SYSOFT SA
116, Ave. des Champs Elysées
75008 Paris, France
Tel. (33) (1) 47 20 65 34
Fax. (33) (1) 47 20 44 52

Doc : ARTRDEN2.DOC
Version : A1.01
Author : Ion CARTIANT
Compuserve : 100060,2404
Monday 24 October 1994, 11:42:23

MARCO
- Object architecture method -

MARCO is an Object Architecture method. It suggests a new way of **organizing** the different models concerning systems and objects during the architectural, analysis and design phases. Its purpose is to render a global view about the information systems and its components. This has to contribute (at least in principle) to the success of applications portability and /or reuse of components.

The fundamentals of MARCO are :

- **A parallel between systems and objects.** Each system may be viewed as an object. Each object may be viewed as a system. At a certain analysis level the internal structure of an object may be ignored. Some other times an object may be considered as a complex structure of other objects.
- **Service.** Each system (each object) asks for services or renders services to other objects

A functional domain = A system = An object

The development of a system analyze goes along three axes, each concerning a view of the system portability :

- More details about the process - semantic portability
- More details about the working environment - application portability
- More details about the user presentation - presentation portability

OBJETS AND SYSTEMS

An object is an entity characterized by the following features :

• A frontier	The object is clearly separated from its environment. This frontier is impenetrable, with the exemption of « service points » areas
• Services	The purpose of the object is to execute services. The possible interactions are : • Services are provided by the object at the request of an other object. The object responds with a rendered services, one of a set of multiple possibilities • The object may ask for services at an external object. The rendered service may belong to a set of different possibilities. This notion of service generalizes that of message
• Points of services	The object is capable to interact with the exterior through points of services, a communication gate clustering the related services. A service point may be of server or client type
• A memory	The object is able to preserve the « souvenir » of past activities in attributes. An attribute preserves one of various states.
• A structure	An object may contain other embedded objects

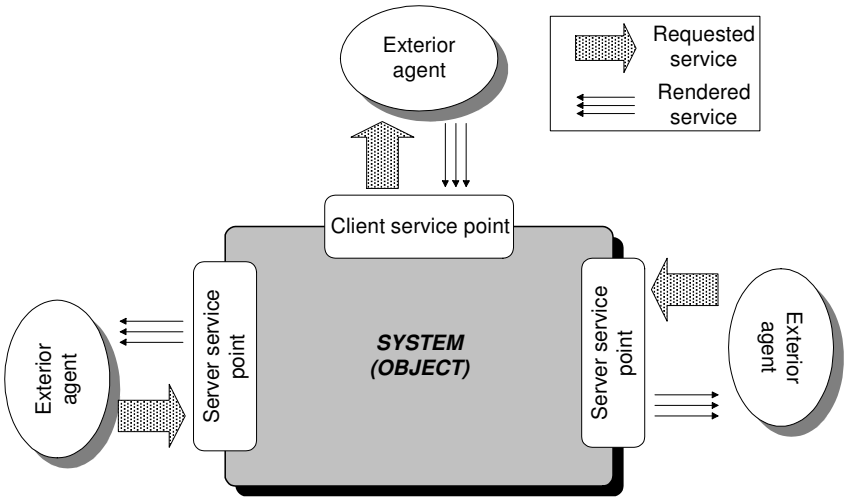

Figure 1. Objet

The object is accessible exclusively through service points. *One can imagine that future (object) operating systems could exploit directly this notion, by providing each object with a protective cover forbidding any access through other areas then the service points.*

An object is defined by various models :

Model	Purpose
• Services model	⇒ Presents services and service points
• Service scenario model	⇒ Presents how the the services are done, viewed from outside
• Inheritance model	⇒ Presents the inheritance relationship of the object
• Dynamic model	⇒ Presents the state change due to external events
• Structure model	⇒ Presents the object structure
• Internal scenario model	⇒ Presents the internals details of the scenario execution

An object presents at the beginning an **external architecture**. It is the vision that the world has about the object. The essential part of this architecture is the presentation of the object services. With this vision only one can explore and decide upon the use or reuse of the object. Then an object, if it is a complex one (and if it is interesting to see how it is built), presents an **internal architecture**. The embedded objects are firstly analyzed only from their external architecture point of view. In our view this completes a « level analysis » .

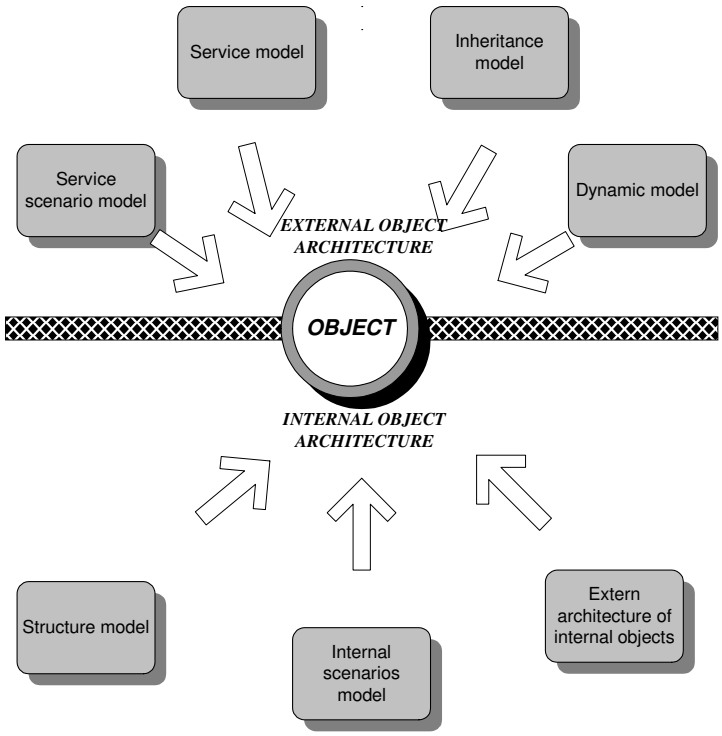

Figure 2 . Object models

EXTERNAL SYSTEM (OBJECT) ARCHITECTURE

An object is first perceived by the views rendered by its external architecture. The models contained in the external architecture are :

- The service model
- The service scenario model
- The dynamic model
- The inheritance model

♦ The service model

The service model is the the most important one for defining an object in MARCO. An object may receive a **service request** from an outside object. To this requested service it responds with a **rendered service**. The rendered service belong to a set of possible responses. The object may also request a service from an external object and getting in return the rendered service returned by the called object.

The requested and rendered services are clustered in **service points**, the communication gate of the object. The service points may be of **client** type (requests sent only) or **server** type (requests received only). The coupling between two objects implies the coupling between the respective client / server service points.

This model defines the service points, the services, the attributes and the status of the object as a whole. A service request does not imply always a rendered service *(for exemple, a service request may indicate an internal state change, without any returned service).* When a rendered service may belong to a set of possibilities, the reason to choose one instead of another is not explored at this point, but in the service scenario model.

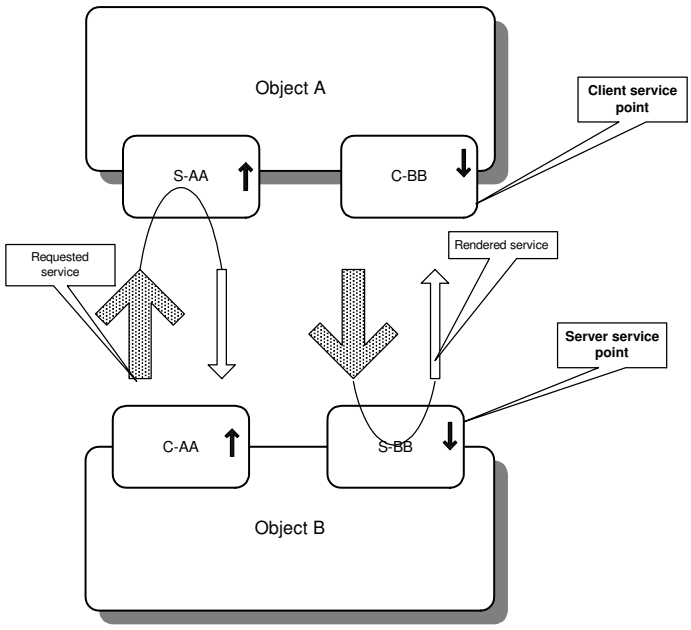

Figure 3. Service points

♦ **Service scenario model**

A service scenario depicts the flow of the object's activities. A scenario is triggered by a service requests and it ends with the rendered service. The initiator of a scenario may be an external object or the studied object (when it issues service requests to exterior objects). In each case a scenario follows the pattern of the service model :

- **scenario without rendered service**
- **scenario with a rendered service.** The choice of the rendered services depends on internal states, on services rendered by request addressed to external objects or of internal processes

♦ **Dynamic model**

An object preserves the memory of its past actions. This memory stores a state of the object, one between various alternatives. The state analysis is part of the behavior analysis of the object. Two techniques may be used to represent this model : state diagram and state machines. The transitions between the states are triggered by events, which are inbound service requests or inbound services rendered to previous outbound service requests.

The service scenario model and the dynamic model describe the behavior of the object.

♦ **Inheritance model**

This model describes the object origins. The actual object is derived from some ancestor. From the outside, the inheritance means adding new service points with new service requests and rendered services for the new object. From an internal point of view, the inheritance means new internal objects with the respective service points.

INTERNAL SYSTEM (OBJECT) ARCHITECTURE

The model contained in the internal architecture are :

- Structure model
- Internal scenario model
- External architecture of each system component

♦ **Structure model**

This model studies the internal object structure. It identifies the component objects. The internal structure may be constituted by a bunch of objects. Depending on the size of the object, the internal objects may be sub-systems, functional domains, elementary objects. The object structure exhibits three objects categories :

- **Exterior (apparent) objects.** This objects gather the external vision of the object (the service points). They are also provided with internal service points, used to interconnect themselves internally
- **Internal objects**, related to the previous exterior object, but invisible from the exterior.
- **A master object.** This object gives a unique sense to the set of internal and exterior objects.

This organization corresponds to the conception that an object is more than its component parts. A special internal object gives the exclusive flavor of the system and discerns it from the trivial sum of internal and exterior objects

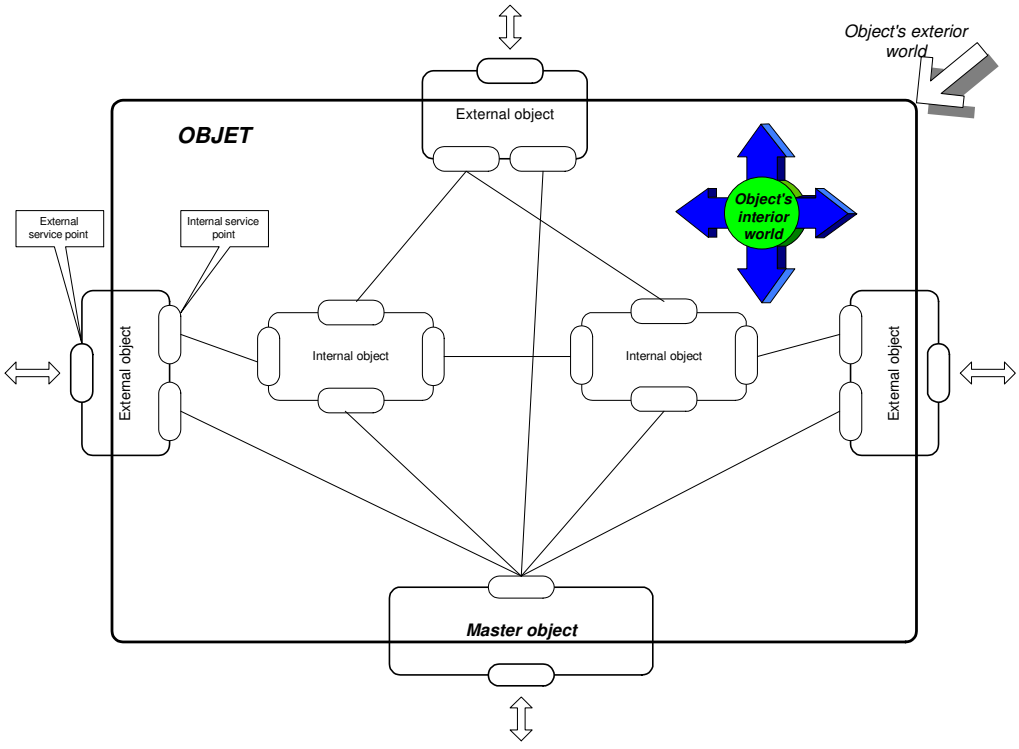

Figure 4.

MARCO recommends that the structure of each object (system) should not expand over ten component objects. This is a choice dictated by the will to manage the complexity of the projects by creating structures easy to understand and related to the application domain :

- Each component object may be built at Its turn by maximum ten component objects, and so on
- The objects study may me delegated easily to various team groups through natural separation lines. The overall system remains coherent.
- This policy leads naturally to the isolation of objects to be reused in other projects

Axiom : an object has maximum 10 component objects

The ten objects limit is a modular criteria (an object collection is an object in this sense). If respected, this rule allows for complexity management by maintaining the comprehension of the system. It makes easy the localization of future changes. The ten object limit is derived from the famous « seven plus / minus two » figure representing a result of the research concerning the capacity of human brain to grasp simultaneously various subjects.

This decomposition rule is applied also to the objects that build the starting object (system), in iterations. At the end we have non-decomposable objects. This limit corresponds to the details judged necessary to understand and to build the application. When going from architecture to analyze and then to design, some non-decomposable objects at this stage may be split in the more detailed stage, following the same procedure : external architecture - internal architecture.

♦ **Internal scenario model**

The internal scenario model correspond to the action flow of the component objects to fulfill a service request. In comparison with the service scenario model, the present model shows the interactions between the service points and the services of the component objects. The ten object limit allows for clarity in this presentation.

♦ **External architecture of component objects**

The component objects are presented through their external architecture, mainly the service model. The internal service model is build by taking into account the service points of the component objects

ARCHITECTURE PROCEEDINGS WITH MARCO

MARCO may be used to define a system and then to refine Its analysis or to synthesize a system starting with some component blocks or both. If we define various steps in the activity of defining and implementing a system, we can see that in MARCO the phases like architecture, analysis, design are differentiated only by the detail degree of the objects. The procedure is the same : external architecture / internal architecture, with the associated model, a seamless procedure.

Generic system analysis process

MARCO suggests the application of the same scheme at each step in the analysis process :

• 1st pass (1st level)	⇒ Define the system (object) external architecture (level 1 object)
• 2nd Pass (2nd level)	⇒ Define the internal architecture of the level 1 object Define the external architecture of the level 2 objects, components of the level 1 object
• ...	...
• « n th » pass (« n th » level)	⇒ Define the internal architecture of the « n th » level objets Define the external architecture of the « n+1 » level objects, components of the « n th » level

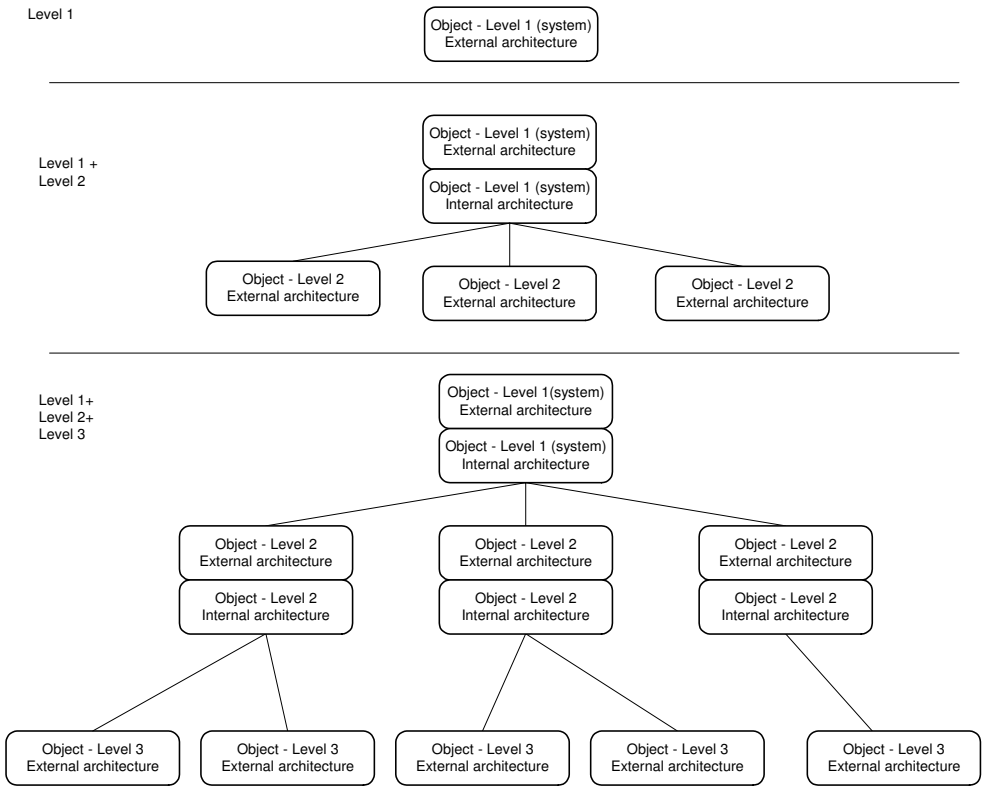

Figure 5.

Generic system synthesis process

The system architecture is elaborated starting with existing objects. Each component is provided with an external architecture. If the component is complex (or interesting) it may have also an internal architecture. The synthesis starts by taking into account the external architecture of the objects constituting the building blocks of the future system.

This objects are provided with server and client service points. The connection of the objects is feasible if this service points are compatible. When various elementary objects are connected in this way, the master object will have to be defined, to obtain in this way the final definition of a complex object (a subsystem). The service points of the subsystem are mainly the service points that remained unmated during the construction phase. This subsystem may be stored for future reuse in other projects. At Its turn it may become a component in a higher level architecture, where it is taken into account only through its external architecture.

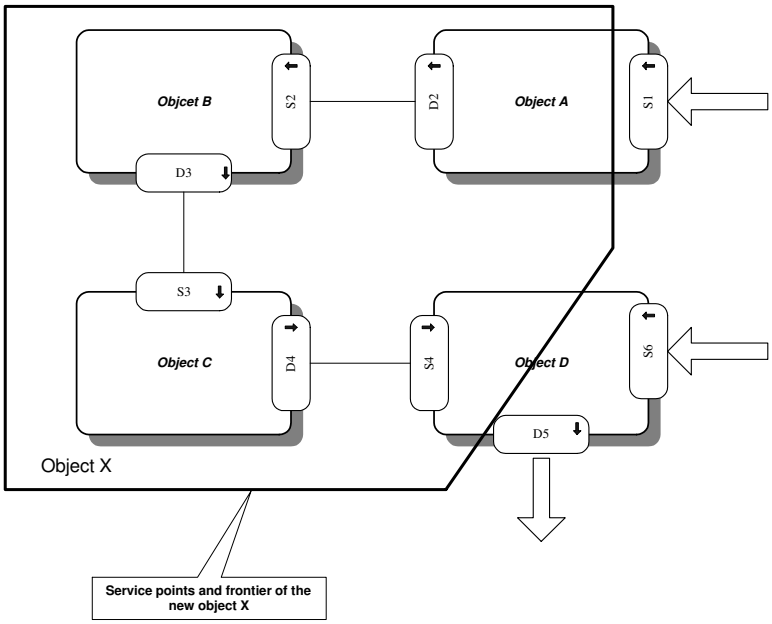

Figure 6. Object construction

Organizing for error processing

The normality of a system is its capacity to process services in a predictive way. An abnormal situation arrives when :

- **The requested service is abnormal**. The service request is off limits
- **The rendered service is abnormal**. To a service request, the rendered services is incongruous (outside the expected norms)
- **The rendered service is absent**. To a service request there is no rendered services after a reasonable waiting time

An error manager must be present to intervene in such cases. This manager is associated to the master object of the system and it processes the error involving the components objects. The error manager is a service point of the master object. It may use the current administrative services. It interacts with the internal objects and with the exterior objects (agents).

A system has to provide for errors processing :

- **Detect the errors.** This situation arrives when a component objects detects an internal abnormality. It will request a service of the error manager to inform it.
- **Signal the abnormality.** If a rendered service is off limits, the object that receives it has to inform the related error manager through a service request
- **Step in** to reestablish the normal use case. If it is not possible, the error manager will call upon the higher error manager

MARCO organizes the error management by identifying an error manager for each system (complex object).

MARCO AND OTHER OO ANALYSIS METHODS

MARCO is essentially an organizing method using different models upon the objects and the systems, aimed at reducing the « details chaos ». The MARCO models are

already used in other methods, but we combine them in a new kind of organization of system analysis and synthesis

- The same approach is used at various analysis levels, from architecture to construction, an elegant way to manage the complexity
- The distinction between external and internal architecture provides for natural integration of subsystems (object) in more complex subsystems (objects). It allows to take into account objects only following their external, visible presentation
- The service points make easy the classification and the identification of the objects for future reuse. The constraints are easily recognized : compatibility between associated client and service points.
- The limitation of component object number helps the understanding and avoids a too quickly atomization of the analyses *(there is no forest, only a group of ten trees, with a leader !)*
- The proposed models are easy to understand. One may go up naturally, to find the global scheme of the system, or go down to see how it is done. The object system architecture is a powerful communication mean to improve the dialogue between the user, the implementor and the manager

Conclusions

In this document we have presented the general lines of our MARCO method. MARCO was developed in the last two years, after becoming aware of the limitations of current methods. Our customers find it quite simply and clear. It is easy to be understood by all levels, managers, implementors and clients. MARCO is presently used in banking applications to help the definition of the next systems and to propose a seamless framework for architecture, analysis and implementation.

□

Ion A. CARTIANT : CompuServe : 100060,2404

- Computer engineer - 1971 - Polytechnics Institute - Timisoara - Romania
- Consulting engineer in communications architectures for banking institutions since 1984
- Software author of various communications packages - about 1000 licenses in Europe
- Author of an OO banking communication server (C++) demonstrated at OOPSLA'92 - Vancouver.
- Presently with SYSOFT SA (owner) - Paris, France - a consulting company

This document is derived from a quite extensive study conducted in the last two years by I. Cartiant. MARCO may be used as a method supporting process innovation.