第32卷 第16期 计 算 机 工 程 2006年8月
Vol.32 № 16 Computer Engineering August 2006
·软件技术与数据库· 文章编号：1000—3428(2006)16—0066—04 文献标识码：A 中图分类号：TP311

基于COM原理的进程结构图呈现技术

史红军，李青山，陈 平，许永峰，夏 辉，李 鹏
(西安电子科技大学软件工程研究所，西安 710071)

摘 要：对于具有分布、并发特征的面向对象系统，抽取和呈现进程间的创建和通信关系，可以从系统的高层体系结构方面帮助用户理解系统。在充分分析Rose扩充技术原理的基础上，给出了反应进程间创建和通信关系的进程结构图的一种呈现技术，通过扩充Rose的版型，将进程结构图在Rose中直观地呈现出来。最后给出了一个分布、并发的软件系统，作为测试案例，对该呈现技术的有效性进行了验证。
关键词：逆向工程；程序理解；进程；进程间通信；COM

Representation Technology of Process Structure Graph Based on COM

SHI Hongjun, LI Qingshan, CHEN Ping, XU Yongfeng, XIA Hui, LI Peng
(Software Engineering Institute, Xidian University, Xi'an 710071)

【Abstract】 As for object-oriented system with characteristics of distribution and concurrency, extraction and representation of creating relations and communication relations among processes can help users to comprehend target system in the view of high-level architecture. After the analysis of principles of Rational Rose extensibility, a kind of representation technology of process structure graph (PSG), which can reflect creating relations and communication relations among processes, is presented in this paper. PSG is directly represented in the environment of Rational Rose by extending stereotypes of Rational Rose. Finally, a software system with features of distribution and concurrency is used as testing case to verify the validity of this representation technology.
【Key words】 Reverse engineering; Program comprehension; Process; Inter-process communication; COM

进程是分布、并发系统中最核心的概念之一，进程间创建关系和进程间通信结构对于用户充分理解系统整体行为特征有非常重要的帮助。进程间的交互关系可以体现系统执行过程中系统结构的一个侧面，它强调系统行为执行的逻辑。进程交互关系信息的提取和抽象，可以从系统的高层体系结构方面帮助用户理解系统，因此在具有分布、并发特征的面向对象系统逆向工程中，抽象进程交互关系具有重要的理论和实际意义。

为了能够将进程结构图直观地呈现出来，需要对Rose进行扩充。Rose提供REI接口，可以方便地将进程结构图呈现到Rose开发环境中，从而将抽象的概念用图形化的符号直观地表示出来，有利于用户对目标系统的理解。同时，在Rose环境中，用户可以对逆向结果进行再加工，抽象出公共部分用于其它的项目，提高软件的复用率。

1 进程结构图

1.1 进程间依赖关系

在具有分布、并发特征的系统中，各进程很少相互独立，它们或多或少有某种关系。进程间主要有两大类关系：创建关系和通信关系。创建关系指一个进程创建另外一个进程，前者称为父进程，后者称为子进程。通信关系指进程之间为达到一定目的而进行数据交换的方式，可以有多种类型的通信手段。

本文主要考虑Unix平台上软件系统进程结构图的呈现技术。在Linux/Unix中，常见的进程间通信方式有管道(FIFO)、消息队列(message queue)、信号量(semaphore)、共享内存(shared memory)、TCP/UDP等[1]。

1.2 进程结构图的定义

为了方便论述进程结构图的呈现，这里给出几个与进程结构图相关的定义[2]。

定义1 进程 是程序的一次运行，它可以和其它的进程并发地执行，用P来表示进程。

定义2 进程模块 对进程多次运行的抽象，对应于程序特定的一段代码。在由相同目标系统得到的进程结构图中，每一个进程模块都有一个标识唯一确定，这里用PM(Process Module)表示进程模块。

定义3 资源模块 对进程间通信方式的抽象，不同的进程通信方式对应相应的资源模块。在由相同目标系统得到的进程结构图中，每一个资源模块都有一个标识唯一确定，这里用RM(Resource Module)表示资源模块。

定义4 进程结构图PSG(Process Structure Graph) 在目标系统中，进程结构图定义为三元组，PSG = (S，T，R)。其中：

S＝{ PM | PM ∈ W }；W是目标系统中所有进程模块的集合。

T＝{ RM | RM ∈ V}；V是目标系统中资源模块的集合。

基金项目：国家自然科学基金资助项目(60473063)；国家教育部博士点基金资助项目(20030701009)；“十五”国防预研项目
作者简介：史红军(1980—)，男，硕士生，主研方向：逆向工程，面向对象技术和软件体系结构；李青山，博士、副教授；陈 平，教授、博导；许永峰、夏 辉、李 鹏，硕士生
收稿日期：2005-11-28 **E-mail：**shi6255@163.com

R＝{ r | r ∈ H ∨ μ ∈ K }；R 表示进程模块之间的关系，H＝{ h | h∈ S→T}是进程模块间的通信关系，表示进程模块和进程模块通过资源模块进行通信，K={ k| k∈ S→S }是进程间的创建关系。

从定义 4 可以看出，进程结构图是一张反映系统进程间创建关系和进程通过资源模块进行通信的关系图，而且进程结构图既吸收了进程间的依赖关系是目标系统的动态行为特征的优点，又剔除了进程交互随目标系统的运行不同而变化的缺点。

2 Rose 扩充的理论基础

Rational Rose 是由 Rational 公司推出的一套可视化建模工具，包括 Rose98、Rose98i、Rose2000、Rose2001 等。这套可视化建模工具支持以 UML 为基础，遵循 RUP 过程框架的软件开发过程。

2.1 Rose 扩充与 COM 原理

从建模工具的可扩展性方面来看，可以把 Rose 看作是一个 UML 开发平台。如果把 UML 看作是独立于开发过程、用于软件开发周期中模型描述的一种建模语言，那么也可以认为 Rose 是独立于处理过程的，是特定于工程的开发工具集合中的一个简单工具。为了支持用户选择的开发过程，或者为了支持与其它开发工具之间的数据交换，从而满足用户二次开发需要，Rose 自然应该具有可扩充性和可适应性。

Rose的扩充性不仅仅是一个脚本语言、一个COM界面或一个模板文件，Rose的扩充性是Rose一些特征的集合，这些特征允许用户改变Rose的外观形象，扩展模型信息，改变Rose的行为以及与其它应用程序交换数据和模型信息，最后，可以将这些扩充打包作为Rose Add-in通过Rose Add-in Manager的管理工具无缝嵌入到Rose中[3]。

COM是由Microsoft提出的组件标准，它不仅定义了组件程序之间进行交互的标准，并且也提供了组件程序运行所需的环境[3]。COM提供了编写组件的一个标准方法。遵循COM标准的组件可以被组合起来以形成应用程序。COM接口是一组逻辑上相互关联的操作，这些操作定义了某种行为，即这组操作的规范，而非特定的实现，实质是接口代表了接口调用者和实现者之间的一种约定。当两个应用程序通信时，一个应用程序初始化二者之间的关系，这个应用程序被称为COM客户机，或COM控制器，而另一个应用程序被称为COM服务器，COM客户机访问由COM服务器提供的服务。

在 Rose 的扩充环境中，简单地说 COM 是一个通信机制，应用程序通过它可以获得对另一个应用程序中的对象的访问，获得对可重用组件的访问。COM 对象是一个应用程序暴露出来的可以被其它应用程序共享的对象。事实上，Rose 是以 COM 组件技术为基础，其核心是一组基于 COM 对象的元模型。Rose 暴露出来的 REI 对象(如模型、类、方法、属性等)都是 COM 对象。Rose 的扩充接口(REI)是按照 COM 的标准制定的接口，所以本质上可以将 Rose REI 看作为 Rose 中提供访问模型信息的一个 COM 服务器。

2.2 REI 体系结构

Rose REI 模型本质上是 Rose 模型的元模型，它将定义和控制 Rose 应用程序及其所有功能的包(package)、类、属性、方法展示出来。用户可以通过 RoseScript 或支持 COM 的外部语言访问这些模型信息。Rose 还提供了类型库支持那些只能早绑定的 COM 客户机(如 Java)。但无论哪种情况，都得使用 Rose Extensibility Model 中定义的 REI 调用。在 Rose 中有 Add-in 的管理工具(Add-In Manager)，可以把编写好的 RoseScript 和 COM 组件(dll 形式)在 Rational Rose 中添加或卸载。图 1 展示了 Rose 中的核心组件和 REI 中的组件以及二者之间的关系。

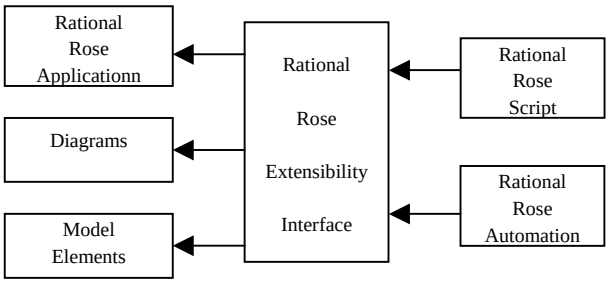

图 1 Rose 中核心组件和 REI 中的组件的关系

进行Rose扩充，通常应该定义一个新的Add-in来容纳这个用户的扩充，为了在项目XDRE[5]中对Rose进行扩充，必须定义一个COM服务器来响应Rose的各种请求。如前面讨论的一样，Add-in简单地说就是扩充Rose的一个封装机制，实际的扩充(如菜单、版型、属性、代码生成器等)都可以在COM服务器、各种ASCII码文件或RoseScript文件中找到。

在本项目[5,6]中，使用 Visual C++ 6.0 定义自己的Add-in，详情请参阅文献[7]。

3 进程结构图的呈现

3.1 进程结构图的信息文件格式设计

在该项目的前期工作中，通过对源程序进行植入[8]，收集动态信息。利用反射植入机制[9,10]，可以收集到进程ID和进程间交互的剧情。更进一步，通过扩充开放编译器[9]，就能收集到进程ID与实例化出该进程的程序代码间的映射关系，从而抽取出进程模块单元。

待呈现的进程结构图的文本信息采用 XML 文件格式。目前几乎所有流行的操作系统平台都支持 XML 文件格式，方便系统的跨平台操作。进程结构图的呈现抽象与进程结构图的抽象部分分离，这样，产生的进程结构图信息可以为其它的应用程序或 Case 工具共享。各个子系统的耦合性也已经降低到最小。进程结构图 XML 文件格式如下所示：

```
<xs:element name="TCP_UDP">
        <xs:complexType>
            <xs:sequence>
            <xs:element ref="identifer"
maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="Create">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="parent_process"
maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="Message_Queue">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="identifer"
maxOccurs="unbounded"/>
```

```
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:element name="Semaphore">
            <xs:complexType>
                <xs:sequence>
                    <xs:element ref="identifer"
maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:complexType>
    </xs:element>
```

从以上代码可以看出，对于各资源模块如 TCP/UDP 资源、消息队列、信号量等，它们的相关进程，都被组织在各资源的子目录下。对于具有创建关系的进程，则将父子进程按创建关系罗列。

3.2 利用 REI 呈现进程结构图

进程结构图的产生是根据植入子系统和动态信息过滤子系统产生的动态信息文件，生成进程结构图。进程结构图的层次化是在进程结构图中添加反映进程功能的类结构信息，这样从进程结构图中，可以直观地展现实现此进程功能的类的信息，包括类的属性和类的方法。

使用形象、直观的图形符号将进程结构图清晰地、准确地展现出来，可以使用户从呆板的文字形式的进程动态信息中解放出来，直接以图形的方式把握系统进程间的创建关系和进程间通信关系。

为了方便地呈现进程结构图，需要对 Rose 的 Stereotype 进行扩充，填加所需要的 Stereotype，待添加的模型元素如图 2 所示。这些版型用于在 Rose 中作为资源模块，分别对应表示进程模块中的通信方式如：TCP/UDP，共享内存，管道，消息队列，信号量等。

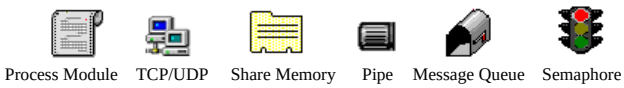

图 2 进程模块和资源模块的标记符号

对于进程模块的创建关系，使用带箭头的虚线表示。箭头从父进程(创建者进程)，指向子进程(被创建的进程)。例如，进程 process1 创建进程 process2，在进程结构图中的表示如图 3 所示。

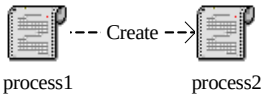

图 3 进程结构图的创建关系

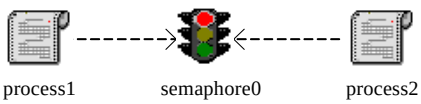

图 4 进程模块之间的通信关系

对于进程模块间的通信关系，使用带箭头的虚线，将多个进程模块同通信资源节点连接，表明这几个进程模块通过此通信资源确定的通信方式进行进程间通信。例如，进程 process1 与进程 process2 通过信号量 semaphore0 进行通信，它们在 Rose 中的进程结构图表现形式如图 4 所示。

3.3 进程结构图的呈现算法

输入 进程结构图 xml 文件。

输出 进程结构图 Rose 模型。

步骤如下：

(1)打开 Rose 模型，寻找 Logical View 的 Main 视图，在当前视图添加 Process Structure Diagram(PSD)包。

(2)载入进程结构图 XML 文件，存储到 PSG 结构中。

遍历结构 PSG 的 lstProcessModules，依次向 PSD 包中加入节点模块包，并设置模块的版型(进程模块或资源模块)，如果是进程模块，向 Node Module 包中添加类切片。

(3)遍历进程模块对应的 lstCls，对于每一个类切片 ca，向 Node Module 包中添加类 cls，设置类的版型为<<Aspect>>，在依次遍历 ca 的 lstAttributes 和 lstMethods，向 cls 类中分别添加属性和方法。

(4)遍历结构 PSG 的 lstArc，依次向 PSD 包添加依赖关系。

4 案例测试分析

本节通过一个经过实践考验的案例对本文工作进行系统验证，使用的测试用例是客户服务中心系统(iCall Center)的核心子系统，iCall Center 是基于 Linux/Unix 操作系统的大型分布、并发面向对象系统，实现机内和机间的进程通信。目前已成功地运行于多家邮政系统。用于测试的子系统封装底层通信功能，为整个 iCall Center 的应用提供通信支撑。

4.1 获取动态信息文件

通过在测试用例源程序中新的位置植入信息，获取到的进程结构图信息文件如下所示(由于篇幅的限制，这里只展示了部分的信息片段)。

```
<Semaphore>
<identifer id="425991">
<process name="./cld"/>
<process name="sei2100"/>
<process name="sei3000"/>
<process name="sei4000"/>
</identifer>
</Semaphore>
<Share_Memory>
<identifer id="491525">
  <process name="./cld"/>
  <process name="sei2100"/>
  <process name="sei3000"/>
  <process name="sei4000"/>
</identifer>
</Share_Memory>
<Create>
  <parent_process name="./cld">
    <child_process name
="./cld_../../src/ProcSubComponent.C_line3473"/>
</parent_process>
</Create>
```

从以上代码可以看出，对各资源模块按 ID 的不同来区分，通过特定资源进行通信的各进程分别记录于该资源的子目录下。对于具有创建关系的进程，则按父子关系顺序组织。按照这种组织格式，进程间的通信关系都已经获得，进程间的创建关系清晰明了。

4.2 进程结构图的呈现结果

对目标系统(iCall Center)进行全植入，运行目标系统，共运行 sei2100、sei3000、sei4000、sei4900、sei1000 和 sei1100 6 种进程组。其逆向得到的进程结构图(部分)如图 5 所示。

恢复进程结构图的目的是帮助用户从进程及进程间依赖关系的角度理解目标系统的高层软件体系结构，因此进程结

构图恢复的结果将直接影响用户对目标系统的理解。

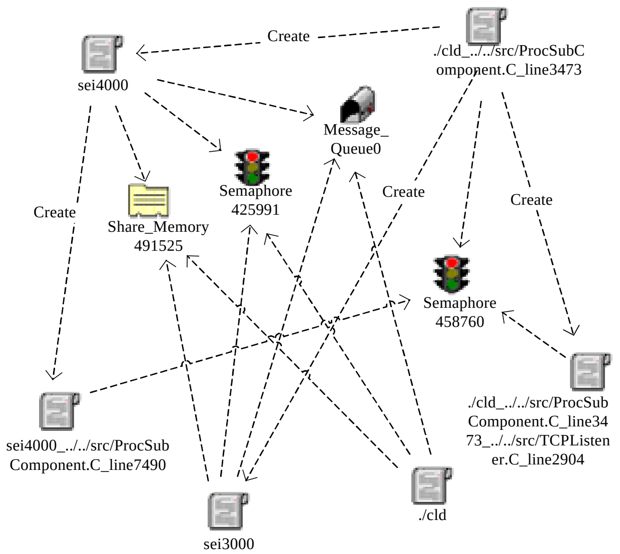

图 5 全植入后得到的进程结构图呈现结果(部分)

评价进程结构图恢复的结果可以从进程结构图是否真正体现实际目标系统的进程间创建关系和进程间通信关系来衡量。从图 5 中可以看出，进程结构图已经把进程的创建关系和通信关系明确地展现出来。从抽象的层面上讲，进程结构图是进程的抽象，使用进程模块的概念来表述进程间的创建和通信关系，以利于用户理解目标系统。

5 结束语

前述各章节讨论了逆向工程中将进程结构图在 Rose 中进行呈现的工作，并使用一个具有相当规模的分布、并发的系统对进程结构图的呈现进行了验证，从试验的结果和表现形式上看，达到了帮助用户从系统的高层体系架构角度理解程序的目的。本文主要工作如下：

(1)利用 UML 和 Rose 扩充机制扩充了 UML 基本模型元素的语法和语义。

(2)将进程结构图利用扩充后的 UML 语法和语义将进程结构图及进程间通信关系在 Rose 中直观地呈现出来。

当然，要使得呈现的进程结构图信息更加方便用户理解目标系统，还有一些待改善的细节部分。在后续工作中，一个可以考虑的完善方向，就是可以考虑呈现进程结构图中更为准确的通信方式。

单方向通信方式(如半双工管道、消息队列)只允许数据的单向流动，而在进程结构图中，仅仅笼统地定义进程模块通过这些通信方式进行通信，并没有体现通信中数据流动的方向，如图 6 所示。

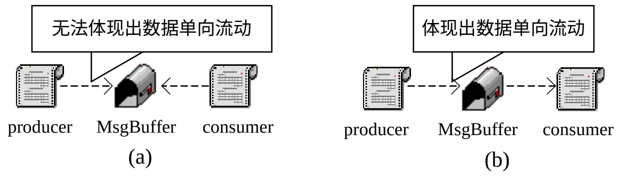

图 6 典型的生产者和消费者问题

图 6(a)语义是生产者和消费者通过消息队列进行进程间消息传递，至于是 producer 或 consumer 向 MsgBuffer 中写入消息，还是 producer 或 consumer 从 MsgBuffer 中读出消息，图中都没有定义，其语义是模糊的。图 6(b)语义是 producer 向消息缓冲区放入消息，consumer 从消息缓冲区取走消息，可以用依赖关系的指向表明数据是单向流动的，这样的进程结构图在语义上更加符合生产者和消费者问题的解空间。显然图 6(b)表现形式更加能够体现出系统的设计特点，对用户理解程序也更有帮助。

当然，该问题的解决，有赖于动态信息的进一步改动。解决方案为在程序运行期间，对于这些单方向数据流动的通信方式，在获取读写信息时，使用不同类型的动态信息，在进程结构图的构造中分别处理这些动态信息。这样问题就可以完全解决。

参考文献

1 刘宗田, 孙志勇, 秦宗贵. 实用 Unix 编程[M]. 北京: 机械工业出版社, 1999.

2 李 鹏. 逆向工程高层软件体系结构的恢复[D]. 西安: 西安电子科技大学, 2003.

3 王玉英. 一种将逆向工程工具无缝嵌入 Rose 的途径[D]. 西安: 西安电子科技大学, 2003.

4 潘爱民. COM 原理及应用[M]. 北京: 清华大学出版社, 1999.

5 陈 平. C3I 系统应用软件逆向工程开发工具研究任务申请书[Z]. 本项目课题组, 2001.

6 李青山, 陈 平, 王 伟. 一种基于反射和开放编译的 C++植入机制[J]. 系统工程与电子技术, 2003, 25(7): 851-855.

7 夏 辉, 陈 平. 基于 COM 原理的 Rose 扩充技术[J]. 微机发展, 2005, 15(5).

8 Rational Software Corp. Online Help Rational Rose[Z]. 2000.

9 陈 平. 反射结构和对象标识的研究[D]. 西安: 西安电子科技大学, 1991.

10 李青山. 面向对象软件的动态模型设计与体系结构抽象[D]. 西安: 西安电子科技大学, 2003.

~~~~~~

(上接第 54 页)

经过优化，我们的实现性能在各个方面超过了 WTK。

5 结论

本文主要介绍了一个完整的 J2ME MIDP 中 RMS 包的实现。包括 RMS 包的概述，两种设计方法以及它们的利弊，设计实现的细节，性能调试与提升，以及最终与 WTK 性能比较的结果。

参考文献

1 Sun Microsystems. Mobile Information Device Profile (JSR-118) Specification[Z]. http://jcp.org/aboutJava/communityprocess/final/jsr118/index.html.

2 Richard Taylor. Taylor Benchmark Home Page[Z]. http://www.poqit.com/midp/bench/index.html.

3 Piroumain V. Wireless J2ME Platform Programming[M]. US: Prentice Hall, 2002.

4 Garcia-Molina H. Ullman J D, Widom J. 数据库系统实现[M]. 北京: 机械工业出版社, 2004.
~~~~~~