# A Knowledge Base for Information Security Modeling

## **Draft Study**

Author: Massimo Coletti

Director Security and Control Systems

Banca Finnat Euramerica S.p.A.

Version draft 1.2 released on June 7, 2006

# 1. Abstract

Compliance with laws, regulations, standards, internal procedures, in the area of information security systems is a complex activity.

One of the issues encountered, is the need to cope with different "models" of the information, processes, system architecture. Often the modeling activity is repeated once for every domain, and is difficult to keep all the developed models consistent and up-to-date.

This document is the story of the development of a unified, modular, knowledge base, with the goal of creating a unique and consistent repository, for the conceptual model, and for the actual configuration of the security system.

The project was developed creating an *ontology* of the concepts analyzed, and using XML as the basic standard for information storage.

## Copyright Notice



## Trademarks

## 2. The approach

My idea was to develop a knowledge base, intended as a repository of conceptual models and detailed information, about our systems. This repository is not a standalone entity, but should be easily fed with data coming from different sources. At the same time, I need to extract, from this knowledge base, data, mainly in human readable format.

After some thoughts, I decided to create the repository relying on XML documents. The reasons are quite straightforward:

- XML is a text-file based format, so it is not limited to some specific operating environments;
- some vulnerability assessment tools (for example nessus or ovaldi) can export their results in XML format;
- conceptual models designed with UML tools can be exported in XML format;
- it is quite easy to transform XML in other presentation formats (HTML, PDF);
- XML documents are managed by all the main development environment (Java, .NET, Ruby, Python, PERL).

The natural consequence of this assumption was to use OWL, as the XML-based standard to be used for the knowledge base representation. Having chosen OWL, the preferred tool to manage OWL files is Protégé, perhaps the leading platform for knowledge engineering and acquisition in OWL environment.

I will discuss at the end of the document the pros and cons resulting from my decision.

### Conceptual reference models

I didn't want to build my conceptual model from scratch, so I used some existing conceptualizations as a reference. In detail:

- MIB, for network architecture;
- the OSI stack;
- the information system conceptual model described in ISM3;
- the suggested reference model of the Italian Privacy Authority for information processing systems;

Using proved, widely accepted, standards as a reference for the design of conceptual models, is a great advantage, towards the goal of a well-engineered knowledge base.

# 3. The problem domain

I am a manager in the banking industry, responsible for the compliance to some rules and laws, and with the duty to set up the security strategy.

This assignment involves managing several sets of information about logical and physical structure of the information system, analyze them, take decisions, and produce reports and other kinds of documents.

The scope of the knowledge base includes initially three domains:

- the physical structure of the network;
- the conceptual model of the information security system, as defined in [ISM3];
- the Privacy Protection Act for Italian companies.

The domains deal with similar set of concepts, but each one has a slightly different perspective. I will try to summarize them.

## The physical network

This perspective is focused on the devices that actually operates in our network. The concept of network is quite extended, including not only the local networks in the different physical locations of the bank, but also the servers hosted outside of our structure, workstations of customers, remote workers, consultants, that operates with our network, from remote locations on the Internet, or other kind of connections.

In this first step, I limited the domain of analysis to the ip network, I know that this approach doesn't cover all kind of network connections.

The conceptual model here is very simple: the network is seen as a series of network segments, with nodes attached on it through interfaces. Each node hosts one or more execution environments. A node, usually, is a physical device (a server, a workstation, a printer, a router); I haven't detailed the case of virtual hosts. This modeling is inspired to the deployment diagrams of UML, described in [UML2], as well as MIB information bases.

### Links with the structure

I use this perspective to discover the "manageable" items in my network. This perspective has links with what I have called (not properly) "structure" information. The link is the information about "where" the nodes are located. This information tells me the "postal address" of the location, and the area (room, floor, etc.), within the facility, where the node is. In this way I can check the physical security against the requirements of the information assets managed by the node. If the facility is not owned by my Company, I can identify the legal owner, and check his responsibilities with the existing agreements.

Linking the logical model with the physical structure allows also to compare the security requirements – deriving from the classification of data hosted in a location – with the physical security measures in place.

## Information Security model

I have decided to follow a security management methodology called "Information Security Management Maturity Model", described in [ISM3]. This methodology if focused on the security of information managed by computer systems; the "information security system" is described in terms of components, that are the "information assets" that I want to protect.

Modeling my system according to the conceptual model of ISM3 helps me to grade the security requirement of each component, in order to choose the appropriate level of security measures.

As this methodology is focused on information security, the conceptual model is not linked with the

architecture of actual computer systems. For instance, there is not the concept of node (host), but instead the concept of *repository,* a store of information. Other structural components, are the interfaces, devices used to transfer information from/to the human users. The channels are the logical links where the information flows, and the borders are the perimeters of my information system. Non structural components (transactional components) are the services, providing some value to the user or other services, and the messages, used to exchange information from the user and the systems, or from service to service.

## Privacy law

The privacy protection rules in EU, and consequently in Italy, are quite advanced and detailed.

The "data Protection Code"[L196] requires that each company managing personal data, takes appropriate security measures to protect integrity, availability and confidentiality of such information. The company should report to the authorities how it is fulfilling this requirement, disclosing some information about the processes that deals with personal data: where the data are stored, how are accessed, from who. On the ground of this disclosure, the company should perform a risk assessment, and evaluate the current security measure against the actual risks.

So we have another conceptual model of the information system. Here – similarly to ISM3 – the focus is on information, and repositories of information. Another significant view is on the organizational aspect of information processing.

# 4. Modularity

One of the requirements of the project is to develop a knowledge base that is reusable in different contexts.

This requirement is pursued through a modular approach in the engineering of the knowledge base. The criteria that is used to evaluate the effectiveness of the modularization effort, is the number of dependencies existing among the different modules.

## Modularization criteria

The following criteria are used to identify the different modules:

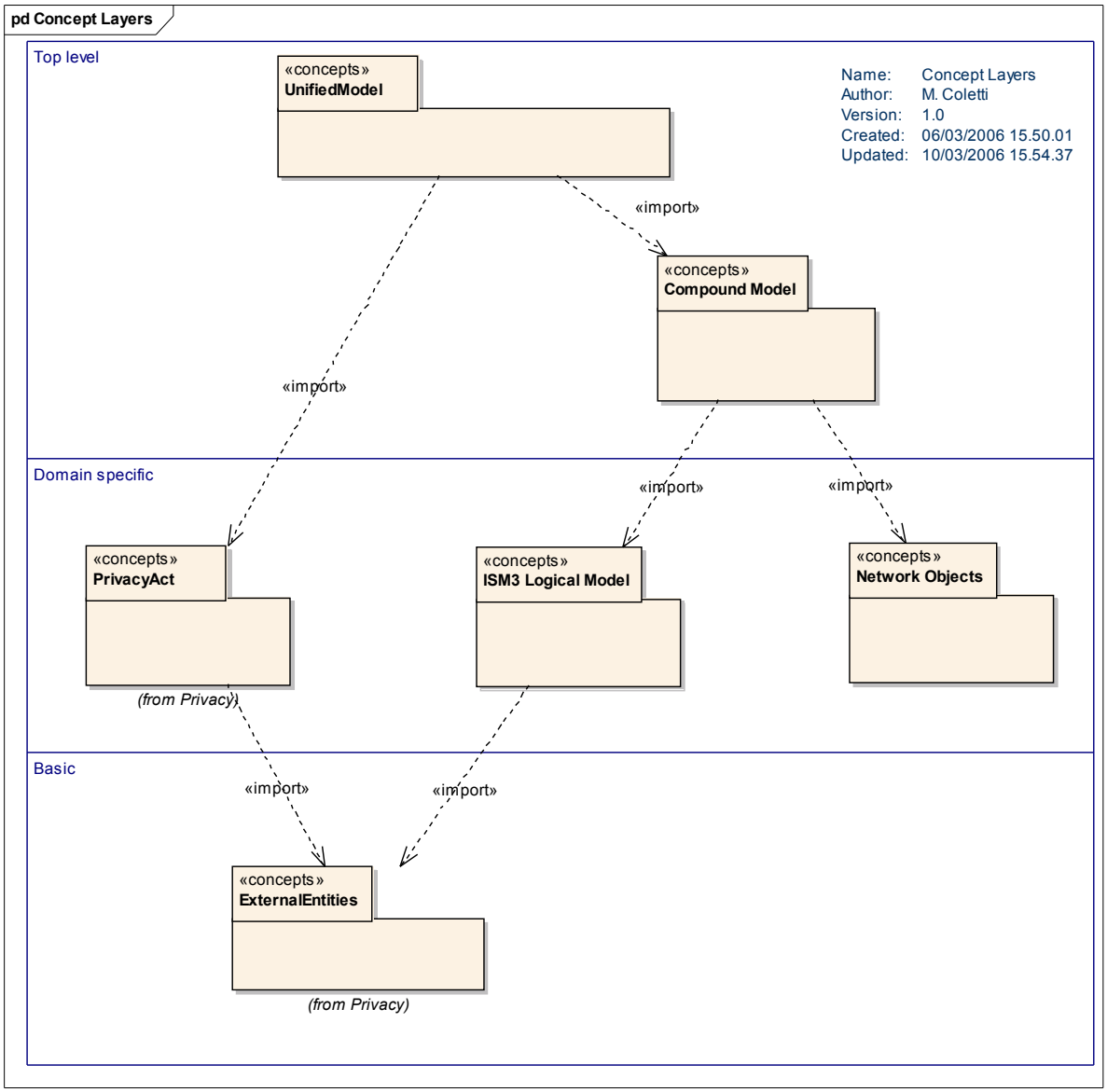
- instances separated from concepts: allows to use the same conceptual model for different realities (or for simulations);
- clear distinction from *single-domain* modules and *multiple-domains* modules. The first kind of module is used to represent concepts and instances pertaining to a single domain of analysis; the second kind is used to links together the single-domain modules.

The rationale behind this decision is to allow the reuse of a single-domain module in different contexts, without any overhead; furthermore, this approach allows to work on a single domain – when needed – having a sliced view of the whole reality: this means less complexity.

The above rules are not followed always. In some situation, I will never code all the real individuals for a given concept, but instead a single "generic" individual. For instance, I will not code all the single customers of our Bank, but only an abstract individual. Those "generics" are coded in the same OWL file with the related concepts; this doesn't break the modularization architecture: a "user" of this system may easily code specific individuals in a separate "instance" OWL file.

## Layers of modularization

The following diagram shows the piratical approach followed to implement the criteria described above:




In the diagram, we see:

- a *domain specific layer*: the modules contained here don't have any dependency on each other. The modules describe the conceptual model of a single view.
- A *basic layer*: here, under a single module, are grouped some "general" ontologies about organizational structure, external entities, that are imported in several upper modules.
- An *integration/top* layer, where the underlying conceptual models are merged together, in different steps, and provide a unified view of the system.

All the modules, stereotyped as UML packages, with a «concepts» stereotype, contain only concepts definition.

The OWL individuals – the instances of those concepts – are in a separate hierarchy of ontologies, with a 1:1 relationship with the "concepts" ontologies.

The following picture shows the resulting import tree in Protégé:


There is at least one more layer missing: the *foundation ontologies layer*. Using a foundation ontology, like DOLCE, could help to obtain an ontological-well formed knowledge base.

The dependency among different *domain* models are represented within the modules of the *top* layer.

## Redundancy issue

One of the (apparent) problems that I see in this approach, is that several "similar" concepts are repeated, with different identities, in the various modules. This leads to a number of properties, having those "similar" concepts as domains and ranges.

This was an apparent issue when I started the knowledge acquisition phase, but it ended in a different way. The "similarity" of concepts masks in the reality different abstractions of things that may happen to have some "physical" overlapping. As an example, the concept of "Repository" in ISM3 is very similar to the concept of "Database" in the Privacy Act domain; but modeling them as the "same" concept is an error: reading carefully the definitions of the two concepts, we can see that their meaning is really different, and overlapping them would result in a conceptual error.

In the current knowledge base, this relationship is modeled as a property "stored_on" having the Privacy concept as domain, and the ISM concept as range. This property, is dependent on two modules in the *domain* layer, for this reason is coded in the "Compound" module, the first step of integration in the *top* layer.

## Importing data from external sources

Some instance information will be manually coded in the knowledge base, but other are available from external sources.

At the present time, the Protégé-OWL systems isn't equipped with powerful tools for importing data from external files (primarily XML files). For this reason, one of the possible approach – involving modularization – is to create a single OWL file from each different XML source, using XSLT transformation or other means. In this way, it will be possible to import the file in the unified knowledge base, replacing it from scratch every time that the source data are refreshed.

## 5. The construction phase

Building the knowledge base was not easy. The development tool, as already mentioned, was Protégé. I believe that it is an exceptional effort, and that the University of Stanford have provided the community of people studying knowledge management with a really great value. When you develop an OWL ontology using Protégé, you have to choose the *language profile to adopt*. OWL comes in different flavors, each providing different expressive levels. My target was the OWL-DL (DL stands for *Description Logics* flavor). This is a level that allows for the interaction with a *reasoner*, a software tool that builds an *inferred model* starting from your knowledge base.

The decision to "limit"myself to the DL profile carries on some limits on the number of information and constraint that I can put inside my knowledge base. For instance, it is impossible to describe *cardinality constraints,* they "upgrade" the level of the language to OWL Full, which is highly expressive, but undecidable.

This decision is still open.

The other decision was about the engineering methodology: is it better to work "horizontally" (complete one module at each time), or "vertically" (build slices of knowledge, integrated across modules).

I have followed mainly the second approach, maybe something resembling the ideas of *agile* methodologies (*Extreme Knowledge Engineering? :-).*

This is not only a joke: I need to validate the modular design, as well as the resulting, integrated, conceptual model. Simple check to ensure these design goals questions like:

- am I able to code only domain-specific information when I am working on a *domain* module, without knowing at all what is related in the other modules?
- Is the complete knowledge base allowing easily a refactoring of the model?
- Am I able to ask relevant queries to the knowledge base, obtaining correct answers?

Coding a "slice" of information allows this kind of "unit testing", also if the conceptual model is not complete.

For instance, one possible query could be: "*List the repositories with customer-sensible data, hosted outside our Bank*". This query doesn't traverse the entire knowledge modular structure, but involves two domains (privacy and ISM3 conceptual model), plus the basic module about structures. I don't need a complete set of information, but I only need the following configurations:

- non-customer repository managed internally;
- non-customer repository managed externally;
- customer repository managed internally;
- customer repository managed externally.

The engineering of a knowledge system is similar to the software development process, but not equal. For sure it is a brand new practice, and reasoning about this is really stimulating. The main difference lays perhaps in the overlapping of the "developer" of the knowledge base and the "data entry" guy. This is why I use the term "engineering" and not developing, willing to emphasize the difference in the role of the developer.

I would like also to call your attention on the "queries". This knowledge base is not designed to elucubrate the philosophy of security system. Our bank needs this system in order to provide different sets of information in a consistent and effective way. This should be the engine driving strategic decision at corporate level, as well as to provide the core information disclosed in the information security reports.

## Gathering information from the real world

Of the different domains described in this document, at least two are mainly "conceptual", this means that the knowledge represented there, is the product of an analysis, a mediation of the knowledge engineer describing a reality made of people, processes, rules. I didn't find a way to extract automatically this information from somewhere.

Things are different when we approach the network domain, or the vulnerability domain (*this is not described in this document, but is part of the "big picture"*).

## Methodology and modularity

A knowledge information system is something that – at large – is highly unstructured, especially when you are engineering the knowledge, i.e. designing the knowledge structure and, at the same time, gathering information that will fill this structure. From another point of view, I think that this process is "data-driven": I discover the entities in the domain of interest, analyze their properties, the relationships with other entities, and derive a conceptual model from this. The conceptual model grows as long as I add more entities in the system.

This doesn't mean that is a ones-shot process; the conceptual model can be reused in a different environment, maybe without changes, applied to a different set of entities or individuals. This is why I am introducing an high degree of modularity in the system.

The modular structure of the knowledge system affects the process of knowledge acquisition. For example, in the Privacy Act domain, the relevant entities depend on a "basis" package of structural information – the organization structure, the external parties, and so on. This dependency should be reflected in the analysis methodology: first sketch a design of your business environment, then start the modeling against the specific domain model, with reference on the basis entities.

## 6. Acknowledgments