

Introducing Engineered Composition (EC): An Approach for Extending the Common Criteria to Better Support Composing Systems

September 30, 2003

Shari Galitzer
sgalitzer@cygnacom.com

Cygnacom Solutions
Suite 100 West
7927 Jones Branch Drive
McLean, VA 22102-3305

1. Introduction

This paper presents an approach, called Engineered Composition (EC), for extending the Common Criteria (CC) to achieve 'seamless composability', i.e. being able to predict that a composition of components will have the desired beneficial properties, with no uncontrollable or unpredictable side effects [12]. In the CC paradigm the components are PPs and STs and the composition is an evaluated system that reuses the components.

EC is a top-down, horizontal decomposition approach, based on information systems object-oriented modeling principles and techniques for large systems that require distributed development. In our experience, it improves the way requirements and capabilities for systems and their components are expressed using the CC because it provides a means to write PPs and STs for component parts that are combinable and comparable. This has a number of advantages:

- making PPs (and STs) combinable provides a means to capture the system properties of the composed components;
- making PPs (and STs) comparable enables them to be composed without first being rationalized;
- through reuse, realizing efficiencies for evaluating the security of systems, in terms of the number of PPs required and the effort to produce them and their respective STs.

A Critical Infrastructure Grants Program (CIPGP) grant from NIST's Computer Security Division supported the development of the approach. The grant provided funding for five months of effort over a two-year period with the goal to validate the EC concept and prepare it to be implemented as part of the CC standard. For more information on the grant please see: <http://csrc.nist.gov/grants/awards.html>.

It's well-established that the Common Criteria currently does not adequately support composition. The CC paradigm currently has three 'work-in-progress-type' of approaches available to apply to the composition issue: 1. component TOEs may be combined into a 'composite TOE' in a PP or ST, however the TOEs may not be comparable and the process has been found to be impractical¹; 2. component TOEs may be combined by users who rely on the TOEs' IT environment requirements as a composition mechanism for the dependent functions, however the

¹ At ICC2 in Spring 2001, in "Engineered Composition for Secure Systems Using the Common Criteria", I presented my analysis from comparing the evaluation process requirements for systems using Composite TOE vs. an early version of the EC approach.

TOEs may not be comparable and there is no mechanisms for the system-level properties; and 3. a TOE may be a system without regard to its component parts.

Industry discussions and the International Common Criteria Conference (ICCC) proceedings continually express the pressing need to improve and add adequate composition support to the Common Criteria (CC), and the ISO/IEC PDTR 15446 'Guide for Production of Protection Profiles and Security Targets' supports this position. Regarding PPs and STs for composite TOEs, Chapter 10 provides guidance related to the specific issues raised by the notion of composability:

...where a PP or ST is being written for a *composite TOE*, that is a TOE that is composed of two or more components, each of which has its own individual PP or ST (termed *component PP* or *component ST* within this Guide);... This general approach will be particularly appropriate for large system architectures that contain many components... It should be noted that the area of composability is one for which to date there has been little practical experience. Further guidance will be provided in future versions of the Guide as and when practical experience is gained in this area [7].

There is now over 4four years of practical experience with the current version of the CC, and this experience has clarified the composability issues. To address them EC proposes to use information systems engineering modeling methods and techniques. Modeling is a proven and well-accepted information systems engineering tool and the EC approach applies it to the CC to facilitate:

- the expression of the security requirements of both a system and its components:
 - capturing the security requirements of a component;
 - capturing how the component contributes to the system security requirements;
 - exposing security issues that result from aggregation of the components;
- making combinable and comparable independently developed PPs and STs;
- developing PPs more efficiently by:
 - making PPs 'refinements' for specific environments rather than a 're-creation' for each specific environment;
 - requiring fewer PPs to meet users needs;
- developing STs more efficiently by:
 - making STs combinable so it's possible to claim compliance to multiple PPs; currently it's not practical;
 - being able to define a scope for an ST that is a component of a set of requirements and be able to reuse those requirements in the ST.
 - reuse of PP specifications for an ST that's a component of a PP;
- supporting the fluid nature of systems.

1.1 What is the EC Approach

The EC approach began as a theory: the composition problem of ensuring (i.e., being able to predict) that a set of components will have certain desired system properties when used to build systems is primarily a modeling problem with security as its subject domain. Therefore information systems modeling methods should solve the problem, and the modeling methods used to support distributed development of large and complex information systems should be particularly appropriate and useful.

The theory then evolved, over two years of study and analysis, into an approach that would provide support for composition in the CC context. The EC approach includes:

- an overarching instruction to use information systems engineering modeling principles and techniques, and not 'reinvent the wheel';

- recommendations to add constructs to the CC paradigm that are currently used in information system modeling and designed to address composition concerns. These include *framework, collaboration, interface, and aspect*.
- A recommendation to include evaluation criteria to ensure the quality of the model constructs, i.e., to ensure they are useful. The evaluation should include well-established modeling methods and metrics that address and analyze the properties of the models including *separation-of-concerns, cohesion, and coupling* properties.
- A recommendation to include evaluation criteria that rigorously enforces consistent terminology among the CC artifacts and documents. The terminology should evolve to be consistent with well-established information systems object modeling technology.

1.2 Contents

Section 2 provides background information that includes the origin of the theory, the method used to research the theory, and observations from analyzing and applying the research. Section 3 discusses the EC approach and its recommendations for the CC paradigm. Section 4 looks ahead.

1.3 Understanding the EC Approach

Understanding how the EC approach provides composition support to the CC depends on understanding that modeling techniques are well-proven and useful engineering tools for supporting information systems life-cycle processes.

This dependency emerged during the numerous discussions, emails, conference calls, and presentations throughout the project. To people acquainted with modeling, the EC approach seemed to be a viable path to address composition, so discussions focused on recommended construct details and secondary benefits to the approach. To people unfamiliar with modeling, explaining the approach was more challenging and required a primer on the effectiveness of modeling as information systems engineering tool. Most attempts to discuss the EC approach for these individuals without going through the fundamentals of modeling were very difficult.

A primer on the effectiveness of modeling as an information systems engineering tool is not a trivial document and is beyond the scope of this document. The background information in Section 2 defines modeling and its benefits. Along with the references in this paper, the following web sites may be useful: <http://aosd.net/>, <http://www.omg.org/uml/>.

2. Background

2.1 EC's Origin

The premise of the EC approach is that the composition problem is primarily a modeling problem with security as its subject domain. Therefore information systems modeling methods should solve the problem.

This theory emerged from the combined experience of working with problems in defining requirements and articulating product capabilities for systems and their components using the CC, and of realizing solutions (implemented and operational) in the information systems development industry for similar types of problems.

As a CC lab, we find that STs and PPs are not comparable and combinable. This hinders our ability to advise and develop PPs and STs that are practical for our users. The issue is pervasive and is manifested by the following problems:

- it is impractical to rationalize multiple PPs;
- it is impractical for distributed authors of component PPs and STs to provide a coherent set of requirements and capabilities – requires coordination methods not available;
- there are too many STs and PPs to write and evaluate, even if systems were static; and

- it is difficult to meet customer expectations; both vendors and end-users expect the CC to provide more.

Developing large systems and their components is an inherently distributed process that requires independent and concurrent efforts. Components are developed independently and composed into large systems – the same is true for component PPs and STs. Information systems engineering modeling methods exist, and are continuously being improved, to support distributed, independent and concurrent component development and then composition for information systems in general. These same modeling methods should work for the associated PP and ST development process.

Solutions realized by modeling techniques for composition in the information systems development industry seemed applicable to the CC composition problem given the following:

- the CC is an object-oriented semi-formal modeling language;
- writing STs and PPs is a distributed development process;
- modeling is a well-established information systems engineering tool;
- modeling exists to simplify: a security principal is 'strive for simplicity' [16];
- the CC's customers (developers and end-users) use modeling as a tool;
- my own professional experience has proven many times the value of modeling complex domains to support information systems development.

2.2 Moving from Theory to Recommendations

The process for exploring our EC theory had three iterative phases: the research; the case study; and the technology transfer.

In the research phase, I studied best practices, innovations and new research papers and journals in the information systems modeling industry, and studied publications and discussed the topic with experts in the information systems security industry.

The research included participation in the ACSA Workshop on the Application of Engineering Principles to System Security Design in Nov 2002 at Boston, MA, <http://www.acsac.org/waepssd>, and the third meeting of the NIST Ad Hoc Working Group on Composability of Evaluated Products in April 2002. These provided the opportunity to discuss EC with groups of well-renown security experts in a dynamic setting. The research phase also included ensuring a valid starting point from the CC perspective through discussions with Kris Britton as the Lead Technical Advisor for NIAP.

In the case-study phase, I applied the knowledge obtained from the research phase to the composition problem in the CC. As mentioned above, this was an iterative process so the research continued while the case-study was in development and was directed by questions raised by the case-study.

The domain for the case study was to define the security requirements for a Public Key Infrastructure (PKI) and its Directory component in a manner that allows its component parts, PPs and STs, to be comparable and combinable. I used Uniform Modeling Language (UML) to express the security requirements for the domain. UML was selected because it's a common semi-formal modeling language with which I was already familiar. The domain of the model, a PKI and its Directory component, was appropriate to use since we were already familiar with its security requirements and because of useful characteristics including:

- its large and complex with heterogeneous components, platforms, and administration;
- its self-referential structure – the infrastructure itself uses its own systems, e.g., certificate-based authentication mechanisms must be able to use the infrastructure's directory system;
- it includes recursive systems– the directory is itself a system of components;
- the fact that as an infrastructure it has defined external interfaces to help bound the effort.

The final step in the case-study phase was to translate the UML model to recommendations for evolving the CC to include composition.

In the technology transfer phase, I included discussions, documentation, and presentations. I presented EC at the ICC2, 3 and 4. In addition to this paper, a paper of EC's work-in-progress was submitted to the ASE draft v.6 comment process, and a subsequent revision to the ACSA Workshop on the Application of Engineering Principles to System Security Design in Nov 2002 at Boston, MA, <http://www.acsac.org/waepssd>.

2.3 Observations From Analyzing and Applying the Research

Observation 1: The advent of aspect-oriented modeling should make the benefits of modeling more accessible to the information systems security industry. Based on my experience in general and my recent research, modeling is a mainstream tool for the information systems industry but not for the security specialty within the industry. An explanation for this may be that security as a whole and many of its mechanisms are cross-cutting concerns. As described in the next section cross-cutting concerns are the system properties that apply across several functions or components within a system and do not decompose neatly. These are challenging to model effectively with object-oriented methods.

However, with the advent of *aspect-oriented* modeling methods to augment object-oriented methods, *cross-cutting concerns* are more straightforward to model. It's conceivable that when the security experts and technicians previously explored object-oriented modeling to support their efforts the *cross-cutting* nature of the security requirements may have obscured some of the benefits of modeling. Now, modeling techniques have evolved to better express security requirements and should make the benefits of modeling more accessible to security experts and technicians.

Observation 2: Current research is using modeling concepts. Research in security for large-scale systems has shown that although direct references to object-oriented modeling are scarce, the concepts are being used. For example, Neumann's work describes the necessity of a 'framework' and some characteristics of its elements. While he uses different language, the analysis relies on object-oriented modeling concepts (object-oriented modeling language is noted in *italics*):

Given a conceptual understanding of a set of system requirements, or even a detailed set of requirements, one of the most important architectural problems is to establish a workable structure [*framework*] of the system that can evolve into a successful implementation. The architectural decomposition of a network into sub networks, a system into subsystems, or subsystems into components [*interfaces, packages, objects, components*] can benefit greatly from modularity [*encapsulation*], hierarchical layering, constructive uses of redundancy [*aggregation and objects*], and separation of concerns [*encapsulation*]. These are design principles that can pervasively affect the decomposability of a system design – and thereby the modular composability.[12]

Observation 3: Top-down decomposition is necessary. The question of whether top-down analysis is a required component of any composition approach was consistently raised in the initial discussions of my research. Given the CC paradigm currently supports only bottom-up 'work-in-progress-type' of approaches² this question is important.

² The three approaches are: 1. component TOEs may be combined into 'composite TOEs in a PP or ST; 2. component TOEs may be combined by users who rely on the TOEs IT environment

The results of my research phase found, without exception, that top-down analysis (i.e., decomposition) is a critical component to any approach to composition. [10,11,15,17]. This is not new to the information systems engineering composition issue. Decomposition is a fundamental software engineering principle to controlling complexity that was well-established in early work by Dijkstra[4] and Parnas[14]. Their seminal work on *decomposition* and *separation of concerns* is consistently cited in current information systems security engineering research.

In another example, Tinto characterized the problems presented by large, complex systems as problems of:

- 'scope of comprehensibility' - where one is faced with comprehending both the details of a large number of functional elements and the global system properties that those elements exhibit when they all work together.
- 'intellectual gap', i.e. the level of abstraction presented by the elements (implementation specifics, functional characteristics) is different from the level of abstraction that is of interest (security). [17]

His work showed that that decomposition is essential to addressing the problems associated with developing and evaluating large, complex systems.

Observation 4: EC is inclusive and provides coordination for the various composition efforts. The purpose of modeling is to simplify and to organize complex problems. As presented and discussed at ICC3 and ICC4, there are many efforts to solve the composition problem based on different and useful perspectives. The models recommended by the EC approach can serve as an organizing tool for the various composition efforts. All valid perspectives should compliment each other and bring synergistic benefits.

2.4 What Modeling Provides to the Composition Problem

The composition problem can be viewed as partly a definitional problem, and partly a methodological one. Definitional concerns involve a statement of the nature of a thing. For the composition problem this is how to specify the component and system requirements and capabilities so component PPs and STs are combinable. Methodological concerns involve the way of doing something. For the composition problem this how to coordinate the distributed, independent and concurrent development efforts so component PPs and STs are comparable.

Modeling provides a simplification of reality to manage complexity and address the 'scope of comprehensibility' and 'intellectual gap' system problems characterized by Tinto. Models are semantically closed abstractions created to solve both the methodological and definitional aspects of the composition problem.

Models provide an abstraction for a purpose and can provide a central coordination point for the development methods. They also provide a means to focus on areas of risk to better enable the application of specialized techniques and tools. Modeling methods and their techniques address the methodological aspect of the problem. To address the definitional problem, the models provide an abstraction that captures and provides focus on the critical particular realities, sometimes referred to as the *essence*, of the security concern. These abstractions define the appropriate representations and abstractions for the security concerns and establish coherent terminology.

Modeling for information systems engineering is designed to achieve four aims:

1. to visualize a system as it is or should be;
to specify the structure of the behavior of a system;

requirements for composition hooks; and 3. a TOE may be a system without regard to its component parts.

2. to have a template that guides in constructing a system;
3. to document the decisions.[2]

2.5 Three Important Modeling Concepts for our Purposes

Three important modeling concepts to help solve the CC composition problem emerged from the research phase: *patterns*, *aspects-oriented* modeling methods, and *interfaces*. [1,2,8,15]:

Patterns solve a common problem in a common way. When modeling a system and its components a modeling technique is to identify and specify *patterns*. Two important kinds of patterns for our purposes are *frameworks* and *collaborations*³. *Frameworks* capture architectural patterns that specify the structure and behavior of an entire system, and *collaborations* capture design patterns that specify a set of abstractions that work together to carry out some common and interesting behavior.

In general patterns:

- provide an extensible template;
- solve a common problem in a common way;
- expose the slots, tabs, knobs & dials that must be adapted to apply to a context;
- are atomic, not easily broken into smaller patterns;
- tend to cut across
- individual abstractions in the system (i.e., horizontal decomposition).

Aspect-oriented modeling is a newly emerging extension to object-oriented modeling methods to better achieve *separation of concerns* for complex systems. Aspect modeling enhances the expressiveness of object-oriented modeling to address the complication of *cross-cutting concerns*. ‘Concerns’ are the primary motivation for organizing and decomposing software into manageable and comprehensible parts. ‘Cross-cutting concerns’ are the parts that do not decompose neatly. They are system properties that apply across several functions or components within a system and can range from high-level security notions like audit, cascading risk or ‘whole-program effects’ to low-level notions like redundant copies of data for fault tolerance or synchronization policies requiring a set of operations that follow locking protocol[6].

The relationship between the object-oriented and aspect-oriented methods can be characterized as one where the object-oriented methods find commonality among classes and push it up the inheritance tree, and aspect-oriented methods realize scattered concerns as first-class elements and eject them horizontally from the object structure [6].

The publications on aspect-oriented modeling consistently cite security in general and many security mechanisms in particular, e.g., authentication and audit, as examples of a cross-cutting concerns which have been a challenge for object-oriented modeling methods. The aspect-oriented extension to the object-oriented modeling methods is designed to address this challenge and make the benefits of modeling more accessible to domains with cross-cutting concerns.

Interfaces of a system are a critical object-oriented modeling concept for distributed information systems. In a model an interface is a collection of operations that are used to specify a service of a class or a component, and represent the major seams in the system that permits component parts to change independently. [2]

Interfaces provide the basis for the recursive property of a system where collaborations and components of a system are themselves made of component parts. The interfaces are

³ The term ‘mechanism’ is also used to describe a design pattern with term ‘collaboration’ used to describe its logical representation. For this paper the just ‘collaboration’ is used to represent design patterns.

represented in the component view of the model. This view exposes the seams of the system and the glue that binds the components. The interfaces that are realized by the cross-cutting properties of the system and represented by the aspect-oriented constructs are called *join points*.

3. The EC Approach and Specific Recommendations

The EC Approach includes:

- an overarching instruction to use information systems engineering modeling principles and techniques, and not 'reinvent the wheel';
- recommendations to add constructs to the CC paradigm that are currently used in information system modeling and designed to address composition concerns. These include *framework, collaboration, interface, and aspect*.
- A recommendation to include evaluation criteria to ensure the quality of the *framework*, i.e., to ensure the models are useful. The evaluation should include well-established modeling methods and metrics that address and analyze the properties of the models including *separation-of-concerns, cohesion, and coupling* properties.
- A recommendation to include evaluation criteria that rigorously enforces consistent terminology among the CC artifacts and documents. The terminology should evolve to be consistent with well-established information systems object modeling technology.

3.1 Recommendation 1 - Use Information Systems Engineering Modeling Principles and Techniques

The CC itself is an object-oriented modeling toolset for a specific context. Therefore the information systems modeling industry is an intrinsic resource for the CC to evolve to better serve its user community.

A well-established information systems engineering challenge is how to compose independently develop components for systems that continuously evolve. This challenge is being successfully met today using object-oriented augmented with aspect-oriented modeling as a keystone to their success. Therefore by 're-using the wheel' and looking at the information systems modeling techniques and applying them to the CC paradigm the following recommendations emerge:

- The CC should have a *framework* construct for system requirements where it's desired to have component PPs and STs for the system be combinable and comparable.
- The *framework* should include constructs and views currently used in information system modeling that are designed to address composition concerns, these include: *collaborations, mechanisms, interfaces, components, and aspects*.

3.2 Recommendation 2 - Add a CC Composition Framework to the CC Paradigm as the Foundation for Composition support

What Does the CC Composition Framework and its component constructs look like? As introduced in Section 2, a framework is an architectural pattern that provides an extensible template for applications within a domain. This template specifies a set of mechanisms that define a skeleton of an architecture together with the 'slots, tabs, knobs, and dials' that are exposed to adapt the framework to a specific context.[2]

Defining the framework for a PKI and its directory components using UML and aspect-oriented extensions, while challenging to do well, was a moderately straightforward process. Translating the models into a format consistent with current CC constructs was more challenging. Therefore the following first presents the contents of the framework first and then separately looks at how to effectively format the contents.

The CC Composition Framework includes:

1. **A domain framework definition** that specifies the scope and purpose of the framework. The definition should:
 - Provide a text description of the general technologies and services of the domain. This should be a few sentences, including a diagram, and written from the perspective of bringing together the producers and consumer of the framework and its attendant PPs and STs, e.g., as would appear in a trade journal.

The following is an example from the PKI domain: The domain for the PKI Composition Framework includes the high-level requirements for the following PKI services and specific requirements for a repository component for these services: Creating and Revoking Certificates, Disseminating Certificates and Revocation Lists, Disseminating Certificate Status, and limited Cryptographic Key Management for the components. The domain does not include the following PKI services: Non-repudiation, Notarization, Secure Timestamp, Cryptographic Key backup and recovery, Automatic Key Update, and Key History Management.

Examples of other domains include the IATF Framework groupings, technology consortium efforts e.g., Smart Card Security User Group (SCSUG) SmartCard PP, Forum on Privacy and Security in Healthcare, and Open Group efforts, including the Partitioning Kernel PP.

- Identify the domain security services and capabilities of the domain. The security services are those provided to its users. In the PKI example this includes creating certificates, distributing certificates and revocation lists, and generating audit records. The security capabilities are the security functions the domain depends upon from itself to provide the services e.g., non-bypassability and domain separation.
 - Identify the domain users (referred to in models as actors) including both the trusted and untrusted users, e.g., PKI relying parties, security administrators, trusted data administrators. The notion of trust vs. untrusted should be from the system domain perspective and consistently maintained for each component.
 - Establish the domain specific terminology. The EC Approach requires the terminology for the CC to be specified and used with more rigor, therefore the terminology in the domain framework definition does not repeat the CC general terminology. The terminology only includes refinements to the general terms for the specific context, and domain and technology specific terminology.
2. **A framework model that captures 3 levels of decomposition and 2 views of interest.** The CC Composition Framework comprises three kinds of inter-connected models that capture a static view of the system from a perspective that supports composition of its component parts. The models are: a system-level object model, a set of object models for the collaborations, i.e., services, identified in the domain framework definition, and a component model. The two specified views of the framework are the threats and policies view, and the requirements view.

The system-level object model captures all system level characteristics, e.g. requirements and patterns, that exist only at the system level and that can be inherited by the system components. The set of object models for the collaborations provide a second level of decomposition and abstract the important characteristics for the collaborations. These abstractions can be at various levels of detail depending on the level at which the critical information resides. These collaboration models inherit the system-level characteristics and then refine them as appropriate for their

purpose, and add new detail relevant to the collaboration. Finally a component model is included to both refine and add component-level information and to represent the *interfaces* for the system identifying the physical seams of the system.

In addition to the views required to communicate these models the framework model should also include a view that maps these models to the domain threats and policies, and a view that maps requirements to the objects and components. The following provides more information on these models and views.

- the system-level object model includes:
 - The set of *classes* organizing objects, attributes, operations, relationships and semantics of the system and establishing its vocabulary.
 - An example of the classes from the PKI case study is a class 'Data' with an attribute of TSF/User data. Other classes that are kinds of Data (and therefore inherit its attributes and operations) are 'Transmitteddata' 'Storeddata' 'Auditdata', 'Authorizationdata'.
 - Control mechanisms are also captured in classes. TSF function is a class with kinds of TSFFunctions including 'I&Adecision', 'AccessControlDecision&Enforcement'.
 - An example of the representation of a relationship is the *association* between the 'AccessControlDecision&Enforcement' class and the 'authorizationdata' class.
 - The requirements for the semantics of the certificates supported by the PKI are captured in the *certificate* class (a kind of 'Data').
 - Special classes are used to capture the system-level *aspects* e.g., audit, cascading risk, protocol or standard requirements.
 - the *collaborations* object models:
 - Inherit the information from the system level so these elements should not be repeated
 - Capture the *mechanisms* patterns.
 - Add refinements and details relevant for the collaboration
 - Capture the collaboration *aspects* in special object classes.
 - the *component* model:
 - Provides the representation for the physical seams of the system;
 - It is created from modeling the *interfaces*;
 - The architectural requirements are specified here, e.g., any existing elements that need to be incorporated;
 - It inherits the information from its collaborations;
 - It adds the details relevant for the component;
 - The *join points* are specified, these are the *interfaces* for the *cross-cutting concerns*. This provides additional mapping for the aspects.
 - Views to facilitate comparable and combinable PPs and STs:
 - The above models include the threat and policy information that drives the requirements. The framework should include a separate view that provides an abstraction to show the relationship between the threats and policies and objects and components.
 - The above models also include CC requirements as attributes of the objects and components. The framework should include a separate view that provides an abstraction that focuses on the mapping of the objects to their requirements.
3. **A new CC composition class of requirement components.** This class captures and organizes special composition requirements:
- captures composition attributes of the model;
 - an interface specification attribute;

- an attribute to identify 'areas of risk' regarding composition.
4. **A Section for Administrative information.** This is less developed but includes:
- procedures for framework life-cycle management;
 - profile specifications or references identified by policy requirements;
 - policy information on STs claiming compliance to the framework.

The format of the CC Composition Framework needs to be established. When considering the format three factors were considered. The first factor was its being accepted by the CC community and the learning curve for incorporating it into practice. The second factor was the challenge of mapping the many-to-many relationships of the threats, policies, and requirements for the system, its collaborations, and components. The final factor was how the format facilitates the goal of modeling, 'to simplify and manage the complexity in order to better understand the system being created'.

Three formats were considered for the framework: a modified PP format, UML or other modeling method format, and a hypertext document. No conclusive recommendation has emerged at this point.

Initially a modified PP format was envisioned. It leverages the existing knowledge base of the community and, at first glance, supports reuse for its constituent PPs and STs. However, PPs specify environment specific information that is not included in the *framework*. This environment information is well presented by text phrases and therefore the text document format of PP serves it well. However, the object models with their aspects and various views are 3-dimensional concepts and are better presented graphically rather than the linear presentation of text phrases. In particular, the challenge of 'flattening the models' became a bigger problem when *aspects* were discovered and used in the framework. At this stage of the analysis while the acceptance factor is favorable the complexity of mapping the many-to-many relationships and providing a simplified view of the domain in a modified PP format does not seem practical

Given the problems with 'flattening the models' in terms of effort, usability of the result, and loss of information through the process, keeping the models in a native object modeling format, e.g. UML, should be considered. The models inherently provide the many-to-many mapping, and a good quality model simplifies and manages the complexity. The concern with using the UML format is resistance to its acceptance because of the perception of having too large of a learning curve.

While there would be learning curve to developing and using UML models, I believe their simplicity in presentation would be worth the investment to learn. Different amounts of knowledge would be required depending a persons role. People developing and evaluating the CC Composition Framework would need to be experts or work with modeling experts, while those using the framework to write a component PP or ST would need only minimal knowledge, in fact ease of use is a characteristic of a good framework. The need for specialized knowledge to work with an aspect of the CC is already part of the paradigm. Requiring expertise in modeling can be considered analogous to the requirement for expertise in formal methods for those developing and evaluating aspects of high-assurance products.

The UML models can be produced using standard desktop software, or specialized modeling software and delivered in standard desktop formats, e.g., Word documents. To shorten the learning curve it's possible to define a CC abstraction of UML. Also, an individual CC Composition Framework can use any modeling method that is optimal for its domain.

Finally, the SCSUG SmartCard PP effort has provided an alternative way forward that is worth exploring; using hypertext. This provides a 3-dimensional presentation, reflecting the concept, and may have a shorter learning curve. How it addresses the many-to-many mapping challenge and simplified representation hasn't been explored in this context.

3.2 Recommendation 3 - Evaluate the Quality of the CC Composition Framework

The quality of the CC Composition Framework must be evaluated since it's the template for its constituent PPs and STs. An error in it would have exponential effects throughout all processes in the CC paradigm. Therefore an APE requirement to evaluate the quality of the framework must be defined.

The framework quality evaluation needs to address two aspects: 1. that it presents a coherent set of security services and capabilities, and 2. that the framework and its models are valid from a modeling perspective. The CC already contains the requirements to address the first part. For the second part the definition of quality models and its associated metrics need to be established. Here, we can again look to the information systems engineering modeling industry for guidance. As a starting point it must verify the basic object modeling principles: *separation of concerns, collaboration, encapsulation, aggregation, cohesion*.

Lifecycle support for the framework must also be considered to support the evolving nature of systems. Again, we can look to the information systems engineering modeling industry for guidance. After selecting the best process for our purpose, the associated policy can be established, e.g., regarding compliance claims.

3.4 Recommendation 4 - Institute Rigorous Terminology in CC Artifacts and Documents

The EC approach generates well-defined terms for a system domain and depends on well-defined terms from the CC. However, as presented at the recent ICC4 by Dr. Dirk-Jan Out, there are unclear and inconsistently used critical CC terminology, examples involve SF, SFR, SFP, SEF, SPM, TSF, TSC, TOE, TSP, Product, System.[22]

Naaman identifies terminology along different levels of abstraction, different concepts of users, subjects and operations and different assurances as the reasons component PPs and STs currently cannot be compose. His work goes on to propose a threat taxonomy to address many composition issues.[9]

These efforts compliment the EC approach and are examples of Observation 4, that EC is inclusive and provides coordination for the various composition efforts. Improving the CC terminology is priority for the EC approach. Since the framework inherits the CC terms, it depends on accurate terminology. The EC approach can also help the efforts to improve the terminology since the modeling process identifies ill-defined terms and helps put rigor to their definition.

4. Going Forward

The work so far shows that the EC approach is a viable way to extend the CC to provide better composition support, and provides a start for the CC community to evolve the CC to better serve its users. The following lists some next steps to continue the evolution.

- Establish the format.
- Continue to evolve the PKI framework to:
 - extend the component information beyond the Directory;
 - improve the models based on further analysis and expert input;
 - increase the level of confidence in the approach;
 - provide the benefits from having its constituent PPs and STs be combinable and comparable;

- Extend the case-study to another domain and include the constituent PP and ST development process.
- Educate the CC community to better understand the role of modeling security in large systems development processes.
- Build modeling expertise in the CC community.
- Continue to track the information systems modeling industry, particularly methods related to aspect-oriented modeling.
- Investigate the availability of a modeling tool for the CC and defining a CC abstraction to shorten the learning curve.
- Consider the benefits of allowing an ST to claim compliance to a CC Composition Framework. Based on my experience, I'm confident the requirements specified in the CC Composition Framework would adequately specify component requirements for a significant percent of user environments.
- Continue and coordinate the related CC composition efforts.

5. References

5.1 Publications

1. Clarke, S., Walker, R.J., *Composition Patterns: An Approach to designing reusable aspects in Proceedings International Conference on Software Engineering*, p. 5-14: (Toronto, Canada, 2001).
2. Booch, Grady, Rumbaugh, James, and Jacobson, Ivar, *The Unified Modeling Language User Guide* (Reading, MA: Addison-Wesley, 1998).
3. Brewer, D., Want, C. Tsai P., *Proving Protection Profile Compliance for the CCL/ITRI Visa Open Platform Smart Card*, 3rd ICCS, May 2002.
4. Dijkstra, E.W., *A Discipline of Programming*, Prentice Hall, Englewood Cliffs, NJ, 1976.
5. De Win, Bart, Piessens, Joosen, Verhanneman, *On the Important of the Separation-of-concerns Principle in Secure Software Engineering*, (ACSA Workshop on the Application of Engineering Principles to System Security Design, November 2002, Boston MA).
6. Elrad, Tzilla, Filman, Robert E., Bader, Atef, *Aspect-oriented Programming*, (Communications of the ACM, October 2001/Volume 44, No.10).
7. *Guide for the Production of Protection Profiles and Security Targets*, ISO/IEC PDTR 15446, January 2002.
8. Jacobson, Ivan, Palmkvist, Karin, and Dyrhage, Susanne, *Systems of Interconnected Systems*. In Bowman, Charles F., *Wisdom of the Gurus* p. 81-91 (New York, NY: SIGS Books, 1996).
9. Naaman, Nir, *Some Ideas on Improving Common Criteria v2.1 Using a Programming Language Paradigm*. (ACSA Workshop on the Application of Engineering Principles to System Security Design, November 2002, Boston MA).
10. National Computer Center Technical Report-002, *Use of Trusted Computer System Evaluation Criteria (TCSEC) for Complex, Evolving Multipolicy Systems* (MD, National Computer Security Center, 1994)
11. National Computer Center Technical Report-003, *Turning Multiple Evaluated Products into Trusted Systems* (MD, National Computer Security Center, 1994)
12. Neumann, Peter G., *Principled Assuredly Trustworthy Composable Architectures: First-Year Interim Report* (CA, SRI International, 2002).
13. Nierstrasz, Oscar, and Meijler, Theo Dirk , *Requirements for a Composition Language*. In Ciancarini, Paolo, Nierstrasz, Oscar, and Yonezawa, Akinori (Eds.), *Object-Based Models and languages for Concurrent Systems* p. 147-161 (Selected papers from ECOOP'94 Workshop on Models and languages for Coordination of Parallelism and Distribution, Bologna, Italy: 1994.
14. Parnas, D.L., *On the Criteria to be used in Decomposing Systems into Modules*, (Communications of the ACM 1972/ Volume 15, No. 12).

15. Sigfried, Stefan, *Understanding Object-Oriented Software Engineering* (Piscataway, NJ: IEEE Press, 1996).
16. Stoneburner, Gary, *Underlying Technical Models for Information Technology Security*, NIST Special Publication 800-33, (MD, National Institute of Standards and Technology, Information Technology Laboratory, Computer Security Division, December 2001).
17. Tinto, Mario, *The Design and Evaluation of INFOSEC Systems: The Computer Security Contribution to the Composition Discussion* (MD, National Computer Security Center C TECHNICAL REPORT 32-92, 1992).

5.2 Documents

18. *Critical Infrastructure Protection Grants Program*, Department of Commerce National Institute of Standards and Technology, Federal Register / Vol. 66, Number 72 / Friday, April 13 2001/ Notices.
19. Elliott III, Ken, *Proposed Requirements for Composition Guidance*, ICC3, May 2002.
20. Holm, Straw, Tinto, *NIAP Conference Panel Discussion on Composition of Evaluated Products*, ICC3, May 2000.
21. NIST PKI Program PKI Documents. <http://csrc.nist.gov/pki/documents/>
22. Out, Dirk-Jan, *CC Terminology: In Search of Common Meaning*, ICC4, September 2003.
23. *Protection Profile (PP) Consistency Guidance for Medium Robustness Draft*, United States National Institute of Standards and Technology and the National Security Agency Protection Profile Review Board (PPRB), 23 July 2002.
24. Supplement: ASE – Security Target Evaluation, version 0.65, CCIMB-2002-04-011, May 2002.

5.3 PPs

25. *Certificate Issuing and Management Components (CIMC) Family of Protection Profiles, Draft* Version 1.0, 4 September 2001.
26. *Federal Public Key Infrastructure Directory Profile*, Version 2.0 Draft, Booz Allen Hamilton, March 14, 2002
27. *Public Key-Enabled Application Family of Protection Profiles, (USMCPKEPP_V2.5)*, Version 2.5, October 2002
28. *Smart Card Security User Group Smart Card Protection Profile (SCSUG-SCPP)*, Version 3.0, 9 September 2001.
29. *U.S. Department of Defense Directory Protection Profile for Medium Robustness Environments Draft*, Version 4.0, 9 August 2003