










LENGUAJES Y  
CIENCIAS DE LA  
COMPUTACIÓN  
UNIVERSIDAD DE MÁLAGA

# Maintenance of Reliable Distributed Applications with Open Source Middleware: Fifteen years later

Manuel Díaz, [Daniel Garrido](#)

-  Introduction
-  Background
-  CORBA Overview
-  Maintenance Experiences
-  Patching the middleware
-  Improving performance
-  Conclusions

- ▣ Maintenance of critical applications
  - Long life cycle (several years)
  - An important choice
    - Hardware
    - Software
  - No real data available for new technologies
    - Technologies are constantly evolving
- ▣ At the beginning of the previous decade
  - Middleware as a promising technology
    - CORBA, Java-RMI, DCOM
  - CORBA to solve all heterogeneity problems
    - Different languages, operating systems, platforms
    - RT-CORBA, FT-CORBA and Minimum CORBA

## Advantages

- Some complexity is hidden by the middleware
  - Low level details
  - Network complexity
- Better interoperability
- Focus on functionality

## Disadvantages

- The middleware has the “control” of the application
- We depend on the middleware and its updates
- How can this affect the maintenance? (specially how it affects **software reliability**)
  - Changes in operating system, languages, patches, ...

## Current situation

### > CORBA is not as popular as expected

#### ↗ Several reasons

- ↗ Internet (Web services, REST, ...)
- ↗ New languages and platforms (e.g. C#, .NET)
- ↗ Not widely accepted by the industry and users (learning curve)
  - ↗ A niche in several sectors: telecommunications, defense, simulation, ...

## But, some projects have already started...

### > And they are being exploited















#### ↗ 24/7

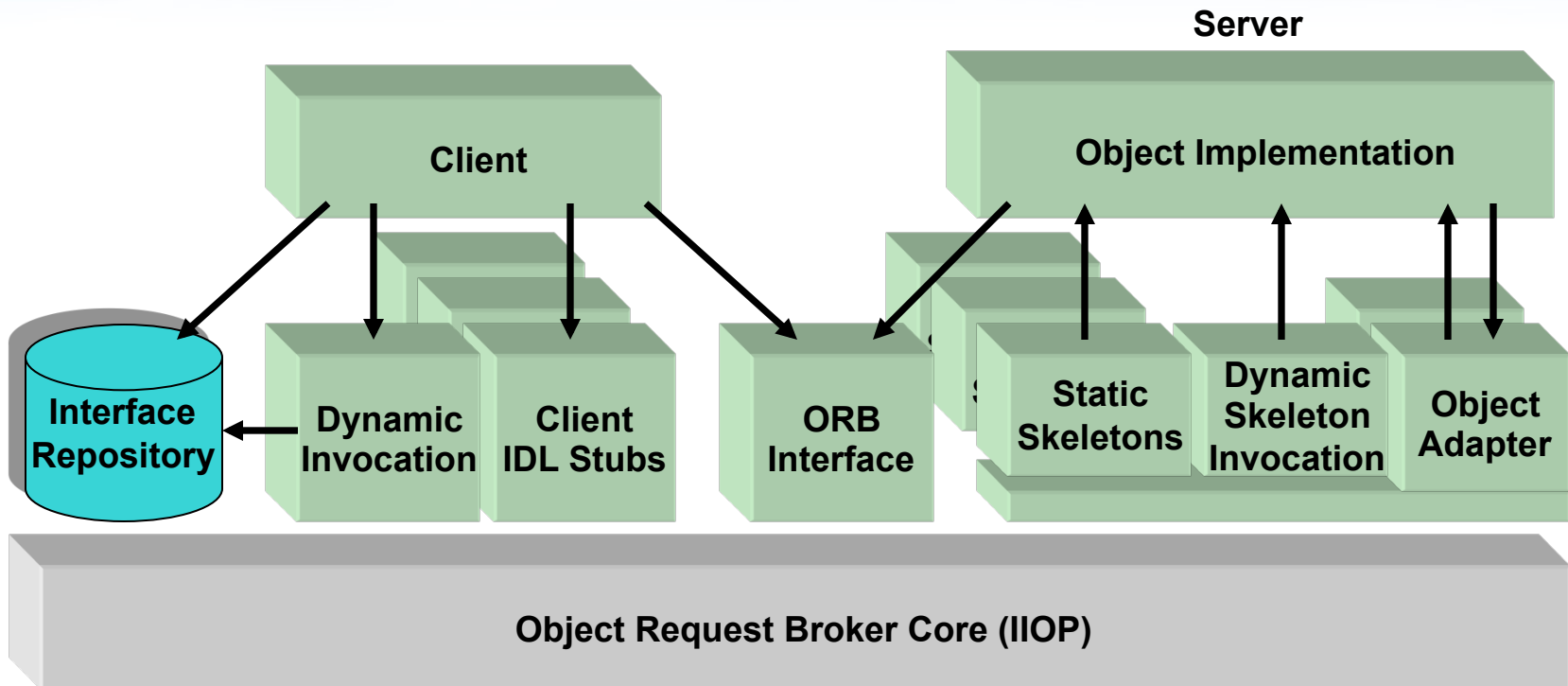
### > Two possibilities

#### ↗ To change technologies

- ↗ But to change is not always a possibility

#### ↗ Maintenance activities

-  Development of applications and communications
  -  UML, software components and CORBA
-  Joint projects with several companies
-  UM-RTCOM was presented
  -  RT-CORBA based
  -  Component model
  -  Real-time support
  -  Higher abstraction level
  -  Some experiences
    -  Distributed simulators with soft real-time constraints
    -  SMEPP European Project
-  Two principally used CORBA implementations
  -  TAO (C++)
  -  JacORB (Java)

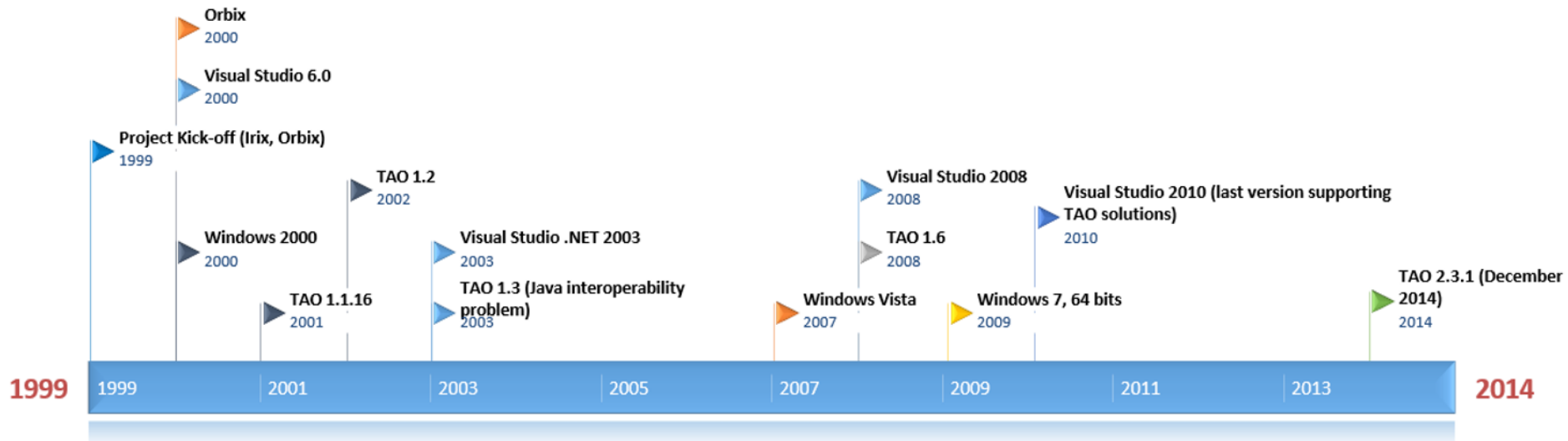


- ▣ Last CORBA version
  - ▣ 3.3, November 2012
- ▣ Many CORBA implementations are currently active (free and commercial)
  - ▣ TAO
  - ▣ JacORB
  - ▣ Orbix, Orbacus, ...
  - ▣ Java SDK implementation
- ▣ New languages have been added to CORBA
  - ▣ Python, Ruby

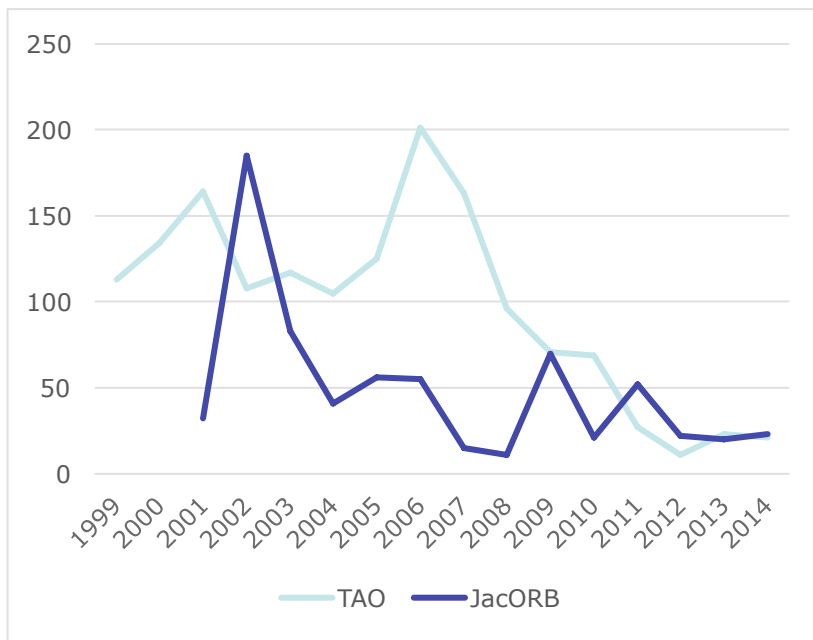


## Main milestones:

- Windows Vista
- C#/.NET
- 64 bits



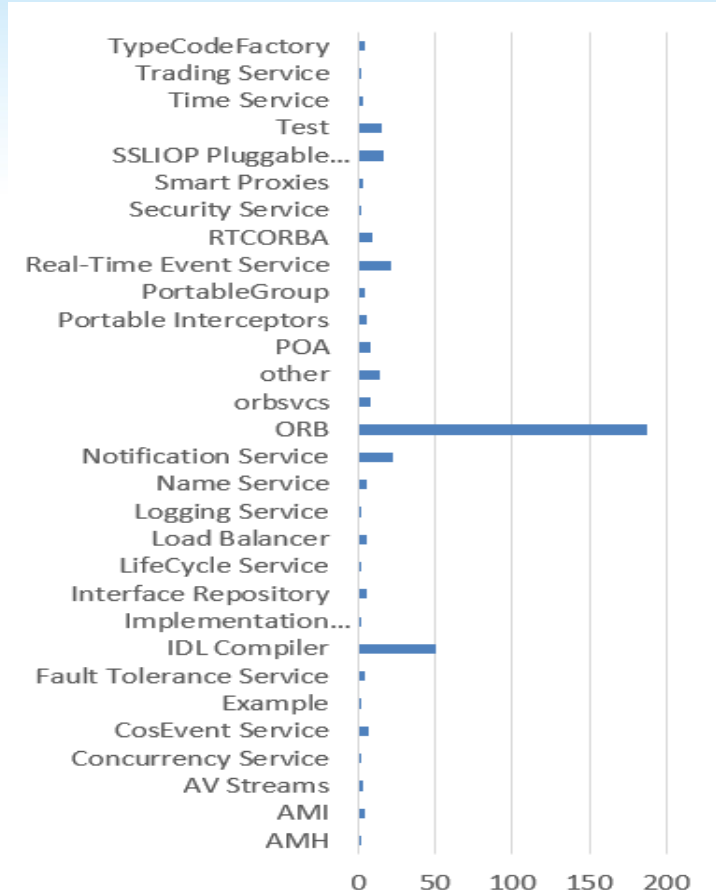
# Maintenance Experiences



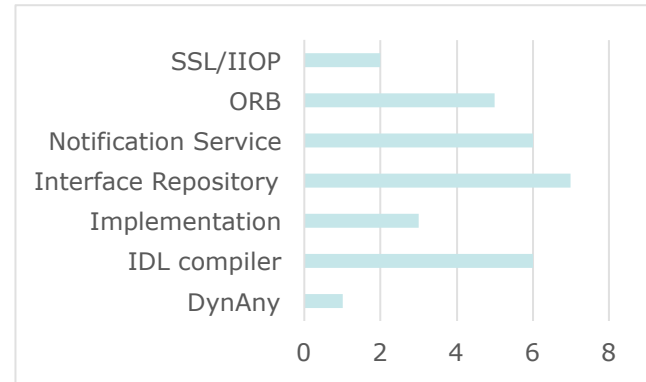
Number of bugs fixed 1999-2014

- ▶ Decreasing activity over the last few years
- ▶ A significant number of bugs remain
- ▶ Latest TAO version (2.3.2)
  - ▶ May 2015
- ▶ Latest JacORB version (3.6.1)
  - ▶ May 2015

# Maintenance Experiences



TAO active bugs  
(end 2014)



JacORB active bugs  
(end 2014)

TAO and JacORB philosophy:

- Bugzilla: bug tracking system
- Users can contribute with solutions and improvements

Alternatively : commercial support (OCI, Remedy IT)

But, when a change is required or a fault is detected...

	<i>Files</i>	<i>Lines</i>	<i>Statements</i>	<i>Class Defs</i>
<i>ACE+TAO</i>	22,915	1,859,251	616,530	8,078
<i>JacORB</i>	2,138	300,129	115,547	2,532

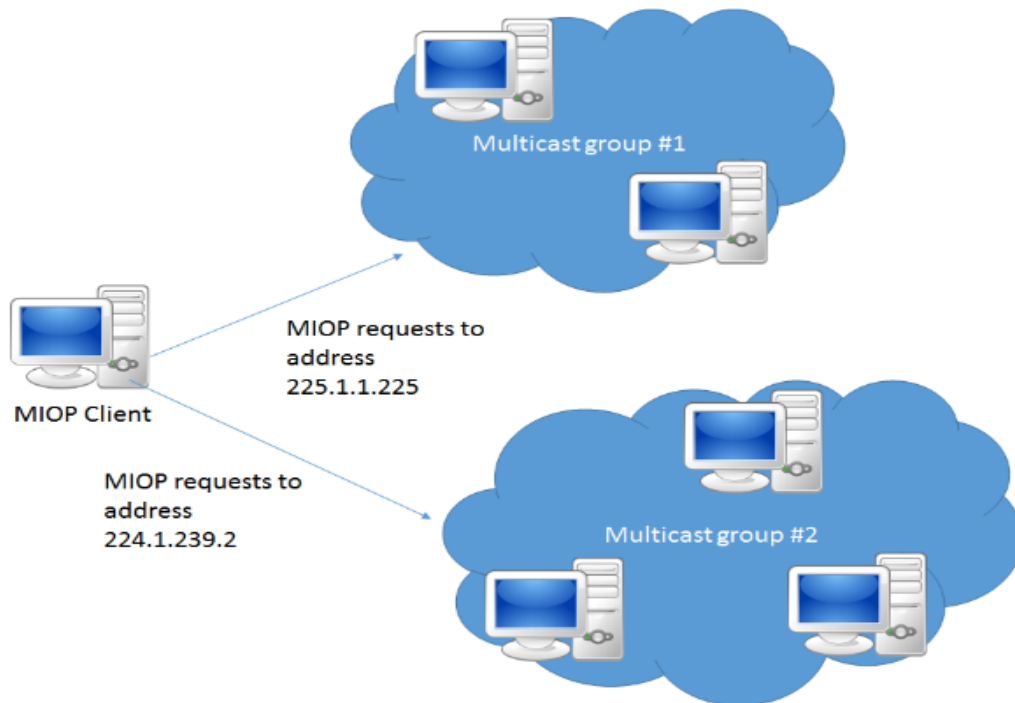
ACE+TAO 6.3.1 and JacORB metrics

- ▣ What about testing?
  - ▣ Automatic testing is supported
    - ▣ More than 1,000 tests are included with ACE+TAO
    - ▣ Different categories
      - ▣ ACE
      - ▣ TAO
      - ▣ ORB services
    - ▣ But this is not enough in critical systems

- ▣ Improvements over time
  - ▣ Service orientation is the new trend
  - ▣ Heterogeneity is a fact
  - ▣ Communication very well encapsulated
  - ▣ Dynamism
  - ▣ Performance
  - ▣ Reliability
  - ▣ Scalability
- ▣ Different kind of changes
  - ▣ Patching the middleware
  - ▣ Improving performance
- ▣ TAO is very extensible and configurable
  - ▣ Several design patterns
  - ▣ Many configuration options

# Maintenance Experiences

- ▣ Improvement performance (multicast)
  - ▣ MIOP/UIPMC protocols



dynamic UIPMC\_Factory  
 Service\_Object \*  
 TAO\_PortableGroup:\_make  
 \_TAO\_UIPMC\_Protocol\_Fa  
 ctory()

"-ORBListenOnAll 0

-ORBListenerInterfaces  
 224.1.239.2=192.168.20.13  
 5"

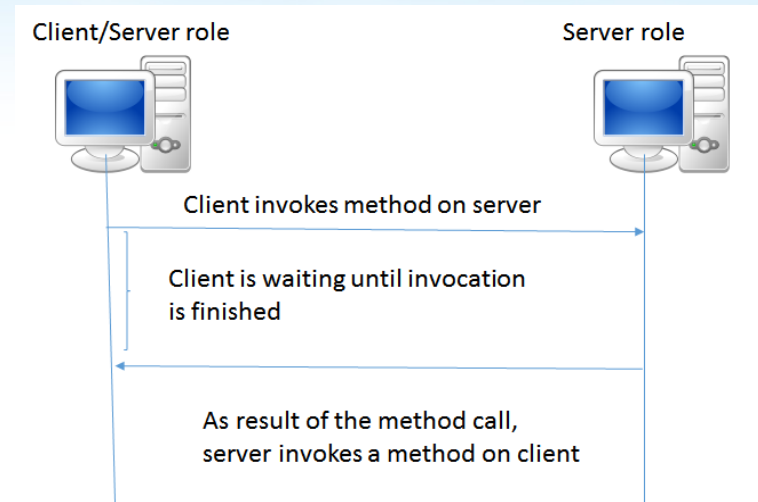
## Improvement scalability (concurrency control)

### On the server side:

- ORB controlled model: requests are attended to in the order specified by the ORB.
- Single thread model.

### On the client side:



- Leader-follower: while waiting, client threads can be reused to process other requests.
- Reactive: thread provided by the TAO Reactor
- Blocking: the client is blocked until the connection finishes.



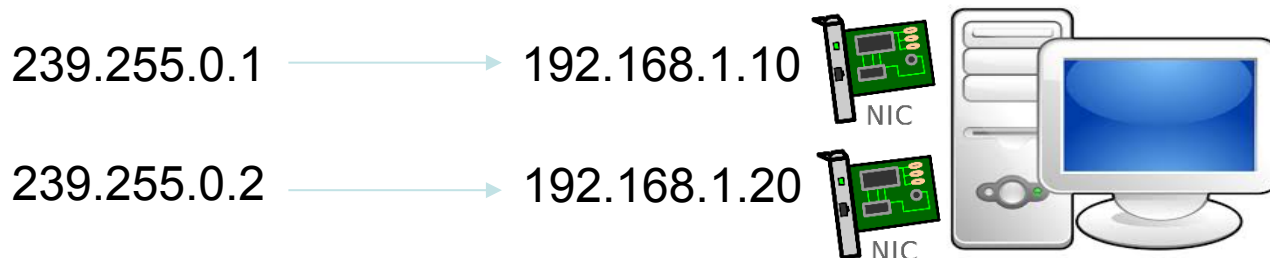
Deadlock  
problems with  
nested upcalls



# Patching the middleware

- 
 Customer request: *“When using IPv6 multicast addresses in a PC equipped with multiple network cards, TAO ignores user preferences and it always uses the first network interface. In addition, we have a strange message”*
- 
 TAO configuration says: user can select in which network interface requests are attended.
 

*“for a machine with two network cards identified by the ip addresses 192.168.1.10 and 192.168.1.20, you can use the single directive -ORBListenerInterfaces 239.255.\*=\*10,224.255.\*=\*20...”*



# Patching the middleware

- ▶ How the problem was detected...
  - ▶ TAO logs showed user preferences, but these logs were false!
  - ▶ Netstat -g shows which network interface was really used
- ▶ Where we should look for... several candidates
  - ▶ ACE sockets
  - ▶ TAO protocols
  - ▶ Configuration
  - ▶ Experiences with TAO helps a little, but a slow process

- ▶ What we found...
  - ▶ A “strange” error message related to IPv6 addresses
    - ↗ *"ACE\_INET\_Addr::get\_ip\_address: address is a IPv6 address not IPv4"*
    - ↗ Reason: call to *addr.get\_ip\_address ()*, method only available for IPv4
    - ↗ We detected unused code. In fact, two operations can be removed from ACE+TAO
      - ↗ *ACE\_UINT32 uint\_ip\_addr (void) const;*
      - ↗ *void uint\_ip\_addr (ACE\_UINT32 ip\_addr);*
  - ▶ Second step: when this code was removed, we obtained a “core”. Reason:
    - ↗ Buffer sizes when using IPv6 string addresses
      - ↗ A constant (MAX\_ADDR\_LENGTH) was defined with size 32, which is not valid for IPv6 address. An example:
      - ↗ 2001:0db8:85a3:0042:1000:8a2e:192.168.158.190

## What else?

- When these problems were solved, the user configuration continued to be ignored
- Reason: TAO can use the following configuration syntax:  

```
./server -ORBId ORB_LAN_1 -  
ORBAllowZIOPNoServerPolicies 1 -ORBListenEndpoints  
iiop://[2001:db8:0:f101::1] -ORBPreferredInterfaces  
*=eth2 -ORBEnforcePreferredInterfaces 1 -ORBDebugLevel  
10
```
- Network interface is selected using "*=interface\_name*"
- Finally, we found...

```
int ACE_SOCK_Dgram::make_multicast_ifaddr6 (ipv6_mreq
*ret_mreq, const ACE_INET_Addr &mcast_addr,
      const ACE_TCHAR *net_if)
{
...
  Imreq.ipv6mr_interface =
  ACE_OS::if_nametoindex(ACE_TEXT_ALWAYS_CHAR(net_if));
...
}
```

- ❏ ACE\_OS::if\_nametoindex calls the standard function if\_nametoindex(), that returns the index of the network interface corresponding to the name ifname
- ❏ TAO was using "if=ethX" as name of the network interface!!!
- When this code was changed, Voilà! It worked

# Improving performance

- ❑ Some improvements can be applied to the code in order to get a better performance
- ❑ The following is ongoing work
- ❑ Customer request: *CORBA sequences performance does not seem to be very efficient. The memset function is intensively used.*

 IDL CORBA test interface:

```
module MyModule {
    const long MSG_MAX_DATA_SIZE=52428800;
    typedef sequence<string,MSG_MAX_DATA_SIZE>
        ByteSequence;
    valuetype mivaluetype {
        public ByteSequence data;
    };

    interface Basic {
        void receivevt(in any vt);
        void shutdown();
    };
};
```

❏ Client fragment:

```
OBV_MyModule::mivaluetype msg;  
CORBA::Any miany;
```

```
miany <<= &msg;
```

```
for(int i=0;i<100; i++) {  
    tst->receivevt(miany);  
}
```

❏ Servant fragment

```
void basic_i::receivevt (const CORBA::Any &vt) {  
    MyModule::mivaluetype *msg;  
    vt >>= msg;  
}
```



What valgrind and callgrind tools say:

```
3,145,730,764   ??? :CORBA::string_free(char*) [/home/
dgarrido/ACE_wrappers/TAO/tao/libTAO.so.2.3.2]
```

```
2,621,440,370   ??? :OBV_mimodulo::mivaluetype::~~mivaluetyp
e()'2 [/home/dgarrido/src/secuencias_ms/server]
```

```
393,222,493   ??? :__GI_memset [/lib64/libc-2.12.so]
```

Reason: a myriad of C++ templates defining other templates, using templates, .... Very hard to read! Finally, a memset operation is performed

What will happen with this? The memset function is expected to be very efficient

- ❑ Problem: when the sequence is extracted in the servant, the memset function is called using the **MAXIMUM** size of the sequence without taking into account the actual sequence length (even when the actual length is zero!)

```
inline static void zero_range(  
    char_type ** begin, char_type ** end)  
{  
    ACE_OS::memset (begin, 0, (end - begin) * sizeof  
                    (char_type*));  
}
```

- ❑ And this is a very costly operation when repeated several times per second with a huge buffer

- ▶ Fifteen years of experiences using open-source middleware
- ▶ Not a bad experience
- ▶ The importance of the choice
  - ▶ Changes in operating system, languages, platform
- ▶ New trends in middleware technology
  - ▶ Data Distribution Service (OMG)
    - ▶ Publish-subscribe mechanism
    - ▶ Not very novel (from 2003!), but the life cycle of critical applications seem different from other kinds of applications