业务逻辑编译器的设计与实现

范 晖 夏清国
(西北工业大学计算机学院 西安 710072)

摘 要 本文介绍了一种应用程序中业务逻辑的实现方案。该方案使用脚本语言来定义业务逻辑,由业务逻辑编译器来生成代码,从而提高业务开发的效率和扩展性。业务逻辑编译器由业务解释模块和业务执行模块组成。该方案已在电信领域投入使用,取得了预期的效果。

关键词 分析程序生成器,扫描程序生成器,业务逻辑

Design and Implementation of the Service Logical Compiler

FAN Hui XIA Qing-Guo
(Department of Computer, Northwestern Polytechnical University, Xi'an 710072)

Abstract A method to implement service logic is proposed. Script language is adopted in this system. A Service logic compiler that generates codes is discussed in detail. It consists of service interpretation module and service implementation module. By now it has been used in telecommunication.

Keywords Parser generator, Scanner generator, Service logic

1 引言

分析程序生成器(parser generator)是一个指定某个格式中的一种语言的语法作为它的输入,并为该种语言产生分析过程以作为它的输出的程序。其中最著名的是 Yacc(yet another compiler-compiler)。用户定义语法规则和语法执行动作,Yacc 把词法分析程序返回的单词和单词属性依据语法规则进行匹配和规约,当匹配到某条语法规则后,执行此规则对应的执行动作。执行动作主要是对信息表的管理、中间代码的生成以及错误检查和处理。

有穷自动机的研究也发展了另一种称为扫描程序生成器(scanner generator)的工具 Lex(lexical analyzer)。Lex 最常见的版本是 Flex(Fast Lex)。用户定义词法分析规则和分析动作,将需要分析的业务逻辑文件打开,并将文件句柄赋值给 Flex 中定义的变量 FILE * yyin。Flex 读取 yyin 文件中的数据,与用户定义的规则逐一匹配,如果匹配成功,执行用户对此规则定义的分析动作。语法分析程序 Yacc 就是不断调用函数 yylex(),与 Flex 进行交互。

Flex 和 Yacc 是 UNIX 两个非常重要的、功能强大的工具。它们的强大功能可以编写任何类型编译器。

目前应用程序普遍采用多层的分布式模型。应用程序的逻辑根据其实现的不同功能被封装到不同组件中,应用程序中最复杂、最难编写的是业务逻辑。业务逻辑通俗地讲是业务上的处理规则 或流程,例如数据处理公式等经过分析后整理成的程序语言。业务随着用户的需求不断地发生变换,例如增加新的业务,修改以前的业务处理逻辑等等。为了满足用户的需求,应用程序需要不断的修改,这无疑加大了程序员的负担,增加了维护和开发成本。虽然可以使用 J2EE 体系中的 EJB 来解决,但是现实中的大部分项目都是单服务器的轻量级项目,这时用 EJB 就像是大炮打蚊子,蚊子没打到,房子却被打破了个洞,而且 EJB 的笨重和复杂是出了名的,很难使用。再则 EJB 也不一定适合每个行业的实际需要,价格也比较昂贵。

2 业务逻辑编辑器的设计及实现

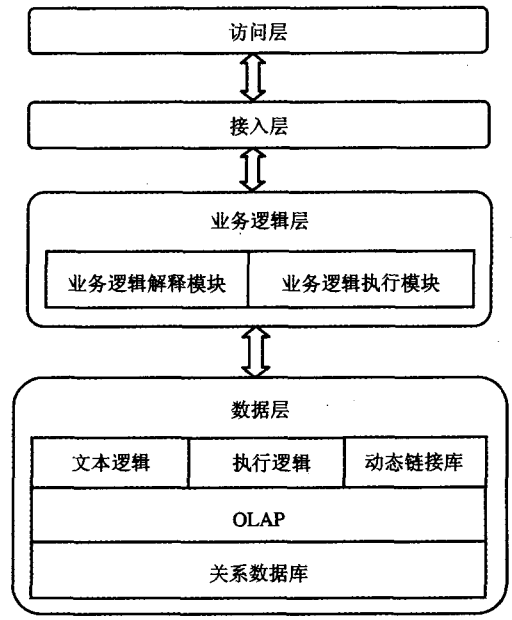

图 1 应用系统的架构

针对以上的问题,本文提出了一种解决方案,将编译原理的思想引入到业务的逻辑开发中,采用分析程序生成器来设计脚本语言,开发一个业务逻辑编译器,结合应用系统的体系结构,能够较好地避免这些问题的发生。应用程序中涉及的业务逻辑由业务逻辑层来进行抽象和处理,开发人员或者用

范 晖 硕士研究生,研究方向:分布计算及应用;夏清国 副教授,博士研究生,主要研究方向:多媒体通信及其服务质量。

户只需使用简单的脚本语言来定义业务流程，就可以方便地完成业务的处理过程。使用脚本语言可以灵活地开发出各种业务逻辑，支持根据需要方便灵活地实施与拓展业务，具有很好的可扩展性。

整个架构方案跨越访问层、接入层、业务逻辑层和数据层。访问层允许互联网用户和内部网用户的访问。接入层中通过实施防火墙和非军事区建立企业门户安全机制，通过设置 Web 服务器建立企业门户。业务逻辑层允许实施多种业务逻辑应用，并根据需要使用不同存储资源（数据库、文件和动态链接库等）。业务逻辑层分为业务逻辑解释模块和业务逻辑执行模块两部分。该层完成业务逻辑编译器的功能。业务逻辑解释模块接受来自接入层的用户请求，将其转化为业务执行模块能够理解的语言并输出到业务的执行逻辑文件中。业务逻辑执行模块根据业务逻辑有序地向数据层发送数据请求，并将数据层返回的数据进行解析，组合成用户所需信息，返回给接入层。业务逻辑执行模块可以调用外部的动态链接库，以完成复杂的功能。在数据层，提供集中的数据管理，支持多种商业应用。数据层主要包括业务逻辑层进行操作的数据如文件、动态链接库和数据库等，完成数据的存取控制。

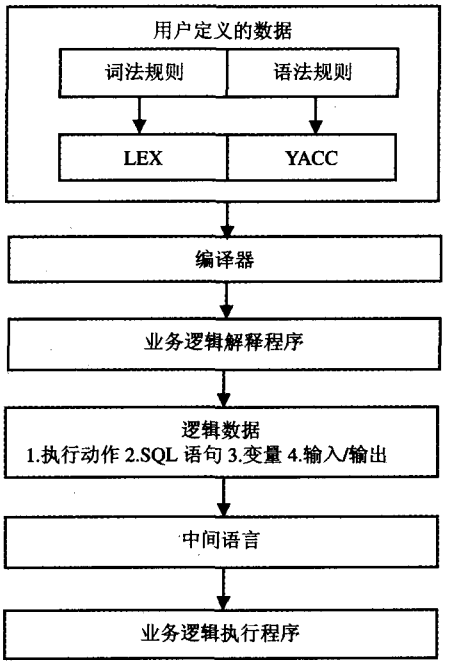

图 2　业务逻辑层的结构

2.1　业务逻辑设计

业务逻辑是用户定制的一系列业务处理动作的集合。它由业务解释程序翻译成业务执行程序，可以识别且高效执行中间语言。

业务逻辑的语法是 C 语言语法的一个子集。为了增强其表达能力，业务逻辑语法又在上面子集的基础上进行了扩展，增加了 SQL 语句、输入协议解析/输出协议和调用外部 C 程序函数语法。以上逻辑由 Lex 和 Yacc 进行分析，形成变量表、SQL 语句表、执行动作表。

业务逻辑规则由各种语句规则和表达式规则构成，另外增加了 SQL 语句描述规则。

业务逻辑数据存贮设计包含以下个方面：变量表、SQL 语句、输入消息、输出消息、表达式、语句。

中间语言由两个部分组成：执行动作和数据。执行动作由一些四元式（一个运算符和至多三个操作数）组成。数据也就是符号表。

2.2　业务执行模块

提供业务逻辑的运行环境，提供业务逻辑的动态加载、卸载等操作，与其它实体进行交互，提供数据库的接口功能。

业务执行程序的业务逻辑数据全部来自业务解释程序。业务解释程序与业务执行程序之间的接口为某种格式的数据文件，抽象后的接口数据格式如图 3 所示。

名称	类型	变量表	STRUCT
优先级	INT	输入参数个数	INT
版本标志	INT	输入参数表	STRUCT
事件类型	INT	输出参数个数	INT
事件号	INT	输出参数表	STRUCT
业务键	INT	SQL 语句个数	INT
变量个数	INT	SQL 语句	STRUCT

图 3　接口数据格式

读取并初始化业务逻辑解释程序送来的业务数据文件，接受外部应用程序发送来的业务请求，根据业务请求中的数据计算请求关键字，根据关键字获得此请求对应的业务数据（执行动作、符号表、SQL 语句表、输入输出协议描述）存放地址。根据业务脚本执行业务逻辑，回送响应消息。支持业务数据的动态加载、删除、更新。

加载业务解释程序的业务数据时，应该根据数据动态分配内存，使得业务数据的存放缓冲区可以随意指定，以避免根据宏定义预设方式带来的缓冲区不足引起程序另编译问题。

更改业务时，应将新更改的业务数据存放到另外的缓冲区，当新业务请求进来时，根据新缓冲区中的业务数据处理业务，并及时监视系统，当所有在旧业务数据上进行的业务处理完毕后，释放该缓冲区。

3　应用实例

上述方案，在中国网通 SCDMA（大灵通）系统中得到了实现。在 SCDMA 网络系统的建设过程中，网通为了吸引更多用户使用 SCDMA 网络，要求我们实现多种新业务（如灵通呼、虚拟同线等等），但由于交换系统是一个复杂而庞大的分布式系统，业务、交换、无线网和一些相关的智能设备之间相互依存、协同工作，因此业务的修改往往比较麻烦，稍不留意就会产生意想不到的结果。曾经一度我们整天忙着不停地修改相关的程序，结果问题也出现了不少，软件质量不能得到有效的保证，不能按期交付用户使用。采用了业务逻辑编辑器模式后，经过在 SCDMA 归属位置寄存器（HLR）和核心网的实际使用中，取得了不错的效果。采用这种模式进行新业务的开发比较简单，只需按照预定的业务流程用设计的业务逻辑语法进行精确的描述，然后解释程序自动生成相关的中间语言，最后交由业务执行程序根据业务规则下发到相关的物理通信实体进行执行即可，而且不会影响旧的业务，开发周期得到了缩短，软件的质量也得到了很好的保证，受到了用户的肯定。

结束语　随着应用环境的发展和变化，用户对业务的需求也不断发生着变化。由于业务逻辑与使用这些逻辑的应用程序之间的依赖性较强，应用软件开发人员不得不经常对软件进行修改，维护量大且开发成本很高。本文提出了一种有

（下转第 267 页）

下公式来度量候选构件对整个软件系统的影响：

$$\text{BackupQuality}=(1-\frac{CP(bc)-CP(kc)}{CP(kc)})\times 100\%$$

其中，bc 表示候选构件，而 kc 表示关键构件。

很显然，如果 BackupQuality 过大的话，我们将重新考虑做 DAR，所以我们在为关键构件进行候选构件的选取的时候必须满足这样的条件：

$BC=\{bc|(I_{kc}\Rightarrow I_{bc})\wedge(I_{kc}\Delta O_{kc}\Rightarrow O_{kc})$OR $satisfy$ (BackupQuality)$\}$

5 提取构件组

通过前面构件之间的度量，我们可以得出单个构件和所有其他构件之间的关联 CP，很显然我们会得到一个 CP 值最大的构件，我们可以关联度上认为该构件为关键构件，也就是说这个构件于其他构件发生的关系最多，一旦此构件出现问题，我们软件遭受致命性损害的可能性也就最大。保证这个构件运行良好，是我们在规避以后软件开发中发生重大风险的很好的手段。

在我们度量构件之间关联度的同时，我们可以看到一张以构件为点，构件之间的关联度为边的构件关联图的形成。

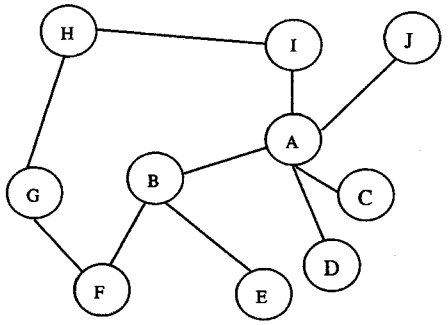

图 3 构件关联图

在图 3 中，我们可以看到边是由之前所度量出来的 BCP 来构成的，比如 A 和 B 的边，也就是 BCP(A,B)，那么在这样一个构造出来的图中，我们一定就可以找到关键构件，利用该关键构件我们可以自动构造出一个构件组，这个构件组可以在软件 SQA 审核之后重用，具体这个构件组识别的算法如下：

算法描述：

```
CS Identification (Graph g)
begin
C kc=getMax(g);
//C 表示构件，以上是获取关键构件
CS cs=getInitialCSByRadom(kc);
//从 kc 的关联构件随机选取一个构件作为初始构件组
float sumcs=sumExtenal(cs);
//获取 cs 中与其他构件关联总和
CS cstemp=getComponentByRadom(cs);
//获取与 cs 相关联的一个构件，将其加入构件组临时集合 cstemp
float sumcstemp=sumExtenal(cstemp);
//计算 cstemp 与其他构件关联的总和，将其存入一个变量 sumcstemp
While(sumcstemp<sumcs)
begin
cs=cstemp;
sumcs=sumExtenal(cs);
CS cstemp=getComponentByRadom(cs);
sumcstemp=sumExtenal(cstemp);
end
return cs;
end
```

通过以上的算法我们可以找出构件组 CS，通过 SQA 的审核之后，这个构件组就可以拿来进行重用了，当然可以在原图中去处 CS 集合的成员，继续寻找新的构件组。

同样，我们可以度量构件组的关联度 S，如果 S 越小，我们就认为该构件组越稳定。

结论 在基于构件的软件开发应用越来越广泛的情况下，软件的稳定性更是应该得到十分的注意。通过本文的方法，我们不仅可以看出软件的稳定性，对软件的风险分析和制定有很好的辅助作用，同时我们还可以看出软件对哪个模块最有关联，最具有哪种类型的关联，在软件设计时，需要对哪个模块进行良好分析和设计。当软件产生问题的时候，可以迅速对问题模块进行替换。运用该方法可以找出合理可重用的构件组，可以对现有的风险制定给予补充和支持，更有力地辅助项目过程的实施。

参 考 文 献

1 Chidamber R, Kemerer F. A Metrics Suite for Object-Oriented Design. IEEE Trans. Softw. Eng., 1994,20(6):476～479

2 Harrison R,Counsell SJ, Nithi RV. An Evaluation of the MOOD Set of Object-Oriented Software Metrics[J]. IEEE Trans. Softw. Eng.,1998,24(6):491～496

3 杨芙清，梅宏，李克勤. 软件复用与软件构件技术. 电子学报，1999,27(2):68～75

4 Szyperski C. Component Software: Beyond Object Oriented Programming. Harlow: Addison-Wesley Longman, Inc., 1997. 1～70

5 Washizaki H, Yamamoto H, Fukazawa Y. A Metrics Suite for Measuring Reusability of Software Components 1530-1435/03 2003 IEEE

6 Denning A. ActiveX Controls Inside Out. Microsoft Press, 1997

7 Hamilton G. JavaBeans Specification 1.01, Sun Microsystems, 1997

8 Penix J, Alexander P. Using Formal Speci_cations for Component Retrieval and Reuse 1060-3425/98 1998 IEEE

(上接第 263 页)

效的方法，大大提高了软件应用的灵活性和可扩展性。通过在某通信设备制造商归属位置寄存器开发中的应用，较好地满足了设计的需求，缩短了开发业务的周期，具有较好的灵活性，取得了不错的应用价值。

参 考 文 献

1 Aho, A V, Sethi R. Compilers: Principles, Techniques, And Tools[J]. Addison-Wesley, Reading, MA, 1986

2 Lee, Peter. Topics in Advanced Language Implementation, MIT Press, Cambridge, MA, 1991

3 Aho A V, Sethi R, Ullman J D. 李建中译. 编译原理[M]. 北京：机械工业出版社，2005

4 Levine J R, Mason T, Brown D. 杨作梅译. lex 与 yacc(第二版)[M]. 北京：机械工业出版社，2003

5 IBM 中国技术支持. 电子商务运行环境模型[EB/OL]. http://www-900.ibm.com/cn/support/guide/blueebook4_1.shtml. 2005-10-8

6 郑先容，黄杰. 基于 CORBA 构件模型的编译器的研究与实现[J]. 计算机应用，2005(25):91～92