

# Use of Reflective Programming Language on Developing CORBA Applications

Kazuhiro Fujieda Takuo Watanabe Koichiro Ochimizu  
Japan Advanced Institute of Science and Technology, Hokuriku  
{fujieda,takuo,ochimizu}@jaist.ac.jp

February 14, 2000

## 1 Large Cost for Development Procedures in using CORBA

In application development using CORBA, the cost of programming is reduced with the use of stubs and skeletons generated by an IDL compiler. In contrast, the cost of development procedures is augmented by these files. These procedures include instructing the features supported by stubs and skeletons to an IDL compiler, and linking them into applications or deploying them to the runtime environments of applications. They become a heavy burden to developers in the situation that interfaces of distributed objects described by IDL or usage patterns of the objects by applications are changeable.

Some of configurability and features provided by ORB depends on extra codes generated in stubs and skeletons. Such extra codes lead extra runtime costs or turnaround time. So developers need to specify which codes an IDL compiler should generate. In general implementations, developers can specify only whether stubs include the codes handling the Any type to an IDL compiler. In some implementations, extra codes are necessary to increase configurability of ORB, such as smart stubs and object wrappers in VisiBroker [1]. CORBA Messaging [2] that will be introduced into CORBA 3.0 needs vast codes in stubs to simplify programming to invoke operations of distributed objects asynchronously.

Integrated development environments supporting CORBA can reduce the costs of development procedures to a certain degree. But such environments restrict development style excessively and cannot automate whole development procedures without instructions by developers on their GUI.

## 2 Automatic Runtime Stub Generator Using the Reflection

For solving these problems, we realized the automatic runtime generator of stubs and skeletons as a library incorporated into CORBA applications. The generator is realized by making use of the power of reflective programming languages, so it is available only when applications are implemented with such languages. It obtains specifications described by IDL necessary to generate stubs and skeletons from the Interface Repository.

Reflective programming languages allow programs to inspect and modify themselves or generating new one, and to use and modify mechanisms of execution environment of them.

One major role of stubs and skeletons is handling network protocols such as mutual conversion between values of complex data types and network streams. This role can be implemented as a common routine using the inspection of data types or class definitions.

Another role of stubs and skeletons is making the type or class definitions correspond to user-defined types or exceptions defined by IDL available to application programs. Because of this role, the ability to generate type or class definitions dynamically is indispensable for realizing the generator. Application programs may use the class definition corresponding to a user-defined type to create an argument value of an operation. So the approach using only customizability of mechanisms invoking methods to map local invocations to request protocols [3] cannot fully support programming in CORBA applications.

For detecting the stubs and skeletons necessary to an application program, the generator requires customizability of mechanisms resolving undefined names of class/type names or module/package names. For generating extra codes to support configurability or features provided by ORB just in time, it also requires customizability of mechanisms invoking methods or functions. Moreover, it requires ability to add methods to existing classes in object-oriented languages.

For example, the polling model of CORBA Messaging requires the generated method of which the name consist of the prefix `sendp_` and the name of the operation invoked asynchronously. It also requires the generated class corresponding to the poller value-type used for receiving the result of the operation later. If the generator can customize the mechanisms resolving undefined class names and invoking methods, it generates these classes and methods in the customized behavior.

### 3 Conclusion

This position paper has described the problems of development procedures in application development using CORBA, and presented the automatic runtime stub generator using reflective programming languages as their solution. It can automatically generate extra codes necessary to applications with various requirements, such as configuring the behavior of ORB or invoking operations with various styles. This paper has also presented the abilities provided by reflective languages necessary to realize the generator.

### References

- [1] INPRISE Corporation, Inc. *VisiBroker for Java 3.3 Programmer's Guide*, 1998. available at <http://www.inprise.com/techpubs/books/vbj/vbj33/index.html>.
- [2] Object Management Group. *CORBA Messaging*, May 1998. OMG TC Document orbos/98-05-06.
- [3] Roberto Ierusalimsky, Renato Cerqueira, and Noemi Rodriguez. Using reflexivity to interface with CORBA. In *IEEE International Conference on Computer Languages (ICCL'98)*, May 1998.