>
<xs:element name="SelfContained">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Dependant"/>
    </xs:sequence>
    <xs:attribute ref="rollup" use="optional"/>
    <xs:attribute ref="weight" use="optional"/>
  </xs:complexType>
</xs:element>
<xs:element name="Dependant" type="AttributesScoreType" default="0"/>
<xs:element name="Reliability">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="FaultTolerance"/>
      <xs:element ref="Recoverability"/>
      <xs:element ref="Maturity"/>
    </xs:sequence>
    <xs:attribute ref="rollup" use="optional"/>
    <xs:attribute ref="weight" use="optional"/>
  </xs:complexType>
</xs:element>
<xs:element name="FaultTolerance">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="MechanismAvailable"/>
      <xs:element ref="MechanismEfficiency"/>
    </xs:sequence>
    <xs:attribute ref="rollup" use="optional"/>
    <xs:attribute ref="weight" use="optional"/>
  </xs:complexType>
</xs:element>
<xs:element name="MechanismAvailable" type="AttributesScoreType" default="0"/>
<xs:element name="MechanismEfficiency" type="AttributesScoreType" default="0"/>
<xs:element name="Recoverability">
```

```
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="MechanismAvailable"/>
        <xs:element ref="MechanismEfficiency"/>
      </xs:sequence>
      <xs:attribute ref="rollup" use="optional"/>
      <xs:attribute ref="weight" use="optional"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="Maturity">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Volatility"/>
        <xs:element ref="FailureRemoval"/>
      </xs:sequence>
      <xs:attribute ref="rollup" use="optional"/>
      <xs:attribute ref="weight" use="optional"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="Volatility" type="AttributesScoreType" default="0"/>
  <xs:element name="FailureRemoval" type="AttributesScoreType" default="0"/>
  <xs:element name="CompositionUsability">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Configurability"/>
        <xs:element ref="Understandability"/>
        <xs:element name="Operability">
          <xs:complexType>
            <xs:sequence>
              <xs:element ref="Parameters"/>
              <xs:element ref="RequiredParameters"/>
              <xs:element ref="EffortForOperating"/>
            </xs:sequence>
            <xs:attribute ref="rollup" use="optional"/>
            <xs:attribute ref="weight" use="optional"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute ref="rollup" use="optional"/>
      <xs:attribute ref="weight" use="optional"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="Configurability">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="EffortToConfigure"/>
      </xs:sequence>
      <xs:attribute ref="rollup" use="optional"/>
      <xs:attribute ref="weight" use="optional"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="EffortToConfigure" type="AttributesScoreType" default="0"/>
  <xs:element name="Understandability">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="DocumentationQuality"/>
      </xs:sequence>
      <xs:attribute ref="rollup" use="optional"/>
```

```
    <xs:attribute ref="weight" use="optional"/>
  </xs:complexType>
 </xs:element>
 <xs:element name="DocumentationQuality" type="AttributesScoreType" default="0"/>
 <xs:element name="Parameters" type="AttributesScoreType" default="0"/>
 <xs:element name="RequiredParameters" type="AttributesScoreType" default="0"/>
 <xs:element name="EffortForOperating" type="AttributesScoreType" default="0"/>
 <xs:element name="CapabilityUsability">
  <xs:complexType>
   <xs:sequence>
    <xs:element ref="Configurability"/>
    <xs:element ref="Understandability"/>
    <xs:element name="Operability">
     <xs:complexType>
      <xs:sequence>
       <xs:element ref="Parameters"/>
       <xs:element ref="RequiredParameters"/>
      </xs:sequence>
      <xs:attribute ref="rollup" use="optional"/>
      <xs:attribute ref="weight" use="optional"/>
     </xs:complexType>
    </xs:element>
   </xs:sequence>
   <xs:attribute ref="rollup" use="optional"/>
   <xs:attribute ref="weight" use="optional"/>
  </xs:complexType>
 </xs:element>
 <xs:element name="Efficiency">
  <xs:complexType>
   <xs:sequence>
    <xs:element ref="TimeBehavior"/>
    <xs:element ref="ResourceUtilization"/>
    <xs:element ref="Scalability"/>
   </xs:sequence>
   <xs:attribute ref="rollup" use="optional"/>
   <xs:attribute ref="weight" use="optional"/>
  </xs:complexType>
 </xs:element>
 <xs:element name="TimeBehavior">
  <xs:complexType>
   <xs:sequence>
    <xs:element ref="ResponseTime"/>
    <xs:element ref="Throughput"/>
   </xs:sequence>
   <xs:attribute ref="rollup" use="optional"/>
   <xs:attribute ref="weight" use="optional"/>
  </xs:complexType>
 </xs:element>
 <xs:element name="ResponseTime" type="AttributesScoreType" default="0"/>
 <xs:element name="Throughput" type="AttributesScoreType" default="0"/>
 <xs:element name="ResourceUtilization">
  <xs:complexType>
   <xs:sequence>
    <xs:element ref="MemoryUsage"/>
    <xs:element ref="DiskUsage"/>
    <xs:element ref="NetworkUsage"/>
   </xs:sequence>
   <xs:attribute ref="rollup" use="optional"/>
```

```
      <xs:attribute ref="weight" use="optional"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="MemoryUsage" type="AttributesScoreType" default="0"/>
  <xs:element name="DiskUsage" type="AttributesScoreType" default="0"/>
  <xs:element name="NetworkUsage" type="AttributesScoreType" default="0"/>
  <xs:element name="Scalability">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="ProcessingCapacity"/>
      </xs:sequence>
      <xs:attribute ref="rollup" use="optional"/>
      <xs:attribute ref="weight" use="optional"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="ProcessingCapacity" type="AttributesScoreType" default="0"/>
  <xs:element name="Stability">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="ModificationsOverTime"/>
      </xs:sequence>
      <xs:attribute ref="rollup" use="optional"/>
      <xs:attribute ref="weight" use="optional"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="ModificationsOverTime" type="AttributesScoreType" default="0"/>
  <xs:element name="ComplexityLevel" type="AttributesScoreType" default="0"/>
  <xs:element name="Cohesion" type="AttributesScoreType" default="0"/>
  <xs:element name="Coupling" type="AttributesScoreType" default="0"/>
  <xs:element name="TestCasesProofs" type="AttributesScoreType" default="0"/>
  <xs:element name="Analyzeability" type="AttributesScoreType" default="0"/>
  <xs:element name="Portability">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="CompositionDeployability"/>
        <xs:element ref="CapabilityDeployability"/>
        <xs:element ref="Coexistence"/>
      </xs:sequence>
      <xs:attribute ref="rollup" use="optional"/>
      <xs:attribute ref="weight" use="optional"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="CompositionDeployability">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="DataCompatibility"/>
        <xs:element ref="CompositionDeploymentAsWebserviceComplexity"/>
      </xs:sequence>
      <xs:attribute ref="rollup" use="optional"/>
      <xs:attribute ref="weight" use="optional"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="CompositionDeploymentAsWebserviceComplexity" type="AttributesScoreType"
default="0"/>
  <xs:element name="CapabilityDeployability">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="DataCompatibility"/>
```

```
      <xs:element ref="CapabilityDeploymentDifficulty"/>
    </xs:sequence>
    <xs:attribute ref="rollup" use="optional"/>
    <xs:attribute ref="weight" use="optional"/>
  </xs:complexType>
</xs:element>
<xs:element name="CapabilityDeploymentDifficulty" type="AttributesScoreType" default="0"/>
<xs:element name="Coexistence" type="AttributesScoreType" default="0"/>
<xs:element name="QualityInUse">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="CompositionEffectiveness"/>
      <xs:element ref="CapabilityEffectiveness"/>
      <xs:element ref="CapabilityProductivity"/>
      <xs:element ref="CapabilitySatisfaction"/>
      <xs:element ref="CapabilityAttractiveness"/>
    </xs:sequence>
    <xs:attribute ref="rollup" use="optional"/>
    <xs:attribute ref="weight" use="optional"/>
  </xs:complexType>
</xs:element>
<xs:element name="CompositionEffectiveness" type="AttributesScoreType" default="0"/>
<xs:element name="CapabilityEffectiveness" type="AttributesScoreType" default="0"/>
<xs:element name="CapabilityProductivity" type="AttributesScoreType" default="0"/>
<xs:element name="CapabilitySatisfaction" type="AttributesScoreType" default="0"/>
<xs:element name="CapabilityAttractiveness" type="AttributesScoreType" default="0"/>
<xs:element name="Composer">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Name"/>
      <xs:element name="Proficiency">
        <xs:complexType>
          <xs:sequence>
            <xs:element ref="TrainingPerformed"/>
            <xs:element ref="AverageCompositionComplexity"/>
          </xs:sequence>
          <xs:attribute ref="rollup" use="optional"/>
          <xs:attribute ref="weight" use="optional"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute ref="rollup" use="optional"/>
    <xs:attribute ref="weight" use="optional"/>
  </xs:complexType>
</xs:element>
<xs:element name="TrainingPerformed" type="AttributesScoreType" default="0"/>
<xs:element name="AverageCompositionComplexity" type="AttributesScoreType" default="0"/>
<xs:element name="MaintenanceEvent">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Type"/>
      <xs:element ref="Date"/>
      <xs:element ref="ComponentCompositionMaintainer"/>
    </xs:sequence>
    <xs:attribute ref="rollup" use="optional"/>
    <xs:attribute ref="weight" use="optional"/>
  </xs:complexType>
</xs:element>
```

```
<xs:element name="Type" type="xs:string"/>
<xs:element name="Date" type="xs:date"/>
<xs:element name="ComponentCompositionMaintainer">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="Name"/>
   <xs:element name="Proficiency">
    <xs:complexType>
     <xs:sequence>
      <xs:element ref="TrainingPerformed"/>
      <xs:element ref="AverageComponentComplexity"/>
      <xs:element ref="AverageCompositionComplexity"/>
     </xs:sequence>
     <xs:attribute ref="rollup" use="optional"/>
     <xs:attribute ref="weight" use="optional"/>
    </xs:complexType>
   </xs:element>
  </xs:sequence>
  <xs:attribute ref="rollup" use="optional"/>
  <xs:attribute ref="weight" use="optional"/>
 </xs:complexType>
</xs:element>
<xs:element name="AverageComponentComplexity" type="AttributesScoreType" default="0"/>
<xs:element name="Component">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="Name"/>
   <xs:element name="Functionality">
    <xs:complexType>
     <xs:sequence>
      <xs:element ref="Accuracy"/>
      <xs:element name="Suitability">
       <xs:complexType>
        <xs:sequence>
         <xs:element ref="Coverage"/>
         <xs:element ref="Completeness"/>
         <xs:element ref="PrePostConditioned"/>
         <xs:element ref="PrePostConditionsProofs"/>
        </xs:sequence>
        <xs:attribute ref="rollup" use="optional"/>
        <xs:attribute ref="weight" use="optional"/>
       </xs:complexType>
      </xs:element>
      <xs:element ref="FunctionalityCompliance"/>
      <xs:element ref="SelfContained"/>
      <xs:element ref="InternalInteroperability"/>
     </xs:sequence>
     <xs:attribute ref="rollup" use="optional"/>
     <xs:attribute ref="weight" use="optional"/>
    </xs:complexType>
   </xs:element>
   <xs:element ref="Security"/>
   <xs:element ref="Reliability"/>
   <xs:element ref="Usability"/>
   <xs:element ref="Efficiency"/>
   <xs:element name="Maintainability">
    <xs:complexType>
     <xs:sequence>
```

```
            <xs:element ref="Stability"/>
            <xs:element name="Changeability">
              <xs:complexType>
                <xs:sequence>
                  <xs:element ref="ComplexityLevel"/>
                  <xs:element ref="Cohesion"/>
                  <xs:element ref="Coupling"/>
                </xs:sequence>
                <xs:attribute ref="rollup" use="optional"/>
                <xs:attribute ref="weight" use="optional"/>
              </xs:complexType>
            </xs:element>
            <xs:element name="Testability">
              <xs:complexType>
                <xs:sequence>
                  <xs:element ref="ExtensiveTestCases"/>
                </xs:sequence>
                <xs:attribute ref="rollup" use="optional"/>
                <xs:attribute ref="weight" use="optional"/>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
          <xs:attribute ref="rollup" use="optional"/>
          <xs:attribute ref="weight" use="optional"/>
        </xs:complexType>
      </xs:element>
      <xs:element name="QualityInUse" minOccurs="0" maxOccurs="1">
        <xs:complexType>
          <xs:sequence>
            <xs:element ref="Effectiveness"/>
          </xs:sequence>
          <xs:attribute ref="rollup" use="optional"/>
          <xs:attribute ref="weight" use="optional"/>
        </xs:complexType>
      </xs:element>
      <xs:element ref="ComponentDeveloper" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="MaintenanceEvent" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute ref="rollup" use="optional"/>
    <xs:attribute ref="weight" use="optional"/>
  </xs:complexType>
</xs:element>
<xs:element name="Security">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="RunTimeSecureness"/>
    </xs:sequence>
    <xs:attribute ref="rollup" use="optional"/>
    <xs:attribute ref="weight" use="optional"/>
  </xs:complexType>
</xs:element>
<xs:element name="RunTimeSecureness">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="InsecureComponentInteraction"/>
      <xs:element ref="RiskyResourceManagement"/>
      <xs:element ref="PreventionOfPorousDefenses"/>
    </xs:sequence>
```

```
      <xs:attribute ref="rollup" use="optional"/>
      <xs:attribute ref="weight" use="optional"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="InsecureComponentInteraction">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="CrossSiteScripting"/>
        <xs:element ref="SQLInjection"/>
        <xs:element ref="CrossSiteRequestForgery"/>
        <xs:element ref="UnrestrictedUploadOfDangerousFileType"/>
        <xs:element ref="OSCommandInjection"/>
        <xs:element ref="ErrorMessageInformationExposure"/>
        <xs:element ref="URLRedirectionToUntrustedSite"/>
        <xs:element ref="RaceCondition"/>
      </xs:sequence>
      <xs:attribute ref="rollup" use="optional"/>
      <xs:attribute ref="weight" use="optional"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="CrossSiteScripting" type="AttributesScoreType" default="0"/>
  <xs:element name="SQLInjection" type="AttributesScoreType" default="0"/>
  <xs:element name="CrossSiteRequestForgery" type="AttributesScoreType" default="0"/>
  <xs:element name="UnrestrictedUploadOfDangerousFileType" type="AttributesScoreType" default="0"/>
  <xs:element name="OSCommandInjection" type="AttributesScoreType" default="0"/>
  <xs:element name="ErrorMessageInformationExposure" type="AttributesScoreType" default="0"/>
  <xs:element name="URLRedirectionToUntrustedSite" type="AttributesScoreType" default="0"/>
  <xs:element name="RaceCondition" type="AttributesScoreType" default="0"/>
  <xs:element name="RiskyResourceManagement">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="ClassicBufferOverflow"/>
        <xs:element ref="PathTraversal"/>
        <xs:element ref="BufferAccessWithIncorrectLengthValue"/>
        <xs:element ref="ImproperCheckForExceptionalConditions"/>
        <xs:element ref="PHPFileInclusion"/>
        <xs:element ref="ImproperValidationOfArrayIndex"/>
        <xs:element ref="IntegerOverflowOrWraparound"/>
        <xs:element ref="IncorrectCalculationOfBufferSize"/>
        <xs:element ref="DownloadOfCodeWithoutIntegrityCheck"/>
        <xs:element ref="AllocationOfResourcesWithoutLimitsOrThrottling"/>
      </xs:sequence>
      <xs:attribute ref="rollup" use="optional"/>
      <xs:attribute ref="weight" use="optional"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="ClassicBufferOverflow" type="AttributesScoreType" default="0"/>
  <xs:element name="PathTraversal" type="AttributesScoreType" default="0"/>
  <xs:element name="BufferAccessWithIncorrectLengthValue" type="AttributesScoreType" default="0"/>
  <xs:element name="ImproperCheckForExceptionalConditions" type="AttributesScoreType" default="0"/>
  <xs:element name="PHPFileInclusion" type="AttributesScoreType" default="0"/>
  <xs:element name="ImproperValidationOfArrayIndex" type="AttributesScoreType" default="0"/>
  <xs:element name="IntegerOverflowOrWraparound" type="AttributesScoreType" default="0"/>
  <xs:element name="IncorrectCalculationOfBufferSize" type="AttributesScoreType" default="0"/>
  <xs:element name="DownloadOfCodeWithoutIntegrityCheck" type="AttributesScoreType" default="0"/>
  <xs:element name="AllocationOfResourcesWithoutLimitsOrThrottling" type="AttributesScoreType"
default="0"/>
  <xs:element name="PreventionOfPorousDefenses">
```

```
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="ImproperAccessControl"/>
        <xs:element ref="RelianceOnUntrustedInputsInSecurityDecision"/>
        <xs:element ref="MissingEncryptionOfSensitiveData"/>
        <xs:element ref="UseOfHardcodedCredentials"/>
        <xs:element ref="MissingAuthenticationForCriticalFunction"/>
        <xs:element ref="IncorrectPermissionAssignmentForCriticalResources"/>
        <xs:element ref="UseOfInsecureCryptographicAlgorithm"/>
      </xs:sequence>
      <xs:attribute ref="rollup" use="optional"/>
      <xs:attribute ref="weight" use="optional"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="ImproperAccessControl" type="AttributesScoreType" default="0"/>
  <xs:element name="RelianceOnUntrustedInputsInSecurityDecision" type="AttributesScoreType" default="0"/>
  <xs:element name="MissingEncryptionOfSensitiveData" type="AttributesScoreType" default="0"/>
  <xs:element name="UseOfHardcodedCredentials" type="AttributesScoreType" default="0"/>
  <xs:element name="MissingAuthenticationForCriticalFunction" type="AttributesScoreType" default="0"/>
  <xs:element name="IncorrectPermissionAssignmentForCriticalResources" type="AttributesScoreType"
default="0"/>
  <xs:element name="UseOfInsecureCryptographicAlgorithm" type="AttributesScoreType" default="0"/>
  <xs:element name="Effectiveness" type="AttributesScoreType" default="0"/>
  <xs:element name="InternalInteroperability">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="DataCompatibility"/>
        <xs:element ref="InterfaceComplexity"/>
      </xs:sequence>
      <xs:attribute ref="rollup" use="optional"/>
      <xs:attribute ref="weight" use="optional"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="Usability">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Configurability"/>
        <xs:element ref="Understandability"/>
        <xs:element name="Operability">
          <xs:complexType>
            <xs:sequence>
              <xs:element ref="Parameters"/>
              <xs:element ref="RequiredParameters"/>
              <xs:element ref="EffortForOperating"/>
            </xs:sequence>
            <xs:attribute ref="rollup" use="optional"/>
            <xs:attribute ref="weight" use="optional"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute ref="rollup" use="optional"/>
      <xs:attribute ref="weight" use="optional"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="ExtensiveTestCases" type="AttributesScoreType" default="0"/>
  <xs:element name="ComponentDeveloper">
    <xs:complexType>
      <xs:sequence>
```

```
      <xs:element ref="Name"/>
      <xs:element name="Proficiency">
       <xs:complexType>
        <xs:sequence>
         <xs:element ref="TrainingPerformed"/>
         <xs:element ref="AverageComponentComplexity"/>
        </xs:sequence>
        <xs:attribute ref="rollup" use="optional"/>
        <xs:attribute ref="weight" use="optional"/>
       </xs:complexType>
      </xs:element>
     </xs:sequence>
     <xs:attribute ref="rollup" use="optional"/>
     <xs:attribute ref="weight" use="optional"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="CCODEnvironment">
   <xs:complexType>
    <xs:sequence>
     <xs:element ref="Name"/>
     <xs:element name="Functionality">
      <xs:complexType>
       <xs:sequence>
        <xs:element ref="Accuracy"/>
        <xs:element name="Suitability">
         <xs:complexType>
          <xs:sequence>
           <xs:element ref="Coverage"/>
           <xs:element ref="Completeness"/>
          </xs:sequence>
          <xs:attribute ref="rollup" use="optional"/>
          <xs:attribute ref="weight" use="optional"/>
         </xs:complexType>
        </xs:element>
        <xs:element ref="FunctionalityCompliance"/>
        <xs:element name="InternalInteroperability">
         <xs:complexType>
          <xs:sequence>
           <xs:element ref="DataCompatibility"/>
           <xs:element ref="InteroperationComplexity"/>
          </xs:sequence>
          <xs:attribute ref="rollup" use="optional"/>
          <xs:attribute ref="weight" use="optional"/>
         </xs:complexType>
        </xs:element>
        <xs:element name="ExternalInteroperability">
         <xs:complexType>
          <xs:sequence>
           <xs:element ref="DataCompatibility"/>
           <xs:element ref="InteroperationComplexity"/>
          </xs:sequence>
          <xs:attribute ref="rollup" use="optional"/>
          <xs:attribute ref="weight" use="optional"/>
         </xs:complexType>
        </xs:element>
        <xs:element ref="LifeCycleManagement"/>
        <xs:element ref="ConfigurationManagement"/>
        <xs:element ref="PolicyManagement"/>
```

```
    </xs:sequence>
    <xs:attribute ref="rollup" use="optional"/>
    <xs:attribute ref="weight" use="optional"/>
   </xs:complexType>
  </xs:element>
  <xs:element name="Security">
   <xs:complexType>
    <xs:sequence>
     <xs:element ref="AccessControl"/>
     <xs:element ref="AuditAndAccountability"/>
     <xs:element ref="IdentificationAndAuthentication"/>
     <xs:element ref="SystemAndCommunicationsProtection"/>
     <xs:element ref="SystemAndInformationIntegrity"/>
     <xs:element ref="RunTimeSecureness"/>
    </xs:sequence>
    <xs:attribute ref="rollup" use="optional"/>
    <xs:attribute ref="weight" use="optional"/>
  </xs:complexType>
  </xs:element>
  <xs:element name="Reliability">
   <xs:complexType>
    <xs:sequence>
     <xs:element ref="FaultTolerance"/>
     <xs:element ref="Recoverability"/>
     <xs:element ref="Maturity"/>
     <xs:element ref="ReliabilityCompliance"/>
     <xs:element ref="EventMonitoring"/>
    </xs:sequence>
    <xs:attribute ref="rollup" use="optional"/>
    <xs:attribute ref="weight" use="optional"/>
   </xs:complexType>
  </xs:element>
  <xs:element name="Usability">
   <xs:complexType>
    <xs:sequence>
     <xs:element ref="Configurability"/>
     <xs:element ref="Understandability"/>
     <xs:element ref="Learnability"/>
     <xs:element name="Operability">
      <xs:complexType>
       <xs:sequence>
        <xs:element ref="ComplexityLevel"/>
       </xs:sequence>
       <xs:attribute ref="rollup" use="optional"/>
       <xs:attribute ref="weight" use="optional"/>
      </xs:complexType>
     </xs:element>
     <xs:element ref="DiscoverabilityComponentsCompositions"/>
     <xs:element ref="DiscoverabilityCapabilities"/>
     <xs:element ref="UsabilityCompliance"/>
    </xs:sequence>
    <xs:attribute ref="rollup" use="optional"/>
    <xs:attribute ref="weight" use="optional"/>
   </xs:complexType>
  </xs:element>
  <xs:element name="Efficiency">
   <xs:complexType>
    <xs:sequence>
```

```
            <xs:element ref="TimeBehavior"/>
            <xs:element ref="ResourceUtilization"/>
            <xs:element ref="EfficiencyCompliance"/>
            <xs:element ref="Accounting"/>
            <xs:element ref="QualityOfServiceManagement"/>
          </xs:sequence>
          <xs:attribute ref="rollup" use="optional"/>
          <xs:attribute ref="weight" use="optional"/>
        </xs:complexType>
      </xs:element>
      <xs:element name="Maintainability">
        <xs:complexType>
          <xs:sequence>
            <xs:element ref="Stability"/>
            <xs:element name="Changeability">
              <xs:complexType>
                <xs:sequence>
                  <xs:element ref="LevelofCustomizability"/>
                </xs:sequence>
                <xs:attribute ref="rollup" use="optional"/>
                <xs:attribute ref="weight" use="optional"/>
              </xs:complexType>
            </xs:element>
            <xs:element ref="TestabilityComponentsCompositions"/>
            <xs:element ref="MaintainabilityCompliance"/>
          </xs:sequence>
          <xs:attribute ref="rollup" use="optional"/>
          <xs:attribute ref="weight" use="optional"/>
        </xs:complexType>
      </xs:element>
      <xs:element name="Portability" type="AttributesScoreType" default="0"/>
      <xs:element name="QualityInUse" minOccurs="0" maxOccurs="1">
        <xs:complexType>
          <xs:sequence>
            <xs:element ref="Effectiveness"/>
            <xs:element ref="Productivity"/>
            <xs:element ref="Satisfaction"/>
            <xs:element ref="Attractiveness"/>
          </xs:sequence>
          <xs:attribute ref="rollup" use="optional"/>
          <xs:attribute ref="weight" use="optional"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute ref="rollup" use="optional"/>
    <xs:attribute ref="weight" use="optional"/>
  </xs:complexType>
</xs:element>
<xs:element name="InteroperationComplexity" type="AttributesScoreType" default="0"/>
<xs:element name="LifeCycleManagement" type="AttributesScoreType" default="0"/>
<xs:element name="ConfigurationManagement" type="AttributesScoreType" default="0"/>
<xs:element name="PolicyManagement" type="AttributesScoreType" default="0"/>
<xs:element name="AccessControl">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="AccountManagement"/>
      <xs:element ref="AccessEnforcement"/>
      <xs:element ref="InformationFlowEnforcement"/>
```

```
      <xs:element ref="UnsuccessfulLoginAttemptsLimitation"/>
      <xs:element ref="SystemUseNotification"/>
      <xs:element ref="PreviousLogonNotification"/>
      <xs:element ref="ConcurrentSessionControl"/>
      <xs:element ref="SessionLockMechanism"/>
      <xs:element ref="SecurityAttributeManagement"/>
    </xs:sequence>
    <xs:attribute ref="rollup" use="optional"/>
    <xs:attribute ref="weight" use="optional"/>
  </xs:complexType>
</xs:element>
<xs:element name="AccountManagement">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="AccountTypeIdentification"/>
      <xs:element ref="AuthorizedUserIdentification"/>
      <xs:element ref="AccountManagementMechanism"/>
      <xs:element ref="GuestAndTemporaryAccounts"/>
      <xs:element ref="AccessControlMechanism"/>
      <xs:element ref="InactiveAccounts"/>
      <xs:element ref="AccountAuditing"/>
    </xs:sequence>
    <xs:attribute ref="rollup" use="optional"/>
    <xs:attribute ref="weight" use="optional"/>
  </xs:complexType>
</xs:element>
<xs:element name="AccountTypeIdentification" type="AttributesScoreType" default="0"/>
<xs:element name="AuthorizedUserIdentification" type="AttributesScoreType" default="0"/>
<xs:element name="AccountManagementMechanism" type="AttributesScoreType" default="0"/>
<xs:element name="GuestAndTemporaryAccounts" type="AttributesScoreType" default="0"/>
<xs:element name="AccessControlMechanism" type="AttributesScoreType" default="0"/>
<xs:element name="InactiveAccounts" type="AttributesScoreType" default="0"/>
<xs:element name="AccountAuditing" type="AttributesScoreType" default="0"/>
<xs:element name="AccessEnforcement">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="RoleBasedAccessControl"/>
    </xs:sequence>
    <xs:attribute ref="rollup" use="optional"/>
    <xs:attribute ref="weight" use="optional"/>
  </xs:complexType>
</xs:element>
<xs:element name="RoleBasedAccessControl" type="AttributesScoreType" default="0"/>
<xs:element name="InformationFlowEnforcement">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="InformationFlowEnforcementMechanism"/>
      <xs:element ref="ProtectedProcessingDomains"/>
      <xs:element ref="DynamicFlowControl"/>
      <xs:element ref="EncryptedDataBypass"/>
      <xs:element ref="EmbeddedDataTypes"/>
      <xs:element ref="MetadataFlowControl"/>
      <xs:element ref="SecurityPolicyFilters"/>
      <xs:element ref="HumanReviewMechanism"/>
      <xs:element ref="PolicyFilterControl"/>
      <xs:element ref="InterconnectedSystems"/>
      <xs:element ref="SecurityAttributes"/>
    </xs:sequence>
```

```
    <xs:attribute ref="rollup" use="optional"/>
    <xs:attribute ref="weight" use="optional"/>
   </xs:complexType>
  </xs:element>
  <xs:element name="InformationFlowEnforcementMechanism" type="AttributesScoreType" default="0"/>
  <xs:element name="ProtectedProcessingDomains" type="AttributesScoreType" default="0"/>
  <xs:element name="DynamicFlowControl" type="AttributesScoreType" default="0"/>
  <xs:element name="EncryptedDataBypass" type="AttributesScoreType" default="0"/>
  <xs:element name="EmbeddedDataTypes" type="AttributesScoreType" default="0"/>
  <xs:element name="MetadataFlowControl" type="AttributesScoreType" default="0"/>
  <xs:element name="SecurityPolicyFilters" type="AttributesScoreType" default="0"/>
  <xs:element name="HumanReviewMechanism" type="AttributesScoreType" default="0"/>
  <xs:element name="PolicyFilterControl" type="AttributesScoreType" default="0"/>
  <xs:element name="InterconnectedSystems" type="AttributesScoreType" default="0"/>
  <xs:element name="SecurityAttributes" type="AttributesScoreType" default="0"/>
  <xs:element name="UnsuccessfulLoginAttemptsLimitation" type="AttributesScoreType" default="0"/>
  <xs:element name="SystemUseNotification">
   <xs:complexType>
    <xs:sequence>
     <xs:element ref="NotificationMessageMechanism"/>
     <xs:element ref="RetentionMechanism"/>
     <xs:element ref="PublicAccessMechanism"/>
    </xs:sequence>
    <xs:attribute ref="rollup" use="optional"/>
    <xs:attribute ref="weight" use="optional"/>
   </xs:complexType>
  </xs:element>
  <xs:element name="NotificationMessageMechanism" type="AttributesScoreType" default="0"/>
  <xs:element name="RetentionMechanism" type="AttributesScoreType" default="0"/>
  <xs:element name="PublicAccessMechanism" type="AttributesScoreType" default="0"/>
  <xs:element name="PreviousLogonNotification">
   <xs:complexType>
    <xs:sequence>
     <xs:element ref="LastLogonNotification"/>
     <xs:element ref="UnsuccessfulLogonNotification"/>
     <xs:element ref="AttemptsOverTime"/>
     <xs:element ref="AccountChangeNotification"/>
    </xs:sequence>
    <xs:attribute ref="rollup" use="optional"/>
    <xs:attribute ref="weight" use="optional"/>
   </xs:complexType>
  </xs:element>
  <xs:element name="LastLogonNotification" type="AttributesScoreType" default="0"/>
  <xs:element name="UnsuccessfulLogonNotification" type="AttributesScoreType" default="0"/>
  <xs:element name="AttemptsOverTime" type="AttributesScoreType" default="0"/>
  <xs:element name="AccountChangeNotification" type="AttributesScoreType" default="0"/>
  <xs:element name="ConcurrentSessionControl" type="AttributesScoreType" default="0"/>
  <xs:element name="SessionLockMechanism" type="AttributesScoreType" default="0"/>
  <xs:element name="SecurityAttributeManagement">
   <xs:complexType>
    <xs:sequence>
     <xs:element ref="BindingSecurityAttributes"/>
     <xs:element ref="DynamicReconfigurationSecurityAttributes"/>
     <xs:element ref="ChangeToSecurityAttributes"/>
     <xs:element ref="AssuranceForSecurityAttributes"/>
     <xs:element ref="AssociationOfSecurityAttributes"/>
     <xs:element ref="DisplaySecurityAttributes"/>
    </xs:sequence>
```

```
    <xs:attribute ref="rollup" use="optional"/>
    <xs:attribute ref="weight" use="optional"/>
  </xs:complexType>
</xs:element>
<xs:element name="BindingSecurityAttributes" type="AttributesScoreType" default="0"/>
<xs:element name="DynamicReconfigurationSecurityAttributes" type="AttributesScoreType" default="0"/>
<xs:element name="ChangeToSecurityAttributes" type="AttributesScoreType" default="0"/>
<xs:element name="AssuranceForSecurityAttributes" type="AttributesScoreType" default="0"/>
<xs:element name="AssociationOfSecurityAttributes" type="AttributesScoreType" default="0"/>
<xs:element name="DisplaySecurityAttributes" type="AttributesScoreType" default="0"/>
<xs:element name="AuditAndAccountability">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="ContentOfAuditRecords"/>
      <xs:element ref="ResponseToAuditProcessingFailures"/>
      <xs:element ref="AuditReviewAnalysisReporting"/>
      <xs:element ref="AuditReductionAndReportGeneration"/>
      <xs:element ref="AuditRecordTimeStamp"/>
      <xs:element ref="ProtectionOfAuditInformation"/>
      <xs:element ref="NonRepudiation"/>
      <xs:element ref="AuditGeneration"/>
      <xs:element ref="SessionAudit"/>
    </xs:sequence>
    <xs:attribute ref="rollup" use="optional"/>
    <xs:attribute ref="weight" use="optional"/>
  </xs:complexType>
</xs:element>
<xs:element name="ContentOfAuditRecords" type="AttributesScoreType" default="0"/>
<xs:element name="ResponseToAuditProcessingFailures">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="AuditStorageFailure"/>
      <xs:element ref="AuditFailureAlert"/>
      <xs:element ref="AuditTrafficControl"/>
      <xs:element ref="AuditFailureResponse"/>
    </xs:sequence>
    <xs:attribute ref="rollup" use="optional"/>
    <xs:attribute ref="weight" use="optional"/>
  </xs:complexType>
</xs:element>
<xs:element name="AuditStorageFailure" type="AttributesScoreType" default="0"/>
<xs:element name="AuditFailureAlert" type="AttributesScoreType" default="0"/>
<xs:element name="AuditTrafficControl" type="AttributesScoreType" default="0"/>
<xs:element name="AuditFailureResponse" type="AttributesScoreType" default="0"/>
<xs:element name="AuditReviewAnalysisReporting">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="AuditIntegration"/>
      <xs:element ref="AuditRecordCentralization"/>
    </xs:sequence>
    <xs:attribute ref="rollup" use="optional"/>
    <xs:attribute ref="weight" use="optional"/>
  </xs:complexType>
</xs:element>
<xs:element name="AuditIntegration" type="AttributesScoreType" default="0"/>
<xs:element name="AuditRecordCentralization" type="AttributesScoreType" default="0"/>
<xs:element name="AuditReductionAndReportGeneration" type="AttributesScoreType" default="0"/>
<xs:element name="AuditRecordTimeStamp" type="AttributesScoreType" default="0"/>
```

```
<xs:element name="ProtectionOfAuditInformation">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="ProtectionMechanism"/>
      <xs:element ref="WriteOnceMedia"/>
      <xs:element ref="BackupMechanism"/>
      <xs:element ref="EncryptionMechanism"/>
    </xs:sequence>
    <xs:attribute ref="rollup" use="optional"/>
    <xs:attribute ref="weight" use="optional"/>
  </xs:complexType>
</xs:element>
<xs:element name="ProtectionMechanism" type="AttributesScoreType" default="0"/>
<xs:element name="WriteOnceMedia" type="AttributesScoreType" default="0"/>
<xs:element name="BackupMechanism" type="AttributesScoreType" default="0"/>
<xs:element name="EncryptionMechanism" type="AttributesScoreType" default="0"/>
<xs:element name="NonRepudiation">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="ProducerIdentityMechanism"/>
      <xs:element ref="ProducerIdentityBindingValidation"/>
      <xs:element ref="ReviewerIdentityMechanism"/>
      <xs:element ref="ReviewerIdentityBindingValidation"/>
    </xs:sequence>
    <xs:attribute ref="rollup" use="optional"/>
    <xs:attribute ref="weight" use="optional"/>
  </xs:complexType>
</xs:element>
<xs:element name="ProducerIdentityMechanism" type="AttributesScoreType" default="0"/>
<xs:element name="ProducerIdentityBindingValidation" type="AttributesScoreType" default="0"/>
<xs:element name="ReviewerIdentityMechanism" type="AttributesScoreType" default="0"/>
<xs:element name="ReviewerIdentityBindingValidation" type="AttributesScoreType" default="0"/>
<xs:element name="AuditGeneration">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="ContentControlMechanism"/>
      <xs:element ref="TimeCorrelationMechanism"/>
      <xs:element ref="StandardizedFormatMechanism"/>
    </xs:sequence>
    <xs:attribute ref="rollup" use="optional"/>
    <xs:attribute ref="weight" use="optional"/>
  </xs:complexType>
</xs:element>
<xs:element name="ContentControlMechanism" type="AttributesScoreType" default="0"/>
<xs:element name="TimeCorrelationMechanism" type="AttributesScoreType" default="0"/>
<xs:element name="StandardizedFormatMechanism" type="AttributesScoreType" default="0"/>
<xs:element name="SessionAudit" type="AttributesScoreType" default="0"/>
<xs:element name="IdentificationAndAuthentication">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="IdentificationAndAuthenticationMechanisms"/>
      <xs:element ref="AuthenticatorManagement"/>
      <xs:element ref="AuthenticatorFeedbackMechanism"/>
    </xs:sequence>
    <xs:attribute ref="rollup" use="optional"/>
    <xs:attribute ref="weight" use="optional"/>
  </xs:complexType>
</xs:element>
```

```
<xs:element name="IdentificationAndAuthenticationMechanisms">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="PrivilegedNetworkAccess"/>
      <xs:element ref="NonprivilegedNetworkAccess"/>
      <xs:element ref="PrivilegedLocalAccess"/>
      <xs:element ref="NonprivilegedLocalAccess"/>
      <xs:element ref="PrivilegedNetworkAccessReplayResistance"/>
      <xs:element ref="NonprivilegedNetworkAccessReplayResistance"/>
    </xs:sequence>
    <xs:attribute ref="rollup" use="optional"/>
    <xs:attribute ref="weight" use="optional"/>
  </xs:complexType>
</xs:element>
<xs:element name="PrivilegedNetworkAccess" type="AttributesScoreType" default="0"/>
<xs:element name="NonprivilegedNetworkAccess" type="AttributesScoreType" default="0"/>
<xs:element name="PrivilegedLocalAccess" type="AttributesScoreType" default="0"/>
<xs:element name="NonprivilegedLocalAccess" type="AttributesScoreType" default="0"/>
<xs:element name="PrivilegedNetworkAccessReplayResistance" type="AttributesScoreType" default="0"/>
<xs:element name="NonprivilegedNetworkAccessReplayResistance" type="AttributesScoreType" default="0"/>
<xs:element name="AuthenticatorManagement" type="AttributesScoreType" default="0"/>
<xs:element name="AuthenticatorFeedbackMechanism" type="AttributesScoreType" default="0"/>
<xs:element name="SystemAndCommunicationsProtection">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="ApplicationPartitioning"/>
      <xs:element ref="InformationInSharedResources"/>
      <xs:element ref="DenialOfServiceProtection"/>
      <xs:element ref="ResourcePriority"/>
      <xs:element ref="BoundaryProtection"/>
      <xs:element ref="NetworkDisconnect"/>
      <xs:element ref="UseOfCryptography"/>
      <xs:element ref="FailInKnownState"/>
      <xs:element ref="OperatingSystemIndependence"/>
    </xs:sequence>
    <xs:attribute ref="rollup" use="optional"/>
    <xs:attribute ref="weight" use="optional"/>
  </xs:complexType>
</xs:element>
<xs:element name="ApplicationPartitioning" type="AttributesScoreType" default="0"/>
<xs:element name="InformationInSharedResources" type="AttributesScoreType" default="0"/>
<xs:element name="DenialOfServiceProtection">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="RestrictionMechanism"/>
      <xs:element ref="ResourceManagementMechanism"/>
    </xs:sequence>
    <xs:attribute ref="rollup" use="optional"/>
    <xs:attribute ref="weight" use="optional"/>
  </xs:complexType>
</xs:element>
<xs:element name="RestrictionMechanism" type="AttributesScoreType" default="0"/>
<xs:element name="ResourceManagementMechanism" type="AttributesScoreType" default="0"/>
<xs:element name="ResourcePriority" type="AttributesScoreType" default="0"/>
<xs:element name="BoundaryProtection">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="IncomingFlowProtection"/>
```

```
      <xs:element ref="OutgoingFlowProtection"/>
      <xs:element ref="IncomingValidationMechanism"/>
      <xs:element ref="PrivilegedFlowMechanism"/>
    </xs:sequence>
    <xs:attribute ref="rollup" use="optional"/>
    <xs:attribute ref="weight" use="optional"/>
  </xs:complexType>
</xs:element>
<xs:element name="IncomingFlowProtection" type="AttributesScoreType" default="0"/>
<xs:element name="OutgoingFlowProtection" type="AttributesScoreType" default="0"/>
<xs:element name="IncomingValidationMechanism" type="AttributesScoreType" default="0"/>
<xs:element name="PrivilegedFlowMechanism" type="AttributesScoreType" default="0"/>
<xs:element name="NetworkDisconnect" type="AttributesScoreType" default="0"/>
<xs:element name="UseOfCryptography" type="AttributesScoreType" default="0"/>
<xs:element name="FailInKnownState" type="AttributesScoreType" default="0"/>
<xs:element name="OperatingSystemIndependence" type="AttributesScoreType" default="0"/>
<xs:element name="SystemAndInformationIntegrity">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="InformationSystemMonitoring"/>
      <xs:element ref="SecurityFunctionalityVerification"/>
      <xs:element ref="SoftwareAndInformationIntegrity"/>
      <xs:element ref="ErrorHandling"/>
    </xs:sequence>
    <xs:attribute ref="rollup" use="optional"/>
    <xs:attribute ref="weight" use="optional"/>
  </xs:complexType>
</xs:element>
<xs:element name="InformationSystemMonitoring">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="CommunicationMonitoring"/>
      <xs:element ref="RealTimeAlertMechanism"/>
      <xs:element ref="CircumventionMechanism"/>
      <xs:element ref="SuspiciousEventsMechanism"/>
    </xs:sequence>
    <xs:attribute ref="rollup" use="optional"/>
    <xs:attribute ref="weight" use="optional"/>
  </xs:complexType>
</xs:element>
<xs:element name="CommunicationMonitoring" type="AttributesScoreType" default="0"/>
<xs:element name="RealTimeAlertMechanism" type="AttributesScoreType" default="0"/>
<xs:element name="CircumventionMechanism" type="AttributesScoreType" default="0"/>
<xs:element name="SuspiciousEventsMechanism" type="AttributesScoreType" default="0"/>
<xs:element name="SecurityFunctionalityVerification" type="AttributesScoreType" default="0"/>
<xs:element name="SoftwareAndInformationIntegrity" type="AttributesScoreType" default="0"/>
<xs:element name="ErrorHandling" type="AttributesScoreType" default="0"/>
<xs:element name="ReliabilityCompliance" type="AttributesScoreType" default="0"/>
<xs:element name="EventMonitoring" type="AttributesScoreType" default="0"/>
<xs:element name="Learnability">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="TrainingMaterialQuality"/>
      <xs:element ref="CommonDevelopmentLanguage"/>
    </xs:sequence>
    <xs:attribute ref="rollup" use="optional"/>
    <xs:attribute ref="weight" use="optional"/>
  </xs:complexType>
```

```
  </xs:element>
  <xs:element name="TrainingMaterialQuality" type="AttributesScoreType" default="0"/>
  <xs:element name="CommonDevelopmentLanguage" type="AttributesScoreType" default="0"/>
  <xs:element name="DiscoverabilityComponentsCompositions" type="AttributesScoreType" default="0"/>
  <xs:element name="DiscoverabilityCapabilities" type="AttributesScoreType" default="0"/>
  <xs:element name="UsabilityCompliance" type="AttributesScoreType" default="0"/>
  <xs:element name="EfficiencyCompliance" type="AttributesScoreType" default="0"/>
  <xs:element name="Accounting" type="AttributesScoreType" default="0"/>
  <xs:element name="QualityOfServiceManagement" type="AttributesScoreType" default="0"/>
  <xs:element name="LevelofCustomizability" type="AttributesScoreType" default="0"/>
  <xs:element name="TestabilityComponentsCompositions">
   <xs:complexType>
    <xs:sequence>
     <xs:element ref="TestSuiteCapability"/>
    </xs:sequence>
    <xs:attribute ref="rollup" use="optional"/>
    <xs:attribute ref="weight" use="optional"/>
   </xs:complexType>
  </xs:element>
  <xs:element name="TestSuiteCapability" type="AttributesScoreType" default="0"/>
  <xs:element name="MaintainabilityCompliance" type="AttributesScoreType" default="0"/>
  <xs:element name="Productivity" type="AttributesScoreType" default="0"/>
  <xs:element name="Satisfaction" type="AttributesScoreType" default="0"/>
  <xs:element name="Attractiveness" type="AttributesScoreType" default="0"/>
  <xs:simpleType name="ScoreType">
   <xs:restriction base="xs:integer">
    <xs:minInclusive value="0"/>
    <xs:maxInclusive value="5"/>
   </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="AttributesScoreType">
   <xs:simpleContent>
    <xs:extension base="ScoreType">
     <xs:attribute ref="weight" use="optional"/>
    </xs:extension>
   </xs:simpleContent>
  </xs:complexType>
  <xs:simpleType name="WeightType">
   <xs:restriction base="xs:integer">
   </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="RollupType">
   <xs:restriction base="xs:integer">
   </xs:restriction>
  </xs:simpleType>
  <xs:attribute name="weight" type="WeightType"/>
  <xs:attribute name="rollup" type="RollupType"/>
</xs:schema>
```

E.2 XML Document for trust taxonomy metric values, rollups and weights

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<Composition xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="file:///C:/GovernanceCCOD/CCOD_Trust_Taxonomy_2_20.xsd">
  <Name>text</Name>
  <Functionality rollup="0" weight="0">
    <Accuracy rollup="0" weight="0">
      <Correctness weight="0">0</Correctness>
    </Accuracy>
    <Suitability rollup="0" weight="0">
      <Coverage weight="0">0</Coverage>
      <Completeness weight="0">0</Completeness>
      <PrePostConditioned weight="0">0</PrePostConditioned>
    </Suitability>
    <InternalInteroperability rollup="0" weight="0">
      <DataCompatibility weight="0">0</DataCompatibility>
      <InterfaceComplexity weight="0">0</InterfaceComplexity>
    </InternalInteroperability>
    <CapabilityInteroperability rollup="0" weight="0">
      <DataCompatibility weight="0">0</DataCompatibility>
      <InterfaceComplexity weight="0">0</InterfaceComplexity>
    </CapabilityInteroperability>
    <FunctionalityCompliance rollup="0" weight="0">
      <Standardization weight="0">0</Standardization>
    </FunctionalityCompliance>
    <SelfContained rollup="0" weight="0">
      <Dependant weight="0">0</Dependant>
    </SelfContained>
  </Functionality>
  <Reliability rollup="0" weight="0">
    <FaultTolerance rollup="0" weight="0">
      <MechanismAvailable weight="0">0</MechanismAvailable>
      <MechanismEfficiency weight="0">0</MechanismEfficiency>
    </FaultTolerance>
    <Recoverability rollup="0" weight="0">
      <MechanismAvailable weight="0">0</MechanismAvailable>
      <MechanismEfficiency weight="0">0</MechanismEfficiency>
    </Recoverability>
    <Maturity rollup="0" weight="0">
      <Volatility weight="0">0</Volatility>
      <FailureRemoval weight="0">0</FailureRemoval>
    </Maturity>
  </Reliability>
  <CompositionUsability rollup="0" weight="0">
    <Configurability rollup="0" weight="0">
      <EffortToConfigure weight="0">0</EffortToConfigure>
    </Configurability>
    <Understandability rollup="0" weight="0">
      <DocumentationQuality weight="0">0</DocumentationQuality>
    </Understandability>
    <Operability rollup="0" weight="0">
      <Parameters weight="0">0</Parameters>
      <RequiredParameters weight="0">0</RequiredParameters>
      <EffortForOperating weight="0">0</EffortForOperating>
    </Operability>
  </CompositionUsability>
  <CapabilityUsability rollup="0" weight="0">
```

```
    <Configurability rollup="0" weight="0">
      <EffortToConfigure weight="0">0</EffortToConfigure>
    </Configurability>
    <Understandability rollup="0" weight="0">
      <DocumentationQuality weight="0">0</DocumentationQuality>
    </Understandability>
    <Operability rollup="0" weight="0">
      <Parameters weight="0">0</Parameters>
      <RequiredParameters weight="0">0</RequiredParameters>
    </Operability>
  </CapabilityUsability>
  <Efficiency rollup="0" weight="0">
    <TimeBehavior rollup="0" weight="0">
      <ResponseTime weight="0">0</ResponseTime>
      <Throughput weight="0">0</Throughput>
    </TimeBehavior>
    <ResourceUtilization rollup="0" weight="0">
      <MemoryUsage weight="0">0</MemoryUsage>
      <DiskUsage weight="0">0</DiskUsage>
      <NetworkUsage weight="0">0</NetworkUsage>
    </ResourceUtilization>
    <Scalability rollup="0" weight="0">
      <ProcessingCapacity weight="0">0</ProcessingCapacity>
    </Scalability>
  </Efficiency>
  <Maintainability rollup="0" weight="0">
    <Stability rollup="0" weight="0">
      <ModificationsOverTime weight="0">0</ModificationsOverTime>
    </Stability>
    <Changeability rollup="0" weight="0">
      <ComplexityLevel weight="0">0</ComplexityLevel>
      <Cohesion weight="0">0</Cohesion>
      <Coupling weight="0">0</Coupling>
    </Changeability>
    <Testability rollup="0" weight="0">
      <TestCasesProofs weight="0">0</TestCasesProofs>
    </Testability>
    <Analyzeability weight="0">0</Analyzeability>
  </Maintainability>
  <Portability rollup="0" weight="0">
    <CompositionDeployability rollup="0" weight="0">
      <DataCompatibility weight="0">0</DataCompatibility>
      <CompositionDeploymentAsWebserviceComplexity
weight="0">0</CompositionDeploymentAsWebserviceComplexity>
    </CompositionDeployability>
    <CapabilityDeployability rollup="0" weight="0">
      <DataCompatibility weight="0">0</DataCompatibility>
      <CapabilityDeploymentDifficulty weight="0">0</CapabilityDeploymentDifficulty>
    </CapabilityDeployability>
    <Coexistence weight="0">0</Coexistence>
  </Portability>
  <QualityInUse rollup="0" weight="0">
    <CompositionEffectiveness weight="0">0</CompositionEffectiveness>
    <CapabilityEffectiveness weight="0">0</CapabilityEffectiveness>
    <CapabilityProductivity weight="0">0</CapabilityProductivity>
    <CapabilitySatisfaction weight="0">0</CapabilitySatisfaction>
    <CapabilityAttractiveness weight="0">0</CapabilityAttractiveness>
  </QualityInUse>
```

```
<Composer rollup="0" weight="0">
  <Name>text</Name>
  <Proficiency rollup="0" weight="0">
    <TrainingPerformed weight="0">0</TrainingPerformed>
    <AverageCompositionComplexity weight="0">0</AverageCompositionComplexity>
  </Proficiency>
</Composer>
<MaintenanceEvent rollup="0" weight="0">
  <Type>text</Type>
  <Date>2010-06-01</Date>
  <ComponentCompositionMaintainer rollup="0" weight="0">
    <Name>text</Name>
    <Proficiency rollup="0" weight="0">
      <TrainingPerformed weight="0">0</TrainingPerformed>
      <AverageComponentComplexity weight="0">0</AverageComponentComplexity>
      <AverageCompositionComplexity weight="0">0</AverageCompositionComplexity>
    </Proficiency>
  </ComponentCompositionMaintainer>
</MaintenanceEvent>
<Composition rollup="0" weight="0">
  <Name>text</Name>
  <Functionality rollup="0" weight="0">
    <Accuracy rollup="0" weight="0">
      <Correctness weight="0">0</Correctness>
    </Accuracy>
    <Suitability rollup="0" weight="0">
      <Coverage weight="0">0</Coverage>
      <Completeness weight="0">0</Completeness>
      <PrePostConditioned weight="0">0</PrePostConditioned>
    </Suitability>
    <InternalInteroperability rollup="0" weight="0">
      <DataCompatibility weight="0">0</DataCompatibility>
      <InterfaceComplexity weight="0">0</InterfaceComplexity>
    </InternalInteroperability>
    <CapabilityInteroperability rollup="0" weight="0">
      <DataCompatibility weight="0">0</DataCompatibility>
      <InterfaceComplexity weight="0">0</InterfaceComplexity>
    </CapabilityInteroperability>
    <FunctionalityCompliance rollup="0" weight="0">
      <Standardization weight="0">0</Standardization>
    </FunctionalityCompliance>
    <SelfContained rollup="0" weight="0">
      <Dependant weight="0">0</Dependant>
    </SelfContained>
  </Functionality>
  <Reliability rollup="0" weight="0">
    <FaultTolerance rollup="0" weight="0">
      <MechanismAvailable weight="0">0</MechanismAvailable>
      <MechanismEfficiency weight="0">0</MechanismEfficiency>
    </FaultTolerance>
    <Recoverability rollup="0" weight="0">
      <MechanismAvailable weight="0">0</MechanismAvailable>
      <MechanismEfficiency weight="0">0</MechanismEfficiency>
    </Recoverability>
    <Maturity rollup="0" weight="0">
      <Volatility weight="0">0</Volatility>
      <FailureRemoval weight="0">0</FailureRemoval>
    </Maturity>
```

```
  </Reliability>
  <CompositionUsability rollup="0" weight="0">
    <Configurability rollup="0" weight="0">
      <EffortToConfigure weight="0">0</EffortToConfigure>
    </Configurability>
    <Understandability rollup="0" weight="0">
      <DocumentationQuality weight="0">0</DocumentationQuality>
    </Understandability>
    <Operability rollup="0" weight="0">
      <Parameters weight="0">0</Parameters>
      <RequiredParameters weight="0">0</RequiredParameters>
      <EffortForOperating weight="0">0</EffortForOperating>
    </Operability>
  </CompositionUsability>
  <CapabilityUsability rollup="0" weight="0">
    <Configurability rollup="0" weight="0">
      <EffortToConfigure weight="0">0</EffortToConfigure>
    </Configurability>
    <Understandability rollup="0" weight="0">
      <DocumentationQuality weight="0">0</DocumentationQuality>
    </Understandability>
    <Operability rollup="0" weight="0">
      <Parameters weight="0">0</Parameters>
      <RequiredParameters weight="0">0</RequiredParameters>
    </Operability>
  </CapabilityUsability>
  <Efficiency rollup="0" weight="0">
    <TimeBehavior rollup="0" weight="0">
      <ResponseTime weight="0">0</ResponseTime>
      <Throughput weight="0">0</Throughput>
    </TimeBehavior>
    <ResourceUtilization rollup="0" weight="0">
      <MemoryUsage weight="0">0</MemoryUsage>
      <DiskUsage weight="0">0</DiskUsage>
      <NetworkUsage weight="0">0</NetworkUsage>
    </ResourceUtilization>
    <Scalability rollup="0" weight="0">
      <ProcessingCapacity weight="0">0</ProcessingCapacity>
    </Scalability>
  </Efficiency>
  <Maintainability rollup="0" weight="0">
    <Stability rollup="0" weight="0">
      <ModificationsOverTime weight="0">0</ModificationsOverTime>
    </Stability>
    <Changeability rollup="0" weight="0">
      <ComplexityLevel weight="0">0</ComplexityLevel>
      <Cohesion weight="0">0</Cohesion>
      <Coupling weight="0">0</Coupling>
    </Changeability>
    <Testability rollup="0" weight="0">
      <TestCasesProofs weight="0">0</TestCasesProofs>
    </Testability>
    <Analyzeability weight="0">0</Analyzeability>
  </Maintainability>
  <Portability rollup="0" weight="0">
    <CompositionDeployability rollup="0" weight="0">
      <DataCompatibility weight="0">0</DataCompatibility>
```

```
      <CompositionDeploymentAsWebserviceComplexity
weight="0">0</CompositionDeploymentAsWebserviceComplexity>
    </CompositionDeployability>
    <CapabilityDeployability rollup="0" weight="0">
     <DataCompatibility weight="0">0</DataCompatibility>
     <CapabilityDeploymentDifficulty weight="0">0</CapabilityDeploymentDifficulty>
    </CapabilityDeployability>
    <Coexistence weight="0">0</Coexistence>
   </Portability>
   <QualityInUse rollup="0" weight="0">
    <CompositionEffectiveness weight="0">0</CompositionEffectiveness>
    <CapabilityEffectiveness weight="0">0</CapabilityEffectiveness>
    <CapabilityProductivity weight="0">0</CapabilityProductivity>
    <CapabilitySatisfaction weight="0">0</CapabilitySatisfaction>
    <CapabilityAttractiveness weight="0">0</CapabilityAttractiveness>
   </QualityInUse>
   <Composer rollup="0" weight="0">
    <Name>text</Name>
    <Proficiency rollup="0" weight="0">
     <TrainingPerformed weight="0">0</TrainingPerformed>
     <AverageCompositionComplexity weight="0">0</AverageCompositionComplexity>
    </Proficiency>
   </Composer>
   <MaintenanceEvent rollup="0" weight="0">
    <Type>text</Type>
    <Date>2010-06-01</Date>
    <ComponentCompositionMaintainer rollup="0" weight="0">
     <Name>text</Name>
     <Proficiency rollup="0" weight="0">
      <TrainingPerformed weight="0">0</TrainingPerformed>
      <AverageComponentComplexity weight="0">0</AverageComponentComplexity>
      <AverageCompositionComplexity weight="0">0</AverageCompositionComplexity>
     </Proficiency>
    </ComponentCompositionMaintainer>
   </MaintenanceEvent>
   <Component rollup="0" weight="0">
    <Name>text</Name>
    <Functionality rollup="0" weight="0">
     <Accuracy rollup="0" weight="0">
      <Correctness weight="0">0</Correctness>
     </Accuracy>
     <Suitability rollup="0" weight="0">
      <Coverage weight="0">0</Coverage>
      <Completeness weight="0">0</Completeness>
      <PrePostConditioned weight="0">0</PrePostConditioned>
      <PrePostConditionsProofs weight="0">0</PrePostConditionsProofs>
     </Suitability>
     <FunctionalityCompliance rollup="0" weight="0">
      <Standardization weight="0">0</Standardization>
      <Certification weight="0">0</Certification>
     </FunctionalityCompliance>
     <SelfContained rollup="0" weight="0">
      <Dependant weight="0">0</Dependant>
     </SelfContained>
     <InternalInteroperability rollup="0" weight="0">
      <DataCompatibility weight="0">0</DataCompatibility>
      <InterfaceComplexity weight="0">0</InterfaceComplexity>
     </InternalInteroperability>
```

```
    </Functionality>
    <Security rollup="0" weight="0">
      <RunTimeSecureness rollup="0" weight="0">
        <InsecureComponentInteraction rollup="0" weight="0">
          <CrossSiteScripting weight="0">0</CrossSiteScripting>
          <SQLInjection weight="0">0</SQLInjection>
          <CrossSiteRequestForgery weight="0">0</CrossSiteRequestForgery>
          <UnrestrictedUploadOfDangerousFileType weight="0">0</UnrestrictedUploadOfDangerousFileType>
          <OSCommandInjection weight="0">0</OSCommandInjection>
          <ErrorMessageInformationExposure weight="0">0</ErrorMessageInformationExposure>
          <URLRedirectionToUntrustedSite weight="0">0</URLRedirectionToUntrustedSite>
          <RaceCondition weight="0">0</RaceCondition>
        </InsecureComponentInteraction>
        <RiskyResourceManagement rollup="0" weight="0">
          <ClassicBufferOverflow weight="0">0</ClassicBufferOverflow>
          <PathTraversal weight="0">0</PathTraversal>
          <BufferAccessWithIncorrectLengthValue weight="0">0</BufferAccessWithIncorrectLengthValue>
          <ImproperCheckForExceptionalConditions weight="0">0</ImproperCheckForExceptionalConditions>
          <PHPFileInclusion weight="0">0</PHPFileInclusion>
          <ImproperValidationOfArrayIndex weight="0">0</ImproperValidationOfArrayIndex>
          <IntegerOverflowOrWraparound weight="0">0</IntegerOverflowOrWraparound>
          <IncorrectCalculationOfBufferSize weight="0">0</IncorrectCalculationOfBufferSize>
          <DownloadOfCodeWithoutIntegrityCheck weight="0">0</DownloadOfCodeWithoutIntegrityCheck>
          <AllocationOfResourcesWithoutLimitsOrThrottling
weight="0">0</AllocationOfResourcesWithoutLimitsOrThrottling>
        </RiskyResourceManagement>
        <PreventionOfPorousDefenses rollup="0" weight="0">
          <ImproperAccessControl weight="0">0</ImproperAccessControl>
          <RelianceOnUntrustedInputsInSecurityDecision
weight="0">0</RelianceOnUntrustedInputsInSecurityDecision>
          <MissingEncryptionOfSensitiveData weight="0">0</MissingEncryptionOfSensitiveData>
          <UseOfHardcodedCredentials weight="0">0</UseOfHardcodedCredentials>
          <MissingAuthenticationForCriticalFunction weight="0">0</MissingAuthenticationForCriticalFunction>
          <IncorrectPermissionAssignmentForCriticalResources
weight="0">0</IncorrectPermissionAssignmentForCriticalResources>
          <UseOfInsecureCryptographicAlgorithm weight="0">0</UseOfInsecureCryptographicAlgorithm>
        </PreventionOfPorousDefenses>
      </RunTimeSecureness>
    </Security>
    <Reliability rollup="0" weight="0">
      <FaultTolerance rollup="0" weight="0">
        <MechanismAvailable weight="0">0</MechanismAvailable>
        <MechanismEfficiency weight="0">0</MechanismEfficiency>
      </FaultTolerance>
      <Recoverability rollup="0" weight="0">
        <MechanismAvailable weight="0">0</MechanismAvailable>
        <MechanismEfficiency weight="0">0</MechanismEfficiency>
      </Recoverability>
      <Maturity rollup="0" weight="0">
        <Volatility weight="0">0</Volatility>
        <FailureRemoval weight="0">0</FailureRemoval>
      </Maturity>
    </Reliability>
    <Usability rollup="0" weight="0">
      <Configurability rollup="0" weight="0">
        <EffortToConfigure weight="0">0</EffortToConfigure>
      </Configurability>
      <Understandability rollup="0" weight="0">
```

```
      <DocumentationQuality weight="0">0</DocumentationQuality>
    </Understandability>
    <Operability rollup="0" weight="0">
      <Parameters weight="0">0</Parameters>
      <RequiredParameters weight="0">0</RequiredParameters>
      <EffortForOperating weight="0">0</EffortForOperating>
    </Operability>
  </Usability>
  <Efficiency rollup="0" weight="0">
    <TimeBehavior rollup="0" weight="0">
      <ResponseTime weight="0">0</ResponseTime>
      <Throughput weight="0">0</Throughput>
    </TimeBehavior>
    <ResourceUtilization rollup="0" weight="0">
      <MemoryUsage weight="0">0</MemoryUsage>
      <DiskUsage weight="0">0</DiskUsage>
      <NetworkUsage weight="0">0</NetworkUsage>
    </ResourceUtilization>
    <Scalability rollup="0" weight="0">
      <ProcessingCapacity weight="0">0</ProcessingCapacity>
    </Scalability>
  </Efficiency>
  <Maintainability rollup="0" weight="0">
    <Stability rollup="0" weight="0">
      <ModificationsOverTime weight="0">0</ModificationsOverTime>
    </Stability>
    <Changeability rollup="0" weight="0">
      <ComplexityLevel weight="0">0</ComplexityLevel>
      <Cohesion weight="0">0</Cohesion>
      <Coupling weight="0">0</Coupling>
    </Changeability>
    <Testability rollup="0" weight="0">
      <ExtensiveTestCases weight="0">0</ExtensiveTestCases>
    </Testability>
  </Maintainability>
  <QualityInUse rollup="0" weight="0">
    <Effectiveness weight="0">0</Effectiveness>
  </QualityInUse>
  <ComponentDeveloper rollup="0" weight="0">
    <Name>text</Name>
    <Proficiency rollup="0" weight="0">
      <TrainingPerformed weight="0">0</TrainingPerformed>
      <AverageComponentComplexity weight="0">0</AverageComponentComplexity>
    </Proficiency>
  </ComponentDeveloper>
  <MaintenanceEvent rollup="0" weight="0">
    <Type>text</Type>
    <Date>2010-06-01</Date>
    <ComponentCompositionMaintainer rollup="0" weight="0">
      <Name>text</Name>
      <Proficiency rollup="0" weight="0">
        <TrainingPerformed weight="0">0</TrainingPerformed>
        <AverageComponentComplexity weight="0">0</AverageComponentComplexity>
        <AverageCompositionComplexity weight="0">0</AverageCompositionComplexity>
      </Proficiency>
    </ComponentCompositionMaintainer>
  </MaintenanceEvent>
</Component>
```

```
<CCODEnvironment rollup="0" weight="0">
  <Name>text</Name>
  <Functionality rollup="0" weight="0">
    <Accuracy rollup="0" weight="0">
      <Correctness weight="0">0</Correctness>
    </Accuracy>
    <Suitability rollup="0" weight="0">
      <Coverage weight="0">0</Coverage>
      <Completeness weight="0">0</Completeness>
    </Suitability>
    <FunctionalityCompliance rollup="0" weight="0">
      <Standardization weight="0">0</Standardization>
      <Certification weight="0">0</Certification>
    </FunctionalityCompliance>
    <InternalInteroperability rollup="0" weight="0">
      <DataCompatibility weight="0">0</DataCompatibility>
      <InteroperationComplexity weight="0">0</InteroperationComplexity>
    </InternalInteroperability>
    <ExternalInteroperability rollup="0" weight="0">
      <DataCompatibility weight="0">0</DataCompatibility>
      <InteroperationComplexity weight="0">0</InteroperationComplexity>
    </ExternalInteroperability>
    <LifeCycleManagement weight="0">0</LifeCycleManagement>
    <ConfigurationManagement weight="0">0</ConfigurationManagement>
    <PolicyManagement weight="0">0</PolicyManagement>
  </Functionality>
  <Security rollup="0" weight="0">
    <AccessControl rollup="0" weight="0">
      <AccountManagement rollup="0" weight="0">
        <AccountTypeIdentification weight="0">0</AccountTypeIdentification>
        <AuthorizedUserIdentification weight="0">0</AuthorizedUserIdentification>
        <AccountManagementMechanism weight="0">0</AccountManagementMechanism>
        <GuestAndTemporaryAccounts weight="0">0</GuestAndTemporaryAccounts>
        <AccessControlMechanism weight="0">0</AccessControlMechanism>
        <InactiveAccounts weight="0">0</InactiveAccounts>
        <AccountAuditing weight="0">0</AccountAuditing>
      </AccountManagement>
      <AccessEnforcement rollup="0" weight="0">
        <RoleBasedAccessControl weight="0">0</RoleBasedAccessControl>
      </AccessEnforcement>
      <InformationFlowEnforcement rollup="0" weight="0">
        <InformationFlowEnforcementMechanism weight="0">0</InformationFlowEnforcementMechanism>
        <ProtectedProcessingDomains weight="0">0</ProtectedProcessingDomains>
        <DynamicFlowControl weight="0">0</DynamicFlowControl>
        <EncryptedDataBypass weight="0">0</EncryptedDataBypass>
        <EmbeddedDataTypes weight="0">0</EmbeddedDataTypes>
        <MetadataFlowControl weight="0">0</MetadataFlowControl>
        <SecurityPolicyFilters weight="0">0</SecurityPolicyFilters>
        <HumanReviewMechanism weight="0">0</HumanReviewMechanism>
        <PolicyFilterControl weight="0">0</PolicyFilterControl>
        <InterconnectedSystems weight="0">0</InterconnectedSystems>
        <SecurityAttributes weight="0">0</SecurityAttributes>
      </InformationFlowEnforcement>
      <UnsuccessfulLoginAttemptsLimitation weight="0">0</UnsuccessfulLoginAttemptsLimitation>
      <SystemUseNotification rollup="0" weight="0">
        <NotificationMessageMechanism weight="0">0</NotificationMessageMechanism>
        <RetentionMechanism weight="0">0</RetentionMechanism>
        <PublicAccessMechanism weight="0">0</PublicAccessMechanism>
```

```
    </SystemUseNotification>
    <PreviousLogonNotification rollup="0" weight="0">
      <LastLogonNotification weight="0">0</LastLogonNotification>
      <UnsuccessfulLogonNotification weight="0">0</UnsuccessfulLogonNotification>
      <AttemptsOverTime weight="0">0</AttemptsOverTime>
      <AccountChangeNotification weight="0">0</AccountChangeNotification>
    </PreviousLogonNotification>
    <ConcurrentSessionControl weight="0">0</ConcurrentSessionControl>
    <SessionLockMechanism weight="0">0</SessionLockMechanism>
    <SecurityAttributeManagement rollup="0" weight="0">
      <BindingSecurityAttributes weight="0">0</BindingSecurityAttributes>
      <DynamicReconfigurationSecurityAttributes weight="0">0</DynamicReconfigurationSecurityAttributes>
      <ChangeToSecurityAttributes weight="0">0</ChangeToSecurityAttributes>
      <AssuranceForSecurityAttributes weight="0">0</AssuranceForSecurityAttributes>
      <AssociationOfSecurityAttributes weight="0">0</AssociationOfSecurityAttributes>
      <DisplaySecurityAttributes weight="0">0</DisplaySecurityAttributes>
    </SecurityAttributeManagement>
  </AccessControl>
  <AuditAndAccountability rollup="0" weight="0">
    <ContentOfAuditRecords weight="0">0</ContentOfAuditRecords>
    <ResponseToAuditProcessingFailures rollup="0" weight="0">
      <AuditStorageFailure weight="0">0</AuditStorageFailure>
      <AuditFailureAlert weight="0">0</AuditFailureAlert>
      <AuditTrafficControl weight="0">0</AuditTrafficControl>
      <AuditFailureResponse weight="0">0</AuditFailureResponse>
    </ResponseToAuditProcessingFailures>
    <AuditReviewAnalysisReporting rollup="0" weight="0">
      <AuditIntegration weight="0">0</AuditIntegration>
      <AuditRecordCentralization weight="0">0</AuditRecordCentralization>
    </AuditReviewAnalysisReporting>
    <AuditReductionAndReportGeneration weight="0">0</AuditReductionAndReportGeneration>
    <AuditRecordTimeStamp weight="0">0</AuditRecordTimeStamp>
    <ProtectionOfAuditInformation rollup="0" weight="0">
      <ProtectionMechanism weight="0">0</ProtectionMechanism>
      <WriteOnceMedia weight="0">0</WriteOnceMedia>
      <BackupMechanism weight="0">0</BackupMechanism>
      <EncryptionMechanism weight="0">0</EncryptionMechanism>
    </ProtectionOfAuditInformation>
    <NonRepudiation rollup="0" weight="0">
      <ProducerIdentityMechanism weight="0">0</ProducerIdentityMechanism>
      <ProducerIdentityBindingValidation weight="0">0</ProducerIdentityBindingValidation>
      <ReviewerIdentityMechanism weight="0">0</ReviewerIdentityMechanism>
      <ReviewerIdentityBindingValidation weight="0">0</ReviewerIdentityBindingValidation>
    </NonRepudiation>
    <AuditGeneration rollup="0" weight="0">
      <ContentControlMechanism weight="0">0</ContentControlMechanism>
      <TimeCorrelationMechanism weight="0">0</TimeCorrelationMechanism>
      <StandardizedFormatMechanism weight="0">0</StandardizedFormatMechanism>
    </AuditGeneration>
    <SessionAudit weight="0">0</SessionAudit>
  </AuditAndAccountability>
  <IdentificationAndAuthentication rollup="0" weight="0">
    <IdentificationAndAuthenticationMechanisms rollup="0" weight="0">
      <PrivilegedNetworkAccess weight="0">0</PrivilegedNetworkAccess>
      <NonprivilegedNetworkAccess weight="0">0</NonprivilegedNetworkAccess>
      <PrivilegedLocalAccess weight="0">0</PrivilegedLocalAccess>
      <NonprivilegedLocalAccess weight="0">0</NonprivilegedLocalAccess>
      <PrivilegedNetworkAccessReplayResistance weight="0">0</PrivilegedNetworkAccessReplayResistance>
```

```
          <NonprivilegedNetworkAccessReplayResistance
weight="0">0</NonprivilegedNetworkAccessReplayResistance>
        </IdentificationAndAuthenticationMechanisms>
        <AuthenticatorManagement weight="0">0</AuthenticatorManagement>
        <AuthenticatorFeedbackMechanism weight="0">0</AuthenticatorFeedbackMechanism>
      </IdentificationAndAuthentication>
      <SystemAndCommunicationsProtection rollup="0" weight="0">
        <ApplicationPartitioning weight="0">0</ApplicationPartitioning>
        <InformationInSharedResources weight="0">0</InformationInSharedResources>
        <DenialOfServiceProtection rollup="0" weight="0">
          <RestrictionMechanism weight="0">0</RestrictionMechanism>
          <ResourceManagementMechanism weight="0">0</ResourceManagementMechanism>
        </DenialOfServiceProtection>
        <ResourcePriority weight="0">0</ResourcePriority>
        <BoundaryProtection rollup="0" weight="0">
          <IncomingFlowProtection weight="0">0</IncomingFlowProtection>
          <OutgoingFlowProtection weight="0">0</OutgoingFlowProtection>
          <IncomingValidationMechanism weight="0">0</IncomingValidationMechanism>
          <PrivilegedFlowMechanism weight="0">0</PrivilegedFlowMechanism>
        </BoundaryProtection>
        <NetworkDisconnect weight="0">0</NetworkDisconnect>
        <UseOfCryptography weight="0">0</UseOfCryptography>
        <FailInKnownState weight="0">0</FailInKnownState>
        <OperatingSystemIndependence weight="0">0</OperatingSystemIndependence>
      </SystemAndCommunicationsProtection>
      <SystemAndInformationIntegrity rollup="0" weight="0">
        <InformationSystemMonitoring rollup="0" weight="0">
          <CommunicationMonitoring weight="0">0</CommunicationMonitoring>
          <RealTimeAlertMechanism weight="0">0</RealTimeAlertMechanism>
          <CircumventionMechanism weight="0">0</CircumventionMechanism>
          <SuspiciousEventsMechanism weight="0">0</SuspiciousEventsMechanism>
        </InformationSystemMonitoring>
        <SecurityFunctionalityVerification weight="0">0</SecurityFunctionalityVerification>
        <SoftwareAndInformationIntegrity weight="0">0</SoftwareAndInformationIntegrity>
        <ErrorHandling weight="0">0</ErrorHandling>
      </SystemAndInformationIntegrity>
      <RunTimeSecureness rollup="0" weight="0">
        <InsecureComponentInteraction rollup="0" weight="0">
          <CrossSiteScripting weight="0">0</CrossSiteScripting>
          <SQLInjection weight="0">0</SQLInjection>
          <CrossSiteRequestForgery weight="0">0</CrossSiteRequestForgery>
          <UnrestrictedUploadOfDangerousFileType weight="0">0</UnrestrictedUploadOfDangerousFileType>
          <OSCommandInjection weight="0">0</OSCommandInjection>
          <ErrorMessageInformationExposure weight="0">0</ErrorMessageInformationExposure>
          <URLRedirectionToUntrustedSite weight="0">0</URLRedirectionToUntrustedSite>
          <RaceCondition weight="0">0</RaceCondition>
        </InsecureComponentInteraction>
        <RiskyResourceManagement rollup="0" weight="0">
          <ClassicBufferOverflow weight="0">0</ClassicBufferOverflow>
          <PathTraversal weight="0">0</PathTraversal>
          <BufferAccessWithIncorrectLengthValue weight="0">0</BufferAccessWithIncorrectLengthValue>
          <ImproperCheckForExceptionalConditions weight="0">0</ImproperCheckForExceptionalConditions>
          <PHPFileInclusion weight="0">0</PHPFileInclusion>
          <ImproperValidationOfArrayIndex weight="0">0</ImproperValidationOfArrayIndex>
          <IntegerOverflowOrWraparound weight="0">0</IntegerOverflowOrWraparound>
          <IncorrectCalculationOfBufferSize weight="0">0</IncorrectCalculationOfBufferSize>
          <DownloadOfCodeWithoutIntegrityCheck weight="0">0</DownloadOfCodeWithoutIntegrityCheck>
```

```
          <AllocationOfResourcesWithoutLimitsOrThrottling
weight="0">0</AllocationOfResourcesWithoutLimitsOrThrottling>
        </RiskyResourceManagement>
        <PreventionOfPorousDefenses rollup="0" weight="0">
          <ImproperAccessControl weight="0">0</ImproperAccessControl>
          <RelianceOnUntrustedInputsInSecurityDecision
weight="0">0</RelianceOnUntrustedInputsInSecurityDecision>
          <MissingEncryptionOfSensitiveData weight="0">0</MissingEncryptionOfSensitiveData>
          <UseOfHardcodedCredentials weight="0">0</UseOfHardcodedCredentials>
          <MissingAuthenticationForCriticalFunction weight="0">0</MissingAuthenticationForCriticalFunction>
          <IncorrectPermissionAssignmentForCriticalResources
weight="0">0</IncorrectPermissionAssignmentForCriticalResources>
          <UseOfInsecureCryptographicAlgorithm weight="0">0</UseOfInsecureCryptographicAlgorithm>
        </PreventionOfPorousDefenses>
      </RunTimeSecureness>
    </Security>
    <Reliability rollup="0" weight="0">
      <FaultTolerance rollup="0" weight="0">
        <MechanismAvailable weight="0">0</MechanismAvailable>
        <MechanismEfficiency weight="0">0</MechanismEfficiency>
      </FaultTolerance>
      <Recoverability rollup="0" weight="0">
        <MechanismAvailable weight="0">0</MechanismAvailable>
        <MechanismEfficiency weight="0">0</MechanismEfficiency>
      </Recoverability>
      <Maturity rollup="0" weight="0">
        <Volatility weight="0">0</Volatility>
        <FailureRemoval weight="0">0</FailureRemoval>
      </Maturity>
      <ReliabilityCompliance weight="0">0</ReliabilityCompliance>
      <EventMonitoring weight="0">0</EventMonitoring>
    </Reliability>
    <Usability rollup="0" weight="0">
      <Configurability rollup="0" weight="0">
        <EffortToConfigure weight="0">0</EffortToConfigure>
      </Configurability>
      <Understandability rollup="0" weight="0">
        <DocumentationQuality weight="0">0</DocumentationQuality>
      </Understandability>
      <Learnability rollup="0" weight="0">
        <TrainingMaterialQuality weight="0">0</TrainingMaterialQuality>
        <CommonDevelopmentLanguage weight="0">0</CommonDevelopmentLanguage>
      </Learnability>
      <Operability rollup="0" weight="0">
        <ComplexityLevel weight="0">0</ComplexityLevel>
      </Operability>
      <DiscoverabilityComponentsCompositions weight="0">0</DiscoverabilityComponentsCompositions>
      <DiscoverabilityCapabilities weight="0">0</DiscoverabilityCapabilities>
      <UsabilityCompliance weight="0">0</UsabilityCompliance>
    </Usability>
    <Efficiency rollup="0" weight="0">
      <TimeBehavior rollup="0" weight="0">
        <ResponseTime weight="0">0</ResponseTime>
        <Throughput weight="0">0</Throughput>
      </TimeBehavior>
      <ResourceUtilization rollup="0" weight="0">
        <MemoryUsage weight="0">0</MemoryUsage>
        <DiskUsage weight="0">0</DiskUsage>
```

```
        <NetworkUsage weight="0">0</NetworkUsage>
      </ResourceUtilization>
      <EfficiencyCompliance weight="0">0</EfficiencyCompliance>
      <Accounting weight="0">0</Accounting>
      <QualityOfServiceManagement weight="0">0</QualityOfServiceManagement>
    </Efficiency>
    <Maintainability rollup="0" weight="0">
      <Stability rollup="0" weight="0">
        <ModificationsOverTime weight="0">0</ModificationsOverTime>
      </Stability>
      <Changeability rollup="0" weight="0">
        <LevelofCustomizability weight="0">0</LevelofCustomizability>
      </Changeability>
      <TestabilityComponentsCompositions rollup="0" weight="0">
        <TestSuiteCapability weight="0">0</TestSuiteCapability>
      </TestabilityComponentsCompositions>
      <MaintainabilityCompliance weight="0">0</MaintainabilityCompliance>
    </Maintainability>
    <Portability weight="0">0</Portability>
    <QualityInUse rollup="0" weight="0">
      <Effectiveness weight="0">0</Effectiveness>
      <Productivity weight="0">0</Productivity>
      <Satisfaction weight="0">0</Satisfaction>
      <Attractiveness weight="0">0</Attractiveness>
    </QualityInUse>
  </CCODEnvironment>
</Composition>
<Component rollup="0" weight="0">
  <Name>text</Name>
  <Functionality rollup="0" weight="0">
    <Accuracy rollup="0" weight="0">
      <Correctness weight="0">0</Correctness>
    </Accuracy>
    <Suitability rollup="0" weight="0">
      <Coverage weight="0">0</Coverage>
      <Completeness weight="0">0</Completeness>
      <PrePostConditioned weight="0">0</PrePostConditioned>
      <PrePostConditionsProofs weight="0">0</PrePostConditionsProofs>
    </Suitability>
    <FunctionalityCompliance rollup="0" weight="0">
      <Standardization weight="0">0</Standardization>
      <Certification weight="0">0</Certification>
    </FunctionalityCompliance>
    <SelfContained rollup="0" weight="0">
      <Dependant weight="0">0</Dependant>
    </SelfContained>
    <InternalInteroperability rollup="0" weight="0">
      <DataCompatibility weight="0">0</DataCompatibility>
      <InterfaceComplexity weight="0">0</InterfaceComplexity>
    </InternalInteroperability>
  </Functionality>
  <Security rollup="0" weight="0">
    <RunTimeSecureness rollup="0" weight="0">
      <InsecureComponentInteraction rollup="0" weight="0">
        <CrossSiteScripting weight="0">0</CrossSiteScripting>
        <SQLInjection weight="0">0</SQLInjection>
        <CrossSiteRequestForgery weight="0">0</CrossSiteRequestForgery>
        <UnrestrictedUploadOfDangerousFileType weight="0">0</UnrestrictedUploadOfDangerousFileType>
```

```
        <OSCommandInjection weight="0">0</OSCommandInjection>
        <ErrorMessageInformationExposure weight="0">0</ErrorMessageInformationExposure>
        <URLRedirectionToUntrustedSite weight="0">0</URLRedirectionToUntrustedSite>
        <RaceCondition weight="0">0</RaceCondition>
       </InsecureComponentInteraction>
       <RiskyResourceManagement rollup="0" weight="0">
        <ClassicBufferOverflow weight="0">0</ClassicBufferOverflow>
        <PathTraversal weight="0">0</PathTraversal>
        <BufferAccessWithIncorrectLengthValue weight="0">0</BufferAccessWithIncorrectLengthValue>
        <ImproperCheckForExceptionalConditions weight="0">0</ImproperCheckForExceptionalConditions>
        <PHPFileInclusion weight="0">0</PHPFileInclusion>
        <ImproperValidationOfArrayIndex weight="0">0</ImproperValidationOfArrayIndex>
        <IntegerOverflowOrWraparound weight="0">0</IntegerOverflowOrWraparound>
        <IncorrectCalculationOfBufferSize weight="0">0</IncorrectCalculationOfBufferSize>
        <DownloadOfCodeWithoutIntegrityCheck weight="0">0</DownloadOfCodeWithoutIntegrityCheck>
        <AllocationOfResourcesWithoutLimitsOrThrottling
weight="0">0</AllocationOfResourcesWithoutLimitsOrThrottling>
       </RiskyResourceManagement>
       <PreventionOfPorousDefenses rollup="0" weight="0">
        <ImproperAccessControl weight="0">0</ImproperAccessControl>
        <RelianceOnUntrustedInputsInSecurityDecision
weight="0">0</RelianceOnUntrustedInputsInSecurityDecision>
        <MissingEncryptionOfSensitiveData weight="0">0</MissingEncryptionOfSensitiveData>
        <UseOfHardcodedCredentials weight="0">0</UseOfHardcodedCredentials>
        <MissingAuthenticationForCriticalFunction weight="0">0</MissingAuthenticationForCriticalFunction>
        <IncorrectPermissionAssignmentForCriticalResources
weight="0">0</IncorrectPermissionAssignmentForCriticalResources>
        <UseOfInsecureCryptographicAlgorithm weight="0">0</UseOfInsecureCryptographicAlgorithm>
       </PreventionOfPorousDefenses>
     </RunTimeSecureness>
    </Security>
    <Reliability rollup="0" weight="0">
     <FaultTolerance rollup="0" weight="0">
      <MechanismAvailable weight="0">0</MechanismAvailable>
      <MechanismEfficiency weight="0">0</MechanismEfficiency>
     </FaultTolerance>
     <Recoverability rollup="0" weight="0">
      <MechanismAvailable weight="0">0</MechanismAvailable>
      <MechanismEfficiency weight="0">0</MechanismEfficiency>
     </Recoverability>
     <Maturity rollup="0" weight="0">
      <Volatility weight="0">0</Volatility>
      <FailureRemoval weight="0">0</FailureRemoval>
     </Maturity>
    </Reliability>
    <Usability rollup="0" weight="0">
     <Configurability rollup="0" weight="0">
      <EffortToConfigure weight="0">0</EffortToConfigure>
     </Configurability>
     <Understandability rollup="0" weight="0">
      <DocumentationQuality weight="0">0</DocumentationQuality>
     </Understandability>
     <Operability rollup="0" weight="0">
      <Parameters weight="0">0</Parameters>
      <RequiredParameters weight="0">0</RequiredParameters>
      <EffortForOperating weight="0">0</EffortForOperating>
     </Operability>
    </Usability>
```

```
    <Efficiency rollup="0" weight="0">
      <TimeBehavior rollup="0" weight="0">
        <ResponseTime weight="0">0</ResponseTime>
        <Throughput weight="0">0</Throughput>
      </TimeBehavior>
      <ResourceUtilization rollup="0" weight="0">
        <MemoryUsage weight="0">0</MemoryUsage>
        <DiskUsage weight="0">0</DiskUsage>
        <NetworkUsage weight="0">0</NetworkUsage>
      </ResourceUtilization>
      <Scalability rollup="0" weight="0">
        <ProcessingCapacity weight="0">0</ProcessingCapacity>
      </Scalability>
    </Efficiency>
    <Maintainability rollup="0" weight="0">
      <Stability rollup="0" weight="0">
        <ModificationsOverTime weight="0">0</ModificationsOverTime>
      </Stability>
      <Changeability rollup="0" weight="0">
        <ComplexityLevel weight="0">0</ComplexityLevel>
        <Cohesion weight="0">0</Cohesion>
        <Coupling weight="0">0</Coupling>
      </Changeability>
      <Testability rollup="0" weight="0">
        <ExtensiveTestCases weight="0">0</ExtensiveTestCases>
      </Testability>
    </Maintainability>
    <QualityInUse rollup="0" weight="0">
      <Effectiveness weight="0">0</Effectiveness>
    </QualityInUse>
    <ComponentDeveloper rollup="0" weight="0">
      <Name>text</Name>
      <Proficiency rollup="0" weight="0">
        <TrainingPerformed weight="0">0</TrainingPerformed>
        <AverageComponentComplexity weight="0">0</AverageComponentComplexity>
      </Proficiency>
    </ComponentDeveloper>
    <MaintenanceEvent rollup="0" weight="0">
      <Type>text</Type>
      <Date>2010-06-01</Date>
      <ComponentCompositionMaintainer rollup="0" weight="0">
        <Name>text</Name>
        <Proficiency rollup="0" weight="0">
          <TrainingPerformed weight="0">0</TrainingPerformed>
          <AverageComponentComplexity weight="0">0</AverageComponentComplexity>
          <AverageCompositionComplexity weight="0">0</AverageCompositionComplexity>
        </Proficiency>
      </ComponentCompositionMaintainer>
    </MaintenanceEvent>
  </Component>
  <CCODEnvironment rollup="0" weight="0">
    <Name>text</Name>
    <Functionality rollup="0" weight="0">
      <Accuracy rollup="0" weight="0">
        <Correctness weight="0">0</Correctness>
      </Accuracy>
      <Suitability rollup="0" weight="0">
        <Coverage weight="0">0</Coverage>
```

```
      <Completeness weight="0">0</Completeness>
    </Suitability>
    <FunctionalityCompliance rollup="0" weight="0">
      <Standardization weight="0">0</Standardization>
      <Certification weight="0">0</Certification>
    </FunctionalityCompliance>
    <InternalInteroperability rollup="0" weight="0">
      <DataCompatibility weight="0">0</DataCompatibility>
      <InteroperationComplexity weight="0">0</InteroperationComplexity>
    </InternalInteroperability>
    <ExternalInteroperability rollup="0" weight="0">
      <DataCompatibility weight="0">0</DataCompatibility>
      <InteroperationComplexity weight="0">0</InteroperationComplexity>
    </ExternalInteroperability>
    <LifeCycleManagement weight="0">0</LifeCycleManagement>
    <ConfigurationManagement weight="0">0</ConfigurationManagement>
    <PolicyManagement weight="0">0</PolicyManagement>
  </Functionality>
  <Security rollup="0" weight="0">
    <AccessControl rollup="0" weight="0">
      <AccountManagement rollup="0" weight="0">
        <AccountTypeIdentification weight="0">0</AccountTypeIdentification>
        <AuthorizedUserIdentification weight="0">0</AuthorizedUserIdentification>
        <AccountManagementMechanism weight="0">0</AccountManagementMechanism>
        <GuestAndTemporaryAccounts weight="0">0</GuestAndTemporaryAccounts>
        <AccessControlMechanism weight="0">0</AccessControlMechanism>
        <InactiveAccounts weight="0">0</InactiveAccounts>
        <AccountAuditing weight="0">0</AccountAuditing>
      </AccountManagement>
      <AccessEnforcement rollup="0" weight="0">
        <RoleBasedAccessControl weight="0">0</RoleBasedAccessControl>
      </AccessEnforcement>
      <InformationFlowEnforcement rollup="0" weight="0">
        <InformationFlowEnforcementMechanism weight="0">0</InformationFlowEnforcementMechanism>
        <ProtectedProcessingDomains weight="0">0</ProtectedProcessingDomains>
        <DynamicFlowControl weight="0">0</DynamicFlowControl>
        <EncryptedDataBypass weight="0">0</EncryptedDataBypass>
        <EmbeddedDataTypes weight="0">0</EmbeddedDataTypes>
        <MetadataFlowControl weight="0">0</MetadataFlowControl>
        <SecurityPolicyFilters weight="0">0</SecurityPolicyFilters>
        <HumanReviewMechanism weight="0">0</HumanReviewMechanism>
        <PolicyFilterControl weight="0">0</PolicyFilterControl>
        <InterconnectedSystems weight="0">0</InterconnectedSystems>
        <SecurityAttributes weight="0">0</SecurityAttributes>
      </InformationFlowEnforcement>
      <UnsuccessfulLoginAttemptsLimitation weight="0">0</UnsuccessfulLoginAttemptsLimitation>
      <SystemUseNotification rollup="0" weight="0">
        <NotificationMessageMechanism weight="0">0</NotificationMessageMechanism>
        <RetentionMechanism weight="0">0</RetentionMechanism>
        <PublicAccessMechanism weight="0">0</PublicAccessMechanism>
      </SystemUseNotification>
      <PreviousLogonNotification rollup="0" weight="0">
        <LastLogonNotification weight="0">0</LastLogonNotification>
        <UnsuccessfulLogonNotification weight="0">0</UnsuccessfulLogonNotification>
        <AttemptsOverTime weight="0">0</AttemptsOverTime>
        <AccountChangeNotification weight="0">0</AccountChangeNotification>
      </PreviousLogonNotification>
      <ConcurrentSessionControl weight="0">0</ConcurrentSessionControl>
```

```
      <SessionLockMechanism weight="0">0</SessionLockMechanism>
      <SecurityAttributeManagement rollup="0" weight="0">
       <BindingSecurityAttributes weight="0">0</BindingSecurityAttributes>
       <DynamicReconfigurationSecurityAttributes weight="0">0</DynamicReconfigurationSecurityAttributes>
       <ChangeToSecurityAttributes weight="0">0</ChangeToSecurityAttributes>
       <AssuranceForSecurityAttributes weight="0">0</AssuranceForSecurityAttributes>
       <AssociationOfSecurityAttributes weight="0">0</AssociationOfSecurityAttributes>
       <DisplaySecurityAttributes weight="0">0</DisplaySecurityAttributes>
      </SecurityAttributeManagement>
     </AccessControl>
     <AuditAndAccountability rollup="0" weight="0">
      <ContentOfAuditRecords weight="0">0</ContentOfAuditRecords>
      <ResponseToAuditProcessingFailures rollup="0" weight="0">
       <AuditStorageFailure weight="0">0</AuditStorageFailure>
       <AuditFailureAlert weight="0">0</AuditFailureAlert>
       <AuditTrafficControl weight="0">0</AuditTrafficControl>
       <AuditFailureResponse weight="0">0</AuditFailureResponse>
      </ResponseToAuditProcessingFailures>
      <AuditReviewAnalysisReporting rollup="0" weight="0">
       <AuditIntegration weight="0">0</AuditIntegration>
       <AuditRecordCentralization weight="0">0</AuditRecordCentralization>
      </AuditReviewAnalysisReporting>
      <AuditReductionAndReportGeneration weight="0">0</AuditReductionAndReportGeneration>
      <AuditRecordTimeStamp weight="0">0</AuditRecordTimeStamp>
      <ProtectionOfAuditInformation rollup="0" weight="0">
       <ProtectionMechanism weight="0">0</ProtectionMechanism>
       <WriteOnceMedia weight="0">0</WriteOnceMedia>
       <BackupMechanism weight="0">0</BackupMechanism>
       <EncryptionMechanism weight="0">0</EncryptionMechanism>
      </ProtectionOfAuditInformation>
      <NonRepudiation rollup="0" weight="0">
       <ProducerIdentityMechanism weight="0">0</ProducerIdentityMechanism>
       <ProducerIdentityBindingValidation weight="0">0</ProducerIdentityBindingValidation>
       <ReviewerIdentityMechanism weight="0">0</ReviewerIdentityMechanism>
       <ReviewerIdentityBindingValidation weight="0">0</ReviewerIdentityBindingValidation>
      </NonRepudiation>
      <AuditGeneration rollup="0" weight="0">
       <ContentControlMechanism weight="0">0</ContentControlMechanism>
       <TimeCorrelationMechanism weight="0">0</TimeCorrelationMechanism>
       <StandardizedFormatMechanism weight="0">0</StandardizedFormatMechanism>
      </AuditGeneration>
      <SessionAudit weight="0">0</SessionAudit>
     </AuditAndAccountability>
     <IdentificationAndAuthentication rollup="0" weight="0">
      <IdentificationAndAuthenticationMechanisms rollup="0" weight="0">
       <PrivilegedNetworkAccess weight="0">0</PrivilegedNetworkAccess>
       <NonprivilegedNetworkAccess weight="0">0</NonprivilegedNetworkAccess>
       <PrivilegedLocalAccess weight="0">0</PrivilegedLocalAccess>
       <NonprivilegedLocalAccess weight="0">0</NonprivilegedLocalAccess>
       <PrivilegedNetworkAccessReplayResistance weight="0">0</PrivilegedNetworkAccessReplayResistance>
       <NonprivilegedNetworkAccessReplayResistance
weight="0">0</NonprivilegedNetworkAccessReplayResistance>
      </IdentificationAndAuthenticationMechanisms>
      <AuthenticatorManagement weight="0">0</AuthenticatorManagement>
      <AuthenticatorFeedbackMechanism weight="0">0</AuthenticatorFeedbackMechanism>
     </IdentificationAndAuthentication>
     <SystemAndCommunicationsProtection rollup="0" weight="0">
      <ApplicationPartitioning weight="0">0</ApplicationPartitioning>
```

```
      <InformationInSharedResources weight="0">0</InformationInSharedResources>
      <DenialOfServiceProtection rollup="0" weight="0">
       <RestrictionMechanism weight="0">0</RestrictionMechanism>
       <ResourceManagementMechanism weight="0">0</ResourceManagementMechanism>
      </DenialOfServiceProtection>
      <ResourcePriority weight="0">0</ResourcePriority>
      <BoundaryProtection rollup="0" weight="0">
       <IncomingFlowProtection weight="0">0</IncomingFlowProtection>
       <OutgoingFlowProtection weight="0">0</OutgoingFlowProtection>
       <IncomingValidationMechanism weight="0">0</IncomingValidationMechanism>
       <PrivilegedFlowMechanism weight="0">0</PrivilegedFlowMechanism>
      </BoundaryProtection>
      <NetworkDisconnect weight="0">0</NetworkDisconnect>
      <UseOfCryptography weight="0">0</UseOfCryptography>
      <FailInKnownState weight="0">0</FailInKnownState>
      <OperatingSystemIndependence weight="0">0</OperatingSystemIndependence>
     </SystemAndCommunicationsProtection>
     <SystemAndInformationIntegrity rollup="0" weight="0">
      <InformationSystemMonitoring rollup="0" weight="0">
       <CommunicationMonitoring weight="0">0</CommunicationMonitoring>
       <RealTimeAlertMechanism weight="0">0</RealTimeAlertMechanism>
       <CircumventionMechanism weight="0">0</CircumventionMechanism>
       <SuspiciousEventsMechanism weight="0">0</SuspiciousEventsMechanism>
      </InformationSystemMonitoring>
      <SecurityFunctionalityVerification weight="0">0</SecurityFunctionalityVerification>
      <SoftwareAndInformationIntegrity weight="0">0</SoftwareAndInformationIntegrity>
      <ErrorHandling weight="0">0</ErrorHandling>
     </SystemAndInformationIntegrity>
     <RunTimeSecureness rollup="0" weight="0">
      <InsecureComponentInteraction rollup="0" weight="0">
       <CrossSiteScripting weight="0">0</CrossSiteScripting>
       <SQLInjection weight="0">0</SQLInjection>
       <CrossSiteRequestForgery weight="0">0</CrossSiteRequestForgery>
       <UnrestrictedUploadOfDangerousFileType weight="0">0</UnrestrictedUploadOfDangerousFileType>
       <OSCommandInjection weight="0">0</OSCommandInjection>
       <ErrorMessageInformationExposure weight="0">0</ErrorMessageInformationExposure>
       <URLRedirectionToUntrustedSite weight="0">0</URLRedirectionToUntrustedSite>
       <RaceCondition weight="0">0</RaceCondition>
      </InsecureComponentInteraction>
      <RiskyResourceManagement rollup="0" weight="0">
       <ClassicBufferOverflow weight="0">0</ClassicBufferOverflow>
       <PathTraversal weight="0">0</PathTraversal>
       <BufferAccessWithIncorrectLengthValue weight="0">0</BufferAccessWithIncorrectLengthValue>
       <ImproperCheckForExceptionalConditions weight="0">0</ImproperCheckForExceptionalConditions>
       <PHPFileInclusion weight="0">0</PHPFileInclusion>
       <ImproperValidationOfArrayIndex weight="0">0</ImproperValidationOfArrayIndex>
       <IntegerOverflowOrWraparound weight="0">0</IntegerOverflowOrWraparound>
       <IncorrectCalculationOfBufferSize weight="0">0</IncorrectCalculationOfBufferSize>
       <DownloadOfCodeWithoutIntegrityCheck weight="0">0</DownloadOfCodeWithoutIntegrityCheck>
       <AllocationOfResourcesWithoutLimitsOrThrottling
weight="0">0</AllocationOfResourcesWithoutLimitsOrThrottling>
      </RiskyResourceManagement>
      <PreventionOfPorousDefenses rollup="0" weight="0">
       <ImproperAccessControl weight="0">0</ImproperAccessControl>
       <RelianceOnUntrustedInputsInSecurityDecision
weight="0">0</RelianceOnUntrustedInputsInSecurityDecision>
       <MissingEncryptionOfSensitiveData weight="0">0</MissingEncryptionOfSensitiveData>
       <UseOfHardcodedCredentials weight="0">0</UseOfHardcodedCredentials>
```

```
        <MissingAuthenticationForCriticalFunction weight="0">0</MissingAuthenticationForCriticalFunction>
        <IncorrectPermissionAssignmentForCriticalResources
weight="0">0</IncorrectPermissionAssignmentForCriticalResources>
        <UseOfInsecureCryptographicAlgorithm weight="0">0</UseOfInsecureCryptographicAlgorithm>
       </PreventionOfPorousDefenses>
     </RunTimeSecureness>
   </Security>
   <Reliability rollup="0" weight="0">
     <FaultTolerance rollup="0" weight="0">
       <MechanismAvailable weight="0">0</MechanismAvailable>
       <MechanismEfficiency weight="0">0</MechanismEfficiency>
     </FaultTolerance>
     <Recoverability rollup="0" weight="0">
       <MechanismAvailable weight="0">0</MechanismAvailable>
       <MechanismEfficiency weight="0">0</MechanismEfficiency>
     </Recoverability>
     <Maturity rollup="0" weight="0">
       <Volatility weight="0">0</Volatility>
       <FailureRemoval weight="0">0</FailureRemoval>
     </Maturity>
     <ReliabilityCompliance weight="0">0</ReliabilityCompliance>
     <EventMonitoring weight="0">0</EventMonitoring>
   </Reliability>
   <Usability rollup="0" weight="0">
     <Configurability rollup="0" weight="0">
       <EffortToConfigure weight="0">0</EffortToConfigure>
     </Configurability>
     <Understandability rollup="0" weight="0">
       <DocumentationQuality weight="0">0</DocumentationQuality>
     </Understandability>
     <Learnability rollup="0" weight="0">
       <TrainingMaterialQuality weight="0">0</TrainingMaterialQuality>
       <CommonDevelopmentLanguage weight="0">0</CommonDevelopmentLanguage>
     </Learnability>
     <Operability rollup="0" weight="0">
       <ComplexityLevel weight="0">0</ComplexityLevel>
     </Operability>
     <DiscoverabilityComponentsCompositions weight="0">0</DiscoverabilityComponentsCompositions>
     <DiscoverabilityCapabilities weight="0">0</DiscoverabilityCapabilities>
     <UsabilityCompliance weight="0">0</UsabilityCompliance>
   </Usability>
   <Efficiency rollup="0" weight="0">
     <TimeBehavior rollup="0" weight="0">
       <ResponseTime weight="0">0</ResponseTime>
       <Throughput weight="0">0</Throughput>
     </TimeBehavior>
     <ResourceUtilization rollup="0" weight="0">
       <MemoryUsage weight="0">0</MemoryUsage>
       <DiskUsage weight="0">0</DiskUsage>
       <NetworkUsage weight="0">0</NetworkUsage>
     </ResourceUtilization>
     <EfficiencyCompliance weight="0">0</EfficiencyCompliance>
     <Accounting weight="0">0</Accounting>
     <QualityOfServiceManagement weight="0">0</QualityOfServiceManagement>
   </Efficiency>
   <Maintainability rollup="0" weight="0">
     <Stability rollup="0" weight="0">
       <ModificationsOverTime weight="0">0</ModificationsOverTime>
```

```
      </Stability>
      <Changeability rollup="0" weight="0">
        <LevelofCustomizability weight="0">0</LevelofCustomizability>
      </Changeability>
      <TestabilityComponentsCompositions rollup="0" weight="0">
        <TestSuiteCapability weight="0">0</TestSuiteCapability>
      </TestabilityComponentsCompositions>
      <MaintainabilityCompliance weight="0">0</MaintainabilityCompliance>
    </Maintainability>
    <Portability weight="0">0</Portability>
    <QualityInUse rollup="0" weight="0">
      <Effectiveness weight="0">0</Effectiveness>
      <Productivity weight="0">0</Productivity>
      <Satisfaction weight="0">0</Satisfaction>
      <Attractiveness weight="0">0</Attractiveness>
    </QualityInUse>
  </CCODEnvironment>
</Composition>
```

Appendix F Bibliography

A. Alvaro E.S. Almeida, A.M.L. Vasconcelos, S.R.L. Meira Towards a Software Component Quality Model [Conference] // Proceedings. 31st Euromicro Conference on Software Engineering and Advanced Applications. - [s.l.] : IEEE Computer Society , 2005.

A. Alvaro E.S. Almeida, S.R.L. Meira A Software Component Quality Model: A Preliminary Evaluation [Conference] // Proceedings. 32nd IEEE EUROMICRO Conference on Software Engineering and Advanced Applications. - Cavtat, Dubrovnik, Croatia : IEEE Computer Society, 2006. - pp. 28-37.

A. Alvaro E.S. Almeida, S.R.L. Meira Quality Attributes for a Component Quality Model [Conference] // 10th International Workshop on Component Oriented Programming in conjunction with the 19th ACM European Conference on Object Oriented Programming / ed. W. Weck J. Bosch, R. Reussner and C. Szyperski. - Glasgow, Scotland : Association for Computing Machinery, 2005.

Alexander Ian F. A Taxonomy of Stakeholders: Human Roles in System Development [Journal] // International Journal of Technology and Human Interaction (IJTHI) / ed. Mesquita Anabela. - [s.l.] : Information Resources Management Association, 2005. - 1 : Vol. 1. - pp. 23-59.

Defense Acquisition University 1.1. Integration of the DoD Decision Support Systems [Online] // acc.dau.mil. - September 21, 2010. - https://acc.dau.mil/CommunityBrowser.aspx?id=314713.

ISO01 Software Engineering - Product Quality - Part 1 Quality Model [Report]. - [s.l.] : International Organization for Standardization, 2001. - ISO/IEC 9126-1.

ISO02 Software Engineering - Product Quality - Part 2 External Metrics [Report]. - [s.l.] : International Organization for Standardization, 2001. - ISO/IEC 9126-2.

ISO03 Software Engineering - Product Quality - Part 3 Internal Metrics [Report]. - [s.l.] : International Organization for Standardization, 2001. - ISO/IEC 9126-3.

ISO04 Software Engineering - Product Quality - Part 4 Quality In Use Metrics [Report]. - [s.l.] : International Organization for Standardization, 2001. - ISO/IEC 9126-4 .

IT Governance Institute About It Governance Framework [Online] // www.itgi.org. - September 21, 2010. - http://www.itgi.org/template_ITGIa166.html?Section=About_IT_Governance1&Template=/ContentManagement/HTMLDisplay.cfm&ContentID=19657.

MITRE cwe.mitre.org [Online] // 2010 CWE/SANS Top 25 Most Dangerous Software Errors. - June 29, 2010. - September 21, 2010. - http://cwe.mitre.org/top25/.

NIST Recommended Security Controls for Federal Information Systems and Organizations [Report] / Computer Security Division, Information Technology Laboratory. - Gaithersburg, MD : National Institute of Standards and Technology, 2009. - Special Publication 800-53.

OASIS Reference Architecture Foundation for Service Oriented Architecture, Version 1, Committee Draft 02.0 [Report]. - [s.l.] : Organization for the Advancement of Structured Information Standards, 2009.