一种基于通道的层次布图算法的研究和实现*

欧胜高，刘　超
(北京航空航天大学 软件工程研究所，北京 100083)

摘　要：针对软件分析与测试工具中逆向建模出现的一些复杂情况,如聚合关系和非结构化的关系等。讨论了算法的步骤和关键技术,并给出了相对应的应用实例。该算法具有层次清晰、对称性强、交叉线少、可视化效果好等特点。

关键词：软件测试；软件理解；布图算法
中图法分类号：TP301.6　　文献标识码：A　　文章编号：1001-3695(2004)11-0173-02

Research and Implementation on Channel-based Hiera Chical Layout Algorithm

OU Sheng-gao, LIU Chao
(Institute of Software Engineering, Beijing University of Aeronautics & Astronautics, Beijing 100083, China)

Abstract: According to the complex topology of the software analysis and testing tools based white box, such as association and aggregation in the class diagram etc, the key points of this algorithm is discussed in details. This algorithm has been applied to drawing diagram in SafePro/Java, a kind of testing tools for Java program and SafePro/C++, a kind of testing tools for C++ program. It is proved that graph implemented by this algorithm is Arrangement in focus, Symmetry perfect, seldom cross line and impact visual.

Key words: Software Testing; Software Comprehend; Layout Algorithm

随着软件规模增长和体系结构的复杂化,针对程序源代码的分析、理解,以及测试、维护等工作也随之变得更加困难。因此,在基于白盒方法的软件分析与测试工具中,通过分析程序源代码来逆向生成各种程序结构模型已经成为一项基本功能。然而,这些模型图的自动布图算法直接影响其平面布局的合理性和清晰程度,进而影响用户对程序的分析和理解。勿容置疑,一个好的布图算法将显著增强模型的可视化效果,提高其可读性,从而节省项目进行再开发和维护的成本,提高效率。

目前,布图算法的研究已有一些报道[1,2]。在北京航空航天大学软件工程研究所开发的 SafePro 系列测试工具中,采用一种以方法连接度(扇入/扇出系数和)为特征的广义张量平衡算法绘制出方法间调用关系图[3],采用以类间继承关系为主序的层次型的布图算法等。这些算法虽然解决了复杂模型的层次化布图,以及层间节点的关系连线通过预置通道进行合理布线等基本问题,但是对于一些复杂情况,如类图模型中除了层次化的继承关系以外,还存在复杂的聚合关系和非结构化的关联关系,依然存在一个共性的问题,即对于大型软件,特别是面向对象软件,由于同层图元之间连线过多而导致部分通道拥塞,连线交叉过多,结构不清晰,给用户理解程序结构带来不便。本文参考构造型布局算法、广义力向量松弛布局法、最小分割布局算法、最优通道布线算法、贪婪通道布线算法的思想,通过限定在逆向生成的类图模型中同层图元间无连线等约束,提出了一种基于层次和预定通道的布图算法。本算法布出的图具有层次清晰、对称性强、交叉线少等特点,有效地增强了模型的可视化效果,提高了可读性。

收稿日期：2003-11-15；修返日期：2004-08-27
基金项目：国家“863”计划资助项目(2001AA113100)

1　算法描述

1.1　基本术语

从程序逆向分析的角度讲,一个面向对象程序的模型图是指对该程序中所定义的基本单元及其相互之间的关系的一种图形化表示,如类图是指对该程序中所定义的类及其相互之间的继承关系、组成关系和关联关系等的一种图形化表示。实际上,通过对程序源代码的分析,可以逆向生成多种不同类型的模型图,它们分别从不同侧面表示了程序的某种结构特征。

(1) **节点**:基本图元,代表面向对象程序设计语言中的实体,如类、函数、文件、宏、包等。

(2) **相关度**:所有与该节点之间存在关系的节点的个数,记作 Div(n)。

(3) **边**:节点与节点之间存在的具体关系,如类的继承、关联、聚集,文件的包含等。

(4) **模型图**:$M=(N, L, R_i)$,其中 N 是节点的集合,L 是边的集合,R_i 记录第 i 层所有节点的层次和列号信息。

(5) Div(node, N):表示节点 node 与集合 N 中所有节点的相关度。

(6) U_i 表示 $R_1+\cdots+R_i$ 节点的集合。

1.2　算法基本思想

对一个模型图 $M=(N, L, R_i)$:

(1) 当新的一层开始时,统计所有节点 Div 值。得到 Div 最小值的节点集合 S,从集合 S 中任取一个节点,当作这层的第一个节点 First,加入到这层的集合 R_i;然后依次遍历 S 的剩

余节点, 如果 Div(S_j, R_i) 为 0(S_j 与 R_i 中任意一个节点都不存在边的关系, 保证同一层中任意两个节点相关度为 0) , 则将此节点加入到 R_i 中, 否则将节点加入到集合 P 中, 意味着, 当布置下一层时, 优先考虑 P 中的节点, 从而使存在边的关系的节点尽量在相邻层, 布局更合理。转到(3) 。

(2) 如果 P 不为空, 遍历 P 中的节点, 得到 Div(P_j, U_i) 的最大值集合 Z; 如果 P 为空则遍历 W 中的节点, 得到 Div(W_j, A) 的最大值节点集合 Z。从 Z 中, 任取一个节点当作这层的第一个节点 First, 加入到 R_i 中, 依次遍历 W 中剩余节点, 得到 Div(W_j, U_i) 的最大值节点集合 B, 遍历 B 中的节点, 如果 Div (B_j, R_i) = 0, 则将 B_j 加入到 R_i 中, 否则将 B_j 加入到 P 中。T_i 中节点列号的分配遵循对称性和均衡性原则。具体算法参考 1. 3。转到(3)

(3) 遍历 N 中其他节点, 如果 Div(N_j, R_i) > 0, 则将 N_j 加入到集合 W 中(W 表示下一层备选节点集合) 。从 N 中删除 R。如果 W 为空, 则按(1) , 否则按(2) 进行布置下一层; 如果 N 为空, 算法终止。

1. 3 列号的计算

上一节主要解决模型中节点所在层的问题, 而并没有处理每层中列号分配的问题。列号的计算是否合理, 对于整个布图效果产生直接的影响。下面给出列号算法。

对于第 i 层如果第 i 层中与 U_{i-1} 中所有节点的关联度为 0, 则按(1) 处理, 否则按(2) 处理。

(1) 遍历 T_i 中的节点, 列号为 Div($T_{i(j)}$ * j + 1(即该节点的关联度* 层中序号 + 1) 。

(2) 依次遍历 T_i 中的节点, 得到 Div($T_{i(j)}$, $U_{(i-1)}$) 的最大值集合 B, 遍历 B 中的每个元素 B_j, 从 U_{i-1} 存在边的节点的最大列号值 Max 与最小列号值 Min, 则 B_i 首选的列号为 cur = Avg(min, max) , 分配时存在以下四种情况: ①如果首选未被占用, 则将此位置分配给当前 B_j; ②如果首选已分配给某节点, 则考虑首先左右位置, 如未被占用, 则分配给 B_j; ③如果上面位置都被占用, 则遍历 1 至 i - 1 层中的这三个位置, 如果 Div(B_j, T_h) = 0 且第 h 层的这三个位置中有未分配的, 则分配给 B_j; ④如果上面三个条件都未满足, 则给 B_j 分配第 i 层离首选最近的未分配的列号。

1. 4 坐标的计算

最终, 模型图绘制在屏幕上, 要计算出节点的坐标以及线的坐标, 尤其是线的坐标计算是否合理, 对布图效果会产生直接的影响。

(1) 计算节点的坐标

假设: 起始位置记作(X, Y) , 层之间的距离记作 rowSpace, 列间距离记作 colSpace, 图元的宽度记作 NodeWidth, 图元的高度记作 NodeHeight, maxRowDistance 表示直线相连的最大列距(保证连线不经过图元) 。遍历集合 T, 得到每个节点所在的层数(记作 row) 和列数(记作 col) , 则节点对应的物理坐标为:

X ' = X + (row - 1) · (rowSpace + NodeWidth)

Y ' = Y + (col - 1) · (colSpace + NodeHeight)

(2) 布线(结合最优通道布线和贪婪布线算法) , 即节点之间的关系。对任意一条边, 可以得到其相关联的节点 NODE _ FROM, 与 NODE _ TO, 连线时有以下三种情况: NODE _ FROM 与 NODE _ TO 是相邻层节点, 且其列间距不超过 maxRowDistance, 则直接通过直线连接, 否则进行③; NODE _ FROM 与 NODE _ TO 在同列, 但不是相邻层, 则遍历两节点层之间的同一列的位置。如果在这些位置上都没有节点, 则直接相连, 否则进行③; 折线的坐标点的计算。约定 A(NODENAME) 表示节点 NODENAME 对应的属性值, 如 row(NODE _ FROM) 表示节点 NODE _ FROM 的行值。按下面两步实现:

①确定边的起始点坐标, 记作(startX, startY) , 以及在横和纵坐标的增长方向记作(dx, dy) , 分四种情况考虑:

如果 row(NODE _ FROM) > row(NODE _ TO)

则 dy = - 1 startY = Y(NODE _ FROM)

如果 row(NODE _ FROM) < = row(NODE _ TO)

则 dy = 1; starY = Y(NODE _ FROM) + NodeHeight)

如果 col(NODE _ FROM) > col(NODE _ TO)

则 dy = - 1 startX = X(NODE _ FROM)

如果 col(NODE _ FROM) < col(NODE _ TO)

则 dy = 1 startX = X(NODE _ FROM) + NodeWidth

通过起始点和目标点位置关系, 确定起始点坐标和增长方向。

②确定连线上关键点的坐标, 分两种情况考虑:

如果 abs(row(NODE _ FROM) - row(NODE _ TO)) > = maxRowDistance, 则关键点 X_1, Y_1 计算公式如下

X_1 = X + (abs(row(NODE _ FROM) - row(NODE _ TO)) - 1) · (rowSpace + NodeWidth)

Y_1 = Y + Random(colSpace)

如果 abs(col(NODE _ FROM) - col(NODE _ FROM)) > 1 则关键点 X_1, Y_1 计算公式如下

X_1 = X + Random(rowSpace)

Y_1 = Y + abs(col(NODE _ FROM) - col(NODE _ TO)) - 1) · (colSpace + NodeHeight)

通过产生 Random 随机数, 从而当一个通道有多根连线时, 线之间层次分明。

2 应用实例

以往的层次型布图算法中, 往往出现横向节点过多, 纵向节点不足的情况, 直接影响可视化效果。本算法通过增加存在边的关系的节点不能在同一层, 减少过多的横向节点分布, 增加纵向节点的深度, 并且布出的图交叉线大大减少。在整体上, 图布局具有很强的对称性。实验表明此算法可以满足 SafePro 系列工具各种布图的效果都较好, 如函数调用关系图、类图、文件包含关系图等。

当类之间缺少适当层次的继承关系时, 会导致正给模型缺少层次性, 使得大量节点被排在同一层中, 因而同层中的边过多, 可视性变差。新算法的布图效果图(略) 。

纵向层次明显增多, 整体图形对称而且均衡, 可视性好。

3 小结

本文提出了一种基于通道的层次型布图算法, 并且在 SafePro 测试工具中应用。并且对不同的布图算法进行了简单的比较。该算法布出的图形具有层次清晰、对称性强、交叉线少、可视化效果好特点。

参考文献:

[1] R Gansner, et al. Dag: A Program that Draws (下转第 177 页)

等到网络畅通时, 它会将发送队列的消息取出, 发送到对方机器的接收队列。如果发送成功, 清除自己的发送队列中相应消息, 整个发送过程结束。可见, 在发送消息的过程中, MSMQ 服务器起着决定性的作用。它协调发送队列和接收队列, 保证整个工作的圆满完成。除了 MSMQ 服务器, 还有 MSMQ 发送器和 MSMQ 触发器。如果熟悉 UML, 通过框图符号就可以知道这是两个 COM 组件。MSMQ 发送器是一个启动 MSMQ 服务, 向消息队列发送消息的功能模块。它指定接收队列的地址(消息队列的地址由 IP 地址或机器名及队列名确定), 与 MSMQ 服务器一起, 完成整个消息传递的过程。原有的应用系统, 将在合适的时机调用 MSMQ 发送器, 向其他系统发送消息。MSMQ 触发器同样是一个 COM 组件, 当指定的接收队列有消息到达时, MSMQ 触发器被触发, 完成赋予它的任务。通常, 它的任务之一就是将接收到的消息处理成自己需要的数据写入自己的数据库。至此, 关于 A 系统(集成后) 的部分说明完毕。至于图 2 右边的 B 系统, 情况完全一样。

Biztalk 交换平台部分: 图 2 中的左右两边各有一个 MSMQ 服务器, 用来与 A, B 两个子系统进行通信。关于 MSMQ 服务器前面我们在对 A 系统(集成后) 阐述时已经进行了介绍。这里我们重点说明框图中的中间部分, 即 Biztalk 服务器。Biztalk 的接收功能通常通过接收函数(Receive Function) 来实现。接收函数需要在 Biztalk 中配置, 配置时需要指定接收队列的地址及通道的名称。Biztalk 的最大特点就是用简单的配置代替编程。通道由三个部分组成: 模板一→映射→模板二。模板一和模板二分别是 A, B 两个子系统的模板。模板是一个 XML 文档, 它定义了消息的规范。通常为了方便, 我们可以按照子系统的数据库结构定义模板。模板定义以后, 各子系统发送的消息格式必须要遵守模板的规范。这样各子系统只需按照自己系统的特点定义自己的模板, 而不用花费精力去研究对方的数据库结构(而且, 一般各子系统的数据库结构也不愿意向别人公开)。不同模板之间的转换交给 Biztalk 服务器完成(同样, 模板映射需要在 Biztalk 中配置完成)。发送功能是通过在 Biztalk 中配置一个消息端口来实现的, 消息端口与通道相连接, 同时要指定接收消息队列的地址。Biztalk 服务器的整个活动过程是这样的: 当有消息到达时, 接收函数取走消息并传递到通道, 通道首先检查消息的格式, 是否与模板一符合, 如不符合停止传递, 并将错误信息写入错误日志, 将消息放入 Biztalk 的悬挂队列(Suspended Queue); 如符合, 将消息的格式由模板一的格式转换为模板二的格式, 然后再将转换后的消息发送到另一系统的消息队列中。

下面我们以 A 系统向 B 系统发送消息为例说明整个过程。原有 A 应用系统调用 MSMQ 发送器按照事先定义好的模板格式发送消息(消息为一个 XML 格式的字符串), 消息到达 Biztalk 数据交换平台的消息队列。接收函数接收消息并将消息传递到通道(Channel)。通道根据模板检验消息的数据格式, 然后对消息进行模板转换, 并将转换后的消息发送到 B 系统的接收队列。此时, B 系统的 MSMQ 触发器被触发, 触发器程序执行解析消息、入库及其他业务逻辑处理等操作, 并决定是否要向 A 系统回复消息。同样, B 系统向 A 系统发送消息的过程类似, 在此不再赘述。

限于篇幅, 我们只进行了两个系统的集成。对于更多系统的集成, 也只是量上的差别, 如在图 2 中再重复一个 C 系统(集成后) 或 D 系统(集成后)。无论增加多少, 原理都是一样的。当然, 在具体实施中, 还有很多技术问题, 读者可参考后面列出的参考文献, 它们可以帮助我们掌握相关技术细节。

7　结论

通过上面的讨论不难看出, 应用本方案构建企业应用集成(EAI), 不需要对原应用程序进行多少修改就可以做到无缝集成, 而且可以解决在企业应用集成中遇到的大部分问题。本方案聚集了 MSMQ, XML, Biztalk Server, COM 等当前先进技术, 在 EAI 中, 每一项关键技术都充分展现出了它的魅力, 起着至关重要的作用。当然, 在具体实施中, 上述技术也并非缺一不可, 我们可以根据实际情况对设计方案进行适当修改。比如, 当在 Internet 上进行 B2B 集成时, 我们可用 HTTP 传输取代 MSMQ 传输。再如, 对于企业内部集成, 当需要集成的系统彼此熟知对方的数据结构同时又无保密性可言时, 可以不用 Biztalk Server。但是, 通常情况下, 集成本文提到的各项技术设计出的方案是一个比较不错的选择。

参考文献:

[1] 林锦雀. 最新 XML 入门与应用[M]. 北京: 中国铁道出版社, 2001.

[2] 余英. Visual C++ COM 和 COM+ 篇[M]. 北京: 中国铁道出版社, 2001.

[3] obert J Oberg. 深入学习: COM+ 高级编程[M]. 北京: 电子工业出版社, 2001.

[4] tephen Mohr, Scott Woodgate. Biztalk 高级编程[M]. 北京: 清华大学出版社, 2002.

作者简介:

刘永壮(1976-), 男, 河北乐亭人, 硕士研究生, 主要研究方向为分布式应用及系统集成; 刘萍, 女, 北京人, 副教授, 主要研究方向为计算机仿真及应用。

(上接 174 页) Directed Graphs[J]. Software: Practice and Experience, 1988, 18(11): 1047-1062.

[2] P Luder, R Ernst, S Stille. An Approach to Automatic Display Layout Using Combinatorial Optimization Algorithms[J]. Software: Practice and Experience, 1995, 25(11): 1183-1202.

[3] 孙昌爱, 刘超, 金茂忠. 一种有效的软件结构图的布图算法[J]. 北京航空航天大学学报, 2000, 26(6): 706-708.

作者简介:

欧胜高, 硕士研究生, 主要从事软件工程、软件测试方面的研究; 刘超, 教授, 主要从事软件工程、软件测试和面向对象技术方面的研究。