# 商业构件评价与选择方法研究*)

盛津芳[1] 王 斌[1] 张尧学[2] 陈松乔[1]

(中南大学信息科学与工程学院 长沙 410083)[1] (清华大学计算机系 北京 100084)[2]

**摘 要** 在基于商业构件(COTS)的软件开发中,构件的评价与选择是贯穿整个开发过程的关键步骤。有两类典型的基于COTS的系统,即COTS方案系统和COTS密集型系统。本文首先描述了针对COTS方案系统的单构件评价问题的特点,并对各种评价方法进行了分析、对比。然后将COTS密集型系统的多构件选择问题定义为一个在给定的约束条件下求解最佳构件组合的数学优化问题。最后给出了该优化问题的形式描述及其相应的求解方法。

**关键词** 商业构件,构件评价与选择,多准则决策,数学优化

## Research on COTS Evaluation and Selection Methods

SHENG Jin-Fang[1] WANG Bin[1] ZHANG Yao-Xue[2] CHEN Song-Qiao[1]

(College of Information Engineering in Central South University, Changsha410083)[1]

(Dept. of Computer Science & Technology, Tsinghua University, Beijing 100084)[2]

**Abstract** COTS evaluation and selection is the key process during the development of COTS-based software. There are two typical COTS-based systems, one is COTS Solution System(CSS) and the other COTS Intensive System (CIS). This paper first introduces the characteristics of the individual COTS selection in CSS and gives a detailed analysis and summary to decision-making techniques for this selection problem. Then the multiple COTS selection in CIS is defined as a mathematical optimization problem to determine the most promising COTS combination for maximizing the global fitness while under cost constraints. Finally the paper gives a formal description of the optimization problem and its solution.

**Keywords** COTS, COTS evaluation and selection, Multi-criteria decision making, Mathematical optimization

## 1 引言

与传统的软件开发方法不同,基于商业构件(COTS)的软件开发方法(CBSD)通过组装已有的商业构件来构造大型、复杂的软件系统[1]。这里的商业构件是指由第三方开发、不提供源代码、仅通过接口与其它软件进行集成,并能实现一定功能的软件产品[2]。在CBSD中,构件的评价与选择是贯穿整个软件生命周期的关键活动。而能否快速、有效地识别、评价和选择合适的商业构件是软件系统成功与否的重要因素。

CBSD是一种以购买为中心,而不是定制为中心的系统开发方法。根据系统中购买部分与定制部分的相对比例,有两类典型的基于COTS的系统(CBS)。一类被称为COTS方案系统(COTS Solution System),以单一的COTS产品为中心,结构简单,可靠性强。例如客户关系管理系统、ERP系统、工作流管理系统等。另一类被称为COTS密集型系统(COTS Intensive System),其特点是系统含有多个COTS构件以及部分定制成分,并通过粘连代码(Glue code)组装而成。这类系统中的COTS产品往往由不同的厂商提供,并且独立于特定的应用,构件提供的功能和接口往往不能完全满足某个具体应用的需求。

为了选择或推荐合适的构件,必须根据用户的需求对候选的构件进行评价,并根据评价的结果对构件排序。构件评价是构件选择的基础。对COTS方案系统,由于被评价的对象是单个构件产品,称为单构件评价问题。而对COTS密集型系统,被评价的对象是多个构件的组合,称为多构件评价问题。本文分别对这两类构件评价问题的特点、评价方法以及采用的决策分析手段进行了分析、比较和总结。

## 2 单构件评价与选择

单构件评价是一个多指标综合评价问题,包含3个主要步骤,即确定评价指标集、选择适当的度量方法和综合评价方法。多个评价指标分别描述被评价构件的不同方面,具有不同的性质和量纲。多指标综合评价的前提是将异量纲的指标值转化成无量纲的相对评价值,最后将各评价值综合在一起得到一个整体的评价[3]。

评价的过程是一个复杂的决策分析过程[4],其原因在于:①由于软件自身的特点,对软件功能与质量的评价往往是不确定的。②有些评价指标缺乏完备的定量手段,对定性描述的指标值的获取常常依赖专家的主观判断。③多个评价指标之间具有不可公度性和矛盾性。所谓不可公度性是指各指标间没有统一的度量标准,因而难以进行比较。④评价过程往往是有多位背景不同的专家参与的集体评价,在看法和判断上有相当的差异;⑤针对一组给定的需求,候选的构件数量可能很大。⑥针对一组候选构件,需要考虑的评价指标数量可能很大。⑦构件评价是确定一个构件在一组功能相似的候选构件中的相对地位,这种地位是通过比较而认识的,由于比较的基准物不

*)本文研究由中国高校博士点基金(No. 20030533011),湖南省自然科学基金(No. 05JJ40312),中南大学博士后科学基金资助。盛津芳 讲师,博士研究生,研究方向为基于构件的软件工程。王 斌 副教授,研究方向为安全软件。张尧学 教授,博士生导师,主要研究方向为多媒体网络服务质量控制、网络互联。陈松乔 教授,博导,研究方向为软件工程。

同，所以这种评价是相对的。

现有的各种多指标综合评价方法都蕴含着应当满足的相应的假设条件，使用这些方法进行构件评价时，只有清楚地认识方法的假设与约束条件及其适用范围，才能取得比较满意的结果[5]。

### 2.1 评价指标集的层次结构

评价指标集的确定是综合评价的前提。评价指标由多种需求转化而来，例如：用户需求；设计需求，如软件体系结构的制约、操作系统的制约等；工程需求，如项目开发的进度、预算，以及分阶段的开发计划等；组织结构需求，如软件复用策略、复用能力、管理模式等。其中，用户需求包括对构件的功能需求和非功能需求，例如可靠性、可移植性、可集成性等。

需求的层次性决定了它可以采用树形结构来组合各项评价指标，每一个下级指标只与一个上级指标相关联，如图1所示。以功能需求为例，结构的最上端，即第一层次，是构件向用户提供的总功能，其定义大多具有综合性含义。第二层次可视为由总功能分解而成的子功能。而每个子功能都有可能由更基本的功能构成。相邻的上下层功能间是递阶分解（上对下）与合成（下对上）的关系。同一层次间的功能相互独立或为互补关系。

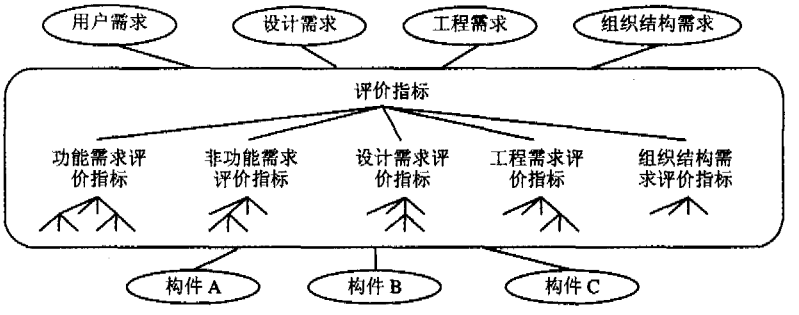


图1 评价指标集的层次结构

与功能需求不同，非功能需求间往往存在一定的依赖或制约关系。NFR框架可用于表示和分析非功能需求[6]。如图2所示，非功能需求“信息安全性”被分解为完整性、可用性和保密性。而“系统性能”进一步分解成为吞吐量和响应时间。需求之间可能是协同关系或冲突关系，例如授权与加密是协同关系，而对信息加密势必降低系统性能，构成冲突关系。

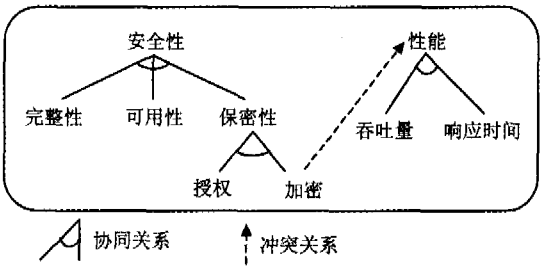


图2 NFR框架表示的非功能需求的分解

### 2.2 综合评价方法

许多在决策支持领域中常用的综合评价方法都可被用于构件的评价，例如：多属性效用理论、加权和法、层次分析法和模糊综合分析法。下面讨论几种主要的方法以及各方法应用于构件评价时的优缺点、约束条件和适用范围。

1. *多属性效用理论* 根据现代效用理论，评价者自觉或不自觉地受制于某种在评价过程中始终能起支配作用的偏好系统。评价者的偏好系统是指评价者根据自己的偏好对方案集中的方案成对比较并区分优劣（或大小），并最终排列各方案的优劣次序。这种偏好系统使得行为主体必然存在一个效用函数，它是客体功能到主体满足性的映射。对于构件的使用者或消费者，其偏好系统中也必然存在一个效用函数。通过它，不同的构件特性均可转换为统一刻画需求满足程度的数量指标，称为功能效用值。在适当选择效用函数原点和比例单位后，不同类属的功能效用值，都将成为在同一标准下可以相加或相乘的纯量。

多属性效用法要求评价者确定每一个评价指标的不同取值的相对优劣程度，以及不同指标的相对重要程度。每个指标上定义了一个效用函数。由单指标效用合成多指标效用有两种运算关系，即加性关系和乘性关系。如果评价指标之间可以相互线性补偿，则适合于加性；如果相互制约则适合于乘性。使用多属性效用理论的困难之处在于确定适当的效用函数。

2. *加权和法* 加权求和法是最常用的一种多指标综合评价方法。它首先要求对指标的取值规范化。例如指标有多种类型。对于效益型指标，取值越大越好；而成本型指标的取值越小越好。不同类型的指标放在一起不便于直接从数值上判断方案的优劣。因此要对数据进行预处理，使得任一指标下性能越优的方案变换后的取值越大。规范化还包括排除采用不同量纲对评价结果的影响，即非量纲化，以及评价值的归一化。

权的引入主要用于解决指标间的矛盾性。权是衡量指标重要性的手段。但是在评价指标较多时，往往难以直接确定每个指标的权重。通常的做法是让评价者把各指标成对比较，然后用一定的方法将成对比较结果聚合起来确定一组权，常用的方法有最小二乘法和本征向量法。对某一候选构件所有评价指标值加权求和得到一个综合评价值。

使用加权和法来评价构件必须基于如下假设：每个指标的边际价值是线形的，即优劣与指标值的大小成比例；任意两个指标价值独立；并且指标间完全可补偿，即一个候选构件的某个特性无论多差都可用其它特性来补偿。但事实上，这些假设往往不成立。比如，指标的边际价值的线形常常是局部的，指标间的可补偿性也只是部分的、有条件的。

3. *层次分析法* 层次分析法首先构造一个由目标层、指标层、方案层（候选构件层）构成的层次结构模型。指标层又分解成为若干有序层次。每一层次中的元素具有大致相等的地位，层次之间按隶属关系建立起一个有序的递阶层次模型。按照对一定客观事实的判断，对每层的重要性以定量的形式加以反映，即通过两两比较判断的方式确定每个层次中元素的相对重要性，并用定量的方法表示，进而建立判断矩阵。

通过对候选构件在各指标下优劣程度的两两比较，求得每个指标下各构件的优先性（即权重），再计算各候选构件的总体

优先性(即总权重),最后根据总体优先性的大小排出构件的优劣。

层次分析法要求评价指标间完全独立。对于软件系统的需求来说很难做到。此外,对于大型、复杂的软件系统,其计算模型中包含的大量的指标之间、构件之间的两两比较,使得该方法难于实施。

4. *模糊综合分析法* 模糊综合分析法首先确定被评价构件的指标集 $U=(x_1,x_2,\cdots,x_m)$ 和评价集合 $V=(v_1,v_2,\cdots,v_n)$,其中 $x_i$ 为各单项指标,$v_i$ 为对 $x_i$ 的评价等级层次,一般可分为五个等级:V={优,良,中等,较差,差}。再分别确定各个指标的权重及它们的隶属度向量,获得模糊评判矩阵。最后把模糊评判矩阵与指标的权重集进行模糊运算并进行归一化,得到模糊评价综合结果。

模糊综合分析法的隶属函数和模糊统计方法为定性指标的量化提供了有效的方法,实现了定性与定量的结合。在将构件特性与用户需求匹配的过程中,二者之间的匹配程度往往不是绝对的肯定或绝对的否定,涉及到模糊因素,而模糊综合分析法可以很好地解决判断的模糊性和不确定性问题。

此方法同样不能解决评价指标间相关带来的评价信息重复问题。当评价指标达到一定数量时,隶属函数的确定过于繁琐,实用性不强。

## 3 多构件评价与选择

COTS密集型系统由多个构件组装而成,构件间需要协作以完成某些功能[7]。与单构件选择不同的是,多构件选择要考虑以下因素:1)利用系统分解降低复杂度。在由多构件组成的系统中,将系统分解成子系统是识别组成构件的必要的先行步骤。在系统分解的过程中,系统功能需求也被分解到各个子系统中去。构件的选择针对子系统的需求进行,进一步缩小了构件的搜索空间。2)考虑不同粒度的构件组合。针对同一组需求,可能存在不同的构件组合。例如图3所示的构件组合{C5,C6,C3,C10,C11}和{C5,C6,C7,C12,C13,C10,C11}都可以完成系统功能。3)考虑构件间的兼容性。造成构件不兼容的原因可能是采用不同的协议、软件体系结构或操作系统等。4)考虑构件的购买成本与组装代价。

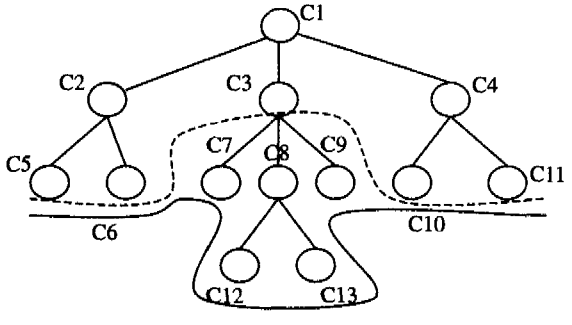


图3 不同的构件组合

多构件评价包含局部评价和全局选择两个阶段。单构件评价方法仍可用于对构件进行局部评价。我们将构件对子系统需求的满足程度称为局部需求满足度,而一组构件组装之后对整个系统的需求满足度称为全局需求满足度。只有当某个构件的局部需求满足度大于系统预定义的一个阈值时,才能进入全局选择。一组局部最优的构件,组装之后却不一定能达到全局的最优。因此,多构件的选择问题应该是一个在给定的约束条件下求解最佳构件组合的优化问题。

为了形式化描述多构件选择问题,首先引入下列符号和假设:

• 系统功能需求分解后得到 $n$ 个功能子集 funct[$i$]($i=1,\cdots,n$),每个功能子集对应一组候选构件 COTS[$i,j$]($j=1,\cdots,m$)。

• 对每个功能子集 funct[$i$]($i=1,\cdots,n$)有一个对应的权重因子 weight[$i$],为该子集的相对重要性。权重因子的总和为1。

• 每个候选构件 COTS[$i,j$]有一个决策变量 $x[i,j]=\{0,1\}$,为1时表示该构件被选中,否则未被选中。如式(1)所描述。

• 对应每个功能子集,最多只能有一个构件被选中。如式(2)所描述。

• 对应每个候选构件,cost[$i,j$]为该构件的购买成本。

• 变量 COST 为系统用于购买构件的成本上限。如式(3)所描述。

• 构件子集 sub[$k$]包括一组不兼容的构件,比如说这组构件不能同时工作。如式(4)所描述。

• 每个构件对应一个需求满足因子 fitness[$i,j$],为该构件相对功能子集 funct[$i$]的满足程度,取值在0到1之间。如该值为1,表示需求满足度为100%。

• 每个构件对应一个组装代价 Int_effort[$i,j$],取值在0到1之间。组装代价用于在局部评价阶段淘汰组装代价超过预定义阈值的构件。

• 该问题的目标是在给定约束条件下(式(1~4))取得最大的全局需求满足度。如式(5)所描述。

基于上述符号和假设,多构件选择问题定义如下:

$$x[i,j]\in\{0,1\}\ \text{for all pairs}[i,j] \tag{1}$$

$$\sum_j x[i,j]\leqslant 1\ \text{for all}\ i=1..n \tag{2}$$

$$\sum_{i=1..n}\sum_j \text{cost}[i,j]x[i,j]\leqslant \text{COST} \tag{3}$$

$$\sum_k\sum_{[i,j]\in \text{sub}[k]} x[i,j]\leqslant 1\ \text{for all}\ k=1..K \tag{4}$$

$$\sum_{i=1,\cdots,n}\text{weight}[i]\sum_j \text{fitness}[i,j]x[i,j]\Rightarrow \text{Max!} \tag{5}$$

我们将全局构件选择定义为一个受约束的优化问题。为解决这个问题,可以首先利用 AMPL(A Modeling Language for Mathematical Programming)[8]对问题进行描述。AMPL 是一种对线性和非线性优化问题建模的代数语言。然后选择一种针对混合整数非线性规划方法的求解器 MINLP[9]来求解。我们在文[7]中介绍了详细的求解过程。

**结束语** COTS方案系统中的构件评价是针对单个构件进行的多指标综合评价。评价指标集具有层次结构,指标间具有互补或制约关系。决策支持领域常用的综合评价方法可用于构件评价,但各有其假设条件和优缺点。针对COTS密集型系统的多构件选择,需要考虑构件的不同组合,以及包括构件间兼容性在内的各种约束条件,因此应该是一个在给定的约束条件下求解最佳构件组合的优化问题。本文给出了该优化问题的形式定义及其相应的求解方法。

## 参 考 文 献

1 Tran V, Liu D, Hummel B. Component-based Systems Development: Challenges and Lessons Learned. In: Proc. of 8th International workshop on Software Technology and Engineering Practice, July 1997. 452~462

2 Wallnau K C, Hissam S A, Seacord R C. Building Systems from Commercial Components, Addison-Wesley, 2001

(下转第277页)

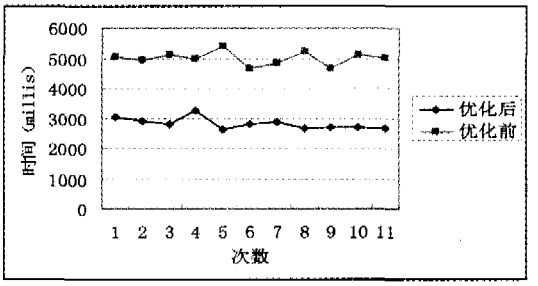


图5 关于节点延迟生成和延迟加载的测试结果

效地提高程序的性能。

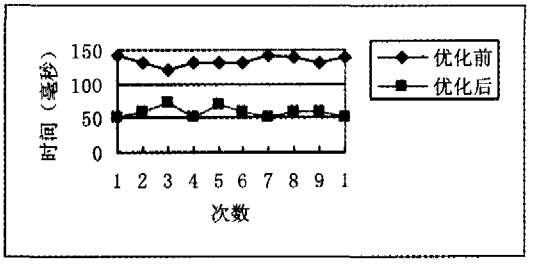


图6 BenchGetElements的结果

### 4.2 查询结果集合的延迟装载

针对查询结果集合优化前后的OnceDOMParser进行了测试。优化前，getElementsByTagName执行的时候一次性地在元素的子节点树中查找所有的元素；优化后，getElementsByTagName执行的时候会生成一个SearchResultNodeList返回给用户，然后用户访问SearchResultNodeList的时候再根据需要从节点树中查找出用户访问的节点。

测试程序解析一个约100k的XML文档，然后，通过调用Element的getElementsByTagName来获取指定的Target元素。执行getElementsByTagName 100次。两种情况所花费的时间如图6。从图中看出，优化后所花费的时间仅为优化前的一半左右，因此，查询结果集合的延迟装载同样可以有

### 4.3 系统吞吐量测试

系统吞吐量测试采用了Sun公司提供的XML Test1.1。系统吞吐量，在Sun XML Test中，是指每分钟执行XML事务的平均数，这里一个XML事务包括解析、访问、修改(包括内容修改和结构修改)、序列化四步。Sun XML Test1.1中给出了六个测试场景，test1、test2、test3分别是访问文档的25%、50%、100%内容，test4测试内容修改，test5测试结构修改，test6综合了内容和结构修改。当然六个测试都包括对文档的解析，并且后三个包含了对修改后的文档的序列化。表1是SUN XML Test1.1的测试结果。可以看出OnceDOMParsr平均比Xerces2.6.2快43.7%以上。

表1 性能测试结果比较

| 测试场景 | 平均性能提升 | Once1st | Once2nd | Once3rd | Xerces1st | Xerces2nd | Xerces3rd |
|---|---|---|---|---|---|---|---|
| Test1 | 43% | 276.52 | 271.08 | 272.87 | 192.95 | 188.65 | 191.35 |
| Test2 | 45% | 267.54 | 267.67 | 266.48 | 187.92 | 179.31 | 182.59 |
| Test3 | 52% | 257.26 | 258.27 | 262.866 | 173.10 | 168.42 | 170.16 |
| Test4 | 38% | 140.17 | 140.31 | 142.33 | 104.80 | 101.91 | 99.37 |
| Test5 | 43% | 142.16 | 140.01 | 141.08 | 98.38 | 100.17 | 97.24 |
| Test6 | 40% | 139.54 | 138.80 | 140.61 | 99.19 | 99.57 | 98.71 |

**小结** 本文在对DOM模型进行研究的基础上，构造了Java运行环境下的XML解析器OnceDOMParser。对其进行的性能测试结果表明OnceDOMParser性能卓越，解析效率要高于目前最流行的XML parser Xerces 43.7%。

下一步工作包括以下几方面：1)增加有效性验证检查；2)扩充OnceDOMParser功能，实现对HTML的支持等；3)针对Java语言本身对系统进行进一步的性能优化。

## 参考文献


1 W3C. Extensible Markup Language (XML) 1.0. http://www.w3.org/TR/1998/REC-xml-19980210, 1998
2 W3C. Extensible Markup Language (XML) 1.1. http://www.w3.org/TR/2004/REC-xml11-20040204, 2004
3 W3C. Document Object Model. http://www.w3.org/DOM/
4 The Apache XML Project: Crimson. http://xml.apache.org/crimson/
5 The Apache XML Project: Xerces Java 2. http://xml.apache.org/xerces2-j/
6 OnceDOMParser详细设计报告. 软件工程技术中心技术文档, 2005
7 OnceStAXParser详细设计报告. 软件工程技术中心技术文档, 2005
8 Java Community Process. JSR 173: Streaming API for XML. http://jcp.org/en/jsr/detail?id=173
9 Java API for XML Processing. http://java.sun.com/xml/jaxp/index.jsp
10 Venners B. Inside the Java Virtual Machine. (Second Edition). 2003
11 W3C. Extensible Markup Language (XML) Conformance Test Suites 20031210. http://www.w3.org/XML/Test/
12 W3C. DOM TS(DOM Conformance Test Suites). http://www.w3.org/DOM/Test/
13 SUN. XMLTest1, 1. http://java.sun.com/performance/reference/codesamples/


(上接第266页)


3 Ncube C, Dean J C. The Limitations of Current Decision-Making Techniques in the Procurement of COTS Software Products. Proceedings of ICCBSS, Orlando, Florida USA, 2002. 176~187
4 Ruhe G. Intelligent Support for Selection of COTS Products. In: Proc. of the Net. ObjectDays 2002, Erfurt, Springer 2003
5 岳超源. 决策理论与方法. 科学出版社, 2003. 170~246
6 Alves C, Castro J. CRE: A Systematic Method for COTS Components Selection. XV Brazilian Symposium on Software Engineering (SBES) Rio de Janeiro, Brazil, October 2001
7 Sheng Jinfang, Chen Songqiao, Wang Bin. COTS Evaluation and Selection Based on Requirements Decomposition, Chinese Journal of Electronics, 2005(1): 62~67
8 Fourer R, Gay D M, Kernighan B W. AMPL: A Modeling Language for Mathematical Programming, Duxbury Press, 2002
9 MINLP World, http://www.gamsworld.org/minlp/index.htm, 2005