In Proceedings of the 43rd Annual ACM Southeast Conference (ACMSE '05),
Vol. 2, pp. 130-135, March 18-20, Kennesaw State Univ. Kennesaw, Georgia. 2005

Common Criteria Requirements Modeling and its Uses for Quality of Information Assurance (QoIA)

Deepak S. Yavagal, Seok Won Lee, Gail-Joon Ahn, Robin A. Gandhi
Dept. of Software and Information Systems
The University of North Carolina at Charlotte
Charlotte, NC 28223-0001, USA.
{dsyavaga, seoklee, gahn, rgandhi}@uncc.edu

ABSTRACT

The Common Criteria for Information Technology Security Evaluation (CCITSE), usually referred to as the Common Criteria (CC), establishes a level of trustworthiness and confidence that should be placed in the security functions of products or systems and the assurance measures applied to them. CC achieves this by evaluating the product or system conformance with a common set of requirements set forth by it. To engineer a product that meets the information assurance goals of CC, a structured and comprehensive methodology is required to drive the activities undertaken in all the stages of the software requirements engineering (RE) process. Such a methodology is inevitable to understand and attain the Quality of Information Assurance (QoIA). As an effort in this direction, we focus on the use of object-oriented ontology modeling as an effective way of representing and enforcing the given common set of requirements established by CC. Our methodology leverages novel techniques from software requirement engineering and knowledge engineering. This paper also describes how this methodology can effectively realize CC-related requirements of the target systems and help evaluate such systems for conformance to the certification and accreditation (C&A) process.

Categories and Subject Descriptors

D.2.1 [**Software Engineering**]: Requirements/Specifications – Elicitation methods, Methodologies, Tools.

General Terms

Management, Design, Reliability, Security, Human Factors.

Keywords

Common Criteria, Requirements Modeling, Information Assurance Metrics & Measures, Ontology.

1. INTRODUCTION

The ever growing need of Information Technology (IT) products in various domains motivates the need for a well defined and comprehensive methodology to establish appropriate assurance levels about their security functions and features. Consumers must able to measure, compare and evaluate various IT products to understand their capabilities and limitations; both functional and non-functional as accurately as possible. National Institute of Standards and Technology (NIST) and National Security Agency (NSA) have recently announced a new collaborative effort to produce comprehensive security requirements and security specifications. These security requirements and security specifications are being developed with significant industry involvement to address the needs of consumers, developers and evaluators and are employed in the new international security standard known as the CC (ISO/IEC 15408) [4].

The information assurance levels assigned by CC are known as Evaluation Assurance Levels (EALs). In order to check system compliance with a particular EAL, a set of activities are prescribed which includes checking if all the necessary security requirements have been identified, evaluating these requirements against the user requirements, and eliciting new requirements based on the objectives to be met. We identify that the current practices followed by CC does not provide a well defined methodology to utilize the requirements enforced by it to drive the activities undertaken in all the stages of RE of an IT product. A well defined methodology is crucial in attaining the QoIA that most consumers demand from an IT product that supports their needs. As an effort in this direction, we propose object-oriented ontology modeling as an effective modeling technique for CC requirements. Using the inherent advantages of this technique we can effectively help consumers to interpret the product specifications, developers to realize the requirements and evaluators to effectively evaluate compliance and practice the C&A process.

The rest of the paper is organized as follows. In Section 2, we provide a brief overview of the CC process, followed by a discussion of the limitations in the existing C&A practices in Section 3. Section 4 discusses the benefits of using ontological [8] engineering processes and how they contribute towards achieving the IA goals set forth by CC for an IT product. Section 5 provides a brief description about the Generic Object Model (GenOM) [7] toolkit. In Section 6 we describe how GenOM can be used to develop CC requirements representations. Section 7 demonstrates, using examples, the ways in which these representations can be utilized. Finally, we present some concluding remarks and a discussion of our future work in Section 8.

2. COMMON CRITERIA OVERVIEW

CC [4] provides a comprehensive list of various security functions that products should include along with a list of methods for assuring their safety. CC produces two types of documents, Protection Profile (PP) and Security Target (ST). PP is a document created by the stakeholder or a consumer that identifies the security properties to be satisfied by the system. ST is a document that identifies the functionalities that need to be supported by the system being developed (called the Target of Evaluation, or TOE). An ST is not required to meet the requirements of a particular PP, but an ST could possibly meet the requirements of one or more PPs.

Process of creation of PP and ST involves identification of organizational security policies, stakeholders needs and assumptions, the environment in which the TOE exists, and the threats and vulnerabilities associated with the system. CC also provides a toolbox that supports the creation of PPs and STs. CC Toolbox™ [3] provides a static template for configuring functional and assurance requirements and their relationships for a specific information assurance product. The security objectives to be satisfied by the TOE are derived from the environment it operates. Finally, the applicable security requirements are selected from a set of given CC security requirements, so that they meet these objectives. The final draft of PP or ST should clearly identify the dependencies between the environment where the system is situated and the security requirements. It should also assist the stakeholders in identifying how these security requirements satisfy the objectives.

The security requirements provided by CC can be further divided into functional requirements and assurance requirements. A functional requirement specifies what the TOE should be able to perform. Assurance requirements provide assurance about how the objectives of the TOE are fulfilled. In CC, the term *class* is used for the most general grouping of functional and assurance requirements. We overview such structures in section 6.

In the next section we discuss some of the limitations of the existing practices followed by CC.

3. LIMITATIONS OF THE EXISTING METHODOLOGY

CC provides a set of security and assurance requirements from the aspects of users, developers, and evaluators that needs to be supported by the system under evaluation. CC requirements have complex dependencies among them. Descriptions provided by CC for the relevant terms are abstract in order to maintain general applicability. This leads to ambiguities and it is often difficult to find the right meaning and usage of the security requirements [9]. Furthermore, CC requires specifications and designs at different levels of abstraction but does not provide any mechanisms to achieve the same. Developers face difficulties in writing the design documents, as making a clear distinction between the details that belong to a higher level and lower level of abstraction is not easy [9]. Hence, we identify the need to represent and organize the security requirements in a way that contributes to effective interpretation and enforcement, with well defined methodology/tool support that facilitates such methods and features.

The software system being evaluated is embedded within an environment that caters to the goals and objectives to be achieved by the organization [6]. Capturing the relationship between the goals/objectives and the operating environment along with the traceability links or dependencies between them is critical for the successful evaluation and accreditation of the system. This is essential for an effective secure software engineering process as security is not an add-on functionality but rather it is an *emergent* feature of the system in relationship with various technical and non-technical factors in the environment that houses it.

We also identify that the current practices of CC methodology do not provide a way to effectively support the RE phases of requirements elicitation, representation, discovery, verification and management. To support these phases of RE we propose the following research objectives based on CC.

- Requirements elicitation & representation: Build an ontology of CC requirements that can transform static CC requirements collections to active one that can be used by people from different interests to gain empirical understanding of the domain.
- Requirements defects discovery: Build methods to discover defects such as *missing requirements* in CC requirements and system requirements.
- Requirements organization & management: The amount of documents perceived during CC evaluation and accreditation process is large [5]. Therefore there is a need to structure and organize such knowledge in a suitable manner so that it can adapt to numerous requirements changes often encountered due to rapid changes in technology and business.
- Requirements verification and validation: Build a methodology that can verify and validate specific PPs and STs and assist their reassurance process.

These objectives are crucial to achieve QoIA, especially in providing effective and efficient ways to incorporate changes in policies, regulations, requirements and new legislation into the assessment process of an IT product to evaluate the conformance to a set of security requirements. In the next section we propose a methodology which is a combination of novel software requirements engineering and knowledge modeling techniques that assists in addressing the research objectives described above.

4. OBJECT-ORIENTED ONTOLOGY MODELING

To assist autonomous object interactions, use of machine understandable ontologies created as a result of the RE process has been pointed out in [1]. In this paper, we propose an object-oriented ontology modeling approach which uses ontologies as its primary method of modeling system requirements. The objective here is to model the CC requirements in a way that not only benefits the analysis and evaluation of the deployed TOE functions and constraints but guides the adopted RE process through all its phases. Through object-oriented ontology modeling techniques we can transform static requirements collections provided by CC, into structures that link the objectives to be achieved by the system, security requirements and the domain objects along with the actions they are able to perform in the environment, from various perspectives. Domain objects are concepts in the environment that interact with each other in order to satisfy

security objectives and requirements of the system. They are interdependent on each other for achieving a particular security objective. The object-oriented ontology modeling process of deriving hierarchical model of CC problem domain is shown in Figure 1. It involves categorization and classification of CC security requirements and related domain knowledge by creating hierarchical representations of CC requirements, producing a structure where the high level requirements identified in the non-leaf nodes are decomposed into specific criteria in the leaf nodes. In the next step the domain objects that help to realize the requirements along with the security objectives that need to be satisfied by the system are identified and modeled using advanced object-oriented knowledge representation techniques. Finally, traceability links are established between requirements, objectives and domain objects and a Problem Domain Ontology (PDO) is created. Such a model provides the necessary means to understand and evaluate the effect of the system functions and constraints in light of the concepts, properties and relationships that exist in the application domain or environment.

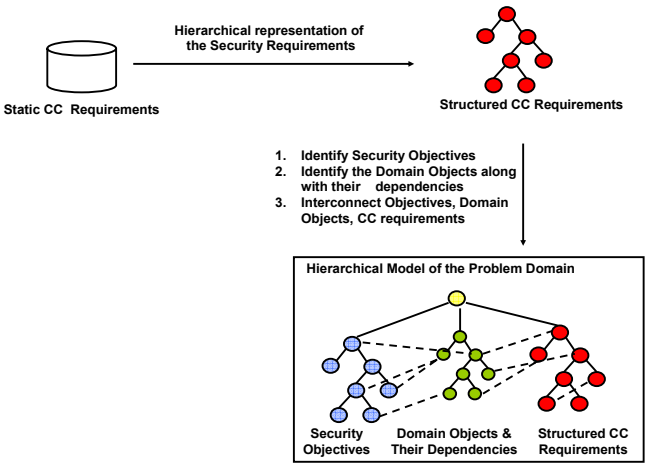

Figure 1: Process to Create Hierarchical Model of CC Requirements

The PDO augments the interpretation and efficiency in enforcement of requirements held within, by giving the opportunity to analyze them from different perspectives and dimensions. It also provides a structured & comprehensive view that helps consumers to interpret the product specifications, developers to realize the requirements and evaluators to effectively evaluate compliance and practice the C&A process. The GenOM toolkit [7], introduced in next section aims at providing complete tool support for creating hierarchical models of the CC PDO using object-oriented ontology modeling technique.

5. THE GenOM TOOLKIT

GenOM toolkit is constructed for the purpose of knowledge acquisition and representation to aid the design and implementation of any *intelligent* software application by using object-oriented technologies. It addresses object modeling in its representation, usage of objects in its application model and its ability to aggregate evidence that supports the analysis of objects' behaviors (through the associated properties and relationships between objects). The harmonization of these characteristics often determines the level of intelligence of the applications. When a software computing paradigm converges toward domain-independent interdisciplinary research, the objects (or models) used in each application model should be interoperable and reusable. GenOM is such an interoperable and reusable object computing model.

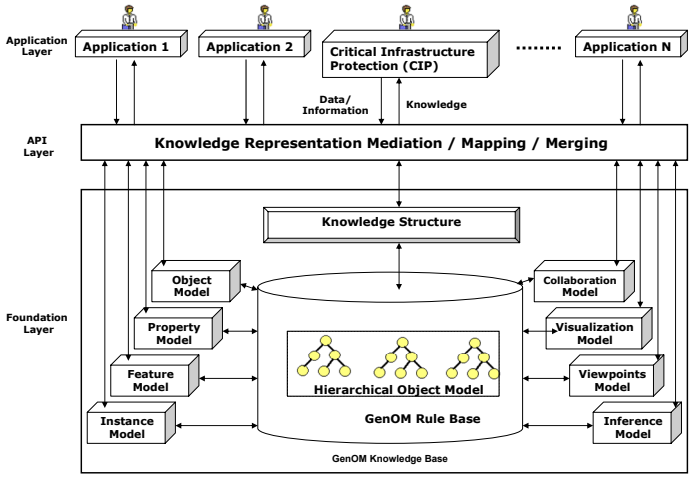

Figure 2: GenOM Conceptual Architecture

GenOM inherits the theoretical foundation of frame representation in artificial intelligence and expands its architecture so that its hierarchical object model can be easily transformable and adaptable to other software design models in various domain applications. It does this by building domain-specific application layers on the GenOM foundation layer, while GenOM itself serves as an integrated environment to create, edit, browse, search and maintain objects. In addition, GenOM provides mechanisms for mediating, mapping, merging and integrating various types of knowledge representation of domain-specific objects in the API layer as shown in Figure 2.

Here we discuss some of the components used in our modeling work. The object model is used to describe the concepts in the domain. It also supports modeling of hierarchical structures and thus provides support for single or multiple inheritance mechanism. The *property* model is used to describe the characteristics or attributes of objects. The *features* model is used to describe the relationship or dependencies that exist between objects. The *instance* model is used to model specific instances for different objects existing in a domain. The visualization components of GenOM are provided for the object modeler and the instance modeler. The visualization component of the *object* modeler facilitates the visualization of the object hierarchies. Visualization for instance modeler facilitates the visualization of instance interdependencies. GenOM is associated with an inference engine which supports reasoning based on the concepts, properties and features defined in the ontology using Jena [2] inference capabilities.

GenOM provides ways for mapping, merging and integrating domain-specific objects and thus serves as a knowledge base for building object-oriented software applications. The next section focuses on how the various concepts in the problem domain of CC can be modeled using ontological engineering process using GenOM toolkit.

6. REQUIREMENTS REPRESENTATION AND ITS USES

In section 4 we discussed our methodology that transforms the static CC functional requirements into a PDO that uses objects, properties, features and instance knowledge modeling components provided by GenOM. Following this methodology and utilizing the inherent structure of the CC requirements, shown in Figure 3, functional classes which are used to categorize requirements in CC are modeled using the *objects* (or Class) knowledge component in GenOM. A resulting example for categorization of different functional classes in CC based on GenOM knowledge modeling components is as shown in Figure 4. Functional classes in CC can be decomposed into several functional families. This relationship can be modeled by expressing the functional families as sub-classes of the class used to model a functional class. Each functional family is associated with a set of attributes to provide information of its behavior, management information and audit data. Each attribute provides information about the functional family from a particular aspect. These attributes can be modeled using the *properties* knowledge component used to describe an *object* knowledge component in GenOM. Each functional family has a set of components associated with it. These components are more specific in describing the requirements of a particular security category of the functional family. Again the components can be modeled using the *sub-class* and *super-class* concepts supported by GenOM. The non-hierarchical relationships between components contained in a functional family can be expressed using the *features* knowledge component in GenOM. The resultant model preserves the structure of the CC requirements as well as adds to their way of representation, making them more amenable to analysis and traceable to and from various related entities in the universe of discourse.

Well annotated user interfaces of GenOM to model *objects* and *features* are shown in Figure 5 and Figure 6 respectively. To represent the dependencies that exist between different components in the functional families, the *features* knowledge modeling component provided by GenOM (Figure 6) can be used. Other dependencies and traceability links that exist between goals/objectives of the system and requirements that help to achieve them can also be modeled in a similar fashion.

The specifications for different CC requirements can be represented using the *instances* knowledge modeling component in GenOM. We make this modeling choice as the specifications describe in detail about how a particular security requirement of a system is satisfied. Instances are used as a mechanism for storing information indicating how the CC requirements are satisfied for a particular security specification. The instances are always traceable to their associated objects in GenOM. Thus it is apparent that GenOM provides the means to logically structure & organize specifications based on the associated requirements.

GenOM also supports visualization of the knowledge modeling components to visually comprehend the models constructed. Figure 7 shows the meta-level visualization of the conceptual model for CC requirements constructed using GenOM. It demonstrates how different concepts described in the CC requirements categorization at the meta-level are related to each other. Such visual representations can also assist consumer, developer or evaluators to gain empirical understanding of the requirements and the interdependencies that exist in the domain to satisfy them. Visualization is also supported at the instance level. This functionality can be used by concerned stakeholders to visualize how specific instances of different requirements in the functional families interact within a particular system, in a graphical manner, allowing them to visualize the directly related instances (security objectives, dependent specifications, etc,.) that will be affected by the change in a particular specification. This assists the stakeholders in adapting to numerous requirement changes, often encountered in the wake of changes in technology and business.

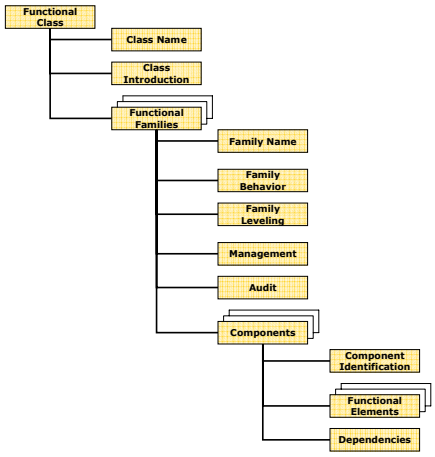

Figure 3: Requirements Structure in CC [4]

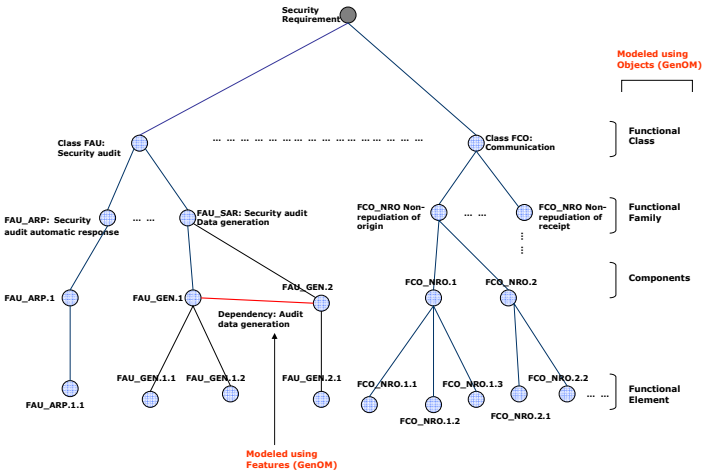

Figure 4: Categorization of Requirements in CC

In the next section we demonstrate the ability of the modeled CC requirements to effectively perform analysis from different aspects using an example concerned with identifying the security requirements for the authentication module for a personal digital assistant (PDA) system.

7. EXAMPLE OF A PDA SYSTEM

It is apparent from the model construction procedure described in the previous section that modeling in a GenOM environment can be performed at two levels; schema/meta-level and instance level. While modeling at the schema/meta-level, concepts operating in the domain are modeled along with their

dependencies. At the instance level, different instances of the concepts expressed at the schema level along with their interactions are modeled. The modeling that takes places at these two levels for the selected example is shown in Figure 8.

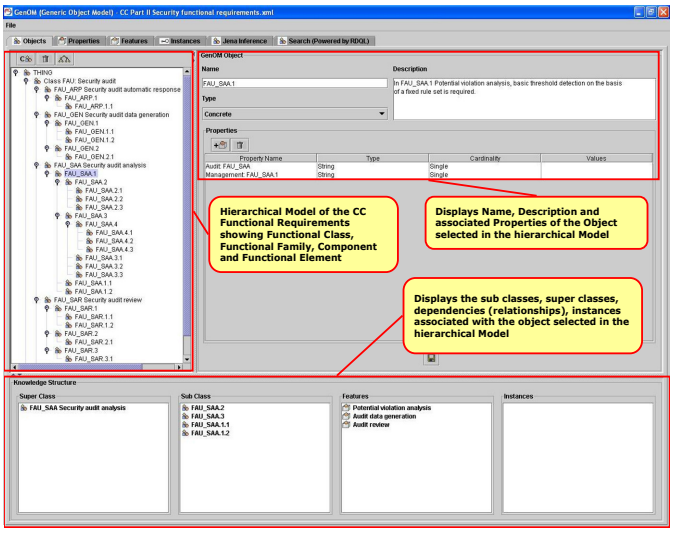

Figure 5: Object Model in GenOM

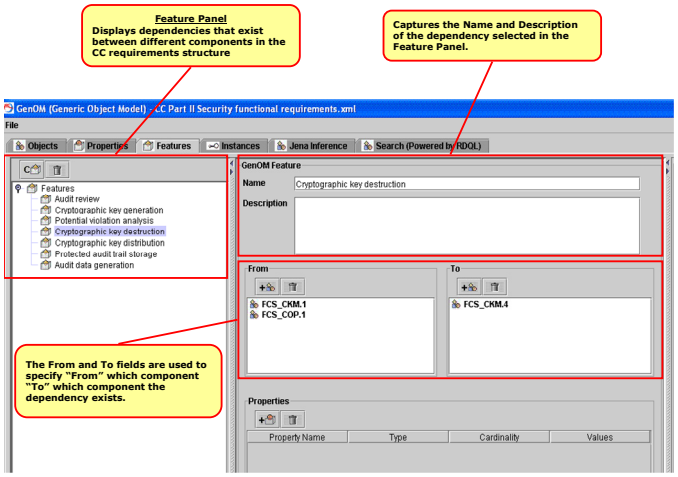

Figure 6: Feature Model in GenOM

For the chosen example, CC problem domain consists of the following concepts or classes:

- *Objectives*: Used to specify the objectives that are to be achieved by the system.
- *Entities*: Specifies the entities that help to achieve the objectives of the system.
- *Security Requirement*: Stores various security requirements to be considered by the system. The security requirements structure provided by CC is modeled under this class.

The dependencies that exist between various classes are modeled using the *feature* knowledge component provided by GenOM. The CC problem domain for this example has the following *features*:

- *Satisfies*: Connects an instance of class *Objective* to itself. The "From" and "To" fields in the GenOM interface (Figure 6) are used to specify the two classes that this relationship connects.
- *Satisfied By*: Connects an instance of class *Security Requirement* to an instance of class *Entity*.

As GenOM models follow an object-oriented approach, instances of any subclass of the class *Entity* can be used to connect to an instance of *Security Requirement* by the inheritance mechanism. Sub-classes inherit all the dependencies from their parents.

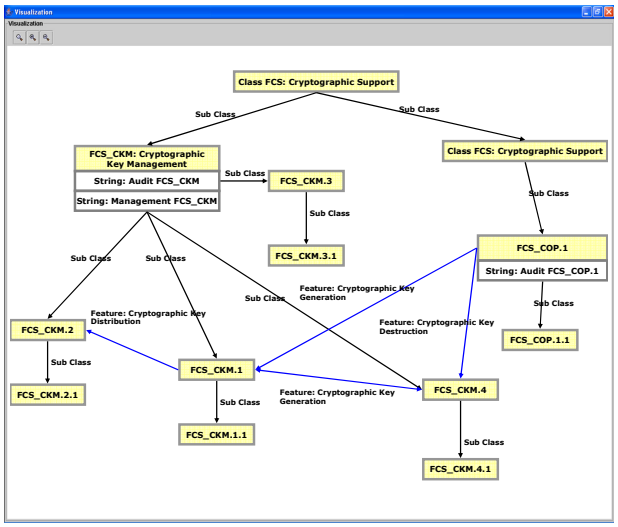

Figure 7: Meta-level Visualization of Conceptual Model

The PDA authentication module strives to achieve secure authentication in a reliable manner without affecting performance. Hence, we identify the three objectives to be satisfied by the system as: "Secure Authentication", "Reliability" and "Performance". The three objectives are modeled as instances of class *Objective* and they are connected by the *Satisfies* relationship. i.e. "Reliability" *Objective Satisfies* (helps in satisfying) "Secure Authentication" *Objective*.

For simplicity, consider a single security requirement: "Achieve password based authentication" for the authentication module of the PDA system. This specification (or requirement) can be modeled at the instance level as an instance of class *Security Requirement*. This particular specification has been identified by browsing and identifying the various CC security requirements needed to satisfy it. The browsing and identification can be achieved by utilizing the *Security Requirement* structure in the object model of GenOM. In order to satisfy this requirement we need two classes *Actor* and *System* and the relationship between these two classes is modeled as a feature *Authenticates* and is read as *System Authenticates Actors*. At the instance level, based on the above schema/meta-level, we create an instance of class *Actor* as "Administrator" and instance of class *System* as "PDA System".

We connect the "PDA system" and "Administrator" using an instance of the feature *Authenticates* in order to satisfy the "Achieve password based authentication" requirement. In order to make this fact more explicit we connect the "Achieve password based authentication" instance with "PDA System" and "Administrator" instances, using instances of the *Satisfied*

By feature defined at the schema level. Based on the type of authentication performed by the PDA system on the administrator, reliability and performance of the system are affected. An instance of feature *Authenticates* between the "PDA system" and "Administrator" instances will affect the reliability and performance of the system, therefore we further connect it with "Reliability" and "Performance" instances using an instance of the feature *Affects*.

GenOM is also associated with an inference engine [2] which can be used for two purposes. First, to identify the emergent behavior the system resulting due to changes in specifications and second, to identify the missing or hidden requirements or specifications that would be necessary to achieve the goals/objectives of the system under consideration.

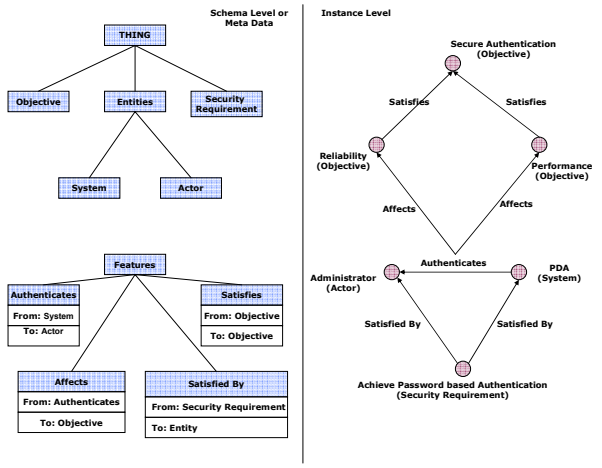

Figure 8: Ontology for a PDA System in GenOM

In the context of the example, if the "Achieve password based authentication" requirement is changed to "Achieve biometric Authentication", then the system needs to be re-evaluated to check if the initial objectives set forth by the system are still satisfied. The change from password based authentication to biometric authentication will cause a change in the authentication type between the "PDA system" and "Administrator" which will in turn affect the "Reliability" (improves reliability) and "Performance" (decreases performance) *Objectives* which will in turn affect the "Secure Authentication" objective of the system. The inference engine can assist the analysis based on such cause-effect reasoning and provide better support for the evaluation process. The inference engine can be also used for analyzing how a particular objective of a system can be achieved using the existing specifications, other objectives and instances modeled in the domain. For example, the inference engine can identify how a set of specifications (instances of *Security Requirement*) and specific actors (instances of *Actor*) can be used to achieve a certain objective. Such information would help the evaluator to verify, if all the necessary pieces of information have been considered to achieve a certain objective of the system.

The links between different instances in the domain act as traceability links between requirements, domain entities and objectives. Following this approach there exist traceability links between the high level objectives of the system to low level requirements or specifications, which aids the consumers, evaluators and developers to better understand, analyze and reason about the requirements. Also, the process of validating the system can be performed in an effective and efficient manner.

8. CONCLUSION AND FUTURE WORK

To create a product that meets the information assurance goals of CC, it is necessary to adopt a structured and comprehensive methodology that systematically utilizes the requirements set forth by CC to drive the activities undertaken in each stage of RE. In this paper we present, object-oriented ontology modeling as an effective modeling technique to represent CC requirements, which can help consumers, developers and evaluators to better understand these requirements in the application domain and effectively enforce the current practices in the C&A process.

Although the process of building a PDO is difficult and time consuming, once we have such a ontology in place, it is reusable across multiple systems. Such a representation can be useful, especially in providing an effective and efficient way to incorporate changes in policies, regulations, functional and non-functional requirements, and new legislations into the assessment process of an IT product. Our future work involves investigating the use of modeling techniques to systematically drive the RE process in a planned and predictable manner by engineering a framework that utilizes the models created so that we can identify and materialize functional requirements as well as assurance requirements to achieve high QoIA.

9. REFERENCES

[1] Breitman, K.K. and Leite, J., Ontology as a Requirements Engineering Product, *In Proceedings of the IEEE Int'l RE Conf., Mini-tutorial on Ontology Development*, 2003

[2] Carroll, J.J., Dickinson, I., Dollin, C., Reynolds, D., Seaborne, A., *Jena: Implementing the Semantic Web Recommendations*, Kevin Wilkinson Digital Media Systems Laboratory HP Laboratories Bristol, 2003

[3] CC ToolBox™. Developed by SPARTA, Inc. for the National Information Assurance Partnership (NIAP) http://cctoolbox.sparta.com

[4] *Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and general model*, , *Part 2: Security functional requirements*, *Part 3: Security assurance requirements*, August 1999 Version 2.1

[5] Hearn, J., Does the common criteria paradigm have a future?, *National Cryptologic Museum, Security & Privacy Magazine, IEEE*, Jan.-Feb. 2004

[6] Jackson, M. The Meaning of Requirements. *Annals of Software Engineering*, Vol. 3, pp: 5-21, Baltzer Science Publishers. 1997

[7] Lee, S.W. and Yavagal, D., *GenOM User's Guide,* Technical Report, Department of Software and Information Systems, University of North Carolina at Charlotte, 2004

[8] Swartout, W. and Tate, A. Ontologies. In Intelligent Systems, *IEEE, 14 (1), pp. 18-19*, Jan/Feb 1999

[9] Vetterling, Monika., Wimmel, G., Wisspeintner, A., Secure systems development based on the common criteria: the PalME project, In *ACM SIGSOFT symposium on Foundations of software engineering, Charleston, SC, USA*, 2002