# 考虑兼容性和重用性的软件组件选择优化模型①

牟立峰, 唐加福
(东北大学流程工业综合自动化教育部重点实验室, 沈阳 110004)

摘要: 基于组件的软件开发 (component based software development, CBSD)方法是一种有效提高软件重用性,降低软件产品开发成本的方法. 在 CBSD过程中很重要的一个环节就是组件的选择; 目前针对这部分的研究多数都是关于组件技术的实现细节,而缺少宏观上的决策指导组件的选择. 针对这种情况,引入了兼容性关系集合的概念,结合非此即彼约束描述软件组件之间的兼容性关系; 提出了一个综合考虑重用性和兼容性的组件选择优化模型; 通过仿真揭示模型中关键参数变化对软件产品总成本目标的影响. 在选择组件开发商参与软件产品的设计和开发过程中,模型可以为软件开发商提供决策支持; 在组件中间商采购组件产品构建组件库过程中,模型也可以提供宏观的指导作用.

关键词: 基于组件的软件开发; 重用性; 兼容性; 组件选择; 组件供应商参与的软件开发
中图分类号: N949, C93 文献标识码: A 文章编号: 1007- 9807(2010)01- 0045- 07

## 0 引 言

随着软件工业的发展,软件产品的实现过程和技术越发复杂化[1-2], 为了实现软件的高度重用性,从而有效降低软件开发成本,出现了基于组件的软件开发 (component based software development, CBSD)方法[3]. 虽然 McIlroy早在 1968年就提出了组件的概念[4], 但当时的 CBSD思想没有得到软件工程学领域的重视,目前仍然存在许多有待解决的 CBSD工程技术问题[5]. 最近 20年,组件技术得到了快速的发展, CBSD技术及软件工程也日益受到了人们的关注和欢迎,许多学者在 CBSD的信息技术实现和软件工程学方面做出了杰出贡献,但是目前的研究均侧重组件技术的实现和对已有技术的比较工作[6-12], 很少有人运用优化理论对软件产品的提取、组织和组装的过程进行优化,特别是在软件产品的组装过程中对组件选择的优化. 文献 [13]考虑组件的开发成本和调整成本的因素,提出了一个构造软件系统过程中的组件选择优化模型,但没有考虑在组装组件过程中的兼容性 (compatibility)问题. 然而,在实际的软件产品结构中,不同平台的组件之间无法组装; 同一组件产品簇中不同版本组件与其他组件之间存在冲突 (conflict), 这就对组件的兼容性提高了要求. 文献 [14] 考虑组件冲突的因素,引入了一个以软件产品整体质量为目标的模型,但没有考虑软件开发的成本因素和重用性因素.

本文以组件供应商参与 CBSD过程为背景,针对软件产品组装过程中组件选择的问题,引入兼容性关系集合的概念,结合非此即彼约束描述软件组件之间的兼容性关系; 提出了一个综合考虑重用性和兼容性的组件选择优化模型; 通过仿真揭示模型中关键参数变化对软件产品总成本目标的影响.

① 收稿日期: 2007- 11- 16; 修订日期: 2009- 09- 20.
基金项目: 国家自然科学基金资助项目 (70721001; 70625001; 70471028); 教育部新世纪优秀人才支持计划资助项目 (NCET- 04- 280).
作者简介: 牟立峰 (1978—), 男, 黑龙江牡丹江人, 博士生. Email: mulifeng@gmail.com

# 1 考虑重用性和兼容性的组件选择模型

## 1.1 考虑重用性和兼容性的软件产品关系模型

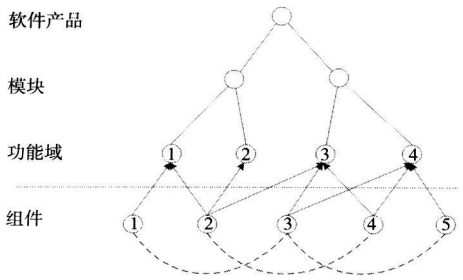


图 1 考虑兼容性和重用性的软件产品层次关系图

Fig. 1 The hierarchy structure of a software product while considering compatibility and reusability

图 1所示是一个简单的同时考虑兼容性和重用性的软件产品层次关系图. 软件产品的功能域可以由若干个可替代组件 (alternative component) 通过调整来实现, 比如可以选择组件 1或者组件 2来实现功能域 1, 可以使用组件 2、3或者组件 4来实现功能域 3等. 每个组件同时也可以通过调整复用到多个功能域中, 比如组件 3可以复用到功能域 3和 4中. 不同的功能域之间的组件存在着兼容性问题, 图 1中通过虚线连接的组件属于同一个兼容集合 (compatibility set), 集合内部的组件之间是相互兼容的, 集合之间组件不兼容, 比如组件 1、3、5之间兼容, 而组件 1与组件 4不兼容. 也就是说, 如果使用组件 1实现功能域 1, 那么必须选择使用组件 3、5分别实现功能域 3、4.

## 1.2 组件兼容关系的数学描述方法

组件之间的兼容关系应该满足兼容自反性 (A兼容 A)和兼容对称性 (A兼容 B, 都有 B兼容 A), 本文假设在满足这两个性质基础上组件兼容关系还满足兼容传递性 (A与 B兼容, B与 C兼容, 则 A与 C兼容). 如果没有这条假设, 建模过程中与兼容性相关的约束条件数量庞大, 给实际建模带来很大困难. 另外这条假设符合大多数实际情况.

为了描述模型, 引入以下数学符号:

$M$ —— 组件供应商提供的可替代组件的总数

$N$ —— 为软件产品功能域的数目

$L$ —— 为兼容集合的总数

$i$, $i'$ = 1, …, $M$ —— 组件的下标

$j$, $j'$ = 1, …, $N$ —— 功能域的下标

$k$ = 1, …, $L$ —— 兼容集合的下标

$c_i$ ($i$ = 1, …, $M$) —— 第 $i$个组件

$f_j$ ($j$ = 1, …, $N$) —— 第 $j$个功能域

$e_k$ ($k$ = 1, …, $L$) —— 第 $k$个兼容集合

$r_{ij}$ ($i$ = 1, …, $M$; $j$ = 1, …, $N$) —— 软件生产商调整 $c_i$实现 $f_j$的调整成本

$d_i$ ($i$ = 1, …, $M$) —— 组件供应商开发 $c_i$的开发成本

$u_{ik}$ ($i$ = 1, …, $M$; $k$ = 1, …, $L$) —— $u_{ik} \in \{0, 1\}$, 1表示 $c_i$的属于 $e_k$, 否则为 0

$x_{ij}$ ($i$ = 1, …, $M$; $j$ = 1, …, $N$) —— 0-1变量, 1表示在软件产品的 $f_j$中选择使用 $c_i$

$y_i$ ($i$ = 1, …, $M$) —— 0-1变量, 1表示选择开发 $c_i$, 0表示不开发

根据以上定义的数学符号, $c_i$和 $c_{i'}$兼容关系可以表示为 $x_{ij} \leqslant x_{i'j}$; $i$, $i'$ = 1, …, $M$; $i \neq i'$; $j$, $j'$ = 1, …, $N$; $j \neq j'$ 表示如果选择 $c_i$实现 $f_j$中, 则一定要选择 $c_{i'}$实现 $f_j$. 兼容性的非此即彼 (either-or constraint) 约束表示形式[11, 15]: 在选择 $c_i$来实现 $f_j$时候, 由于受到实现技术、接口等方面的兼容关系的影响, 必须在 $f_{j'}$, $j' \neq j$中使用与 $c_i$在同一 $e_k$中的 $c_{i'}$或者 $c_{i''}$. 两者是不同组件供应商为实现相近功能而设计开发的组件, 所以虽然两者与 $c_i$均属于 $e_k$, 但只能选择其中一个来实现 $f_j$. 即 $x_{ij}$ = 1时候, 要么 $x_{i'j}$ = 1, 要么 $x_{i''j}$ = 1; 如果 $x_{i'j}$ = $x_{i''j}$ = 0, 必有 $x_{ij}$ = 0. 这种约束可以表示为要么 $x_{ij} \leqslant x_{i'j}$要么 $x_{ij} \leqslant x_{i''j}$. 因为非此即彼的约束表现形式是非线性形式, 所以通过引用辅助 0-1变量 $p_{i'j'}$来克服这种非线性表现形式.

$$p_{i'j'} = \begin{cases} 0 & x_{ij} \leqslant x_{i'j'} \text{ 约束起作用} \\ 1 & x_{ij} \leqslant x_{i'j'} \text{ 约束不起作用} \end{cases} \quad (1)$$

通过引入 $p_{i'j'}$, 考虑兼容性的非此即彼约束可以表示为: 如果 $c_i$和 $c_{i'}$同属于 $e_k$, 则

$$\begin{cases} x_{ij} - x_{i'j'} \leqslant G p_{i'j'} \\ \sum_{i=1}^{M} p_{i'j'} = E - 1 \\ x_{ij}, x_{i'j'}, p_{i'j'} \in \{0, 1\} \quad i \neq i', j \neq j' \end{cases} \quad (2)$$

式 (2)中 $E$表示 $e_k$中可以调整到 $f_j$的组件总

数. 式 (2) 并不是严格的数学表达形式, 说明仅仅通过非此即彼约束和辅助变量 $p_{i'j'}$ 还不能完整地表示组件之间的约束关系. 引入参数 $u_{ik} \in \{0, 1\}$ 用来描述组件与兼容集合之间的关系, 当 $u_{ik} = 1$ 表示 $c_i$ 属于 $e_k$; 否则不属于 $e_k$. 由 $u_{ik}$ 构成的矩阵称为兼容关系矩阵. 因为每个组件只能属于一个兼容集合, 所以 $u_{ik}$ 应当满足 $\sum_{k=1}^{L} u_{ik} = 1,\ i = 1, \cdots, M$, 这一条件可以用来检验兼容关系矩阵的有效性. 当选取 $e_k$ 中 $c_i$ 来实现 $f_j$ 时, 可以用来实现 $f_{j'}$, $j' \neq j$并且与 $c_i$ 兼容的组件数目为 $\sum_{i'=1}^{M} u_{i',k},\ i' \neq i$ 个, 只能在其中选择一个组件来实现 $f_j$. 引入参数 $u_{ik}$ 以后, 可以构造与约束 (2) 意义相同的约束 (3) ~ (5). 当约束 (3) ~ (5) 中正实数 $G$取足够大的时候, 在 $e_k$ 中的 $\sum_{i'=1}^{M} u_{i',k},\ i' \neq i$个可选组件中仅仅有一个被选中.

$$u_{ik}u_{i',k}(x_{ij} - x_{i'j'}) \leqslant u_{ik}u_{i'k}Gp_{i'j'},\quad i, i' \neq 1, \cdots, M;\quad i \neq i'; j, j' = 1, \cdots, N; j \neq j'; k = 1, \cdots L \tag{3}$$

$$\sum_{i=1}^{M} u_{i'k}p_{i'j'} = \sum_{i=1}^{M} u_{i'k} - 1 \tag{4}$$
$$j' = 1, \cdots, N; k = 1, \cdots, L$$
$$x_{ij}, p_{i'j'} \in \{0, 1\}$$
$$i, i' = 1, \cdots, M; j, j' = 1, \cdots, N; (i \neq i'; j \neq j') \tag{5}$$

约束 (3) 中左右两侧的 $u_{ik}u_{i'k}$ 表示只有在 $c_i$ 和 $c_{i'}$ 同时属于 $e_k$ 时, 才检验约束条件 $(x_{ij} - x_{i'j'}) \leqslant Gp_{i'j'}$, 否则就没有必要检验. 约束 (4) 左侧的 $u_{i'k}$ 用来控制只累加 $c_{i'}$ 属于 $e_k$ 的 $p_{i'j'}$.

**1.3** 考虑重用性和兼容性的组件选择数学模型

一种待实现的软件产品包含若干功能域, 如果要实现此软件产品, 需要在组件供应商提供的所有组件中选取组件覆盖并实现所有功能域. 在组件选择过程中要考虑以下两点:

1) 不同组件的兼容性问题, 即只选择并组装相互兼容的组件;

2) 组件的重用性在成本方面的体现. 这种重用性设计直接导致 CBSD 方式中成本构成的特点, 即软件产品开发成本由组件的开发成本和组件的调整成本构成.

根据问题的描述, 考虑重用性和兼容性的组件选择优化的可以描述为如下模型:

$$\text{Min } Goal = \sum_{i=1}^{M}\sum_{j=1}^{N} r_{ij}x_{ij} + \sum_{i=1}^{M} d_i y_i \tag{6}$$

$$\text{s.t.} \sum_{i=1}^{M} x_{ij} = 1 \qquad j = 1, \cdots, N \tag{7}$$

$$x_{ij} \leqslant y_i \quad i = 1, \cdots, M;\ j = 1, \cdots, N \tag{8}$$

$$u_{ik}u_{i'k}(x_{ij} - x_{i'j'}) \leqslant u_{ik}u_{i'k}GP_{i'j'}$$
$$i, i' = 1, \cdots, M;\ i \neq i';\ j, j' = 1, \cdots, N;$$
$$j \neq j';\ k = 1, \cdots, L \tag{9}$$

$$\sum_{i=1}^{M} u_{i'k}p_{i'j'} = \sum_{i=1}^{M} u_{i'k} - 1 \tag{10}$$
$$j' = 1, \cdots, N;\ k = 1, \cdots L$$
$$x_{ij},\ y_i,\ p_{i'j'} \in \{0, 1\}\ i, i' = 1, \cdots, M;$$
$$j, j' = 1, \cdots, N;\ i \neq i';\ j \neq j' \tag{11}$$

模型中的决策变量为 $x_{ij}$ 和 $y_i$. 目标函数 (6) 由两部分组成: $\sum_{i=1}^{M}\sum_{j=1}^{N} r_{ij}x_{ij}$ 为所选组件的调整成本和; $\sum_{i=1}^{M} d_i y_i$ 为所选组件的开发成本和, 目标函数是软件开发总成本最小. 约束条件 (7) 控制每个软件功能要求仅仅由一个组件来实现. 约束 (8) 表示: 选择调整 $c_i$ 来实现 $f_j$, 要以开发 $c_i$ 为前提. (9) 和 (10) 通过引入参数 $u_{ik}$ 结合非此即彼约束表示组件兼容关系约束.

## 2 算例、仿真和数据分析

某一个优化应用软件开发商计划设计一种供应链优化问题解决商业套件产品, 要求优化应用软件产品中有 4个功能要求, 分别是: 优化算法引擎 (D1)、优化建模语言工具 (D2)、生产计划调度应用工具包 (D3) 和物流配送应用工具包 (D4). 组件供应商提供的所有组件中, 有 10个组件可以经过组件调整或者二次开发满足以上 4个功能要求. 分别为物流配送算法组件 (C1)、抽象优化建模语言组件 (C2)、物流问题建模语言组件 (C3)、物流算法引擎组件 (C4)、物流与调度算法引擎组件 (C5)、调度问题建模语言组件 1(C6)、调度问题建模语言组件 2(C7)、物流算法引擎组件 (C8)、通用问题优化组件 1(C9) 和通用问题优化组件 2(C10). 组件 C6和 C7为不同

组件供应商提供的调度问题建模语言组件; 组件 C9和 C10为不同组件供应商提供的通用问题优化组件. 存在 3个兼容集覆盖所有候选组件, 即每个组件属于并且唯一属于 3个兼容集合中的一个. 兼容集内的组件可以无缝构成一个兼容的软件系统, 而与其他兼容集中的组件存在排斥或者不兼容的情况, 无法构成满足顾客需求的软件系统. 组件的开发成本和调整成本如表 1所示. 组件的开发成本和调整成本单位为人月, 实际运用模型过程中也可以转化成为货币成本. 表中符号“∞”表示由于组件所适应的平台类型或者实现技术的特性, 不可能通过调整满足特定的要求, 或者调整的成本已经超过了开发成本, 所以可以认为它的调整成本为无限大. 这种表示方式可以减少解决实际模型的运算量.

表 **1**　可选组件的所属兼容集、开发成本和调整成本

Table 1 The compatibility sets, development cost and adaptation cost of available components

| | 所属兼容集 | 开发成本(人月) | 调整成本(人月) | | | |
|---|---|---|---|---|---|---|
| | | | D1 | D2 | D3 | D4 |
| C1 | 1 | 69 | ∞ | ∞ | ∞ | 7 |
| C2 | 1 | 68 | ∞ | 8 | ∞ | ∞ |
| C3 | 2 | 71 | ∞ | 6 | ∞ | 8 |
| C4 | 2 | 65 | 5 | ∞ | ∞ | 6 |
| C5 | 3 | 44 | 8 | ∞ | 5 | 6 |
| C6 | 1 | 59 | ∞ | 6 | 5 | ∞ |
| C7 | 2 | 75 | ∞ | 8 | 4 | ∞ |
| C8 | 3 | 78 | 6 | ∞ | ∞ | 5 |
| C9 | 2 | 70 | 5 | 7 | ∞ | ∞ |
| C10 | 1 | 72 | 7 | 6 | ∞ | ∞ |

根据 (6) ~ (11) 可以构造模型并运用 ILOG Cplex 求解上述模型, 求得模型目标函数为 163人月. 选择组建供应商提供的 C4和 C7组件, C4通过调整以后实现软件产品的功能要求 D1和 D4, C7通过调整以后实现软件产品的功能要求 D2和 D3.

本文提供了在多组件供应商参与软件产品开发过程中, 对可替代组件的选择模型. 在此模型中影响目标的关键因素有: 可选组件个数、软件产品功能要求个数、组件兼容集合个数、组件开发成本和组件调整成本. 为了研究在软件产品开发过程中, 以上关键因素的变化对产品开发的整体成本的影响, 本文运用模型对大量数据进行仿真、求解和分析, 来观察各个关键变量对目标函数的影响, 以及变量之间的相互关系. 在仿真试验过程中使用实际数据是最理想的方法. 然而 CBSD方法是软件工程理论中的新兴方法; 本文的模型尝试了新的思路, 同时考虑了组件的重用约束和兼容约束对成本目标的影响; 各个组织 (组件供应商、组件中间商和软件开发商 ) 缺少对于模型中的关键变量数据的积累. 基于以上原因, 本文采用仿真数据结合模型进行仿真试验. 仿真所用到的数据选取如下:

表 **2**　实验所用到的仿真数据

Table 2 Simulated data in experiments

| | 仿真数据描述 | 仿真数据取值 |
|---|---|---|
| 1 | 软件产品的功能要求个数 | 5 ~ 20个, 递增步长为 1个 |
| 2 | 可装配到软件产品中的可选组件的个数 | 10 ~ 100个, 递增步长为 5个 |
| 3 | 组件兼容集合个数 | 3 ~ 10个, 递增步长为 1个 |
| 4 | 组件开发成本 | 满足均匀分布 $N(40, 80)$ 的随机数 |
| 5 | 组件调整成本 | 满足均匀分布 $N(10, 20)$ 的随机数 |
| 6 | 组件可调整概率 | 20% ~ 100%, 递增步长为 10% |

表 2中的第 6个仿真数据在模型中没有体现, 是因为在实际的软件产品设计开发过程中, 组件能否通过调整而满足功能要求已经是已知的了. 但是在仿真过程中, 每个组件能否调整到某个功能要求, 我们利用调整概率来决定. 调整概率为 $c_i$ 可以通过调整而实现 $f_j$ 的概率. 同时为了避免仿真数据震动大而影响最终变化趋势的观察, 针对相同组件个数、要求个数和兼容集个数的仿真数据, 随机生成 5组开发成本、调整成本. 分别将这 5组仿真数据带入到模型中求解, 对 5次求解所得到的目标值取平均值.

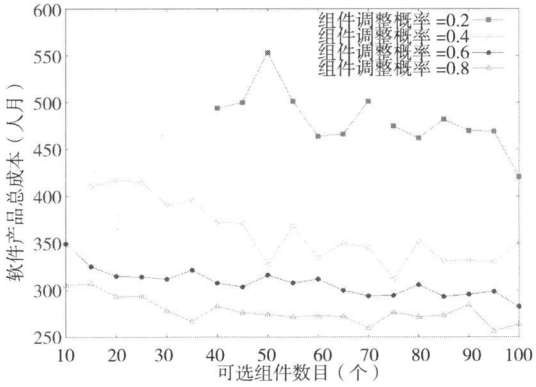


图 2　组件调整概率对软件产品总成本的影响
(软件功能要求 = 15个, 兼容集合数 = 5个)

Fig 2 The influences of probability of component adaptation on the total cost of a software product (the number of software requirements = 15, the number of compatibility sets = 5)

仿真的方法是运用模型求解每组仿真数据, 观察关键变量之间的关系, 以及对目标函数的影响. 这就需要对每组仿真数据带入模型求解. 实验数据一共 16 × 19 × 8 × 9 × 5 = 109 440组数据. 因为模型 (6) ~ (11) 为 0– 1整数规划模型; 而且对于每组数据, 随着参数变大 (组件个数的增大, 要求个数的增大), 求解问题的复杂性便提高. 因此仿真过程选择使用 ILOG OPL建立模型, 运用优化引擎 ILOG CPLEX 对每组数据的模型求解, 通过 Java预处理数据并调用 OPL求解问题模型. 由于仿真过程所用的数据量较大, 所以 OPL的数据源选择 EXCEL数据源, 优化结果也由 EXCEL来管理. 使用 GNUPlot观察仿真结果.

图 2反映了可选组件、组件调整概率和总成本之间的关系. 为了观察三者之间的典型关系, 取软件功能要求数目为 15个, 兼容集合数目为 5个. 通过图 2可知, 市场上存在的可选组件越多, 软件产品供应商对组件的选择余地越大, 软件产品的总体成本越低. 仿真假设组件的开发成本与市场上同类组件的数目无关, 所以这种趋势反映可选组件越多, 越可能选择到调整成本低的组件. 虽然组件的调整成本远远小于组件开发成本 (*N* (10, 20); *N* (40, 80)), 但整体上还是能降低软件开发的总成本. 另外这种下降趋势 (速度) 随着组件调整概率减少而增大, 并且曲线的波动随组件调整概率增大而减小. 说明当组件的重用性越大时, 即越能通过适当的调整满足特定软件功能要求时, 越能降低软件产品的总成本, 而且这通过这种方式降低成本远远比扩大可选组件数目有效. 曲线的波动反映, 在市场上的组件的重用性整体上比较低的情况下, 通过选择正确的组件很可能会大幅度降低软件的开发总成本. 图 2中, 组件调整概率为 20% 的曲线在组件数目为 40之前没有目标值, 表示组件调整概率为 20%, 组件数目小于 40个的时候, 模型很少有解, 即可行方案非常的少.

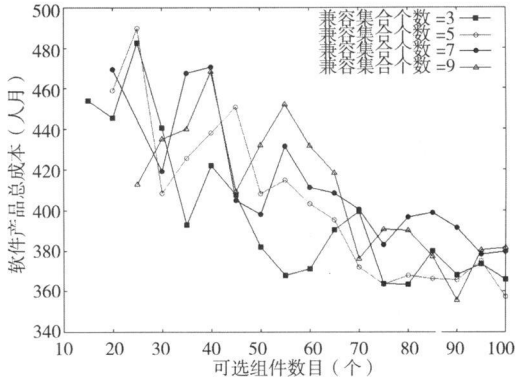


图 3　兼容集合数目对软件产品总成本的影响
(软件功能要求 = 15个, 调整概率 = 30%)

Fig 3 The influence of the number of compatibility sets on the total cost of a software product (the number of software requirements = 15, the probability of adaptation = 30%)

图 3反映了可选组件、组件兼容集合和总成本之间的关系. 为了观察三者之间的关系, 取软件功能要求数目为 15个, 调整概率数目为 30%. 在可选组件个数与总成本之间的关系方面, 图 3反映的内容与图 2相同. 图中的每条曲线均波动很大, 是因为组件调整可能性设置的比较小为 30%. 实际情况中针对一个软件产品项目, 组件的

调整可能性确实比较小. 通过图 3中曲线之间的偏差可知, 兼容集合个数的变化对软件产品总成本影响并不显著. 表示在这种参数环境下, 在降低总成本的作用方面, 可选组件数目的增加比兼容集合的减少更有效.

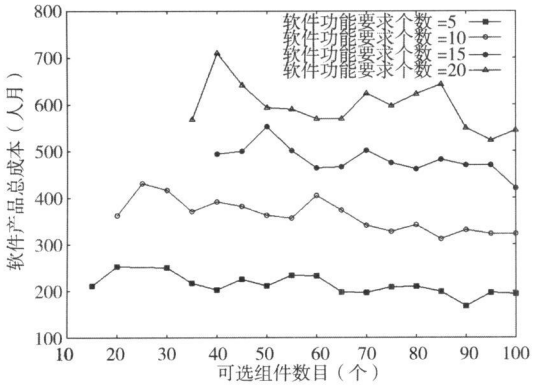


图 4　软件功能要求对软件产品总成本的影响
(兼容集合数 = 5个, 调整概率为 = 20%)

Fig. 4 The influence of the number of software requirements on the total cost of a software product (the number of compatibility sets = 5, the probability of adaptation = 20%)

图 4反映了可选组件、软件功能要求和总成本之间的关系. 为了观察三者之间的典型关系, 取兼容集合个数为 5个, 调整概率为 20%. 首先软件的功能要求数目越多成本越高, 得到模型可行解就要求需要有更多的可选组件; 其次在组件较多, 软件功能要求较少的情况下, 模型最优解不会有太大变化. 相反的情况下, 运用模型求得的结果可能很理想.

## 3　结　论

本文以 CBSD为研究背景, 针对组件供应商参与软件产品开发设计过程中可替代组件选择的问题, 提出了组件选择优化模型, 指导软件中间商和开发商的组件选择决策活动. 本文的特殊贡献在于: 首先将系统工程思想和优化理论应用于 CBSD过程, 并且指导此过程中的组件选择问题的决策; 提出了更符合实际情况的具体问题, 即综合考虑重用性和兼容性的组件选择问题; 优化模型中体现了组件重用性所引起的软件产品不同的成本结构, 综合考虑了组件供应商的组件开发成本和软件开发商的组件调整成本; 模型中体现了组件兼容关系约束的影响; 将模型运用到大量仿真数据中, 揭示模型中关键参数之间的关系和他们对目标值的影响.

参 考 文 献:

[1] Budgen D. Software Design[M]. 2nd edition. Pearson Addison-Wesley, 2003.

[2] Sommerville I. Software Engineering[M]. 6th edition. Harlow: Addison-Wesley, 2001.

[3] Brown A W. Large-scale Component-Based Development[M]. Upper Saddle River: Prentice Hall, 2000.

[4] McIlroy M D, Burton J M, Naur P, *et al*. Mass Produced Software Components[C] // Software Engineering Concepts and Techniques, 1968 NATO Conference on Software Engineering. Petrocelli/Charter, New York, 1969: 88 - 98.

[5] Crnkovic I. Component-based software engineering-new challenges in software development[J]. Software Focus, 2002, 2(4): 127 - 133.

[6] Szyperski C, Gruntz D, Murer S. Component Software: Beyond Object-Oriented Programming[M]. (2nd Edition). Harlow, England: Addison-Wesley, 2004: 10 - 11.

[7] Lanergran R G, Grasso C A. Software engineering with reusable designs and code[J]. IEEE Transactions on Software Engineering, 1984, 10(5): 498 - 501.

[8] Diaz R P, Freeman P. Classifying software for reusability[J]. IEEE Software, 1987, 4(1): 6 - 16.

[9] McArthur K, Saiedian H, Zand M. An evaluation of the impact of component-based architectures on software reusability[J]. Information and Software Technology, 2002, 44(6): 351 - 359.

[10] Boehm B. Managing software productivity and reuse[J]. Computer, 1999, 32(9): 111 - 113.

[11] Wang L, Krishnan P. A Framework for Checking Behavioral Compatibility for Component Selection[C] // Proceedings of the Australian Software Engineering Conference (ASWEC'06), 2006: 49 - 60.

[12] Craig D C, Zuberek W M. Verification of Component Behavioral Compatibility[C] // Proceedings of the 2nd International

Conference on Dependability of Computer Systems, 2007. 294- 304.

[13] Sundarraj R. P. An optimization approach to plan for reusable software components[J]. European Journal of Operational Research, 2002, 142(1): 128- 137.

[14] Jung H W, Choi B. Optimization models for quality and cost of modular software systems[J]. European Journal of Operational Research, 1999, 112(3): 613- 619.

[15] Hiller F, Lieberman G. Introduction to Operations Research[M]. 6th edition. New York: McGraw-Hill, 1995: 459- 557.

# Optimization model for software component selection considering compatibility and reusability

*MU Li-feng, TANG Jia-fu*

Key Laboratory of Integrated Automation of Process Industry of Ministry of Education, Northeastern University, Shenyang 110004, China

**Abstract** Component based software development is well acknowledged as a methodology which establishes reusability of software and reduces development cost effectively. Component selection is a very important part of CBSD process; however most of the research works have focused on details of implementation using information technologies rather than optimization methods. Based on this background, a concept of compatibility set is proposed to present compatibility of software components with either-or constrains; a component selection model is proposed which considers reusability and compatibility; meanwhile relationships between parameters of the model and the objective value of total cost are discussed through simulation. The model can assist software providers in decision making during their selecting of component providers, who will be involved in software product design and development. Component intermediate merchants can utilize the model for instruction function when they build component repository for software markets.

**Key words** component based software development; reusability; compatibility; components selection; software development involved by component suppliers