构件集成算法的研究

蒲海红[1],侯秀萍[1],赵云峰[2]
(1.长春工业大学 计算机科学与工程学院,吉林 长春 130012;
2.装甲兵技术学院 控制工程系,吉林 长春 130117)

摘 要:为解决当前软件开发上存在的工作重复性及功能等问题,通过对构件的研究,提出了一种点对点的构件集成算法,即对外暴露的输入接口和输出接口的集成。具体做法为调用服务的构件的提供服务的接口和接受服务的构件的接受服务的接口相连,并传送服务和数据。这种构件集成方法只与构件的接口有关,而与构件内部功能实现无关,从而达到系统零耦合的目的。通过文中提出的算法,集成已有的或外购的经测试符合系统要求的构件,可以减少软件开发周期,使系统的功能更加完善。

关键词:ERP;构件;集成算法
中图分类号:TP301.6 **文献标识码**:A **文章编号**:1673-629X(2009)05-0075-04

Study of Component Integration Method

PU Hai-hong[1], HOU Xiu-ping[1], ZHAO Yun-feng[2]
(1. College of Computer Science and Engineering, Changchun University of Technology, Changchun 130012, China;
2. Department of Controlling Engineering, Armor Technique Institute of PLA, Changchun 130117, China)

Abstract: Through studying the component, presents point-to-point component integration method which integrates uncovered import interface and export inerface. Concretely describing as follows: the providing services interface(PSI) which is one interface of the transfering services component combines with the accepting services interface(ASI) which is one interface of the accepting services component, and PSI transmits services and data to ASI. This method is not related to the interior function of component, but just related to the interface of component. Consequently system could achieve zero-coupling through this component integration method.

Key words: ERP; component; integration_arithmetic

0 引 言

目前许多大型应用软件在开发方式上存在着很多问题,如开发量大、开发周期长、变化多等突出问题,开发人员在开发软件时,做了很多重复性的工作,如代码级的重复和类的重复,直接延长了开发周期。21世纪将是以知识经济和信息社会为特征的崭新时代,经济与科技向全球化与一体化方向发展日益明显,企业间既有竞争,又有结盟[1]。要求新一代软件系统必须具有动态易变性,能通过快速重组,快速响应市场需求的变化。这就要求采用新的有效的组织与控制决策策略,通过简单的控制规则来实现复杂软件系统的动态重组与运行控制。此外,软件更新或升级可以通过更换现有的部分新的组成部分[2]。构件是软件复用的基础[3],构件具有可组合、封装性、可独立布署、可重复使用和非针对具体情况的特点[4]。"系统构件化"成为系统快速重组的趋势[5,6]。文中根据一汽启明公司开发的ERP产品的实际情况和存在的问题,按照"架构=构件+交互"的思想,设计了适合ERP系统的构件集成算法。

收稿日期:2008-08-20
基金项目:吉林省科技发展计划重大科技攻关项目(吉科合字(20040305))
作者简介:蒲海红(1974-),女,吉林公主岭人,讲师,硕士研究生,研究方向为软件工程;侯秀萍,教授,硕士研究生导师,研究方向为软件工程。

1 相关定义

定义1 构件:

ComponentDefinition::={componentName, Require_interface, provide_Interface, Protocol, order, station}

Protocol::={precondition|postcondition}[7]

其中componentName是构件名,Require_interface是接受服务的接口,provide_Interface是提供服务的接口,Protocol是构件规约,order记载输入描述条件的顺序,

station 为构件状态。

定义 2 在理想状态下,构件(Component)可以抽象为 $C = (R,\bar{R})$,其中 $R = \{R_1,R_2,\cdots,R_n\}$,$\bar{R} = \{\overline{R_1},\overline{R_2},\cdots,\overline{R_m}\}$[8]。$R$ 和 $\bar{R}$ 分别表示输入接口和输出接口。

定义 3 顺序结构构件集成:设 C_1 和 C_2 是两个构件:$C_1 = (R_1,\overline{R_1})$,$C_2 = (R_2,\overline{R_2})$,$C_1$ 和 C_2 串行集成后的构件为 C,$C = (R_1,\overline{R_2})$。即复合构件的对外提供服务的接口为 $\bar{R} = \overline{R_2}$,对外请求服务的接口为 $R = R_1$[7]。

定义 4 分支结构的构件集成:设 C_1 和 C_2 是两个构件:$C_1 = (R_1,\overline{R_1})$,$C_2 = (R_2,\overline{R_2})$,$C_1$ 和 C_2 并行集成后的构件为 C,$C = (R_1,\overline{R_1}) \mid (R_2,\overline{R_2})$。即复合构件的对外提供服务的接口为 $\bar{R} = \overline{R_1}$,对外请求服务的接口为 $R = R_1$,或者复合构件的对外提供服务的接口为 $\bar{R} = \overline{R_2}$,对外请求服务的接口为 $R = R_2$[7]。

定义 5 并行结构的构件集成:设 C_1 和 C_2 是两个构件:$C_1 = (R_1,\overline{R_1})$,$C_2 = (R_2,\overline{R_2})$,$C_1$ 和 C_2 并行集成后的构件为 C,$C = (R_1 \cup R_2,\overline{R_1} \cup \overline{R_2})$。即复合构件的对外提供服务的接口为 $\bar{R} = \overline{R_1} \cup \overline{R_2}$,对外请求服务的接口为 $R = R_1 \cup R_2$[7]。

定义 6 循环结构的构件集成:设构件 C_1:$C_1 = (R_1,\overline{R_1})$,循环集成后的构件为 C,$C = (R_1,\overline{R_1})$。即复合构件的对外提供服务的接口为 $\bar{R} = \overline{R_1}$,对外请求服务的接口为 $R = R_1$[7]。

2 构件集成算法设计

系统根据用户输入的构件描述信息选择最合适的构件,再根据集成算法集成为复合构件。选择构件遵循的原则:①提供服务的构件的后置条件(postCondition)满足接收服务的构件的前置条件(preCondition);②提供服务的构件输出接口提供的方法满足接收服务的构件的输入接口的需求。

这样用户输入一个描述信息以后,可能会同时有不止一个构件满足上述要求,因此要选择一个最合适的构件,遵循的原则:①所选中的构件集中的每个元素即构件都能够运行到,并且都能运行到终点,即每个构件都存在一个从开始到结束的路径(终点构件(station is End)应为用户根据需求最后要实现的功能);②如果有多个构件集能完成上述要求,则选择其中时间复杂度最小即运行所需时间最短的构件集进行集成。

算法描述如下:

g:刚刚加入到构件集中的构件,G_1:备选构件集合,返回结果为复合构件或系统

```
// integration - arithmetic
receive describeInformation
for every g_i in G_1
call moveToIntegration method// 调用选择构件的方法
call selectComponentIntegration method// 调用选择构件集方法
call velidate// 调用验证算法
{switch relationship
case in_ series// 顺序构件集成
    {R = R_1;R̄ = R̄_2}
case divarication// 选择构件集成
    {R = R_1;R̄ = R̄_1} or {R = R_2;R̄ = R̄_2}
case parallel_ connection// 并列构件集成
    {R = R_1 ∪ R_2;R̄ = R̄_1 ∪ R̄_2}
case circulation// 循环构件集成
    {R = R_1;R̄ = R̄_1}
}
```

选择构件应遵循的原则:①提供服务的构件的后置条件(postCondition)能够满足接收服务的构件的前置条件(preCondition)的要求;②提供服务的构件的输出接口能够满足接收服务的构件的输入接口的要求。

概括如下:

postCondition of g fulfil preCondition of g_i

provie_ method of g fulfil require_ method of g_i

根据供求双方构件的前置与后置条件,以及输入输出方法来选择构件,一个构件可能会有多个备选的接收构件,把接收服务的构件看作是提供服务的构件的孩子,则这些建立映射关系的构件可以看成一棵树,并且按照输入描述条件的顺序分别位于树的对应层,每一个构件都是树上的一个节点,其中提供方法的构件是父节点,接受方法的构件是子节点,每一个通向叶子节点的分支都是一种集成选择,对应分支上的节点就是这种选择的构件集合。采用深度优先[9]遍历的方法来验证每一种选择运行时的时间复杂度及运行后的功能,以确定正确的集成选择。例如图 1:

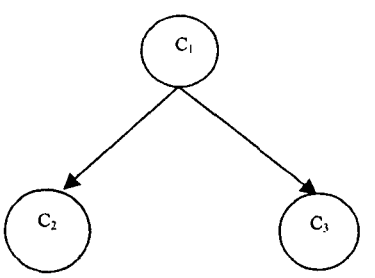

图 1 输入条件与构件生成树的关系

输入第一个条件时找到构件 C_1,输入第二个条件的时候找到构件 C_2 和 C_3,C_2 和 C_3 都可以接收 C_1 提供的服务,因此可以集成 C_1 和 C_2,也可以集成 C_1 和 C_3,即生成两个构件集 $\{C_1,C_2\}$ 和 $\{C_1,C_3\}$,由于系统集成的唯一性,因此要从两个构件集中选择一个作

为最终的构件集。选择遵循的原则是:① 必须能够到达终点构件;② 运行的时间复杂度要相对小。如果用户根据需求确定 C_3 是结束构件,则根据原则 ① 可确定 $\{C_1, C_3\}$ 为最终的构件集。

确定最终构件集的算法描述为:

```
Tt 记录输入描述条件的顺序
//Tt 输入描述条件的顺序
put node s0 to Ta;
while(the table Ta is not empty)
{
take the first node n from the table Ta;
put n to table Tb;
delete node n from the table Ta;
if n is enlarge(可扩展的)
put every child of n to the head of Ta;
}
stop iterator;
define varible i=1;
//开始第六步;
while the table Tb is not empty
{
new queue i;
if the queue i-1 is exist and corresponding node is not empty
{
take the correspond number order1 of queue(i-1);
take the correspond number order2 of the first node of Tb;
if(order1<order2){
take the correspond node c from queue(i-1);
}else if(order1=order2 and order1<=the value of Tt){
take the first node c of the table Tb;
}else if(order1>order2 and order1<=the value of Tt){
i++;
}
put c to queue i;
}
else{
take the the first node a1 from Tb;
take the the second node a2 from Tb;
while (the number of a1 < the number of a2)
{
put a1 to queue i;
delete a1 from Tb;
take the the first node a1 from Tb;
take the the first node a2 from Tb;
}
i++;
}
}
stop iterator and out result;
```

由上述算法形成的 i 个队列,即为输入 Tt 个条件后生成的构件集合。

对于生成的构件集合的合法性需做必要的验证,即从树的根节点开始遍历选中的分枝,如果每个父节点的后置条件都能满足子节点的前置条件,并且每个父节点提供的方法都能满足子节点需要的方法,即每个节点最终都能到达终节点,则称该集成是合法的,否则该集成是不合法的,所有节点都应该至少存在于一条由开始节点到结束节点构成的路径。

验证(velidate)算法描述如下:

```
//check the integration
while(if the Tb is not empty)//如果 Tb 表非空
{
get the node a from the Tb
if a has sub node //如果 a 有子节点
{
insert a to the queue
delete a from the Tb
}
else//说明该节点 a 为终点或是不可到达终点的节点
{
if the a of end point//说明 a 是终点,所以可到达 a 的节点都是合法节点
{
insert a to the queue
delete a from the Tb
Return hasEndPoint=true
}
else
{
delete a from the Tb
return hasEndPoint=false
}
}
}
if the Tb is empty and hasEndPoint=true
return "The integration is legality."
else
return "The integration is lack of legality."
```

3　应用举例

以基于 ERP 系统的报表工具中的固定报表为例,来验证构件集成算法。为降低系统的耦合度,便于系统的开发、测试、升级,把整个系统分解为一个个相对独立的小模块,并依此定制构件。现在假定符合需求的构件都已经存在于构件库中了。系统运行后,在用户界面,用户根据需求输入对构件的功能的简要描述,系统根据用户的描述信息选择最合适的构件,再根据

集成算法集成为复合构件。限于篇幅，文中只列举以下几个主要构件及其接口：

构件库中的构件如下：

$C1::=$ {mainPanel, $R11$, {$\overline{R11}$, $\overline{R12}$, P, O, notEnd}//展现类构件

$\overline{R11}$.method = {createNewTemplate}

$\overline{R12}$.method = {openTemplate}

$P::=$ {null | (null | templateName)}

$C2::=$ {openTemplate, $R21$, $\overline{R21}$, O, notEnd}//打开报表模板

$R21$.method = {setTemplateName}

$\overline{R21}$.method = {getPageSize, getTableMessage, getCellMessage, getHeaderFooter}

$P::=$ {templateName | template}

$C3::=$ {runTemplate, $R31$, {$\overline{R31}$, $\overline{R32}$, $\overline{R33}$}, P, O, notEnd}//运行模板生成报表

$R31$.method = {setDBInterface, setRepTemplet}

$\overline{R31}$.method = {jGrid}

$\overline{R32}$.method = {jGrid}

$\overline{R33}$.method = {jGrid}

$P::=$ {template | (temp;ate | jGrid)}

$C4::=$ {editTemplate, $R41$, $\overline{R41}$, P, O, notEnd}//编辑模板

$R41$.method = {setPageSize, setTableMessage, setCellMessage, setHeaderFooter}

$\overline{R41}$.method = {getPageSize, getTableMessage, getCellMessage, getHeaderFooter}

$P::=$ {template | template}

$C5::=$ {gridToPDF, R51, $\overline{R51}$, P, O, notEnd}// 导出为 EXCEL

$R51$.method = { jGrid}

$\overline{R51}$.method = {getPageSize, getTableMessage, getCellMessage, getHeaderFooter}

$P::=$ {jGrid | excelFile}

$C6::=$ {gridToPDF, $R61$, $\overline{R61}$, P, O, End}//导出为 PDF

$R61$.method = {jGrid}

$\overline{R61}$.method = {getPageSize, getTableMessage, getCellMessage, getHeaderFooter}

$P::=$ {jGrid | pdfFile}

输入描述条件后，根据构件集成算法，生成的树结构如图 2 所示。

构件根据输入的条件的顺序分别位于不同层，其中第四层是叶子节点，只有 C_6是终节点，遍历这棵构件生成树，可以得到 $C_1\ C_2\ C_4\ C_3$，$C_1\ C_2\ C_3\ C_5$，$C_1\ C_2\ C_3\ C_6$，$C_1\ C_2\ C_3\ C_4$，$C_1\ C_2\ C_3\ C_3$五个分支，根据集成算法可以得到 $C_1\ C_2\ C_3\ C_6$是集成的构件，根据验证算法，父节点可以满足子节点的要求，并且存在从根节点到终节点的路径，因此，集成算法是正确的。

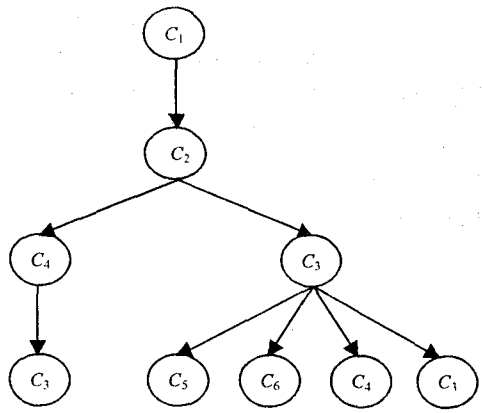

图 2 构件生成树

4 结束语

构件集成算法是软件复用的核心，在构件研究中占有相当大的比重。文中通过对构件的研究，抽象出了切实可行的构件集成算法，使软件开发可以节省相当多的人力及投资，大大缩短软件开发周期。构件集成为软件开发提出了一种挑战，也是软件未来发展的方向，适合软件未来的发展，并且使软件开发步上了一个新的台阶。

参考文献：

[1] 熊有伦，丁 汉，吴 波. 新一代制造系统理论及建模[J]. 中国机械工程，2000，11(1－2)：49－52.

[2] Yau S S, Ning Dong. Integration in Component－Based Software Development Using Design Patterns[C] // Computer Software and Applications Conference. Taipei, Taiwan, China: [s.n.], 2000.

[3] 郭胜旺，葛 玮. 构件及基于构件的开发方法研究[J]. 计算机技术与发展，2005，15(7)：37－39.

[4] Jens－Magnus A, Dibbern J. The Tension between Integration and Fragmentation in a Component Based Software Development Ecosystem[C] // Proceedings of the 39th Hawaii International Conference on System Sciences. Kauai, Hawaii, USA: [s.n.], 2006.

[5] Koren Y. Reconfigurable manufacturing systems[J]. Annals of the CIRP, 1999, 48(2): 1－14.

[6] 罗振壁，盛伯浩，赵晓波，等. 快速重组制造系统[J]. 中国机械工程，2000，11(3)：300－303.

[7] 王志坚，费玉奎，娄渊清. 软件构件技术及其应用[M]. 北京：科学出版社，2005：129－138.

[8] 袁飞云，张 驰，黄广君，等. 基于 COTS 构件组装的系统开发[J]. 微电子学与计算机，?006，23(8)：38－41.

[9] Taleghani A. Using Software Model Checking for Software Component Certification[C] // Software Engineering－Companion Volume, 2007 29th International Conference. Minneapolis, MN: [s.n.], 2007.