基于对象请求代理的程序自动开发方法与工具*

王千祥, 骆永华, 梅 宏, 杨芙清

(北京大学 计算机科学技术系,北京 100871)
E-mail: wqx@cs.pku.edu.cn; yang@cs.pku.edu.cn
http://www.pku.edu.cn

摘要: 随着分布式对象技术的日益流行,越来越多的应用程序开始采用基于对象请求代理(object request broker,简称 ORB)进行开发.从开发人员角度看,目前这类应用程序在开发过程上仍存在一些不足之处,例如,需要掌握的基本概念较多,ORB 产品之间的兼容性不足等.在深入分析了这些问题后,提出了一种改进开发过程的方法,通过代码自动生成,分离特定于 ORB 产品的代码等技术,尽可能提供对其中开发活动的自动支持.实验结果表明,实现上述方法的支撑工具原型系统可以有效地减少代码编写量,并降低编码出错的概率.

关 键 词: 分布式对象技术;对象请求代理;CASE
中图法分类号: TP311 文献标识码: A

分布式对象技术(distributed object technology,简称 DOT)是分布式计算技术和面向对象技术的结合.与传统的分布式程序设计方式相比,例如基于套接字(socket)的技术和基于远程过程调用(remote process call,简称 RPC)的技术,分布式对象技术提供了更高层次上的抽象和更强的协同工作能力;与传统的面向对象技术相比,分布式对象技术支持分布的、异构的对象之间的互操作[1~3].分布式对象技术正趋于成熟,并得到了广泛的应用,目前电信、金融、商业等许多领域皆存在大量基于分布式对象技术开发出来的应用程序.分布式对象技术的流行是基于构件的软件开发(component based software development,简称 CBSD)作为一个研究热点兴起的主要动因之一[4],它在实现层上提供了对软件复用,特别是基于构件的软件复用的技术支撑[8],它所提供的对象模型和互操作能力正在成为构件实现和构件组装的基础.当然,我们也应该看到,CBSD 提供了一条自底向上的支持软件复用的途径,却缺少系统化的方法论来指导 CBSD 过程[5].

从软件工程角度看,分布式对象技术向软件工程提出了新的需求.一方面,分布式对象技术为软件系统的开发带来了明显的益处,例如,提供了实现级的构件模型,提供了必要的公共服务等.另一方面,基于分布式对象技术的应用程序的开发尚存在一些不足之处,例如,开发人员需要了解较多的基本概念,才能较好地掌握分布式对象技术;不同的 ORB 支持产品之间兼容性不足;应用程序的开发与前期分析、设计阶段之间的衔接不够密切等等.这些问题集中表现在,从分布式对象系统开发的全过程看,除编程环节有所改善外,总体开发效率还不够高.这种状况影响了分布式对象技术的推广与应用.

CORBA[6],COM[7]和 EJB[8]是目前在学术界和产业界受到广泛重视的 3 种分布式对象技术规范,它们分别在特定的应用环境得到了不同程度的应用.其中,CORBA 是对象管理组织(object management group,简称 OMG)在对象管理体系结构(object management architecture,简称 OMA)基础上定义的对象请求代理的公共结构.与其他两种分布式对象技术规范相比,CORBA 在开放性、互操作性等方面的特点更为显著,得到了众多软件厂商的

* 收稿日期: 2000-07-20; 修改日期: 2000-11-27
基金项目: 国家自然科学基金资助项目(60103001;60043002);国家教育部青年骨干教师资助项目

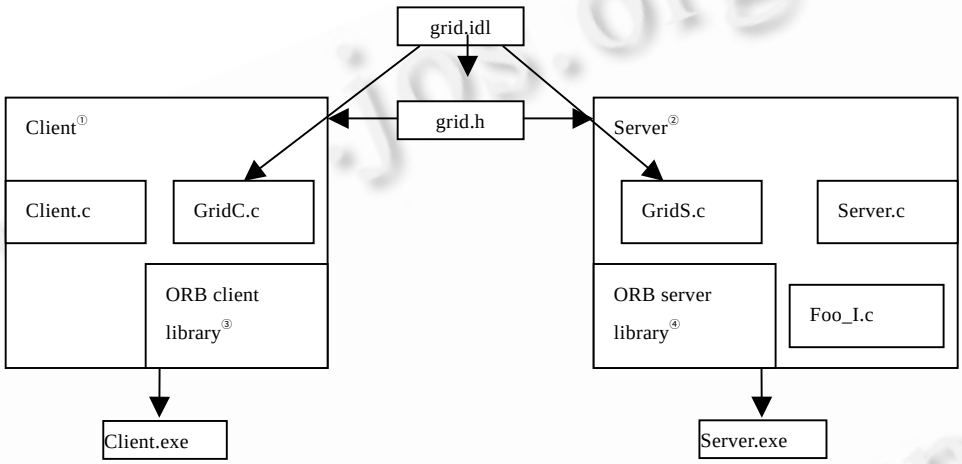
作者简介: 王千祥(1970－),男,山东莱州人,博士,副教授,主要研究领域为软件工程,网络计算环境;骆永华(1977－),男,浙江绍兴人,硕士生,主要研究领域为软件工程;梅宏(1963－)男,重庆人,博士,教授,博士生导师,主要研究领域为软件工程和软件工程环境,软件复用和软件构件技术;杨芙清(1932－)女,江苏无锡人,教授,博士生导师,中国科学院院士,主要研究领域为系统软件,软件工程和软件工程环境,软件复用和软件构件技术,软件工业化生产技术.

支持,因此,我们的研究着重于如何对基于 ORB 的应用程序开发提供自动支持.

1 基于 ORB 的应用程序开发

OMG 定义的 ORB 是一种面向对象中间件,它有效地隔离了互操作的双方:只要客户与服务器对象都遵守由接口定义语言(IDL)定义的共同接口,则客户可以透明地激活一个远地服务器对象.这种机制有力地支持了大型复杂系统的集成,因而 CORBA 在大型业务关键的应用系统中得到了广泛应用.CORBA 规范的重点在于对对象间的互操作进行约束,并对对象的运行过程进行支持,至于 ORB 怎样实现,应用程序怎样开发,则不给予约束.

目前已经有很多厂商提供了支持 CORBA 规范的 ORB 产品,但由于 CORBA 技术发展的历史还不长,多数 ORB 厂商主要致力于对 CORBA 规范的实现,这导致了现有的 ORB 产品在开发过程上基本是类似的,但在编程细节上则差异较大.假定我们以一个网格对象 grid 为核心开发一个分布式对象系统,则基本的开发过程如图 1 所示.



①客户,②服务器,③ORB 客户函数库,④ORB 服务器函数库.

Fig.1 The developing process of ORB-based applications

图 1 基于 ORB 应用程序的开发过程

各步骤的具体内容如下:

(1) 定义对象接口.对于系统中需要利用 ORB 进行交互的对象(grid),首先应将其对外显露的接口进行定义,并生成 IDL 文件,本例中为 grid.idl.

(2) 将 IDL 文件映射为具体语言的指代(stub)/骨架(skeleton).在本例中,它们分别包含在 GridC.cc 与 GridS.cc 中.IDL 是独立于具体编程语言的,而应用程序则一定是由具体语言完成的,因此,必须将 IDL 进行映射,产生由具体语言表示的接口,以供调用.

(3) 编写实现具体服务功能的代码.ORB 提供的仅是对象间互操作的支持,至于对象提供的具体功能,则必须由编程人员实现.在本例中,具体代码为 Server.c.

(4) 编译、连接,产生服务器程序,在本例中为 Server.exe.

(5) 编写调用具体服务功能的代码.与(3)类似,客户如何调用服务的过程由编程人员实现.在本例中,具体代码为 Client.c.

(6) 编译、连接,产生客户程序.在本例中为 Client.exe.

其中,(3)、(5)两部分需要编写具体的代码,而且所编写的代码不仅需要符合 CORBA 规范,还需符合具体 ORB 产品的特性,这就向开发人员提出了较高的要求.由此可以看出,尽管在 ORB 的支持下,开发人员可以将精力集中在系统的逻辑上,简化系统在互操作方面的开发,但是从整个开发过程来看,则仍存在多方面的不足,例如:ORB 的确为基于 ORB 进行分布式系统的开发屏蔽了底层细节,从而在一定程度上降低了系统开发的复杂

度,但是 CORBA 本身引入了许多新的概念,例如:对象引用、代理、对象适配器,以及各种高层服务等,如果编程人员需要开发基于 ORB 的应用程序,则需要熟练掌握这些概念.这对于使用 CORBA 进行应用系统开发的一般用户来说,是一个较大的障碍,如果有工具对开发过程进行支持,则开发人员可以在工具的引导支持下,较快地掌握相关概念,并忽略一些细节与技巧.

由于 OMG 仅制订 CORBA 规范,具体的 ORB 产品由各个厂商分别实现,因此,尽管这些 ORB 产品都声称符合 CORBA 规范,但是各个 ORB 的实现皆表现出一定的特殊性,如果体现这些特殊性的代码能够由工具自动生成,则可以显著地减少开发人员的代码编写量.CORBA 对于目前软件界兴起的基于构件的软件开发提供了有力的支持,但 CORBA 主要解决系统的分布问题和异构问题,对基于构件开发的支持也主要体现在编码阶段,而对于业务逻辑的分析、系统结构的设计等问题,CORBA 并不提供直接支持.

从上面的分析我们可以发现,如何为基于 ORB 应用程序的开发提供更为简洁、高效的支持,是目前尚未得到深入研究的问题,这也正是本文希望解决的.

2 基于 ORB 的应用程序自动支持方法

通过对上述基于 ORB 应用程序开发过程的分析,以及现有产品及平台对开发过程支持不足的现状,本文提出如下的辅助开发方法,以简化开发过程:

(1) 开发一个辅助编程工具(支撑工具),引导开发人员进行基于 ORB 的应用程序开发.这不仅可以降低基于 ORB 应用程序开发的门槛,还可以帮助保证程序代码的规范性、正确性.

(2) 支撑工具提供自动生成框架代码的能力.在一个基于 ORB 的应用程序开发过程中,框架代码的编写比较机械,这部分工作完全可以由工具自动完成,使开发者把注意力进一步集中到特定的应用之上.实际上,这也是技术人员开发 ORB 这一中间件的初衷之一.

(3) 分离特定于具体 ORB 产品的代码,使开发者可以更少地关注具体的 ORB 产品.

从前面对开发步骤的分析我们得知,基于 ORB 应用程序的代码分为两部分,即服务器端代码与客户端代码,因此本文分别研究服务器端与客户端的辅助开发方法.下面我们结合所开发的支持工具原型分别对它们进行介绍.

2.1 服务器端工具

服务器端支持工具的目标是:根据特定于服务器及环境的信息,生成特定于具体 ORB 实现的代码框架,将特定于 ORB 环境的代码与特定于应用的代码分离,并使开发者能够方便地与已有的代码框架进行交互,以构造出符合要求的服务器端应用程序.工具结构如图 2 所示.其中各部分的功能分别为:

(1) 获取接口信息

接口信息可以由开发者手工提供,也可以从设计工具导入.接口信息包括下面几方面的内容:

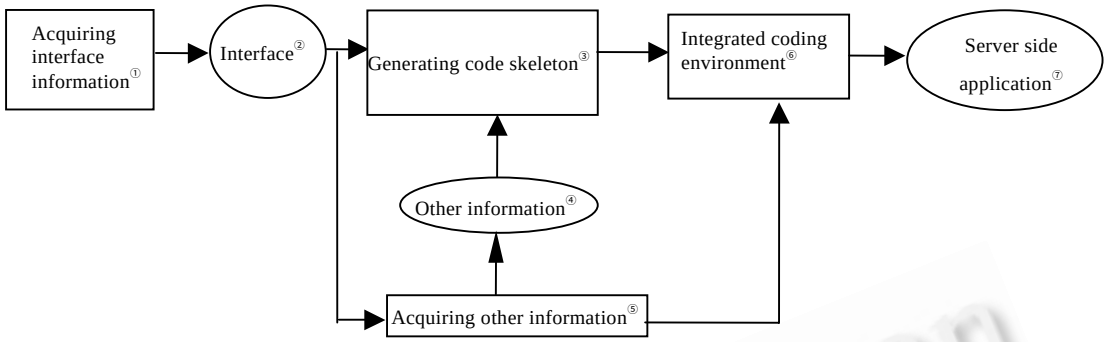
- 服务器所提供服务的 IDL 描述,这是必须包括的接口信息.
- 服务器向客户发出回调(call back)时所需的客户端回调接口的 IDL 描述,这是可选的接口信息.
- 服务器需要请求的其他服务器接口的 IDL 描述,这是可选的接口信息.
- 服务器向请求服务提供的回调接口的 IDL 描述,这是可选的接口信息.

(2) 获取其他信息

接口信息对于代码框架的生成来说不够充分,还需要根据其他信息共同生成代码框架,这些信息包括:

- IDL 编译选项信息

IDL 编译选项是 IDL 编译器在对 IDL 文件进行编译时,针对开发过程中不同的编译需求及代码生成要求所设置的一些开关.不同的 ORB 实现都有自己专有的 IDL 编译器,因此在 IDL 选项的功能和格式上各不相同.通过分析,我们总结出了各种 IDL 编译选项中的一些具有共性的开关设置,例如:对象实现采用 BOA 方式还是 TIE 方式,是否为 IDL 文件中所包含的其他 IDL 文件生成代码等,方便了这类信息的获取.



①获取接口信息,②接口,③生成代码框架,④其他信息,⑤获取其他信息,⑥集成编码环境,⑦服务器端应用程序.

Fig.2 The structure of server side supporting tool

图 2 服务器端支持工具的结构

• CORBA 高层服务信息

这是指一个服务器在自己提供服务的同时,可能需要一些 CORBA 高层服务的支持,如命名服务(naming service)、事件服务(event service)、事务服务(transaction service)等.需要高层服务的应用程序在逻辑上更复杂,对支持工具的要求也更高,目前的支持工具暂时仅考虑支持命名服务.

• 平台信息

平台信息包括两方面的内容:服务器基于哪一种具体的 ORB 产品,以及服务器采用哪一种编程语言环境(以及底层操作系统运行环境).

• Make 信息

目前的 ORB 产品多数采用 makefile 文件对文件之间的依赖关系以及编译链接过程进行控制,在该文件的信息中,一部分与上面提到的信息有关,另外一部分则需要专门指定,例如,所依赖的库函数、生成调试版还是发布版等.

(3) 生成代码框架

代码框架包括实现类的框架、应用程序的主函数、Makefile 文件、与 NamingService 相关的代码.

(4) 集成编码环境

在代码框架生成之后,开发者即可以在这样一个框架结构上加入特定于应用需求的实现代码,与一般的编程语言集成环境类似,本系统的集成编码环境主要的功能包括编辑源代码、服务器信息的进一步扩充(如添加并编辑 naming context graph)、应用程序的编译链接.

提供集成编码环境不仅有利于保证所生成代码框架的一致性和完整性,也为开发者继续在代码框架之上添加自己的实现代码提供了方便.

2.2 客户端工具

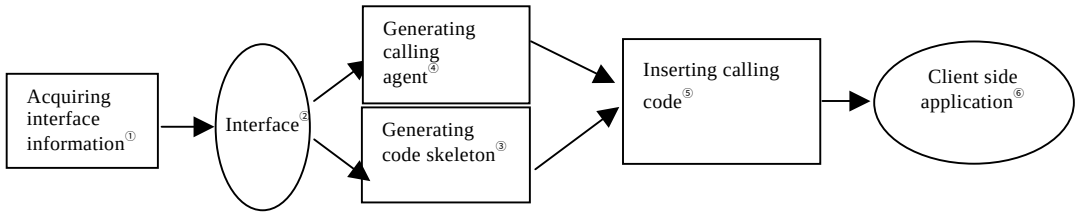
客户端支持工具的设计目标是:使开发者可以方便地浏览接口池提供的接口,并根据业务逻辑的需要通过图形化的操作在本地生成远程对象的代理,自动生成相应的初始化代码,并使开发者可以方便地往特定于应用的代码文件中插装调用语句.工具结构如图 3 所示.其中各部分的功能分别为:

(1) 获取接口信息

客户端的接口信息是通过接口浏览器获取的,该模块负责从接口池服务器和名字服务器中读取信息,并以用户所需要的方式在本地予以显示.

接口浏览器负责实现从保存接口描述信息的接口池服务器中读取接口信息(包括接口属性和操作),并在树形结构的窗口中进行显示.在接口浏览器中显示的信息包括服务器、接口、操作、属性、参数等.

如果需要支持命名服务,名字浏览器负责与命名服务服务器的交互,并将从保存名字及其环境的命名服务服务器中读取名字信息,然后在树形结构的窗口中进行显示.



①获取接口信息,②接口,③生成代码框架,④生成调用代理,⑤调用代码插入,⑥服务器端应用程序.

Fig.3 The structure of client side supporting tool

图 3 客户端支持工具的结构

(2) 生成代码框架

本模块主要完成客户端代理的生成以及 Makefile 文件的生成.客户端代理生成的主要工作是从接口信息中提取用户所需接口的信息,并生成 IDL 文件.然后调用 IDL 编译器编译所生成的 IDL 文件,得到客户端代理.以 Makefile 模板为基础,根据用户指定的一些信息,生成可编译源代码的 Makefile 文件.用户指定的信息有:是否为调试版(用户指定)、是否动态链接(用户指定)、可执行程序的文件名(用户输入)等.

(3) 生成调用代理

调用代理是我们为进一步简化客户端的开发而提出的概念,它封装了与代理密切相关的代码,例如对象的声明、对象的绑定(或创建)等.对于基本的 ORB 编程,可以根据用户选择的接口和用户输入的接口以及服务器所在的主机,生成与该接口交互的所有代理的代码(包括修改和读取接口属性、调用接口所有操作).如果需要支持命名服务,则根据用户选择的名字及其所属的上下文,生成从该名字得到的对象引用的代理代码.

(4) 调用代码插入

对于基本的 ORB 编程,用户可以将已生成代理的接口中的操作或属性“拖”、“放”到源代码中的某个位置,由支持工具在该位置生成调用操作或读取或修改属性的一条语句.具体参数的输入将由系统弹出对话框,列出每个参数的名称、类型以及编辑框.用户可以在编辑框中输入参数,或者直接在源代码中输入系统将生成的简单语句.

如果需要支持命名服务,则在用户将已生成代理的名字(对象)“拖”、“放”到源代码中的特定位置时,系统在该位置上生成调用代理中对象引用的一条语句.

3 实例研究

本节中我们结合一个简单实例进一步介绍工具的支持方法.实例仍然结合上面提到的网格对象进行,其中,服务器端包含一个名为 Grid 的 CORBA 对象,而客户端调用该对象,其中 Grid 的接口定义文件 Grid.idl 已经由建模工具生成,另外,应用程序不使用高层服务.

在工具的支持下,服务器端应用程序的开发步骤为:

(1) 读取建模工具生成的接口信息 Grid.idl:

```
interface grid{
    readonly attribute short height;    //height of the grid
    readonly attribute short width;    //width of the grid
    void set(in short raw, in short col, in long value);    //set the element [raw, col]of the grid
    long get(in short raw, in short col;    //return element [n,m] of the grid
}
```

(2) 在工具的引导下,选择各种必要的信息,例如:对象实现采用 BOA 的方式,不使用高层服务,采用的 ORB 产品为 Orbix,实例初始化参数为(100,100)等.

(3) 编译 Grid.idl,生成 Grid 类的头文件以及部分框架.

(4) 生成应用程序主函数、Makefile 文件等代码(具体代码略).

(5) 编写必须的功能实现代码(黑体部分),其他部分是工具自动生成的:

```
class Grid_i: public virtual GridBOAImpl{
    short m_height, m_width;
      long **m_array;
  ……
}
 virtual CORBA::Short Grid_i::width(CORBA::Environment &) { return m_width; }
 virtual CORBA::Short Grid_i::height(CORBA::Environment &){ return m_height;}
 virtual void Grid_i::set (CORBA::Short row, CORBA::Short col, CORBA::Long value,
                           CORBA::Environment&){ m_array[row][col] = value; }
 CORBA::Long Grid_i::get (short row, short col,
                           CORBA::Environment &){ return m_array[row][col]; }
```

(6) 编译,连接,生成服务器端的应用程序

这样,原来需要手工编写代码的(1)、(2)、(4)全部由工具给予直接支持,不必直接编写代码,而原来需要编写大量代码的(5)的工作量也大为减少.

客户端应用程序的开发步骤为:

(1) 支持工具从接口浏览器中读取建模工具生成的 Grid 对象的接口信息.

(2) 生成客户端代理.

(3) 生成 Makefile 文件(具体代码略).

(4) 生成调用代理(以获取网格的宽度为例):

```
CORBA::Short TOOL_grid_grid_width(){
        gridVar = grid::_bind("id","hostname");
        return(gridVar->width());
}
```

(5) 插入调用代码(仍然以获取网格的宽度为例):

```
TOOL_grid_grid_width();
```

(6) 编译,连接,生成客户端的应用程序 Client.exe.

这样,原来需要手工编写代码的(3)~(5)由工具给予了支持,不仅减少了代码编写量,同时也降低了出错概率.

在上述例子中,辅助工具自动生成 182 行代码,在不包含实现具体功能代码的前提下(该部分代码量随机性较大),手工还需编写 26 行代码.因此,辅助工具自动生成了原来需手工编写代码(上述代码行数之和)的 87.5%,可见,辅助工具对用户编码工作的减轻是十分显著的.

4 结束语

随着软件规模与复杂性的增加,基于 ORB 进行分布式应用系统的开发将为越来越多的开发组织所接受,这为基于 ORB 的应用程序提供了很大的潜在市场.从软件工程的角度提供自动化的支持将在较大程度上提高这类开发工作的效率,并改善开发产品的质量,本文提出了改进这类应用程序的开发过程的一种自动化途径,并开发了相应的支撑工具原型.实验性应用表明,该方法及工具确实可以为开发人员提供方便和帮助,从而提高开发效率.进一步的研究工作包括如下几方面:

(1) 支持更多的高层对象服务.如果没有高层对象服务的支持,开发人员是很难开发出大型而复杂的分布式系统的,因此除现有对命名服务的支持外,工具将对于更多的高层对象服务提供支持.

(2) 实现与特定 ORB 产品的无关性.目前实现的工具原型在这方面做了很多尝试,但是还没有完全做到对特定 ORB 产品的无关性.我们希望最后的工具能允许用户通过参数设置的方式,方便地在不同的 ORB 之间进行切换,不必手工去修改代码,更不必完全重写与应用逻辑相关的代码.

(3) 与建模工具的进一步结合.目前实现的工具原型在 IDL 获取等方面已经初步利用了建模工具的功能,但两个工具之间的衔接还不是很平滑,我们将在如何于编码阶段充分利用设计结果这一问题上进行更为深入的研究.

References:

[1] Schmidt, D.C., Schmidt, S.V. Introduction to distributed object computing. SIGS C++ Report , 1995.

[2] Ron B.N. CORBA: A Guide to Common Object Request Broker Architecture. New York: McGraw-Hill, 1995.

[3] IONA Technologies. Orbix Programmer's Guide. IONA Technologies, 1998.

[4] Yang Fu-qing, Mei Hong, Li Ke-qin. Software reuse and software component. Journal of Electronics, 1999,2:68~75 (in Chinese).

[5] Mei, Hong, Chang, Ji-chuan, Yang, Fu-qing. Composing software components at architectural level. In: Proceedings of the IFIP WCC 2000. Beijing: Publish House of Electronic Industry, 2000. 224~231.

[6] OMG'99, Object Management Group. The Common Object Request Broker: Architecture and Specification, Revision 2.3, 1999.

[7] Dale Rogerson, Inside COM, Microsoft Press, 1997.

[8] DeMichiel, L., Yalcinalp, U., Krishnan, S. Enterprise JavaBean Specification,Version 2.0, 2001. http://java. sun.com/ejb.

附中文参考文献:

[4] 杨芙清,梅宏,李克勤.软件复用与软件构件技术.电子学报,1999,2:68~75.

An Automated Approach to Development of ORB-Based Applications and Its Supporting Tool*

WANG Qian-xiang, LUO Yong-hua, MEI Hong, YANG Fu-qing

(*Department of Computer Science and Technology, Beijing University, Beijing* 100871, *China*)

E-mail: wqx@cs.pku.edu.cn; yang@cs.pku.edu.cn

http://www.pku.edu.cn

Abstract: With the distributed object technology becomes a hot topic, more and more developers begin to use ORBs as the applications' basis of developing and executing. ORB can simplify the development of large application by facilitating the lower-level communication between objects. But from the view of developers, this kind of support is not enough yet, such as the introduction of too many new concepts (especially those related with ORB), and the incompatibility between the ORBs from different vendors. These problems bring difficulties to the application developers for their mastering the ORB and make the applications portable between the different ORB platforms. After analyzing the issues in the ORB-based application development, an automated approach is presented to improve and facilitate the development process. The prototype of the supporting tool shows that by generating skeleton codes and separating ORB product related codes, the proposed method can efficiently reduce the coding work and the number of errors.

Key words: distributed object technology; object request broker; CASE

* Received July 20, 2000; accepted November 27, 2000

Supported by National Natural Science Foundation of China under Grant Nos.60103001,60043002; the Chief Young Teacher Foundation of the Ministry of Education of China