

# A Diagnosis Service for CORBA-Based Applications

Oum-El-Kheir Aktouf-Benkahla and Abdelaziz Gacemi  
LCIS-INPG, 50, rue B. de Laffemas, B.P. 54, 26902 Valence Cedex 9, France  
Oum-El-Kheir.Aktouf@esisar.inpg.fr

## I. INTRODUCTION

Middle-wares are more and more used for dealing with heterogeneity in distributed systems. Due to their large utilization, they should be fault-tolerant as faults in some systems may lead to important economic losses. Recent works have proposed fault-tolerant solutions in middle-wares. Most of the proposed approaches are based on component replication and fault masking. The goal of our work is to investigate fault diagnosis that allows precise location of faulty components. This approach has several benefits. It may complete the fault-masking approaches by allowing them to locate and then fix the faulty components, thus alleviating increasing number of faults that may corrupt their correct functioning. Furthermore, by allowing the precise diagnosis of faulty components, applications are debugged more efficiently. Finally, and for non-critical applications, diagnosis may replace replication-based fault-tolerance allowing a better usage of system resources.

## II. FAULT MASKING AND FAULT DIAGNOSIS

Most proposed fault-tolerance solutions for distributed applications are based on fault masking. Redundancy is widely used to mask the effect of faulty components on the application global functioning without locating these faulty components. There are three main approaches for fault masking depending on the implementation level of fault-tolerance mechanisms. In the first approach, fault-tolerance is implemented into the operating system such as in the DELTA-4 system [1] which defines a generic architecture for fault-tolerant distributed operating systems. The second approach consists of solutions that introduce fault-tolerance on top of the operating system, especially at the Object Request Broker (ORB) level. Some representative examples of this approach are the interception methods which intercept significant events generated by the ORB [2] and the integration methods which consist of modifying the ORB to make it fault-tolerant [3]. The third approach consists of implementing fault-tolerance mechanisms at the application level. Thus, fault-tolerance may be provided as services like in [4]. In all the preceding approaches, researchers have focused on replicating the system or the application components to allow fault masking. Thus, results are guaranteed to be correct though some faults may corrupt the functioning of some system or application components. But such solutions are very costly and resource consuming. This is especially true for non critical applications which may afford some time loss for less costly fault-tolerance.

One cost-effective fault-tolerance method is system diagnosis, which is based on inter-component tests to determine the fault state of the system. Several diagnosis strategies have been proposed for distributed systems. These diagnosis strategies ensure a deep knowledge of the state of system components and communication links between them. Very good reviews have presented the main models and strategies proposed for system diagnosis [5] [6]. The goal of our work is to introduce concepts of system diagnosis into CORBA-based middle-wares to allow precise location of faulty components and avoid costly redundancy approaches. The basic idea is to provide application objects with testing and diagnosis capabilities to let them monitor themselves. Section 3 below presents Object Diagnosis Service (ODS) which is the proposed diagnosis service for CORBA objects.

## III. OBJECT DIAGNOSIS SERVICE (ODS)

### A. Basic Requirements

In a diagnosis process, three aspects are very important. The first one concerns the execution of inter-component tests and the transmission of test results. The second one is the diagnosis process itself which consists of analysing inter-component test results for determining the system global fault state. Finally, the last aspect is related to the transmission of the system state to an external observer to allow correcting and/or repair actions to be conducted. The main difficulty in all these steps is to avoid inconsistent behaviour of faulty components that may corrupt the overall diagnosis process. In the proposed diagnosis service, these three aspects are taken into account. The proposed diagnosis service provides mechanisms for allowing a group of distributed objects to execute mutual tests. Fault-free objects of the object group will then be able to determine the fault state of the group and an external object, called the observer will be able to reliably get the fault state. While designing this service, CORBA requirements have been kept in mind. Thus, ODS is simple, generic and provides interfaces independently from the implementations.

### B. Diagnosis Group

A diagnosis group is defined as a group of objects that use the same diagnosis model and the same diagnosis algorithm. New objects may join the diagnosis group while others may leave it. Every object in a group is notified each time a new object joins the group or a member object leaves the group.

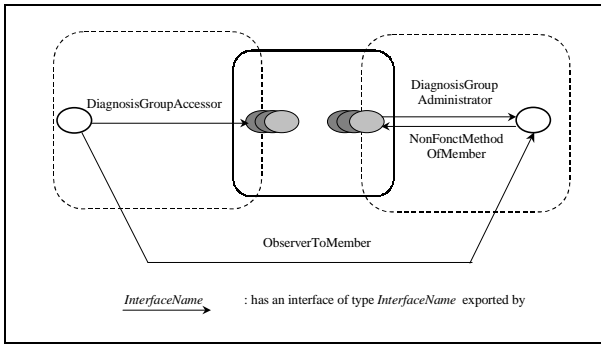


Figure 1. Represents ODS architecture.

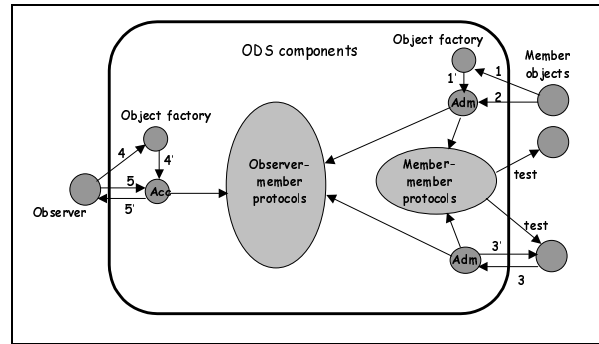


Figure 2. Shows object interactions in ODS.

### C. ODS Architecture

ODS provides a diagnosis service through a set of interfaces that should be inherited by the application using ODS service. Three classes of interfaces may be distinguished (see Fig. 1).

#### 1) Interface of the observer side

This interface is provided to an external object that aims to know the fault state of the diagnosis group. To allow a reliable transmission of the group state to an external observer, algorithms such as those proposed by [7] and [8] should be used. The DiagnosisGroupAccessor interface allows the external observer to access to the object group as a unique diagnosis entity.

#### 2) Interface of the member object side

This interface allows an object to join a diagnosis group or to leave it, and to launch the diagnosis process according to a given diagnosis algorithm. For this purpose, a *diagnose()* method is provided and should be overloaded to implement different diagnosis algorithms.

#### 3) Interface of the service object side

The interface associated to a service object side is inherited by the members of a diagnosis group. This interface provides member objects of a diagnosis group with testing and diagnosis capabilities. A *test()* method is provided. This method may be redefined by the user according to the desired test precision.

### D. Object interactions

Fig. 2 shows interactions involving observer objects, member objects and service objects in ODS. Object *Acc* of type DiagnosisGroupAccessor is the observer. Object *Adm* of type Diagnosisgroup-Administrator interacts directly with member objects. It is located on the same site as the member objects. To join a diagnosis group, an object must create an instance of type DiagnosisGroupAdministrator using Diagnosis-GroupAdministratorFactory (1,1').

This one returns a reference to the created object *Adm*. Using this reference, the server invokes the *joinDiagnosisGroup()* method (2). A group change is then notified to all member objects.

To perform a group diagnosis, a member object invokes the *diagnose()* method of the *Adm* object. *Adm* object gathers the diagnosis information (3, 3'). An observer object wishing to know the state of the group first creates an object *Acc* of type DiagnosisGroupAccessor using DiagnosisGroupAccessor-Factory (4, 4'). It then invokes *getStateDiagnosisGroup()* method of object *Acc* (5). Using protocols like the ones developed in [7] and [8], diagnosis information is transmitted reliably to the observer object (5').

## IV. CONCLUSION

Ongoing work consists of implementing the proposed service and compare its performances to classical fault-tolerance services ones. More details of ODS architecture and obtained results will be related in a future paper.

## V. References

- [1] M. CheÂreÂaque, D. Powell, P. Reynier, J.-L. Richier, and J. Voiron, "Active Replication in Delta-4", *Proc. 22nd Ann. IEEE Int'l Symp. Fault-Tolerant Computing*, 1992.
- [2] L. E. Moser and P. M. Melliar-Smith, "The Interception Approach to Reliable Distributed CORBA Objects", in *3rd USENIX Conference on Object-Oriented Technologies and Systems*, 1997.
- [3] S. Maffei and D. C. Schmidt, "Constructing Reliable Distributed Communication Systems with CORBA", *IEEE Communications Magazine*, vol. 14(2), 1997.
- [4] P. Felber, B. Garbinato and R. Guerraoui, "The Design of a CORBA Group Communication Service", *IEEE Symposium on Reliable Distributed Systems*, 1996.
- [5] M. Barborak, M. Malek and A. Dabhura, "The Consensus Problem in Fault-Tolerant Computing", *ACM Computing Surveys*, vol. 25(2), 1993.
- [6] S. Lee S. and K. G. Shin, "Probabilistic Diagnosis of Multiprocessor Systems", *ACM Computing Surveys*, vol. 26(1), 1994.
- [7] S. Koppolu and A. Chatterjee, "Hierarchical Diagnosis of Identical Units in a System", *IEEE Trans. Comput.*, Vol. 50 (2), 2001.
- [8] S. E. Kreuzer and S. L. Hakimi, "Distributed Diagnosis and the System User", *IEEE Trans. Comput.*, Vol. 37(1), 1988.