Preparing for Large-Scale Investigations with Case Domain Modeling

By

Chris Bogen and David Dampier

From the proceedings of

The Digital Forensic Research Conference

DFRWS 2005 USA

New Orleans, LA (Aug 17th - 19th)

Preparing for Large-Scale Investigations with Case Domain Modeling

A. Chris Bogen
Engineering Research and Development Center, United States Army Corps of Engineers, Vicksburg, Mississippi
Dr. David A. Dampier
Department of Computer Science and Engineering
Mississippi State University
Chris.Bogen@erdc.usace.army.mil, dampier@cse.msstate.edu

Abstract

In any forensic investigation, planning and analysis activities are required in order to determine what digital media will be seized, what types of information will be sought in the examination, and how the examination will be conducted. Existing literature and suggested practices indicate that such planning should occur, but few tools provide support for such activities. Planning an examination may be an essential activity when investigators and technicians are faced with unfamiliar case types or unusually complex, large-scale cases. In complex, large-scale cases it is critical that the investigators provide computer forensics technicians with the appropriate amount of case data supplemented with keyword lists; too much case data or too little case data can make the forensics technician's task very difficult.

This paper presents the concept for a novel application of ontology/domain modeling (known as case domain modeling) as a structured approach for analyzing case facts, identifying the most relevant case concepts, determining the critical relationships between these concepts, and documenting this information. This method may be considered as a foundational analytical technique in computer forensics that may serve as the basis for useful semi-automated tools. An example case domain model is presented, the method for constructing a case domain model is described, and applications for case domain modeling are presented.

1. Introduction

Traditionally, digital forensics practitioners have been recruited from careers in criminal justice and law enforcement with limited previous computer or IT (information technology) experience. However, the increase in the occurrence of cyber crimes and the growing demand for more digital forensics technicians is extending recruitment to persons who originate from careers in computer science, software engineering, and information technology with limited previous criminal justice or law enforcement experience (such as the authors of this paper). As career IT personnel migrate to digital forensics, their problem solving approaches will follow. This paper describes how we have attempted to adapt a software engineering domain analysis method to computer forensics examination planning.

Many investigating agencies, especially in law enforcement, have distinct roles for an investigator and a forensics technician [13]. The investigator executes warrants and subpoenas, interviews persons involved in the case, conducts preliminary forensics tests, and populates the case file. The forensic technician forensically images media, uses forensically-sound examination tools, tags evidence, analyzes evidence, and reports the results. The scope and goals of an examination is initially defined by the investigator and refined by the technician. In large-scale cases, this activity of filtering voluminous case information is critical: Too many case details may overwhelm the forensics technician and lessen his/her productivity, and too few case details could make the forensics technician's task impossible to perform. Documenting the examination scope and goals is also important if the technician begins the examination several weeks after the investigator finishes working the case.

In large-scale cases, there may be an abundance of diverse case information often related to an unfamiliar case domain. Consequently, there is a high degree of uncertainty regarding the goals of a large-scale examination. For example, a computer forensics team may be tasked with imaging and examining more than 30 workstations and a few servers if they conduct white collar crime investigations of corporations or large organizations. In such circumstances it can be difficult to characterize the evidence of a crime and clearly outline the scope/goals of the forensics examination.

Some manuals and best practice guides for digital forensics stress the importance of planning a digital examination [1, 12, 13] and establish fundamental planning concepts. But there seem to be few tools, standards, or structured methods to support such planning activities. We propose that a case domain modeling methodology, adapted from software engineering requirements analysis, could serve as a structured analytical technique for deriving and representing relevant computer forensics case information. This structured derivation of relevant case information could be especially useful for decreasing the uncertainty associated with the goals of large-scale examinations.

The products of software engineering and computer forensics differ significantly. The former delivers a practical software configuration that consists of documentation, computer executable code, and data structures [8], while the latter delivers digital evidence and documentation that indicates the occurrence of a digital event. However, there are significant similarities between the approaches and underlying philosophies of software engineering and computer forensics: a focus on delivering a quality product, the importance of structured and scientific methods, the application of repeatable processes, the application of computer science concepts, and the application of software tools for supporting methods and processes. The remainder of this paper is organized as follows: Section 2 provides an introduction to domain modeling with UML (unified modeling language) and presents an example of a case domain model, Section 3 describes the process of building a case domain model, Section 4 introduces future applications of case domain modeling, and Section 5 presents conclusions and future work.

2. Introduction to Domain Modeling with UML

Software is typically developed for a specific field of discourse known as the application domain. Banking, audio recording, architectural design drafting, photography, and fluid mechanics are all examples of application domains. These application domains are populated by tangible objects, places, organizational hierarchies, processes, jargon, and policies that are often unfamiliar to the software developers. Software developers have varying viewpoints and assumptions regarding the application domain, and according to Uschold and Gruninger [14]: "the consequent lack of a shared understanding leads to poor communication within and between these people and their organizations…[and] difficulties in identifying requirements and thus in the defining of a specification of the system." In software engineering, domain analysis and modeling was proposed as a method for addressing this problem and reaching a shared understanding [4].

Domain modeling (aka ontology modeling) is a method for describing the characteristics of and relationships between concepts in a specific domain or field of discourse. Domain or ontology modeling is rooted in Plato's classical philosophical frameworks [9], and in the 1970s it emerged in artificial intelligence research (knowledge representation and content theory) [3]. Now there are several domain and ontology modeling languages and methodologies in Computer Science and software development: OWL (web ontology language) for Web development [6], UML (unified modeling language) conceptual and class diagrams for software engineering [5], and KIF (knowledge interchange format) for artificial intelligence [7]. The UML conceptual model and its underlying methodology are used as the framework case domain modeling because they provide the necessary expressive abilities while being relatively easy-to-understand. Alternatively, other ontology or domain modeling languages could be used to represent case domain models; ultimately the knowledge gained by building the model is more important than the syntax of the model.

UML conceptual models are developed during the requirements elicitation phase of software projects, and they help developers arrive at a shared understanding of the project's application domain [5]. The UML conceptual model notation is relatively simple as the model is intended to be reviewed by a layperson customer[1]. The foundational element of the UML conceptual model is the concept. A concept represents a "real-world" entity that may contain zero or more attributes that characterize the concept. Larman provides a list of common concept categories that may be used to identify candidate concepts [5]. Table 1 provides an abridged list of Larman's list of concept categories with specific examples of candidate concepts in the digital forensics domain.

Figure 1 provides an example of a case domain model (represented with a UML class diagram) for email threat cases in a university environment. The example case domain model assumes that a student sent his/her professor a threatening email from a public-use

[1] Customers or users may review the domain model to validate the developers' understanding of the problem domain.

Table 1: Concept Categories and Examples

Concept Category	**Examples**
Physical or tangible objects	*Cell phone, Hard Drive, CDR disk*
Descriptions of things	*Marketing Report, Incident Report*
Places	*Home, Street*
Transactions	*Payment, Sale, Money Deposit, Email Transmission*
Roles of people	*Victim, Suspect, Witness*
Containers of things	*Databases, Hard Drives*
Things in a container	*Files, Transactions*
Computer or Electro-mechanical systems	*Internet Store, Credit Card Authorization System*
Abstract noun concepts	*Motive, Alibi, Insanity, Poverty*
Organizations	*Mafia, Corporate Department, Government Organization*
Events	*Robbery, Meeting, Phone Call, File Access*
Rules and policies	*Laws, Procedures*
Records of finance, work, contracts, legal matters	*Employment Contract, Lease, Receipt, Subpoena*
Services	*Internet Service Provider, Telephone Service, Cell Phone Service*
Manuals, Books	*Flight Manual, Explosives Manual*

university lab computer. Concepts are represented by boxes, with the concept name appearing in the top of the box, and concept attributes are listed in the field below the concept name. Lines drawn between concepts indicate a named relationship; for example, *Class is taught by Faculty member*. A line with an "arrowhead" indicates a generalization specialization relationship; for example, *Faculty member* is a specialized type of the general concept, *University personnel*. The case domain model is an abstract representation of case information that is relevant to a specific case type, and this abstract representation is instantiated when applied to specific circumstances. To instantiate the model the investigators or technicians simply replace the attribute names with the specific values dictated by the case circumstances. For example, the *GPA* attribute of *Suspect* can be instantiated with the value 4.0. The following section describes how to build a Case Domain model by adapting the UML methodology for conceptual modeling

3. Building Case Domain Models

As the UML conceptual model allows software engineers to identify relevant concepts in a software problem domain, designing a case domain model is a process that offers a structured approach for analyzing, filtering, organizing, and documenting relevant case information in a computer forensics examination. The process of constructing a case domain model consists of three steps:

1. Select Case Concepts
2. Select Concept Relationships
3. Identify Concept Attributes
4. Instantiate Model

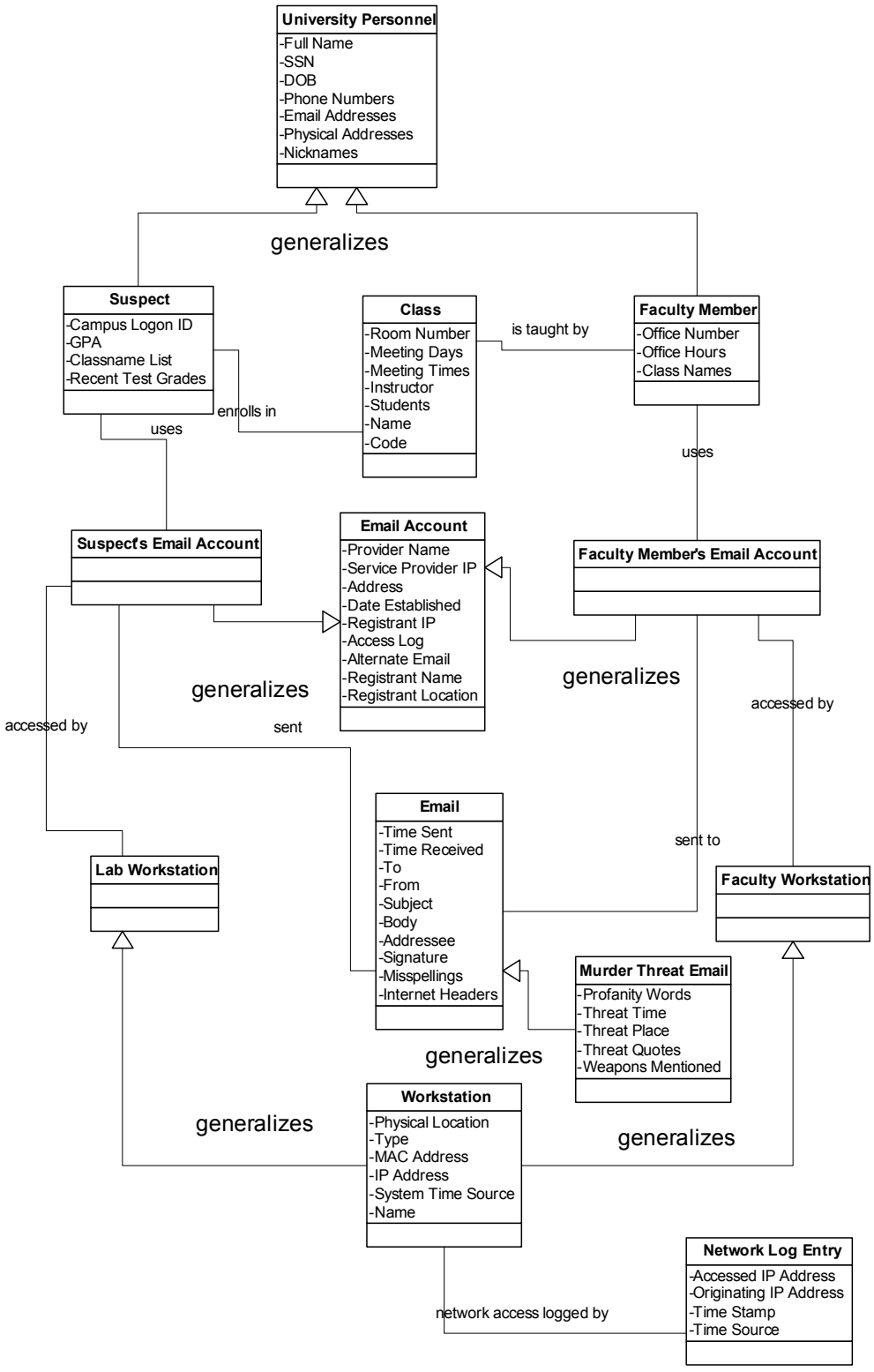

Figure 1: University Email Threat Case Domain Model (in UML notation)

The golden rule for selecting concepts, relationships, or attributes is, "if it is not relevant to the examination, then do not include it in the case domain model." The case domain model should not be a comprehensive representation of all entities/concepts involved in a case. Rather, the case domain model should represent all case concepts that are essential to the forensics examination. In large-scale cases it would be especially counterproductive to model every involved concept; even relatively simple cases could yield an unmanageable case domain model if all concepts are exhausted. Thus, each step in the process of constructing a case domain model must be supported by methods and heuristics for selecting appropriate concepts, attributes, and relationships. The following subsections will provide a brief discussion of methods and heuristics for each step of the model building process. These principles are derivative of Craig Larman's instructions on conceptual modeling in his textbook, *Applying UML and Patterns* [5].

Selecting Case Concepts

A list of candidate concepts may be identified by extracting nouns and verbs (known as noun-verb extraction [5]) from case documents such as underlying facts and circumstances, warrants, subpoenas, arrest reports, incident reports, etc. Additionally, a concept category table may be referenced when selecting concepts (see Table 1). Finally, the USDOJ's *Electronic Crime Scene Investigation a Guide for First Responders* provides a checklist (pp. 42-44) of common evidence entities that should be sought in certain types of investigations [12]; these may be directly mapped to case concepts. It is important to begin with a very exhaustive list of concepts and gradually eliminate concepts that are irrelevant. Some of the eliminated concepts may be modeled as attributes instead of concepts, so it is useful to preserve the candidate list of concepts for later use.

Reusability is an important factor to consider when selecting concepts; reusing concepts can save time when developing future case domain models. A concept name that is more abstract is easier to reuse than a concept name that is more specific. For example, S*uspect* is more general than *Patrick Bateman* and thus is easier to reuse in a later case. An attribute such as *Name* may be included in the S*uspect* concept in order to distinguish between actual instances of the concept.

Identifying Concept Relationships

For the purposes of planning a forensics investigation, the concept names and attributes are the most important items of information; concepts and attributes are the relevant pieces of information that the technician will use to seed the examination plan. However, relating the concepts adds an additional layer of information that can help an outsider understand the background and circumstances of a case. Table 2 lists some typical relationships that may occur between case domain concepts [5]. Such a table may be used as a checklist for identifying potential relationships between selected case concepts.

When too many relationships are selected then the complexity of the case domain model becomes unmanageable; imagine Figure 1 with lines drawn between every pair of concepts. Larman states that, "it is undesirable to overwhelm the conceptual [domain] model with associations [relationships] that are not strongly required and which do not

illuminate our understanding. Too many un-compelling associations obscure rather than clarify," [5]. Thus, redundant and derivable relationships should be avoided in favor of essential relationships that foster an understanding of the case domain. Multiplicity (aka cardinality) constraints may be added to the relationships to specify how many items are involved in the relationship: A *Suspect owns* 0 or more *Vehicle*s. Such constraints may enhance case domain understanding but they are not essential for deriving and identifying important case information.

Table 2: Typical Concept Relationship Categories

Category	**Examples**
A is a physical part of B	DVD Drive – Workstation
A is a logical part of B	Network Mapping – Network Intrusion
A is physically contained in/on B	Used CDR Media – CD Case
A is a description for B	Readme file – Executable Program
A owns B	Suspect – Vehicle
A is a member of B	Suspect – Gang
A is an organizational subunit of B	Information Technology Division – Company
A uses or manages B	Systems Administrator – Company Network
A is a specialized version of the generalized B	Systems Administrator – Company Employee
A communicates with B	Suspect – Associates
A is known/logged/recorded/reported in B	Email Registration – Network Logs

Selecting Concept Attributes

Attributes are the defining characteristics of a concept, and they represent the information that is essential to the computer forensics examination. These attributes may be referred to when constructing keyword searches, examining text documents, examining network logs, etc. For example, when looking for documents that refer to the suspect, the *name* attribute of concept *Suspect* can be elaborated to form a short keyword list that includes initials, nicknames, first name, last name, middle name, etc.

As a minimum, the list of attributes should be exhaustive enough to uniquely distinguish between instances of a concept. For example, the *name* attribute is insufficient for distinguishing between unique instances of a *Suspect* concept. What if two distinct people have an identical name? Appending this attribute list with *social security number* and *birth date* is sufficient information to distinguish between two distinct instances of *Suspect*. Common attribute types that may occur in a case domain include names, phone numbers, IP addresses, physical addresses, account numbers, email addresses, times, and dates. As was the case with other steps in the modeling process, it is important to maintain a moderate approach between providing a comprehensive attribute list and a minimal attribute list.

Instantiating the Model

When the abstract model is complete, then actual values are assigned to the case domain concept attributes. A significant amount of unknown attribute values may indicate the need to revisit pre-forensics investigative efforts.

4. Future Applications of Case Domain Modeling

Thus far, case domain modeling has been presented as an analytical framework for determining and documenting the scope of a forensics examination. Other applications of case domain modeling may include but are not limited to selecting keyword search terms, building expertise and reusing knowledge, providing an intelligent tool infrastructure, and supplementing existing forensics modeling approaches.

Selecting Keyword Search Terms

Keyword lists are often an important artifact for defining the scope of a search warrant and an examination. Given a case domain model a keyword search term list may be developed by following a process of attribute selection and elaboration. Investigators and technicians would select concept attributes that will be included in a keyword search. Each attribute may have its own list of keyword search terms that represent various synonymous permutations of the modeled attribute (e.g. a *date* keyword list should contain several representations of a date). The process of elaborating the attributes may be semi-automated with ontology modeling tools such as Protégé[2].

Knowledge Reuse and Expertise Building

Case domain models present an abstract view of case information, and the model becomes specific when the attributes are instantiated with actual values. These abstract domain models may be used on cases that share common characteristics. Additionally, concepts that are common to many cases may be shared between several case domain models (e.g. *Vehicle, Email,* and *Person*). Novice investigators may develop their investigative skills by attempting to construct case domain models, or by attempting to apply an expert's case domain models. Additionally, the abstract attributes of a Case Domain model may indicate what types of questions should be asked during suspect/witness interviews; i.e. the attributes provide the blanks of information on a form that the interviews attempt to fill in via interviews. Such an application could also extend the usefulness of case domain modeling from large-scale to small-scale cases. Successful reuse of case domain modeling would be highly dependent upon the availability of an effective tool for cataloging and searching case domain models.

Providing infrastructure for intelligent computer forensics software agents

If case domain models are supplemented with inference rules, then they could provide a robust knowledge base for intelligent computer forensics software agents. These intelligent agents could observe a technician's digital examination, refer to a case domain knowledge-base, and offer guidance or automated functionality of examination tasks. Such an application may be unrealized due to the complexity associated with constructing a formal knowledge base for general use. Developing a formal knowledge base for a specific case type is relatively easier, but doing so decreases the number of potential applications for its associated intelligent agents; if the knowledge base is specialized then the intelligent agents are also specialized.

---

[2] Protégé is an open-source ontology modeling tool developed by the Medical Informatics group at Stanford University. Protégé has an easy-to-use graphical interface and there is a terminological enhancement plug-in that could automate the elaboration of attributes.

Supplementing existing computer forensics modeling and analytical approaches

Case domain models can be used to represent the underlying information domain of existing computer forensics models such as Stephenson's DIPL [11], Bruschi and Monga's forensic graphs [2], and Schneier's attack trees [10]. Such supplementary views are common in software engineering modeling languages such as UML. If computer forensics researchers continue to propose modeling methodologies, then someday a comprehensive, unified computer forensics modeling language may emerge from several existing computer forensics modeling approaches.

5. Conclusions & Future Work

With semi-automated support, case domain modeling could increase the amount of evidence recovered without significantly increasing the combined effort of planning and executing a large-scale examination. The value of this additional effort may also be realized when general case domain models are reused on similar cases.

Adoption of case domain modeling and its associated applications will be unrealized until a suitable software tool is developed for designing, representing, and managing reusable case domain models. Such a tool could also include semi-automated support for keyword term selection and other applications. The most significant challenge of producing such a tool would be providing case domain modeling functionalities and interfaces that are practical and comprehensible to a diverse (with respect to professional and educational background) population of computer forensics practitioners. Preliminary research indicates that Stanford Medical Informatics' Protégé, an open-source ontology modeling tool, may be an appropriate starting point for building a prototype case domain modeling tool.

In general, any method or tool should be chosen based on the characteristics of the problem, and case domain modeling is no exception to this rule. Planning and developing an examination strategy are necessary tasks when a high degree of uncertainty is associated with the goals of the forensics examination; such circumstances are ideal for the proposed case domain modeling method. However, case domain modeling may involve too much overhead for straightforward, common examinations that do not require extensive pre-planning or execution effort. Furthermore, the proposed method could be outright ineffective for tasks that involve the recovery of little or no textual data (e.g. recovering illicit pornographic images).

We attempted a preliminary application of case domain modeling on a case at the Mississippi State Attorney General's Cyber Crime Unit. Though no formal experiments were conducted, the case domain modeling principles were successfully applied to a typical, small-scale case. The investigating officer remarked that the modeling framework was simple, appropriate, and would provide a nice enhancement to the typical "dry-erase board" method of outlining case information. Surprisingly, our modeling of the case domain identified a few important pieces of information that the experienced cyber crime investigator had overlooked. This positive anecdotal evidence has encouraged our further exploration of case domain modeling.

Bogen is currently evaluating case domain modeling applied to the selection of keyword search terms (and the automation thereof). Experiments will evaluate the amount of effort required to apply case domain modeling, the amount of evidence recovered using case domain modeling, and other qualitative factors. Additionally, this research may necessitate the construction of a case domain modeling software tool. The empirical results of this research will be included in Bogen's upcoming PhD dissertation (May 2006).

References

[1] Association of Chief Police Officers (AOPO), "Good Practice Guide for Computer Based Electronic Evidence," 2003; http://www.nhtcu.org/ACPO%20Guide%20v3.0.pdf (current 2004 May 31).

[2] D. Bruschi and M. Monga, "How to Reuse Knowledge About Forensic Investigations," presented at Digital Forensics Research Workshop, Linthicum, Maryland, 2004.

[3] B. Chandrasekaran, J. R. Josephson, and V. R. Benjamins, "What are ontologies, and why do we need them?," *Intelligent Systems and Their Applications, IEEE [see also IEEE Intelligent Systems]*, vol. 14, no. 1, 1999, pp. 20-26.

[4] N. Iscoe, G. B. Williams, and G. Arango, "Domain modeling for software engineering," presented at Software Engineering, 1991. Proceedings., 13th International Conference on, 1991.

[5] C. Larman, *Applying UML and Patterns An Introduction to Object-Oriented Analysis and Design*. Upper Saddle River, New Jersey: Prentice Hall, 1998.

[6] D. McGuiness and F. van Harmelen, "OWL Web Ontology Language Overview," 2004; http://www.w3.org/TR/owl-features/ (current

[7] N. Noy and C. Hafner, "The State of the Art in Ontology Design," *AI Magazine*, vol. 18, no. 3, 1997, pp. 53-74.

[8] R. Pressman, *Software Engineering A Practitioner's Approach*, 6th ed. New York, New York: McGraw Hill, 2005.

[9] R. Prieto-Diaz, "A faceted approach to building ontologies," presented at Information Reuse and Integration, 2003. IRI 2003. IEEE International Conference on, 2003.

[10] B. Schneier, "Attack Trees," *Dr. Dobb's Journal*, vol. 24, no. 12, December, 1999, pp. 21-29.

[11] P. Stephenson, "Applying DIPL to an Incident Post Mortem," *Computer Fraud and Security*, vol. 2003, no. 8, August, 2003, pp. 17-20.

[12] United States Department of Justice Office of Justice Programs, "Electronic Crime Scene Investigation a Guide for First Responders," United States Department of Justice, Washington, DC July 2001.

[13] United States Department of Justice Office of Justice Programs Computer Crime and Intellectual Property Section, *Search and Seizure Manual: Searching and Seizing Computers and Obtaining Electronic Evidence in Criminal Investigations*, 1.0 ed. Washington, DC, 2002.

[14] M. Uschold and M. Gruninger, "Ontologies: Principles, Methods, and Applications," *The Knowledge Engineering Review*, vol. 11, no. 2, 1996, pp. 93-136.