CAMELEON: A CIM "Modelware" platform for distributed integrated management

M. Sibilla, A. Barros de Sales, Y. Steff, T. Desprats, D. Marquié
IRIT Laboratory
sibilla @irit.fr

François Jocteur-Monrozier, C. Lasserre
CNES
Francois.Jocteur-Monrozier@cnes.fr

Anne-Isabelle Rivière
Alcatel CIT
Anne-Isabelle.Riviere@alcatel.fr

Abstract

This paper presents the CAMELEON "Modelware" platform result of a R&D project led by CNES (French Space Agency) and developed by Alcatel TITN Answare and IRIT Research laboratory.

CAMELEON is a distributed management platform for complex systems. This project provides an innovative approach for global and distributed management system involving object-oriented technologies such as CIM, CORBA and Java.

1. Towards a new generation of Network and System Management

The requirements of specific user communities is defined by the qualities of service (i.e. secure; safe critical; time critical; easy to use) that best categorise their actual requirements. These requirements imply resource management in end-systems and network nodes.

A distributed system contains a number of distributed applications cooperating with supporting services. The supporting services include all services provided by a traditional operating system in a centralized system (i.e. file storage, user access) and include services necessary for system distribution (i.e. name management, trading and services supporting distribution transparencies). All these services required management together with the applications: the nature of the management functions required to do this will depend on the concerned services or applications.

Management systems should be able to manage entire companies from a global point of view. This means they have to handle and support the management of thousands of entities, computers or devices. This requirement increases by a lot the management information volume.

In network management domain, the informational model has always represented the implemented management system functionality. Implementation details have never been specified, only protocol access. Therefore, management architecture needed to resolve heterogeneity from management protocols and informational models [1], [2].

The network management expertise can be modelled in order to manage complex systems. With the WBEM initiative, and more specifically with CIM schemas, a unified method of gathering management knowledge is born. The most important are the common models for the entire management community.

Nonetheless, some questions on distributed management system arise: how to deal with managed resource heterogeneity? How to share this knowledge between management systems? What about management interaction?

We will present our solution CAMELEON platform and some answers gained from our experience.

The CAMELEON[1] platform is the result of a R&D study, initiated by the French National Space Centre (CNES[2]). CAMELEON (formerly known as *Sumo*: **SU**pervision et **M**aîtrise des **O**pérations spatiales) was a study done between Dec. 1999-June 2001[15]. It was developed in collaboration with ALCATEL CIT and IRIT laboratory. The objective is to manage large numbers of complex space systems. Indeed, in space domain, space ground segments systems have to manage an increasing number of satellites (e.g. constellations of satellites). A new approach is needed to reduce cost and increase autonomy and automation of such complex systems.

The main requirements for the approach CNES wanted to take with CAMELEON are the following:

- Homogenisation of management information systems,
- Improved competitiveness and durability of solutions obtained, through the use of standard technologies for the development of applications and for the management of systems.
- Optimal factorisation and reuse (implementation of generic "technical" or "task" components).
- High integration capability of legacy systems.

In order to achieve such requirements, CAMELEON combines the three following standard technologies to

[1] The term CAMELEON is an analogy to the animal.
[2] Centre National d'Etude Spatiale

provide integrated, scalable management of heterogeneous complex systems:

- The WBEM [3] initiative [3], especially the CIM [4] [4] as a management information model,
- CORBA [5] as communication infrastructure,
- Java as development and prototyping language.

2. CAMELEON : A "Modelware" Platform

2.1. What is "Modelware" platform?

We propose the term "Modelware" that means: "if something can be modelled, the modelware infrastructure can be managed it". This approach combines Software Engineer [9] and Network Management ones. CIM schemas are independently expressed regardless middleware technologies (Platform Independent Model). We propose this term "Modelware" to the French national group OFTA [10].

CAMELEON platform is composed of management knowledge and management entities (Object Managers-OM). The management knowledge is modelled using CIM (schema and instances). An OM has several functions in order for it to be processed (see Figure 1).

Managed objects are abstracted from real managed entities. In our platform, they are java objects gathered together into a conceptual management database (called management information base).

Object Providers (OPs) act as gateways between the management system represented by OMs and the real managed world (WBEM integration viewpoint). The CIM management models give an uniform view of heterogeneous management domains.

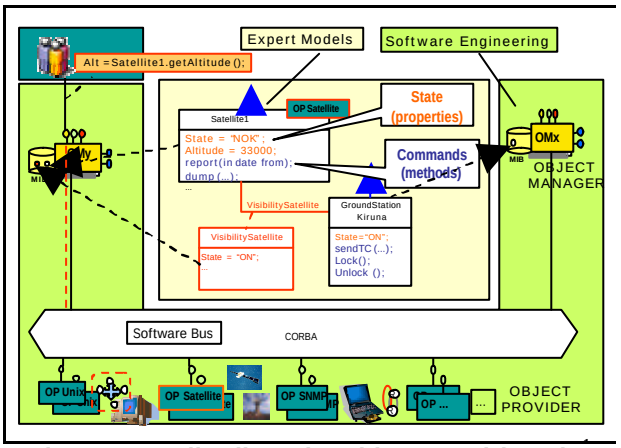

Figure 1 : A distributed architecture driven by CIM models

As represented in figure 1, any OM or OP is a CORBA Object. In consequence:

- Inter-OMs communications use CORBA/IIOP.
- OM-OP communications use also CORBA/IIOP.

Any Object Manager has to offer and implement all of basic management functions (such as CIM-objects mgmt fct, Dependency mgmt fct, Historic mgmt fct, State mgmt fct, Event mgmt fct), whereas additional functions are optional and may be activated when starting the OM (such as Security mgmt fct, Fault Tolerance mgmt fct). These functions represent the executive part of this "Modelware" approach. For more details, please see [6].

In order to illustrate this term Modelware, we shall refer to the following analogy to explain it : CIM models are the blood, CAMELEON platform is the heart !

2.2. CIM Management Knowledge Integration in the CORBA platform

The management knowledge is originally fed to an initialisation mof file, loaded thanks to our CIM parser. Internal representations of CIM objects (classes, instances, properties, methods…) are mapped to Java objects by the *CIM-API* which is developed by Sun [8] (see figure 2). Management functions provide data processing at the OM level.

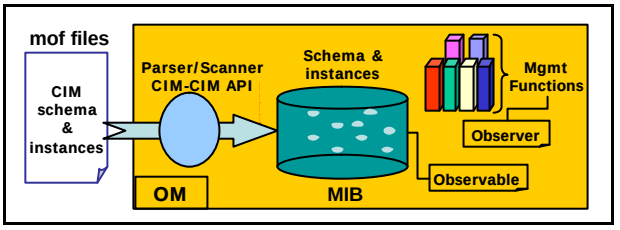

Figure 2 : Management Knowledge Integration

Since all communications between management entities are done through CORBA, most of the information that has to be exchanged is actually CIM information. First mapping proposed from CIM to IDL thanks to which this mapping, management objects (OM and OP) interfaces stay simple. These interfaces are similar to traditional management interface: **get** , **set** and **invoke**.

The mof descriptions of elements to be supervised are spread within each distributed management entity.

Because all the management entities of the CAMELEON architecture are CORBA objects, they can communicate transparently and independently from the location, the machine, the OS in which they are running. However, management knowledge distribution introduce issues that we consider in section 4.

In the following section, we will develop the major CIM modelling contributions and draw out modelling issues.

[3] Web-Based Enterprise Management
[4] Common Information Model

3. CIM Contributions

The CAMELEON platform is the result of the SUMO project lead in parallel with the DMTF works, during these last three years. Some of CAMELEON contributions will be summarized in this section.

The CIM meta-model and the mof notation can be added at the meta-model level (see Figure 3).

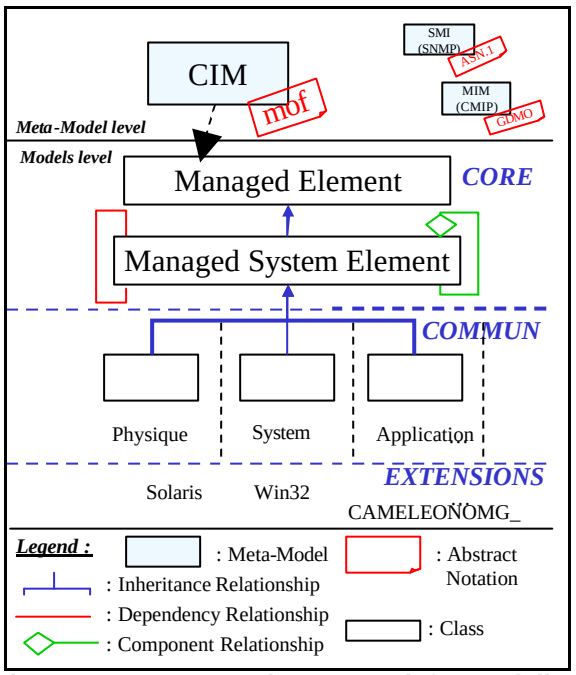

Figure 3: A conceptual Framework for Modeling

The added value of its common models framework is to unify the network and system management modelling at the Models level.

3.1. CIM schema extensions

In the CAMELEOM platform, the major CIM schema extension was done in order to manage the CORBA environment and CAMELEON itself. The schema name CAMELEONOMG_ was created, however it must eventually be replaced by the schema name CORBA_ (at the Extension model level) if it is adopted by the OMG. This work must to be compared to the new version of Application Model [11].

Other extensions made are:

- some CIM classes in the Common model (these class names are preceded by the CAMELEONCIM_ schema name),
- some classes needed by the CAMELEON management entities (they were introduced by the schema name CAMELEON_).

3.2. A new pattern for active dependency

Based on a management point of view, the semantics of CIM_Dependency association class is strong. When we specialize CIM_Dependency association, we define new association classes. We can add properties and methods, but we cannot specify how the Antecedent's evolution affects the Dependent.

For example, if a system (Antecedent) goes down, all the hosted services (Dependents) must be affected.

Here, an event-action correlation should automatically determine if the event "The Antecedent's Status property was updated" occurs, the action to perform is "to update Dependent's Status property".

This example of Dependency correlation could be translated in the class methods, however it should appear as a static and implicit mechanism.

This requirement cannot be expressed by CIM_Action and CIM_Check derived classes, because these classes express the status change of instances of CIM_ManagedSystemElement derived classes.

In order to simplify the designing and to reuse the event-action correlation, we propose to define a new object class CAMELEONCIM_ActionOnDependency which specifies what event is waited, and what action must to be performed when the event occurs. The CAMELEONCIM_DependencyAction association links dependency and action (see Figure 4).

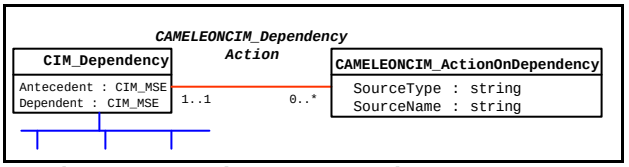

Figure 4 : "Action on Dependency" pattern

The action CAMELEONCIM_ActionOnDependency is an abstract class used to represent action on dependency. The following properties describe the expected event:

- SourceType (an enumerated type that specifies which Antecedent's element is watched: Property, Method or InstanceOfClass),
- SourceName (name of element),
- SourceActionType (an enumerated type that specifies the action on the element that throws the event: Create, Delete, Update, Access, Before, After).

To specify the resulting action to perform on the Dependent, we define several subclasses of the CAMELEONCIM_ActionOnDependency class (see figure 5).

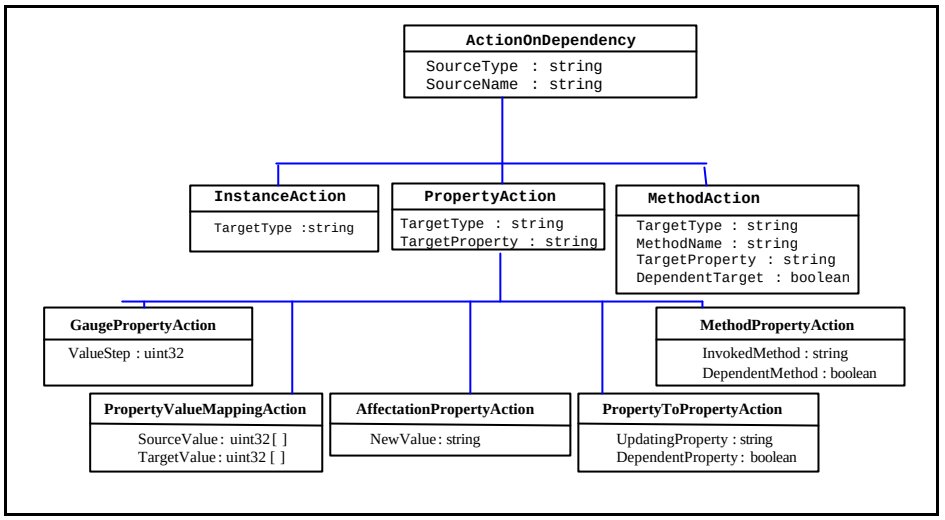

Figure 5: The CAMELEONCIM_ActionOnDependency subclasses

The Dependent's target element affected by the action can be an instance, a property or a method. As the target element, the action may be (enumerated TargetType property):

- Delete, Lock[5], Unlock[6], Update[7], Reset[8] on an instance,
- Set, Update, Reset on a property,
- Invoke, Lock, Unlock on a method.

- To update a property, several actions are identified:
- CAMELEONCIM_GaugePropertyAction
  To increase or decrease a gauge (or a counter),
- CAMELEONCIM_PropertyValueMappingAction
  For enumerated type property (such as status), a value mapping between Antecedent's and Dependent's properties can be identified.
- CAMELEONCIM_AffectationPropertyAction
  A property is updated with a new value.
- CAMELEONCIM_PropertyToPropertyAction
  A Dependent's property is updated by the value of a Dependent's or Antecedent's property.
- CAMELEONCIM_MethodPropertyAction
  A Dependent's property is updated by the return of an invoked method.

See Example of mof CAMELEONCIM_ActionOnDependency instanciation between a System (Antecedent) and a Service (Dependent) in figure 6.

The CAMELEONCIM_DependencyAction association has to be instanced in order to link an instance of the CIM_HostedService Dependency and this Action instance.

```
Instance of CAMELEONCIM_
PropertyToPropertyAction
{Name = "action1";
 SourceType = 1;           // Property
 SourceName = "CIM_System.status";      //
Name of the Antecedent's property
 SourceActionType = 3;     // Update
 TargetType = 1;           // Set
 TargetProperty = "Status"; // Name of the
Dependent's target property
 UpdatingProperty : "CIM_Service.status";//
Name of the property which is used
 DependentProperty = false;
// The UpdatingProperty belongs to the
    Antecedent
}
```

Figure 6: Example of mof CAMELEONCIM_ActionOnDependency

This ActionOnDependency model can be added to schema and instanced into the MIB. It must automatically be taken into account by the ActiveDepedency function which is added to an OM regardless object location.

3.3. A global management point of view

Because of this model-driven approach, the power of expression of a meta-model contributes to enhance the management power.

[5] The element is not accessible
[6] The element becomes accessible
[7] The value(s) of the element is (are) got from the real resource.
[8] The value(s) of the element is (are) reinitialized by its default value.

CIM schemas and instances are abstractions of managed domains. They represent the management knowledge shared between management entities. We situate them at a kernel level (see figure 7).

Thus, the technology level performs management activities.

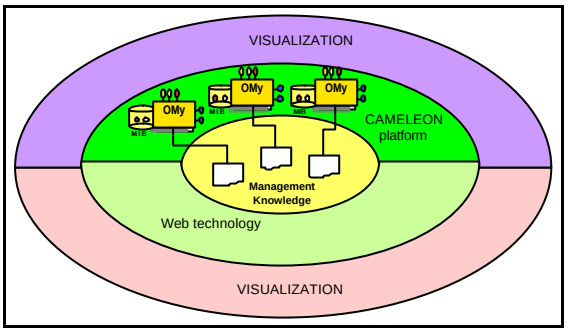

Figure 7: A global management point of view

Finally, graphical user interface tools are provided on the visualization level, such as:

- **The CIMWatch user interface**. The CIMWatch user interface is similar to a MIB browser. It presents all management entities, their managed objects and their data. An automatic refresh is accomplished by using notification exchange.
- **FTWatch**. The FTWatch user interface visualizes the Fault Tolerance organization between CORBA objects when this function is activated. An automatic refresh is accomplishedby using notification exchange.
- **Task Scheduling**. The management application "Task scheduling" manages the running of task planning with Faul Tolerance and security. It combines the use of all platform functionalities.

For the designing step the DMTF MOFEditor tool could be used but this time no tools is provided to generate mof files from classes diagrams. This is an important lack.

We continue investigation a CIM meta model in order to add pre and port condition at the designing level.

4. Management Knowledge Distribution

In the CAMELEON platform, we have solved two issues drawn out by management knowledge distribution.

4.1.1. A federated Naming schema

For the CAMELEON framework to function transparently, independently of OMs and OPs locations, it is necessary to define a strong and efficient naming convention.

- When talking about object locations in CAMELEON, we have to consider two levels:
- location of CORBA objects running as part of the CAMELEON infrastructure (OMs and OPs);
- location of the CIM Objects (mapped to Java objects within OMs).
- CIM Objects in the MIB need to be uniquely identified and referenced for all of the CAMELEON distributed system. Their naming is based on two naming models :
- Symbolic name of the Object Manager having the MIB, based on CORBA naming,
- Object path of the CIM object (i.e. namespace + model path), based on CIM naming.

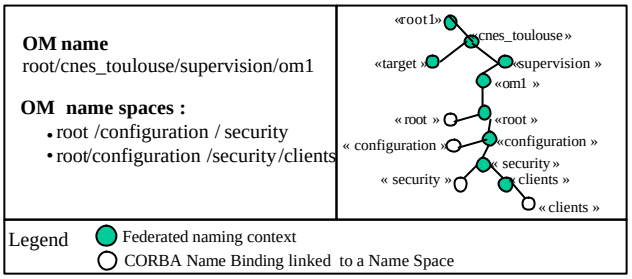

Figure 8: Example of federated symbolic name

By using this convention, any CIM instance can be located anywhere in CAMELEON. To locate the OM responsible for this instance, we look at the first part of the instance identifier and then ask the CORBA naming service to resolve the symbolic name.

There are some solutions to build federated naming services, for instance using CorbaScript [9], but it is always a complex operation. This CAMELEON service proposes to configure and deploy it more easily, by describing the configuration through CIM classes in the schema.

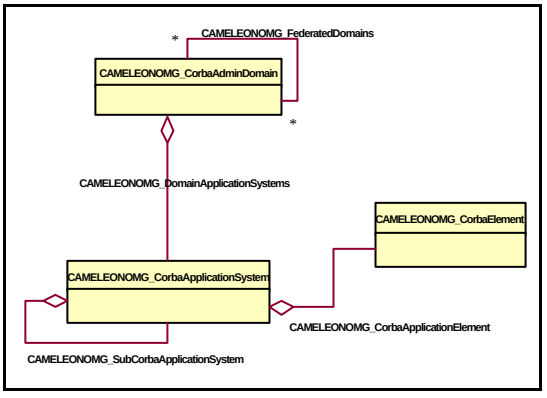

Figure 8: Federated naming classes

For this, we defined amapping between classes and instances of the CIM model and the organisation of federated naming services. Part of the CORBA model used in this mapping is presented in figure 8:

- CAMELEONOMG_CorbaAdminDomain maps to roots of naming services,
- CAMELEONOMG_DomainApplicationSystem maps to context bindings between the root context and any sub-context,
- CAMELEONOMG_CorbaApplicationSystem maps to naming contexts excepted the root one,
- CAMELEONOMG_SubCorbaApplicationSystem maps to bindings between contexts, except the root ones,
- CAMELEONOMG_CorbaApplicationElement maps to object bindings within naming contexts.
- Finally, CAMELEONOMG_FederatedDomains maps to federation contexts and all the bindings to other root of other naming services.

When the CAMELEON Object Manager starts with the Federated Naming Service activated, it reads its MIB and builds the naming service (contexts, sub-contexts and context bindings) and the federation with other naming services (cross-bindings between federation contexts and root contexts for each naming service).

Advantages of naming services federation are numerous: scalability, performances, better fault tolerance, local control…

4.1.2. An external Notification Service-based exchange

Receiving no-solicited event between management entities is mandatory within a management platform.

This requirement concerns events occurred on managed entities represented by objects within a MIB.

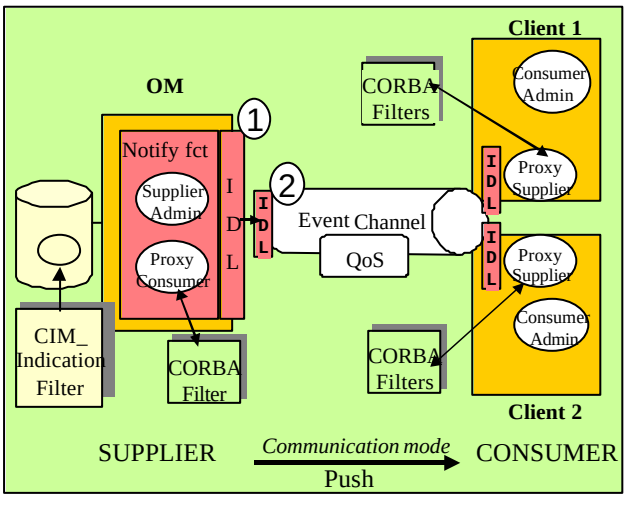

Figure 9: Implementation of Notification functionality

CIMv2.5 models these events as instances of CIM_Indication classes and the subscription mechanism by specifying the following classes: CIM_IndicationFilter, CIM_IndicationHandler and the association CIM_IndicationSubscription.

A supplier/consumer service is required to implement external CIM_Indication exchange between any management entities running on the platform (OM management functions, Visualization application). This requirement is supplied in CORBA by the Notification Service. The CORBA product "ORBacus Notify Service" was chosen because it offers filtering mechanism and QoS parameters [7]. This service allows objects to supply asynchronous events to consumer objects through event channels.

Several choices have to be done concerning the event channel, such as :

- **Event channel configuration** : the parameter `EventReliability` is selected for event persistency and the parameters `ConnectionReliability` and `OrderPolicy` for connection persistency,
- **Event type** : "`Structured Event`" in order to manage filter QoS,
- **Event channel filtering** : at both Supplier and Consumer sides in order to limit event traffic,
- **Communication mode** : push supplier - push.consumer.

Some implementation details are represented in figure 9. Each OM has a Notify function that creates both `Supplier Admin` and `Proxy Consumer` CORBA objects. A client has to create both `Consumer Admin` and `Supplier Proxy` objects to perform Consumer role.

Two IDL interfaces were defined :

- the Notify function interface (an extract is done in figure 10), and
- the NotifyObserver interface (see figure 11).

```
interface Notify_Mgt {
// Types of events that this interface knows
enum EventType {
   INITIALIZATION, // MIB has been initialized
  ACCESS,     // object has been accessed in MIB
  CREATE,     // object has been created in MIB
  UPDATE,     // object has been updated in MIB
   DELETE     // object has been removed from MIB
    };
typedef sequence < EventType >
SeqEventType;
// Types of objects that events can apply to
enum ObservableType {
 MIB,    // Event pertains to the MIB as a whole
 CLASS,  // Event pertains to a StructCIMClass
 INSTANCE,// Event pertains to a StructCIMInstance
 PROPERTY,// Event pertains to a StructCIMProperty
 METHOD  // Event pertains to a StructCIMMethod
 };
// Filter
```

```
//pathFilter : name of class, object path of class,
    // object path of instance or ""
 struct Filter {
  string    pathFilter;
  ObservableType    typeObservable;
  string     nameMethodOrProperty;
  SeqEventType     eventTypes;
  };
 typedef sequence <Filter> SeqFilter;
...
 // Name of Event Channel in Naming
Visibroker
 typedef string NameEventChannel;
//Useful Informations for the creation of Consumer
struct SubscriptionResult{
SubscriptionID uidofSubscription;
//uid of  Subscription
 NameEventChannel eventChannelName;
 };
 typedef sequence <SubscriptionResult>
SeqSubscriptionResult;
SubscriptionResult subscribe(in string
nameClient, in Filter filter); ...}
```

Figure 10: Extrat of Notify function interface

```
public void push_structured_event
(StructuredEvent event)
```

Figure 11: IDL NotifyObserver interface

Finally, Figure 12 summarizes the mapping done between IDL filter/CORBA filter/CIM_IndicationFilter.

IDL Filter	**pathFilter** **typeObservable** **nameMethodOrProperty** **eventTypes**	**pathFilter** ? « Toto » **typeObservable** ? « Instance » **nameMethodOrProperty** ? « » **eventTypes** ? {CREATE,DELETE}
CIM Filter	**Select** . . **Type** **From**	**Select** **Type** *CREATE AND DELETE* **From** *Toto*
CORBA Filter	**Filtered fields :** EventType ObservableType PathFilter NamePropertyOrMethod	**Constraints for each filtered field** (**$EventType** = =' CREATE ') and ($ EventType == 'DELETE ') (**$ObservableType** == ' Instance ') (**$PathFilter** = ='Toto') (**$NamePropertyOrMethod** == ' ')
Note : the symbol '$' is defined in the CORBA filter grammar to indicate a filterable field of the current Structured Event		

Figure 12: IDL filter/CORBA filter/CIM_IndicationFilter. mapping

CIM_Indication classes had to be specialized just in order to override name property by adding the qualifier "key".

5. Conclusion and Issues

The CAMELEON platform proposes a global and distributed approach to manage large and heterogeneous systems. This is made possible thanks to the maturity of object technologies and standards [12].

Some work has been done on CORBA management, and results are promising though tests we made are still applied to a limited number of entities. CAMELEON is already able to offer some services that can hardly be found in current ORB products, especially security and fault tolerance. Moreover, CAMELEON makes using and configuring these services for any CORBA application simpler than usually because of the use of the CIM modelling and instantiation.

Future works will be looking at the advantages of using CAMELEON as a framework for integrating services, especially services addressing CORBA and CORBA objects.

CAMELEON is an attempt to enlarge the WBEM/CIM management approach to more complex systems such as Space systems. We pursue some work in order to get CAMELEON infrastructure used for space constellations management. This is in these complex and large domains to manage that we will see all of the advantages of using CORBA as our communication framework.

The potential web opening is a new way to ensure interoperability and cooperation between management entities.

We follow our investigation on 2 directions:

- End-to-End QoS for wide distributed CORBA environments [13], and
- Web management services.

6. Acknowledgements

We would like to thank all the persons that worked on the CAMELEON project. In addition to the co-authors, they are :

- CNES agency : Florent Maisonneuve;
- Alcatel CIT: Philippe Link, Nicolas Avis, Kristelle Lassagne, Laurent Babarit, Patrick Jimenez;
- Communication & System: Alain Roussel ;
- ATOS ORIGIN, Dominique Benech;
- IRIT laboratory: Pr Yves Raynaud, and
- Members of the national META group

7. References

[1] JSMAN: an open source Java-based Framework for Seamless Integration of equipment, network and service management paradigm.
http://www.loria.fr/equipes/resedas/JSMAN/

[2] GEMINI : A Generic Environment for Management INformation Integration. Anne-Isabelle Rivière. European Doctorate Thesis - Paul Sabatier University - Dec 1997
[3] WBEM Initiative – Distributed Management Task Force (DMTF) http://www.dmtf.org/wbem/index.html
[4] Common Information Model (CIM) Specification – Distributed Management Task Force (DMTF) – Version 2.2 – 14 June 1999
[5] The Common Object Request Broker: Architecture and Specification – Object Management Group (OMG) – Version 2.3.1 – October 1999
[6] "Supervision of the CORBA Environment with CAMELEON: a WBEM/CIM -Based Management Framework", Bénech D., Jocteur-Monrozier F., Rivière A.-I. International Symposium on Distributed Ojects and Applications (DOA' 00). Antwerp, Belgium, September 2000.
[7] ORBacus otification service. www.ooc.com
[8] Solaris 8.0 Operating Environment WBEM – Sun – http://www.sun.com/solaris/wbem/
[9] Model Driven Architecture. Ormsc/2001-07-01. July 9, 2001.
[10] OFTA, www.ofta.net
[11] Towards a CIM Schema for RunTime Application Management. A.Keller, H.Kreger and K.Schopmeyer. DSOM'2001 workshop, 15-17 october, 2001 – Nancy, France.
[12] " Les technologies orientées objet au coeur des systèmes de gestion : modélisation, organisation et répartition " Jocteur-Monrozier F., Benech D., Sibilla M. and A.-I. Rivière. Electronic Journal on Networks and Distributed Processing , V. 11 , p. 211-227. Mars 2001. Accès: http://rerir.univ-pau.fr/rerir2.html
[13] "CORBA QoS Management with CIM/WBEM ", Barros A., Sibilla M., Jocteur-Monrozier F. Workshop on Real-Time and Embedded Distributed Object Computing , June 4-7, 2001. Herndon, VA USA
[14]" Global System Management with Standard Object Oriented Technologies" Rivière A.-I, Sibilla M. and Jocteur-Monrozier F., DAta Systems In Aerospace conference (DASIA), (Canada), May , 2000, http://www.eurospace.org/
[15] www.irit.fr/SUMO