Invalidation for CORBA Caching in Wireless Devices

James Jayaputera and David Taniar

School of Business Systems
Monash University, Clayton
Vic 3800 Australia
{James.Jayaputera,David.Taniar}@infotech.monash.edu.au

Abstract. Wireless devices are widely used to get some information from the Internet. However, limited bandwidth, frequent disconnections and different communication protocols are most common problems in these devices to communicate with fixed network. Common Object Request Broker Architecture (CORBA) is a middleware architecture developed by Object Management Group (OMG) to establish communications amongst some different devices architectures. In this paper, we propose an approach to use Object by Value and Invalidation Report mechanisms to cache a CORBA object in a mobile environment. The aim of our approach is to solve some problems occurred in the existing CORBA caching approach.

1 Introduction

Mobile devices bring an efficiency for human life, such as pocket PC, handheld and mobile phones. Since these devices bring a simpler life, people try to adapt these devices into computer technologies area [1,2]. Unfortunately, this adaption raises some problems, such as frequent disconnection, slow speed transmission and slow processing resources (memory, processing speed). Frequent disconnections are the result of shadow signals, interference, changing base station (handover) and noise [3,4,5,6,7,8]. Providing a caching between mobile devices and servers (static or wireless hosts) might reduce number of disconnections in mobile devices [7,3,6]. Caching can also speed up the connection between a client and a server. In this paper, we propose an approach to use Object by Value and Invalidation Report mechanisms to cache a CORBA object in a mobile environment. The aim of our approach is to solve some problems occurred in the existing CORBA caching approach.

Caching strategies have also been used for database, CORBA or web systems like in [9,10,11,12]. Recent research suggests to reuse the existing caching mechanisms in mobile computing environments [13,7,8], however, only a few works have been done in CORBA [10,14,15]. A common issue in caching is to keep the consistency of a caching. One solution is by sending an updated information of items that have changed during w broadcast interval, where w is the invalidation

L.T. Yang et al. (Eds.): EUC 2004, LNCS 3207, pp. 460–471, 2004.

broadcast window size. This mechanism has been used to handle disconnection in a mobile environment [7,3,6].

The communication between a wireless device and a fixed host from a wired network must be through a Mobile Support Station (MSS) or Base Station (BS). A BS or MSS is a host that performs data transmission and address translation between wired and wireless network. During this communication, a mobile device might moves from one cell to another or within one cell where a cell is a physical area that is scoped by one BS or MSS. A disconnection will occur as a result of this movement, the movement from one cell to another cell is called handover. This MSS might be connected into large computer networks where they are heterogeneous, which consist of a combination of various different of operating systems and programming languages. Therefore, wireless application development must be capable to be executed anywhere.

Common Object Request Broker Architecture (CORBA) [16,17] as one type of heterogeneous applications which is developed by OMG (Object Management Group) is one possible solution. CORBA provides interoperability between objects communications in a heterogeneous system and distributed systems, in the sense that its object implementation is transparent to the developers.

Caching a CORBA object is not a simple issue, because it is a complex object which might contain data, reference, and other object's information. The existing caching CORBA objects are cached the CORBA object reference [10, 14]. However, there are some issues have been raised when we cache the reference of CORBA object. They are lack of portability, lack of support for caching across networks, lack of cache consistency, fine grained data shipping which generates unnecessary remote calls, excessive hand coding leading to software maintenance problems.

The strength feature of CORBA, Passing Object by Reference, is not suitable to be used in mobile computing area. Because this feature has excessive remote invocations to get object implementations based on the object reference. The slow bandwidth and resources (memory and processing speed) of mobile devices will become a bottleneck to this to be done. A new feature from CORBA version 2.3 or later [16,17], passing object by value has been added into it. This feature will help to reduce number of remote invocations, because interfaces and the implementations resides in the object. Therefore, the object by value can be used in mobile computing environment.

However, CORBA applications deal with heterogeneous environment which may contains different programming languages, operating systems and others. Therefore, when an object is received from one server, the object implementations need to be adapted with the current environment.

To the best our observations, passing object by value and invalidation report mechanisms have not been used together to cache the CORBA object within a mobile environment. Therefore, we would like to propose an approach that use both mechanisms together in order to cache CORBA objects within a mobile environment.

In next section, we will have brief discussion about CORBA caching and the Invalidation Report mechanisms. Some related works of this paper are presented in section 3. we will discuss theoretically about our approach to use the Invalidation Report mechanisms in CORBA caching approach in section 4. In section 5, analysis results will be shown. Finally, the last section will summarize the contents of this paper.

2 Background

Caching is a mechanism to store an object to either client, network or server in order to reduce server load [18]. CORBA Caching is an indirect way to get a CORBA object from a server to the client. However, the existing caching the CORBA object mechanism triggers some problems: they are lack of portability, lack of support for caching across networks, lack of cache consistency, fine grained data shipping which generates unnecessary remote calls and excessive hand coding leading to software maintenance problems. One way to solve the caching consistency issue is sending an invalidation message from a server to the cache in order to evict invalid object in the cache, so the object in the cache is same as one in the server. Caching the value of the object is one possible solutions to solve rest of the problems. In this section, two ways on how to cache a CORBA object are first outlined to stimulate our study. Later, some invalidation report mechanisms are described briefly in order to keep the cache consistency.

2.1 Caching by Reference and Value

CORBA applications, as one of middleware architecture and infrastructure that enable heterogeneous applications to work together over the network. Started from CORBA specification version 2.3 [19,16], a CORBA object can be passed by its reference or value. Therefore, we can cache either the references of objects or the value of CORBA objects. The terms of CORBA object is similar to Object Oriented term, which is an identifiable, encapsulated entity that provides one or more services that can be requested by a client [20]. However, the implementations of a CORBA object are separated from their interfaces. The OMG Interface Definition Language (IDL), which is used in this paper, has specified the object interfaces.

Caching the references of a CORBA Object is not adequate to be fitted into a mobile environment, because the number of remote invocations made in order to get the object implementation from the server side will consume a lot of bandwidth. Some considerations have to be taken in order to cache OO objects. Those considerations are data, reference, method level access, type of accessed and object's information [10]. Therefore, passing object by reference is not suggested to be implemented in a mobile environment since this environment has narrow bandwidth and frequent disconnections.

On the other hand, we can cache the value of CORBA object instead of the references which is more efficient in a mobile environment. This is due to

independent implementations between a client and a server, especially in a disconnection mode [19]. The aim of caching the value of CORBA object is to encapsulate data or make a copy of an object [16,17].

Object by value specification [16] adds new keywords to the IDL syntax and new constructs to the grammar. *valuetype* [16], one of the keywords,is a new IDL construct that acts as a bridge between an existing *struct* and an *interface*. *valuetype* supports public and private data members, single or multiple inheritance, description of a complex data type, such as arbitrary graphs with recursion and cycles, and sharing or null semantics.

Passing a *valuetype* as parameter during remote invocation is similar to passing a value as parameter during local invocation. Both are the same, in that they create a copy of the argument passed in the receiving side and the copy of the argument is executed locally. However, they are different, in that they pass a parameter during both invocations. This is because if a parameter is passed during remote invocation, it is passed from one server to another. Both servers could have the same or different environments. On the other hand, if a parameter is passed during local invocation, it is passed in one server. If a copy of the argument is passed remotely, the copy has no effect to the original one, because the behaviours of the local copy are always invoked locally where the *valuetype* exists. It is unlike the operation invocations on CORBA objects, which is invocations involve remotely by transmitting the requests and the replies over the network.

2.2 Invalidation Report

Invalidation Report (IR) [3,7,6] is a mechanism that a server broadcasts a report which only contains a list of data items that have been updated or changed in order to inform their clients that their cached data may have different ages.

There are two ways on how the invalidation report is passed from a server to clients: *Synchronous* and *Stateless server* approaches. The synchronous method is a method which uses a time interval in order to send an invalidation report from a server to clients periodically. It means that a updated information of the items (timestamps, a list of tuples) that have changed during w broadcast interval, where w is the invalidation broadcast window size. On the other hand, the Asynchronous method is a method that the server broadcasts an invalidation message for a given data item immediately if the data item has changed its values. This means that an invalidation message is not broadcast periodically.

On the other hand, a server can act as a *stateful* or *stateless* server. A stateless server is a server that is not aware of the consistency of its clients' caches and the availability of the mobile host in its own cell. On the contrary, a stateful server approach is where the server is aware of the information about all the MHs that are currently within its cell. The server saves the information about which data is currently being cached by which client.

3 Related Work

Sleepers and Workaholics (SW) Approach [7] has been proposed by D. Barbara and T. Imielinski [7]. The MHs are often got disconnection is indicated as *sleepers*. On the other hand, the MHs are often connected at most of the time is called *Workaholics*. In this approach, they used *synchronous* reports and *stateless* server mechanisms to broadcast IR from a server to one or many clients. On the other words, the server will broadcast invalidation reports to the mobile host and not be aware of the availability of the mobile host in its own cell. Updated Invalidation Report (UIR) approach has been presented by Cao [3]. This is the SW improved approach, because the SW approach has a long query latency whereby a client has to listen to next IR and use that report to decide whether its cache is still valid. In addition, an IR consists of information about all the items in the database, resulting in wide bandwidth consumption when an IR is broadcast. Cao improved SW approach by removing the redundant information in the IR, reducing the timestamps overhead of the UIR, efficiently utilizing the broadcast bandwidth

Asynchronous and Stateful (AS) approach [6] is an approach that the server keeps track the information about all the MHs that are currently within its cell and broadcasts an invalidation message for a given data item immediately if the data item has changed its values. The invalidation messages are kept in the MSS until an explicit acknowledgement is received from the specific MH. The purpose of this approach is to buffer the invalidation messages at the MSS in order to minimize bandwidth usage upon validation of a client's cache during reconnection.

In this paper, we propose a CORBA caching that use a Object by value and Invalidation Report. The aim of our propose is to solve the existing issues in CORBA caching.

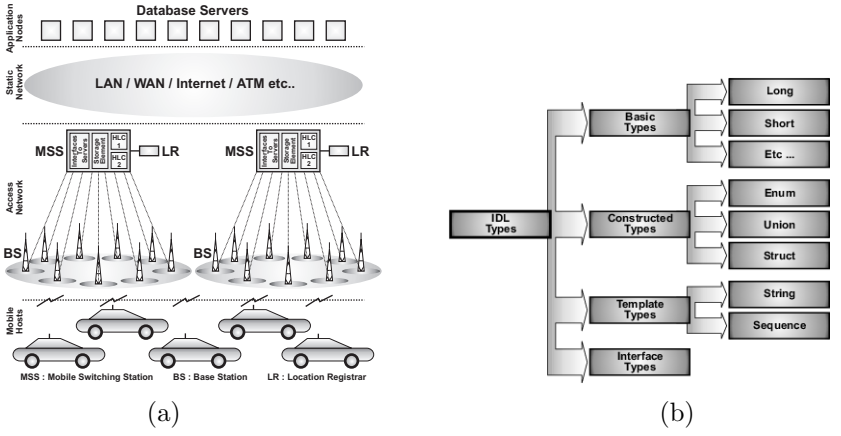
4 Propose CORBA Caching in Wireless Devices

In this section, we propose a new approach to cache the value of CORBA object and maintain the cache consistency by using Invalidation Report mechanisms in a mobile environment. Our approach is similar to the AS approach as shown in Figure 1a, however, we use CORBA object, instead of flat object. Flat object is an object that does not have any encapsulated entities [20].

This section is divided into two parts. The process to cache the value of CORBA object is discussed in the first part, followed by the invalidation process to keep the consistency of the caching in last part.

4.1 The Usage of Object by Value Approach

We have shown two ways on how CORBA objects can be cached: by reference or value, in section 2. Caching reference of CORBA objects is not a simple issue, because of some issues consideration about what kind of the objects that need to



(a) (b)

Fig. 1. (a)Sandeep-Mobile Computing Environment, (b) OMG IDL Types

be cached. Furthermore, if we notice carefully, a number of remote invocations will be expected. It means marshalling / unmarshalling costs and the amount of data transferred become major problems when we need to fetch a reference from the request and find a suitable object based on the request. Therefore, caching the references of CORBA objects are not considered.

In our propose approach, passing object by value mechanism is used to cache the value of CORBA object. It means the implementation and the interfaces will reside inside the cache. Therefore, the number of remote invocations can be reduced.

According to IONA [20], passing an object by value means that the internal state of the object is included in an operation parameter or return value and a copy of the object is constructed in the receiving process. All OMG IDL types, shown in Figure 1b [19], can be passed by value, except the interface type. The interface type only can be passed by reference, because it needs to get the object implementations.

The object by value specification [16] adds new keywords to the IDL syntax and new constructs to the grammar. However, we only concentrate with *valuetype, ValueBase and supports* keywords. We explained briefly the *valuetype* definition in section 2. We elaborate in more details about *valuetype* and the last two keywords in this section.

Valuetype does not have to declare all its operations. It can *support* an interface instead. In addition, a *valuetype* does not have to inherit from **CORBA::Object**, it inherits from CORBA::ValueBase. CORBA::ValueBase is analogous or has a similar role to CORBA::Object, which supports common operations for all value types and operations for marshalling/unmarshalling protocols, and is used to denote the base type.

The basic procedure is similar to procedure to create a CORBA object. However, we use *valuetype* to create a object. Let us consider the following example.

```
interface Account2 {
   String getAddress();
   void setAddress();
}

valuetype valueTypeObject supports Account2 {
  private String Address;
}
```

The example above has a data member called *Address* of valuetype object. After we compiled the IDL above, we will get the generated code below:

```
public class valueTypeObject implements
org.omg.CORBA.portable.ValueBase, AccountOperations
{
   String Address = null;
   protected valueTypeObject() {}
   public String[] _truncatable_ids() {
      return
      valueTypeObjectHelper.get_instance().get_truncatable_base_ids();
   }
   public String getAddress()
   { }
   public void setAddress(String theAddress)
   { }
}
```

As we can see from the generated codes above, the operations from interface **Account2** are inherited to *valueType* Object class when we use keyword *supports*. It makes our codes much simpler if we need to add another value and used the existing the operations. However, when a valuetype supports an ordinary interface type and the formal parameter type is an interface type, like the example above, its instances may also be passed by reference. In this situation, their behaviour likes an ordinary object implementations and must be linked with a POA policy and registered with ORB. Otherwise, an exception error will be raised when the instances of this value type are tried to be used as a remote object. Interested users can read [17] chapter 11 for more details about Portable Object Adapter (POA).

Moreover, using valuetype on the other hand, can avoid excessive marshalling, because a valuetype can only contain structs and the marshalling/unmarshalling process is requested in the first execution time only. Afterwards, the object is ready for a second execution. Therefore, *valuetype* works more efficiently compared to the object reference, especially for recursive operation or cyclic objects, such as trees and lists.

Figure 2a shows the caching process illustration. The cache will reside in the MSS and the MH. When a client asks for a valuetype object to the cache by marshalling the request. The cache unmarshalling the request into its environment. Then, if the cache does not have the object, the cache will forward the

client request to the server by marshalling the request. The server unmarshall the request and find the object requested. After the server found the object, the server sends the object by marshalling the object to the cache. The cache unmarshall the object and store the object. Then, the cache marshall the object upon forwarding the object to the client. The client unmarshall the object.

In this situation, the client and the cache will do remote invocations to the cache and the server respectively. However, the object received by the client is the copy version which means that any modifications on the object at client side will not affect to the object at server side.

After the object has been successfully cached, the cache consistency is considered. If the object is stored in a client cache and accessed locally, the server will inform the client that the object cached has been changed and the client can update its cache straight away. However, if some parts of a CORBA object implementation are transparently located in a heterogeneous system and that CORBA object implementation needs to be fully cached, then all library codes that support the object require to be cached as well. If some libraries that support an object are not available on the client side, then that object cannot be accessed. CORBA developers may also need to write libraries that are not available in the client side. After the object is successfully cached, the developer needs to ensure that the cached objects are consistent with the ones on the server side.

When a client does second execution with same object, the execution can be done locally without remote invocation. Therefore, the client only has less remote invocations to the server in order to get the object implementation. This situation brings a beneficial for wireless applications due to narrow bandwidth and frequent disconnection in wireless environments.

In next section, we adopt the invalidation report mechanisms to maintain the cache consistency when there is a modification on the object at the server side.

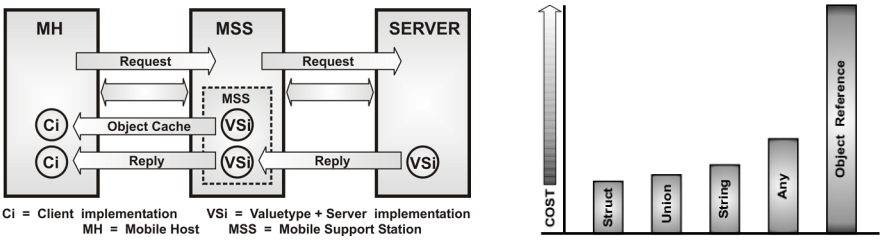
4.2 The Adoption of Invalidation Report Approach

In this section, we discuss how to maintain the cache consistency. In order to enable the MH to be updated everytime whenever there is any change on the server side, we consider to use the AS Invalidation Report approach since this approach does not consume a lot of bandwidth and the server will aware with the client availability in its cell. We assume the MSS will not faulty. In addition, we also consider to cache server functionalities that will be beneficial during disconnection.

When a MH sends a request to a server, the MH will verify its cache first. If the object is not found in the MH's cache, the request will be sent to the MSS to verify the existence and the validation of object requested. If the valid object requested is valid in the MSS, that object is sent to the MH directly. Otherwise, the MSS will find the server that holding the object. The server records the MSS details and sends the object requested to the MSS. The MSS store that object received to be used in the future and forward the object to the MH. The

information about the MH and the timestamp will be stored in the MSS and the MH.

After the object is received, the MH checks the object functionalities. If the server functionalities is needed, the MH sends a request to the MH's cache. If the functionalities is available in MH, it invoke those functionalities locally with no dependencies to the server side. Otherwise, the process of getting server functionalities from a server is similar to requesting the object above. However, the MSS adds some information; object that use that functionalities and timestamps, upon receiving the server functionalities and forward it to MH. The server functionalities received are adapted using a client object adapter or POA before they are used.



(a) (b)

Fig. 2. (a) Caching Process Illustration, (b) Comparison of cost execution of IDL Types

Furthermore, when the MH makes a second request and experiences a disconnection, the request is answered by invoking the valid object locally from its cache. If the object is not available locally and the MH is moving to another MSS (MSS2), then, the MH needs to resend the same request to the MSS2. However, the valid object requested is sent to the MH through the MSS2 at this time. The MSS1 discards the request if the MH is unavailable in its area.

When a data item; valuetype and server functionalities, in the server has been changed, the server broadcasts an IR to the MSSs that ever asked the data item. The MSS forwards the IR to those MHs, which ever asked the data item, by asking their HLCs.If those MHs are active, they mark that data item and then, they send a request for an update for that data item to MSS. MSS forwards the query to the server on behalf of those MHs. Upon receiving the updated data item back from the server, it adds a new entry to HLC for this particular data item and finally, forwards the data item to the requesters (MHs). Therefore, the MH will keep their objects updated during frequent disconnections, especially, if there is an change from the server.

5 Discussions

In order to support wireless applications works in anywhere, the wireless applications should be developed using a middleware application, such as CORBA.

Because the middleware application provides an interoperability between objects in a heterogeneous and distributed environment where it is transparent to the developers. The existing CORBA caching approach cached the references of the CORBA objects for wired network. Figure 2b [21] has shown the excessive cost of passing object by reference for doing marshalling and data transferred. Hence, if we pass a object by reference into a mobile applications, this will lead inefficiency in terms of network latencies and cost of executions.

In order to reduce the execution cost of the CORBA applications, we use the value type to cache CORBA object, because it contains struct, interface and encapsulated data. There is no dependencies between one host and another host, because the instances of valuetype can be executed locally. In our approach, we cache the value of the object instead of the reference of the object. The cost of marshalling and unmarshalling is reduced, because the remote invocation can be done at least one time.

The lack of portability can be prevented when we cached values of the objects, because the implementations of the object are mapped directly to the client environment.

We used the invalidation report approach to keep the cache consistency. When there is any modification of the object cached, the server will inform the caching by sending information of the objects changed. The invalidation report will not be broadcasted periodically due to the narrow bandwidth in the mobile environment. The client will send call back message to retrieve the message missed during disconnection. In this way, the excessive hand-coding can be skipped in order to maintain the existing software development, because software developers needs only change the implementation at server side and the server will inform the client of that modification.

6 Conclusion

In this paper, we have proposed to use an invalidation report mechanism in conjunction with caching of CORBA object in a mobile environment. The aim of our approach is to keep the CORBA object updated in client side during disconnection. Even though, Invalidation Report mechanisms have been proposed in mobile environments, however, those mechanisms do not deal with CORBA object. On the other hand, some CORBA caching approaches have been used either in wired or mobile networks, they have not been used together with Invalidation report mechanisms. Three IR mechanisms which acknowledge clients where there are updates on the server side, are addressed, namely Sleepers and Workaholics, Updated Invalidation Report and Asynchronous Stateful techniques. There are two ways to cache CORBA object: by value or by reference. Caching CORBA object reference is not efficient in a mobile environment, because it has excessive number of remote invocations to the server which consume a lot of bandwidth and frequent disconnections. However, valuetype as a new standard from CORBA which makes it capable to pass an object by value. It is a bridge between the interface and a struct. Caching value of the object is more efficient.

The client invokes remotely once to the server. The value type does not need to be dependent with the server, because it encapsulates the data and operations.

We use the invalidation report mechanisms to keep the cache consistency during frequent disconnections. Amongst the three invalidation report approaches discussed, the AS report is the most efficient approach to be used in a mobile environment. The AS approach broadcasts the invalidation report only when there is a request from the client since this approach keep availability of the client always up-to-date.

References

1. PalmOne: Palm in education: Success stories. http://www.palmone.com/us/education/studies/ (2004)
2. Palmtop: Palmtop news at vcu. http://www.cbil.vcu.edu/pda/ (2001)
3. Cao, G.: A scalable low-latency cache invalidation strategy for mobile environments. In: Proceedings of the sixth annual International Conference on Mobile Computing and Networking, ACM Press, New York, USA (2000) 200–209
4. Lee, S., Hwang, C., Yu, H.: Supporting transactional cache consistency in mobile database systems. In: Proceedings of the ACM international workshop on Data engineering for wireless and mobile access, ACM Press, New York, USA (1999) 6–13
5. Coglianese, M.: Optimistic data replication for mobile applications (2000)
6. Kahol, A., Khurana, S., Gupta, S.K.S., Srimani, P.K.: An efficient cache management scheme for mobile environment. Proceedings of the The 20th International Conference on Distributed Computing Systems (2000) 530–537
7. Barbara, D., Imielinski, T.: Sleepers and workaholics: Caching strategies for mobile environments. In: Proceedings of the 1994 ACM SIGMOD international conference on Management of data, ACM Press, New York, USA (1994) 1–12
8. Sistla, P., Wolfson, O., Huang, Y.: Minimization of communication cost through caching in mobile environments. IEEE transactions on Parallel and Distributed Systems **9** (1998) 378–389
9. Franklin, M., Carey, M.: Client-server caching revisited. In: Proceedings of the International Workshop on Distributed Object Management, ACM Press, New York, USA (1992) 57–78
10. Kordale, R., Ahamad, M., Devarakonda, M.: Object caching in a corba compliant system. Computing Systems **9** (1996) 405
11. Aggarwal, C., Wolf, J., Yu, P.: Caching on the world wide web. IEEE transactions on Knowledge and Data Engineering **11** (1999) 428–441
12. Bennet, J., Carter, J., Zwaenepoel, W.: Adaptive sofware cache management for distributed shared memory architecture. In Proceedings of the 17th ISCA (1990) 125–134
13. Huang, Y., Sistla, P., Wolfson, O.: Data replication for mobile computers. In: Proceedings of the 1994 ACM SIGMOD international conference on Management of data, ACM Press, New York, USA (1994) 13–24
14. Tari, Z., Hammoudi, S., Wagner, S.: A corba object-based caching with consistency. In: Int. Conference on Database and Expert Systems (DEXA), Florence (1999) 322–331

15. Seitz, J., Davies, N., Ebner, M., Friday, A.: A corba-based proxy architecture for mobile multimedia applications. Proceedings of the 2nd IFIP/IEEE International Conference on Management of Multimedia Networks and Services (MMNS'98) (1998)
16. OMG Group: Objects By Value. (1998)
17. OMG Group: CORBA 2.5 Specification. (2001)
18. Milutinovic, V.: Caching in distributed systems. **8** (2000)
19. Rainford, M.: The real value of corba objects. Application Development Advisor, SIGS (1998)
20. OMG: A discussion of the object management architecture. OMG Discussion 1997 (1997)
21. IONA: Mainframe CORBA Concepts Guide. 6 edn. (2003)