# Modelling and developing the information to manage an Internet Data Centre

Jorge E. López·de·Vergara[†], Julio Guijarro[‡], Patrick Goldsack[‡]

[†]Dept. Ingeniería de Sistemas Telemáticos
Universidad Politécnica de Madrid
Av. Complutense, s/n
E - 28040 Madrid. Spain
jlopez@dit.upm.es

[‡]Internet Systems and Storage Laboratory
Hewlett-Packard Laboratories
Filton Road. Stoke Gifford
Bristol, BS34 8QZ. United Kingdom
{julio_guijarro, patrick_goldsack}@ hp.com

## 1 Introduction

### 1.1 Overview

New Internet Data Centres are emerging that contain thousands of servers. In the near future they may contain tens of thousands. Such scale brings with it a fundamental discontinuity in the ways in which these Data Centres are managed. Administering this number of servers, all of them connected to the network, not only implies the control and monitoring of their performance, but also other aspects such as their fault-tolerance, security, accounting and configuration.
This paper discusses the different possible approaches to define and prototype a data model to manage the configuration of such centres in an easy and flexible way so the management system can deal with the required scale.

### 1.2 Structure of the document

This paper introduces first of all the concept of Internet Data Centres, explaining what they are and the needs of managing them. Then, it gives the different approaches for modelling the information to manage them. These include the Common Information Model (CIM) defined by the DMTF (Distributed Management Task Force), and other possibilities such as a tailored data model or a mixed version of those explained before. Another subsection, about how to deal with time and versioning in a configuration using CIM, is also included. Next section explains how a prototype of the data model has been implemented, showing how to change from an Object Oriented Model to a Relational Model, and also how to map an XML structure to a set of database tables. This section also gives a view about how this model works, dealing with both the hardware and configuration. Finally, some conclusions summarize the document.

# 2 Internet Data Centres

## 2.1 What they are

This paper uses the term *data centre* (sometimes spelt *data center* or simply *datacenter*) as a specialized facility that houses web sites and provides data serving and other computing services for other companies. This kind of data centre may contain a Network Operations Centre (NOC), which is a restricted access area containing automated systems that constantly monitor server activity, web traffic, and network performance and report even very slight irregularities to engineers so that they can spot potential problems before they happen. [10]

The importance of the data centres has grown due to the large number of xSP (Service Provider) currently competing in the market. They can be the well known Internet Service Provides (ISPs) or large Network Service Providers (NSPs), but also the new Application Service Providers (ASPs), Capacity Service Providers (CSPs), Content Delivery Service Providers (CDSPs) and Wireless Application Service Providers (WASPs) [2].

In fact, the purpose of an Internet Data Centre (IDC) is to bring a set of services to help people to provide any kind of Internet services [3]. Hewlett-Packard Labs have been involved in the study of these IDCs, trying to find the way of managing them in an automated way.

## 2.2 Why manage them

To provide an IDC service, maximising the resources and minimising the costs, the IDCs should have the following characteristics: [3]

- Easy to administer,
- High performance,
- Capacity and flexibility,
- High availability,
- Scalability,
- Security (data security as well as network security),
- Accounting and billing.

Most of these characteristics match perfectly in the FCAPS (Fault, Configuration, Acounting, Performance and Security) model, proposed by OSI [6] and adopted in the network management world. If the system can be managed covering those areas, then the service will be the best possible, and the IDC will achieve its goals.

# 3 Modelling the information

## 3.1 Why modelling is needed

Most management systems are based on three points [8], depicted in Figure 1:

- The communication protocol used to exchange the managed information. Many standard management protocols have been defined, such as SNMP or CMIP. HTTP can be another approach, as proposed by the DMTF.
- The format of the messages to understand the exchanged information. ASN.1 or XML are different neutral syntax that can be used for this purpose
- The information itself, which has been previously defined. In this case, much standard information has been defined in several MIBs (Management Information Bases). CIM (Common Information Model) is an approach that unifies the different existing MIBs, avoiding framework dependencies.

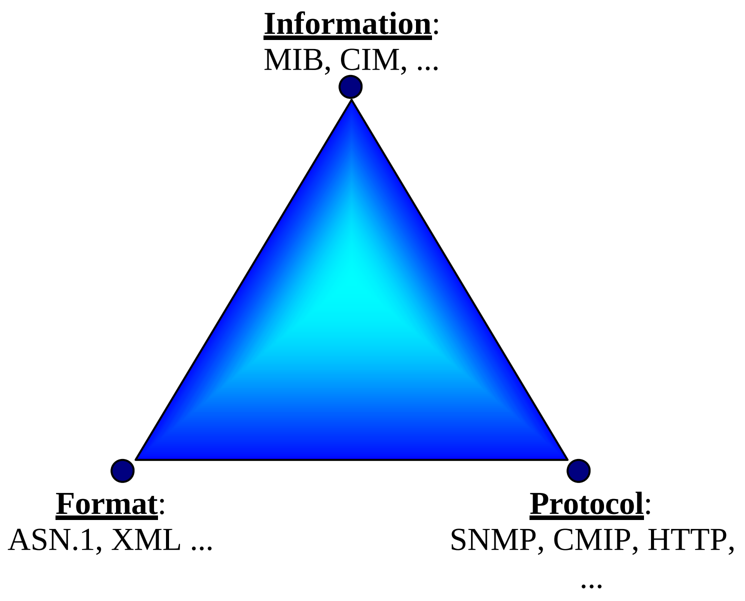


**Figure 1. The vertexes of the managing systems**

Protocol and format are the basement of the management systems, but they have to manage the information. Thus, if the information has not been defined, there is nothing to exchange.

Although much management information has been defined and standardized, it is possible that it is not suitable for the IDCs, due to its novelty. Modelling the information takes a long time, and it is not easy to predict if it is going to be easy to populate or query [9]. Then, it should be useful to study which information is available and if it is useful for managing an IDC.

## *3.2 Different approaches*

### 3.2.1 CIM

The first approach that can be used for modelling the IDC management information is to make use of CIM. CIM [4] is an initiative of the DMTF (Distributed Management Task Force) that tries to unify all the management information defined in different frameworks such as SNMP or DMI.

CIM is an Object Oriented model with a meta-model that can be mapped to a small subset of UML [7]. Thanks to this, very complex models can be created easily. Also, it is simply extensible, using the object-oriented inheritance. In fact, CIM has been conceived to be extended for each particular case.

As DMTF has been made up with the principal vendors, it contains most of the standardized information available. The information is grouped into different schemas, relative to different areas, which are extensions to a very simple core model: systems, networking, devices, users, applications, physical, support and directory integration.

However, the strength of containing so much information can also be a weakness: the model becomes more complex and it includes information that sometimes is useless or too detailed. Meanwhile, other information is already undefined.

It is also difficult to implement a system to handle this model. The easiest way to handle with data is using a database system, and most popular database systems use a relational approach instead of an object-oriented approach. This supposes to fix the *impedance mismatch* between these different data models. This question has been widely studied [1,4], and the results are that there are no recipes: each case requires its own *impedance adaptation*.

Anyway, there are many ideas contained in CIM, about how to organize the information, which should be included in the IDC data model.

### 3.2.2 Tailored model

The other approach would be to create a proprietary IDC model, specifically designed for this purpose. This can result in some problems: The information to be defined will not be standard, with the added cost that this implies, and the updates of the information will have to be included in the model every time a new kind of device is in the market.

The previous solution did not have these problems, as it was already standardized (as most vendors are in the DMTF they will update their new devices) and updated every time (five released versions in two years).

Also, if the model is defined using a relational model, it will be easily implementable in a database, but then, the flexibility given by an object-oriented model is lost.

### 3.2.3 Adopted solution

Finally, the trade solution adopted for the prototype comes from taking best ideas from both proposed models. The issue is to use CIM without all the complexity that it gives. This implies:

- Using databases as the simplest solution to implement the data model, making their population easy.
- Reducing the *impedance mismatch* with the object-oriented model, giving to it the highest possible flexibility.
- Reducing the information contained in CIM, but maintaining the useful hierarchical structure.

With this, from an initial CIM data model with more than one hundred classes interesting for the IDCs domain, the modelling resulted in a split into two submodels, one with twenty tables, related to the hardware inventory of the Data Centre, and very similar to an XML structure generated when scanning that hardware, and another model, with only ten tables (others can be added if needed), related to the configuration and quite similar to how configuration is modelled in CIM.

Section 4 explains more deeply the implemented prototype data model.

## 3.3 *Dealing with time and versioning*

Time and versioning are important items when doing configuration management. Time allows planning and verification of a deployed configuration. The versioning allows the maintenance of different configurations and makes it possible to come back to one of them at any moment. It is also desirable to have an historical log of how the deployment has been done, checking if the desired configuration is the same as the one that is instantiated.

One of the Common Information Model strengths is its wide scope, which covers most of the information required to manage a system. However, its definition of time and versioning is very poor. For instance, it only includes an installation date (InstallDate) for every ManagedSystemElement, so if an element is installed several times, only one of those can be maintained. Then, as time and versioning were needed for the IDC domain, a solution extending the CIM model was defined.

The proposed solution, shown in Figure 2, adds another class to the model, TimeSlot, and an association, TimeContext, which connects several configurations with several times periods.

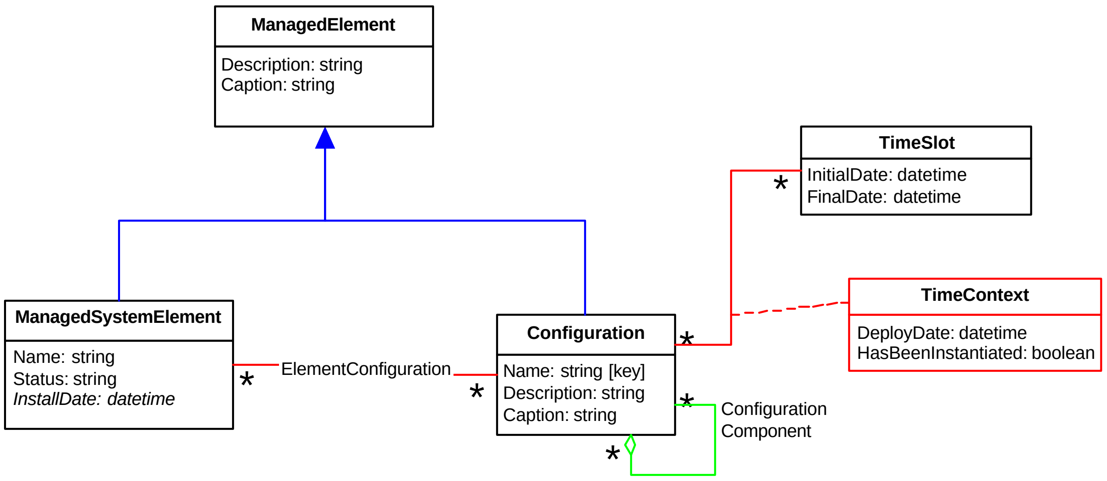


**Figure 2. Adding time to CIM Configuration**

This solution is very simple and flexible, as it allows the definition of several valid configurations for an element associated with different times. This can be useful as a rollback mechanism for returning to a previous well-known configuration.

It also allows the definition of configurations for different elements associated with the same time. For instance, many servers that have to run the same software simultaneously.

Adding a DeployTime in the association of a TimeSlot with a Configuration also allows the definition of a prior time at which the configuration should be prepared. The inclusion of another field, HasBeenInstantiated, could be useful to check if the configuration was instantiated at that time.

To avoid localization problems, all times should be related to UTC, so the location of the system maintaining the information could be independent from where the requests were sent.

# 4 Implementing the Configuration Management

## 4.1 *Mapping formats*

### 4.1.1 Object model to relational model

As noted earlier, mapping an object-oriented model to a relational model can be a hard task. Both models have some similarities, but their differences can be a problem when doing the mapping. Methods are not mappeable, but the relationships such as inheritance or aggregation can be solved in some ways.

The inheritance can be though as an IS-A relationship, in which a table has the foreign key of the parent table. This supposes having a table for each defined class. Other solutions are merging sons and parent tables into one table, but this denormalizes them.

The aggregation can be solved in two manners: the first one is using object relational databases, in which some structures can be defined as column fields. The second one is just thinking of them as associations.

N-to-N associations should be mapped into a table, meanwhile 1-to-N can be a column in the appropriate table.

Figure 3 shows how the object-oriented model of Figure 2 could be mapped into a relational model. In this case, all classes have a surrogate key, called ID, in addition to class properties. Then, descendant tables have that ID as a foreign key of the parent table. As association and aggregation are N-to-N relationships, one table has been defined for each one, referencing the ID field of the associated tables as foreign keys.

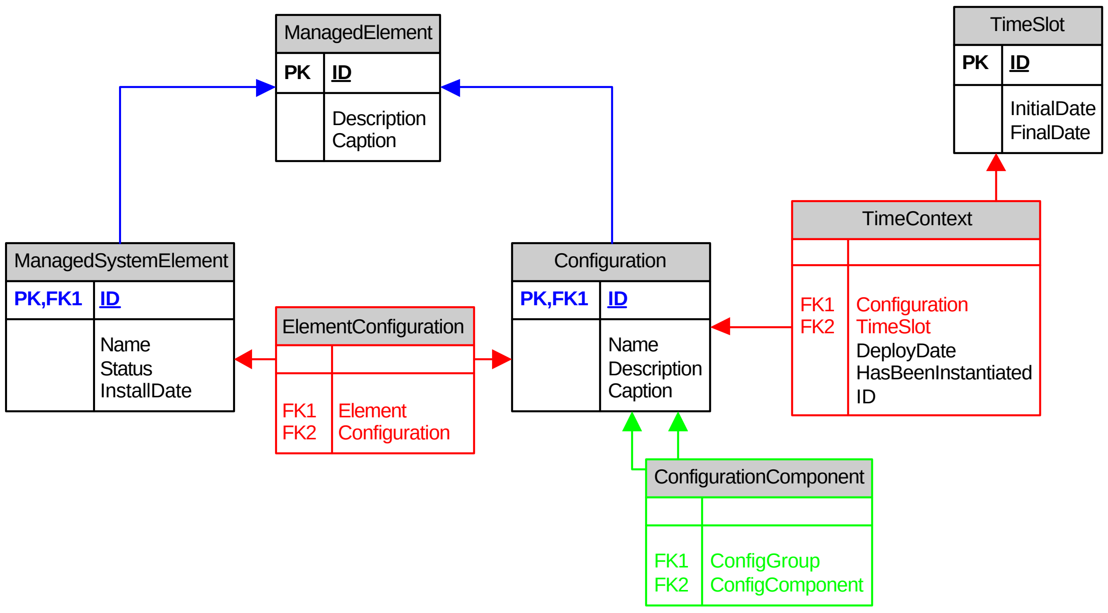


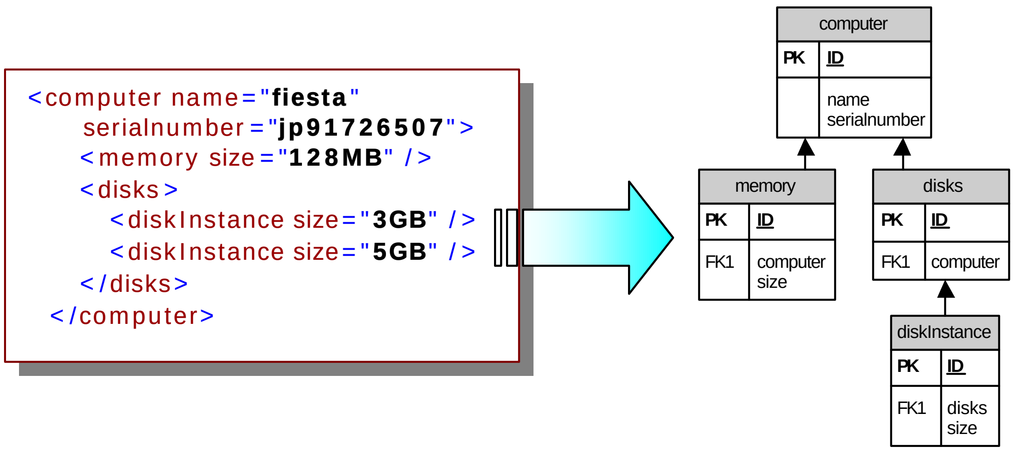
**Figure 3. Mapped relational model**

### 4.1.2 XML to relational model

Although CIM has an XML representation [11], this section is related to a different subject. The IDC developed in HP Labs uses a system that scans all the hardware of the network servers and generates an XML file with all collected information. The XML DTD is different to the CIM DTD, and it has been defined specifically for this purpose. It is, actually, a kind of log file with the hardware information the server had when booting at a certain date.

XML has been proven as a useful syntax to represent the hardware information: The information is easily parseable and the neutral syntax allows the conversion of the information to many formats such as HTML or plain text. However, inserting the information into a database can be a good idea, to connect the hardware information with the configuration information that has been defined using the CIM model.

Then, this database should be defined in a way it can be inserted in the easiest way, which is copying the structure of the XML tags: each tag represents a table, and if one tag is inside the scope of other one, then there will be a relationship between their relative tables. Tables should also have a surrogate key, which will be generated when inserting the information, although this is not necessary in XML. Then, the relationship is done using these keys. Figure 4 shows a simple example with four tags that are mapped to other four connected tables.



```
<computer name="fiesta"
     serialnumber="jp91726507">
     <memory size="128MB" />
     <disks>
        <diskInstance size="3GB" />
        <diskInstance size="5GB" />
     </disks>
  </computer>
```

**Figure 4. Mapping an XML structure in a relational model**

## *4.2 The final prototype*

### 4.2.1 The configuration data model

The final implementation of the configuration data model is shown in Figure 5.

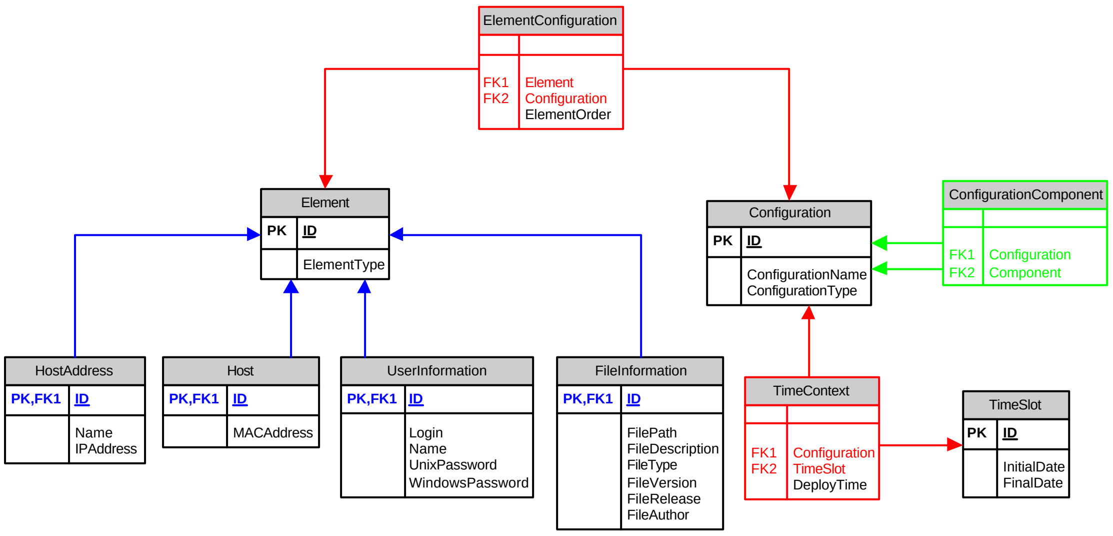


**Figure 5. The configuration data model**

This model is an extrapolation of that one shown in Figure 2 and Figure 3. However, some attributes or classes have been changed. But in general it contains the same ideas: There is an Element table that would be as the ManagedSystemElement class, which is the parent class of every element in the system. Then, there are also Configuration and TimeSlot tables, as well as the different association tables, such as ElementConfiguration, ConfigurationComponent or TimeContext.

The way in which this database must be populated should start introducing the data for the referenced tables (those that have the arrowhead), and then the referencing tables. The most difficult part is when inserting different elements, as they should be inserted previously in the Element table, because their ID is a foreign key contained in such table.

### 4.2.2 The hardware data model

Figure 6 shows the implemented hardware data model. It has not direct relationship with the CIM model, but most of the obtained properties can be mapped in some CIM classes' attributes.

As explained before (see 4.1.2), this model is quite similar to the DTD of the XML files that are generated when scanning the computers. The election of this solution was caused because the fact of that it is very difficult to fill all the information defined in CIM, and that some of the obtained information is defined in several classes or it is not defined at all.

Also, it is very useful to have human-readable XML file with the inventory of the hardware, instead of looking up table by table which elements are part of a proper host.

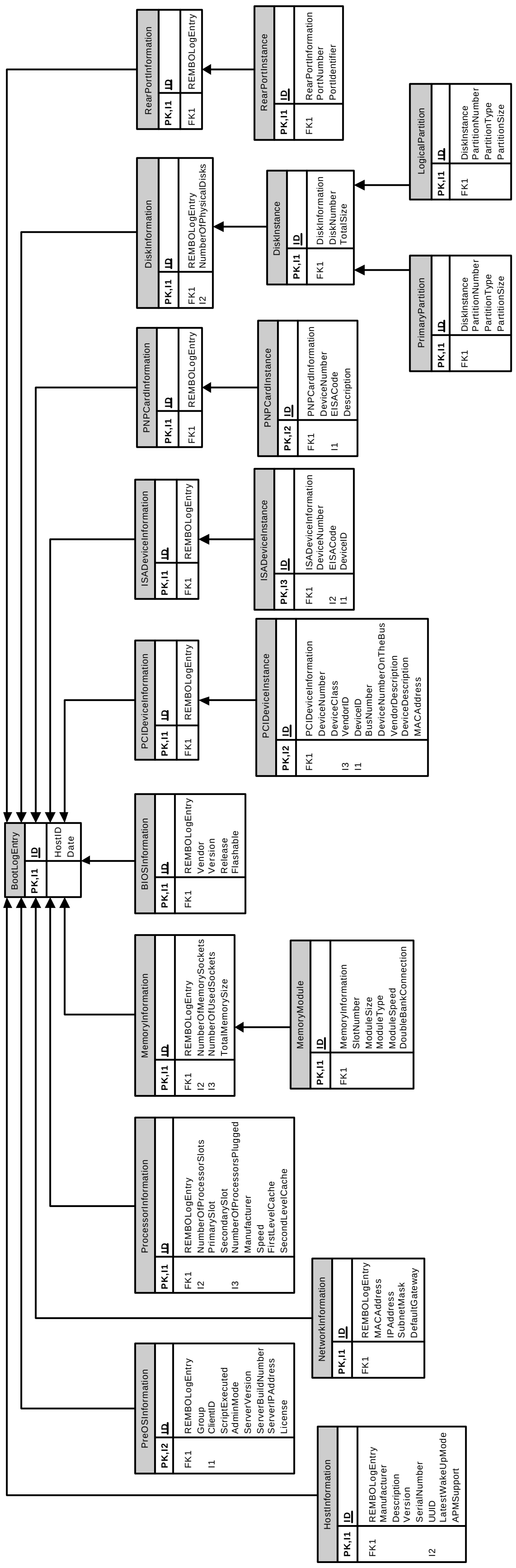


**Figure 6. The hardware information model**

## 5 Conclusions

The Internet Data Centres have come into the market providing a way of scaling the electronic services. These IDCs have to be managed in some way to improve the benefit/cost relationship.

Then, it is important to define the information to be managed because if not, there is nothing to exchange between manager and managed entities. Different approaches can be used for that, such as CIM, a standard unified information model or other non-standard tailored model. They have many advantages and drawbacks, so the trade solution for the implemented prototype has been a mixed approach taking best ideas from both ones.

To implement the data model, some concepts have to be applied to transform an object-oriented model onto a relational one. Also, a way to store an XML structure in a database has been defined.

A prototype has been implemented in two different data models, to reduce the complexity of the CIM model as much as possible. The configuration information takes some ideas from CIM configuration (inheritance, configuration relationships), but reducing the number of classes and adding time considerations to deal with different configurations for the same machine at different dates. The hardware information is based on an XML structure generated when the hardware of a machine is scanned. Both models can be linked using the identifier of the host to look for the hardware that had a certain configuration.

## 6 Acknowledgements

This paper has been done as a part of the Serrano project, focused on the Internet Data Centres. The authors would like to thank all the Serrano team for their ideas and comments about the data model.

## 7 References

1. Scott W. Ambler, ***Mapping Objects To Relational Databases***. Ronin International. July 2000. Available at http://www.AmbySoft.com/mappingObjects.pdf
2. Sujata Banerjee and Xiaoyun Zhu. ***Internet Data Centers: A Survey of Key Players and Market Growth***. Hewlett-Packard, September 25, 2000. Available at http://bisl.hpl.hp.com/projects/idc/docs/market1.doc
3. Sylvain Baudoin, ***Internet Data Center. Network administration and application environment service.*** Ecole Centrale de Lyon. Information and Communication Technologies. April-September 2000.
4. Winston Bumpus (Editor), John W. Sweitzer, Patrick Thompson, Andrea R. Westerinen, Raymond C. Williams, ***Common Information Model: Implementing the Object Model for Enterprise Management.*** John Wiley & Sons, 1999.
5. Cetus Links, ***Databases: Mapping Objects to Relations***. 2000. Available at http://www.cetus-links.org/oo_db_systems_3.html
6. International Standardization Organization. ***ISO 9595. Information Processing Systems - Open Systems Interconnections - Common Management Information Service Definition***, Geneva, Switzerland, 1990.

7. Object Management Group. ***Unified Modeling Language (UML) 1.3 Specification***. OMG document formal/00-03-01. March 2000.
8. William Stallings, ***SNMP, SNMPv2, and CMIP: The Practical Guide to Network Management Standards***, Addison-Wesley, 1993.
9. Patrick Thompson. ***Introduction to CIM Modeling***. DMTF Developers Conference 2000. Available at http://www.dmtf.org/download/presentations/conf2000/c101_102.pdf
10. whatis.com. ***Data Center***. Word suggested by Roger Godinho. Last updated on: Nov 14, 2000. Available at http://whatis.techtarget.com/WhatIs_Definition_Page/0,4152,332661,00.html
11. World Wide Web Consortium. ***Extensible Markup Language (XML) 1.0 (Second Edition)***. October 2000. Available at http://www.w3.org/TR/2000/REC-xml-20001006