The architects of Ada 95 [11] had foreseen this increasing interest in distributed systems. They chose to add a Distributed Systems Annex (DSA in short) in the latest language revision [8]. This annex, while still fully consistent with the rest of the language, defines how subprograms can be called remotely, and how complex data structures such as pointers on remote objects and remote subprograms can be built and used. However, unlike foreign distributed architectures such as CORBA, those facilities preserve the strong type checking and the safety features of the Ada programming language.

SIGAda'99 10/99 Redondo Beach, CA, USA

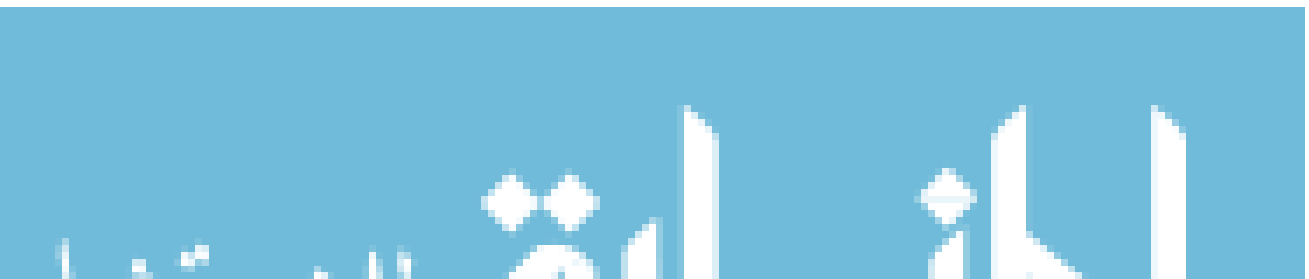
[3]. The DSA is hidden to the RTI programmer, who only needs to use the RTI API, but it takes care of all the communication between RTI nodes, as shown on figure 1.

1.2.3 Education

Ada 83 has been used for years in software engineering classes, because of its high-level features such as genericity, strong-typing, encapsulation and tasking. The fact that an Ada compiler catches most errors at compile time makes it much easier for students to concentrate on the real problem rather than on a trivial mistake uncaught by a C compiler.

Ada 95 extends the power of Ada 83 to object oriented and distributed programming. From our own teaching experience, students enjoy using it when learning the basis of distributed programming (remote subprogram calls, distributed objects) because of its ease of use and its integration in a consistent model. For example, our students have been able to develop a complete multi-users messaging system based on distributed objects in a few hours. It would

Figure 2: Global CORBA architecture



2.

Si
no
ca
re
th

However, the set of entities that can be used remotely could be slightly enlarged by introducing the notion of remote rendez-vous for example. This would require allowing a task declaration in the visible part of a `Remote_Call_Interface` package, as well as remote accesses to task types and objects. Of course, appropriate restrictions must be placed on types of entry parameters, just as those restrictions exist for remote subprograms.

3.1.2 The mid-level layer

What we call the mid-level layer here is the declaration of the `System.RPC` package. As written in section 2.1, the design team of the DSA was willing to ensure a compatibility between any DSA-capable compiler and any PCS through this standardized package.

While this definition allowed us to start quickly the implementation of GLADE because one part of the design was implicitly contained in the annex, we soon realized that the requirement to go through `System.RPC` for every remote call introduces a lot of constraints.

GNAT has been developed as a separate project [1] and can be used independently.

3.3.2 Using a common protocol

We are currently investigating the use of IIOP as the basis of our low-level protocol, to ease the interfacing process between the DSA, CORBA and RMI. The basic idea behind this is effort splitting: implementing a tasking runtime requires a lot of resources from Ada compilers vendors, and so does the implementation of the DSA. If some of the costs could be shared by the CORBA and RMI vendors, there would probably be more implementations of the DSA, as the existing infrastructure could be reused easily.

We already have a full Ada ORB implementation [10], soon to be released as free software. This software, whose code name is Broca, will serve as a basis for both our free software CORBA product, AdaBroker, and a future version of GLADE that will be using IIOP.

Using IIOP as the standard protocol for the DSA would allow accessing CORBA and RMI services directly through the DSA

but would benefit from a standardization of the protocol used to communicate between the partitions. This standardization would ease the development of Ada distributed applications using different Ada compilers, and would open the world of distributed Ada to other languages, without loosing any of the Ada safety and security.

Despite those (hopefully constructive) criticisms, we firmly believe that the presence of the Distributed Systems Annex is a major achievement in the language definition. We want to thank again the design team who chose to make Ada even more powerful and user-friendly by including the DSA in the ISO standard.

References

[1] F. Azavant, J.-M. Cottin, L. Kubler, V. Niebel, and S. Ponce. AdaBroker, using OmniORB2 from Ada. Technical report, ENST Paris, March 1999.

[6] http://www.infres.enst.fr/ANC/