第 27 卷第 1 期
2010 年 1 月

控 制 理 论 与 应 用
Control Theory & Applications

Vol. 27 No. 1
Jan. 2010

文章编号: 1000−8152(2010)01−0069−08

两阶段过程模型下的产品线复用策略优化

吴志樵[1,2], 唐加福[1,2], 周进刚[1,3]

(1. 东北大学 信息科学与工程学院, 辽宁 沈阳 110004; 2. 东北大学 教育部流程工业综合自动化重点实验室, 辽宁 沈阳 110004;
3. 东软集团股份有限公司 基础软件事业部, 辽宁 大连 116085)

摘要: 考虑了软件产品线工程中各软件模块复用策略的选择问题. 在建立软件产品线开发两阶段过程模型的基础上, 归纳出6种典型的复用策略及其实现方式, 提出考虑开发成本、工时、故障率的情况下进行复用策略的选择优化模型, 解决在满足可获得预算、开发周期、系统可靠性需求的约束下产品线质量最优问题. 并基于贝叶斯理论构建易测试函数, 为产品线开发中领域测试强度的估算提供了方法. 最后, 以邮箱服务系统为实例说明该模型的有效性.

关键词: 软件质量; 软件可靠性; 软件复用; 软件产品线; 最优化模型

中图分类号: TP311. 5　　**文献标识码:** A

Selecting optimal reuse scenarios based on the two-stage procedure model in a software product line

WU Zhi-qiao[1,2], TANG Jia-fu[1,2], ZHOU Jin-gang[1,3]

(1. Institute of Systems Engineering, Northeastern University, Shenyang Liaoning 110004, China;
2. Ministry of Education Key Lab of Integrated Automation of Process Industry, Northeastern University, Shenyang Liaoning 110004, China;
3. Business Division, Neusoft Corporation, Dalian Liaoning 116085, China)

Abstract: We investigate the selection of the reuse scenarios on a two-stage software product line for satisfying the given requirements within a budgetary constraint while maximizing the product quality. Based on the two-stage process model, we conclude 6 typical reuse scenarios and their realizations, and propose a method for selecting the best one from them by considering the budget limitations, the development cost and time, the failure rate and reliability to optimize the production quality. Based on the Bayes theory, we developed a simplified testing function for estimating the intensity of domain testing in the developing phase of the product. A mailbox service system is used as a real example to demonstrate the effectiveness of the model.

Key words: software quality; software reliability; software reuse; software product line; optimization models

1　引言(Introduction)

软件复用作为解决“软件危机”的一颗“银弹”, 在软件工程领域研究与实践中颇为活跃. 通过在新的软件产品(或其派生产品)中融入现有软件资产, 软件复用技术可以有效地降低软件开发总成本, 缩短软件开发时间, 提高软件生产效率[1,2]. 目前, 复用技术中所支持复用粒度最大的方式是软件产品线(software product line, SPL)[3]. 通过合并通用功能到软件产品族的高质量内核中, 经选择和(或)修改可变部分, 实现对已有软件资产的复用. 同时, 作为复用技术中粒度最小方式的构件(component)技术[3]的发展促成了软件模块(module)可以快速简便地组装成新的应用, 客观上为软件产品线开发方式提供了可行的技术支持. 随着上述两种技术的逐步成熟, 对软件产品线开发的认识, 已经由传统意义上的编程实现转化为对可复用资产的重组.

近年, 面向产品线的复用优化技术在理论上不断完善. Jung[4]和Berman[5]使用经典的0–1背包模型来考虑质量最高条件下的构件选择问题, 还有学者将该过程视为成本最小问题[6,7]进行选择. 此外, SEI[8]进一步提出将构件的内部开发与市场购买(make or buy)这两种核心资产获得方式相结

收稿日期: 2008−12−05; 收修改稿日期: 2009−04−13.
基金项目: 国家自然科学基金资助项目(70721001,70625001); 教育部博士点专项基金资助项目(20060145009); 中央高校基本科研业务费资助项目(N090604004).

合来考虑整个组织的构件选择优化问题. 实践中, STARS(software technology for adaptable reliable systems)报告[9]显示, 在使用软件产品线技术后的第1个示范系统的开发成本降至以往成本的43%, 第2个系统更降为10%. 另一个案例是惠普公司[10], 产品线技术令其软件开发的错误率降低了15%, 且生产效率提高了57%.

尽管如此, 由于缺乏系统的方法, 未能在软件复用中协同考虑构件和产品线, 致使构件和产品线技术在企业级的开发中屡屡受挫. 此外, 在近期国际会议上多次被提及[11,12]的基于产品线复用开发的高可靠性管理问题也依然未能得到有效解决. 产品线开发成败的核心因素是一个高质量的内核. 若在归入核心资产的构件中存在不易测试的错误, 将导致在单软件开发基数上以几何倍数增加的高故障率. 而可靠性的保证是建立在对构成模块不同以往的高强度测试的基础上, 面向产品线开发若要获得高可靠性保证, 所需要付出的代价比常规开发超出20%∼40%[13]. 如果在领域分析中忽视这部分的高昂代价, 将使开发时间超期、开发成本超额的情况发生的可能性大大增加. 因此, 必须在产品线开发的前期就将领域工程优化模型考虑进来.

本文建立软件产品线开发过程的两阶段模型, 对各种复用方式进行协同考虑, 使开发者能够根据产品的成本—效益对各软件模块进行差异开发, 应对市场做出及时反应. 提出QOCRED(quality optimization under cost, reliability and delivery time constraints)模型进行构件复用策略的选择, 帮助开发者决定买入或开发及以何种开发方式才能实现在满足预算、预期、可靠性的约束下软件产品线的质量最高问题. 并提出易测试函数, 充分考虑高可靠性保证代价这一问题.

2 产品线开发的两阶段过程模型(The two-stage procedure model)

近年来在软件的生产过程中, 尽管基于构件和产品线工程的开发范型[8,14~16]推动了各种层级复用技术的引入和运用. 尽管如此, 在构件和产品线技术尚存在使用时机不当致使不能实现高效率地系统化复用的遗憾. 多数构件技术, 如CORBA(common object request broker architecture), DCOM (distributed component object model), EJB(enterprise javabean)等, 所提供的模型仅将构件视为可执行模块, 在生命周期的最后实现和部署阶段才投入使用; 与之相反, 由于传统的软件实现技术(如编程语言)未能提供有效的机制来支持对已有代码的快速利用, 产品线工程被迫只能在前期活动中运用[3]. 两者都未能实现在全局视角下开发过程的整体优化. 因此, 为了使组织的构件选择更具效率, 达到有效进行系统化复用的目的, 首先需要对构件—产品线实现的相关操作及其影响因素进行深入分析, 抽象出典型的复用策略, 提供一个能够涵盖软件产品线开发全部活动的系统分析模型.

从宏观上讲, 软件产品线的开发由核心资产循环和产品资产循环两部分组成[8]. 本文以此两个循环为切入点, 对涉及的领域工程和应用工程两阶段操作进行定义, 得到软件产品线开发的两阶段模型, 如图1所示.

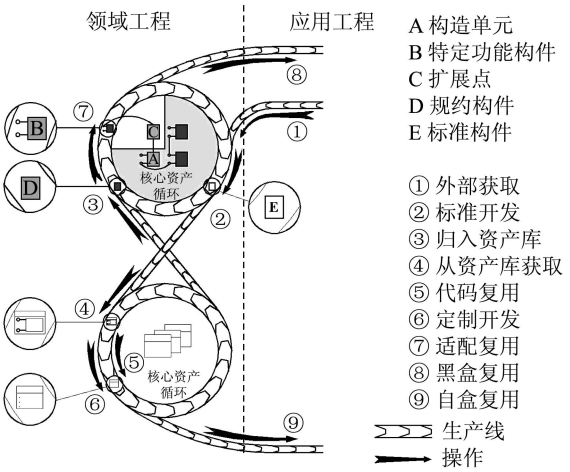

图 1 软件产品线开发的两阶段过程模型

Fig. 1 The two-stage procedure model for reuse scenarios selection in SPL

上图描述了产品线开发的两个主要过程: 领域工程和应用工程. 轨道代表软件产品的虚拟生产线, 承载着从构件选择到虚拟产品线描绘的全过程. 对核心资产的复用⑧是通过组织市场购买外部资产①或从系统外部建立新的可复用资产②获取元件, 经系统适配⑦以形成新的应用; 同理, 对以往成功软件产品的复用⑨可以经由方式⑤和⑥为新产品提供构件. 无论资产以何种方式获取, 其目标都是将其复用⑧和⑨到目标系统中以满足特定的系统需求. 值得一提的是, 此二循环之间并非相互孤立, 而是能够通过标准化或定制化操作③和④交换资产的.

本文的工作在假设核心资产库已建成的基础上展开, 其中的资产可被其他产品存取, 并能挖掘(mining)和规约(cataloging)其他软件产品(包括相关文档)归类到该库中. 对软件产品线开发的两过程分析如下:

2.1 领域工程阶段(Domain engineering)

领域工程是为一组相似或相近系统的应用工程建立基本能力和必备基础的过程[1], 它覆盖了建立

可复用软件资产的所有活动[17]. 产品线方法取得成功的关键因素是优质的可重用“核心”. 而核心至少应包括一个参考框架. 从领域工程的角度来看, 框架是实例化的DSSA(domain specific software architecture), 反映一个软件系统族的体系结构, 并为其提供基本的构造单元(framework component), 同时定义针对特定功能构件(application component)需求在何处进行调整和修改, 即扩展点(hot spots)[18]. 这一过程本文定义7种操作(operations)方式:

1) 外部获取(external utilization) 从组织外部(主要指COTS(commercial off-the-shelf)市场获取符合某产品特定功能接口(或规约)的构件;

2) 标准开发(made for reuse). 自行开发符合标准构件接口定义的可复用构件, 也称面向复用的开发;

3) 归入资产库(mining and cataloging). 将构件通过检索、分析、获取、标准化等一系列过程规约到核心资产库;

4) 从资产库获取(cataloged and commission). 通过检索、评估组织资产库中的构件, 为目标产品选择可复用构件. 这部分成本往往很小, 但需要考虑许可授权的成本(license or royalty fees);

5) 代码复用(copy and paste). 从成功案例中复用代码级资产到目标项目. 从长远角度考虑这一操作对可复用资产的积累没有多少好处;

6) 定制开发(made to order). 为特定产品而使用的从零代码开发产品的方式. 这一操作产品质量高, 但由于开发成本和工时都比较高昂, 目前较少使用这一操作;

7) 适配复用(adaptation for reuse). 对从核心构件库中获取资产进行是系统理解的过程, 包括对运行系统、源代码、设计、分析、文档等的全面理解[1]. 也称其为构件再工程.

2.2 应用工程阶段(Application engineering)

应用工程是对所拥有的可复用构件资产(领域工程输出的成果)在预先分析、设计和编码后, 进行相应的适配、组装、测试等一系列过程, 使之完成特定的目标产品. 从复用角度看, 主要包括两种操作方式:

1) 黑盒复用(black-box reuse). 它与白盒复用的不同之处在于其不需修改构件资产就可以复用于目标产品;

2) 白盒复用(white-box reuse). 对已有资产进行适应性修改, 使其满足目标产品的功能要求, 是代码级复用的主要方式.

2.3 典型复用策略(Typical reuse scenarios)

通过以上对面向产品线的软件开发的领域工程和应用工程两阶段及其主要操作的定义, 可抽象出软件产品线过程中的典型复用策略描述, 如图2所示.

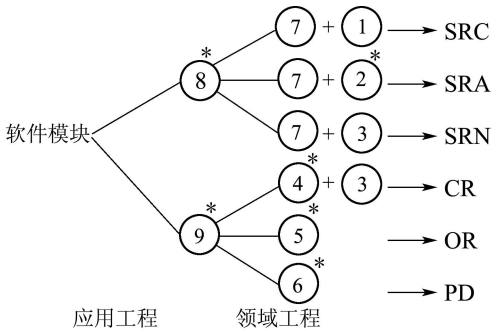

图 2　典型复用策略及其影响因素

Fig. 2　Typical reuse scenarios and effect factors in SPL

其中: CR(controlled reuse), OR(opportunistic reuse)和PD(pure development)为机会主义复用. 这一类复用的特点是成本相对较小, 但由于仅关注于对目标产品特定需求的适配、修改工作, 其复用投资往往不能使产品线中的其他产品受益. SRC(systematic reuse-COTS)、SRA (systematic reuse-adapted)和SRN (systematic reuse-new)为系统化复用, 短期内其成本高于机会主义复用, 但该资产在未来有可能多次被复用, 即可带来平均成本的大幅下降, 故系统化复用是产品线开发的主要复用方式.

每一项操作属性值均由成本、工时、失败率3个系数值构成, 通过各项操作的系数值便可计算出优化模型中的各参数值. 图中带“*”的操作每次复用都需重新进行; 与此相对应, 不带“*”的操作不需重复.

3 产品线复用策略优化数学模型(Mathematical model for reuse scenarios selection)

下面考虑通过领域分析[1]将一组领域产品分为n个相对独立的软件模块来实现, 且每一模块的功能均使用构件技术实例化现有资产来完成的产品线开发优化问题(其中$i = 1, 2, \cdots, n; j = 1, 2, \cdots, 6$).

3.1 参数及变量(Variable and parameter)

参数:

q_{ij}—-第i个模块使用第j种复用策略实例化时实例的质量;

$\bar{q}_{ij'}$—-第i个模块使用第j'个COTS资产完成实例化时实例的质量;

c_{ij}—-第i个模块选择第j种复用策略开发时获取复用资产和开发工时所需要的费用;

$\bar{c}_{ij'}$—-第i个模块选择第j'个COTS资产所需要的

购置费用;

d_i—-第i个实例使用白盒复用策略时所需成本;

$\bar{d}_i$—-第i个实例使用黑盒复用策略时所需成本;

λ_{ij}—-第i个实例使用第j种策略时应用工程过程中的平均失败率;

$\eta_{ij'}$—-第i个模块使用第j'个COTS资产的平均失败率;

t_{ij}—-第i个模块使用第j种策略所需的开发时间;

$\bar{t}_{ij'}$—-第i个模块使用第j'个COTS资产的获取时间.

变量:

$$x_{ij}=\begin{cases}1, & \text{第}i\text{个模块选择第}j\text{种策略;}\\ 0, & \text{否则.}\end{cases}$$

$$\bar{x}_{ij'}=\begin{cases}1, & \text{第}i\text{个模块使用第}j'\text{种COTS资产;}\\ 0, & \text{否则.}\end{cases}$$

$$y_i=\begin{cases}1, & \text{第}i\text{个模块选择白盒复用实现;}\\ 0, & \text{否则.}\end{cases}$$

$$\bar{y}_i=\begin{cases}1, & \text{第}i\text{个模块选择黑盒复用实现;}\\ 0, & \text{否则.}\end{cases}$$

$z_{ij'}$—-第i个模块使用第j'个COTS资产需要的测试次数.

3.2 目标方程(Quality objective function)

由于软件产品线的强大之处在于其对产品重用性的提高. 显然, 系统地开发和复用低质量产品的组织是好景不长的, 这就要求可重用资产应尽可能具有最佳质量. 所以, 产品线质量目标函数被广泛认可的是构成系统的全部构件质量的加权求和形式[7]:

$$\max\sum_{i}^{n}\theta_i\times(\sum_{j=1}^{J}q_{ij}x_{ij}+\sum_{j'=1}^{J'}\bar{q}_{ij'}\bar{x}_{ij'}). \tag{1}$$

对于q_{ij}, $\bar{q}_{ij'}$参数的获取可通过国际软件质量标准ISO/IEC 9126或其面向构件和产品线的拓展标准[19~22]得到. θ_i类似于传统软件产品设计中的模块权重, 这里理解为模块的核心程度. 在面向产品线开发时, 主要反映模块被调用频度.

3.3 成本约束(Budget constraint)

$$\sum_{i}^{n}(\sum_{j=1}^{J}c_{ij}x_{ij}+\sum_{j'=1}^{J'}\bar{c}_{ij'}\bar{x}_{ij'})+\sum_{i=1}^{n}(d_iy_i+\bar{d}_i\bar{y}_i)\leqslant B. \tag{2}$$

其中B为产品线的预算成本值, 对产品线开发成本参数c_{ij}, $\bar{c}_{ij'}$的估算方法与传统开发有所不同, 通常引入成本参数的贴现率(discount rate)、周期成本(periodic costs)、整合计算[23,24]等概念, 不详细展开. 通过(2)可以看出整个产品线的成本是围绕领域工程和应用工程两部分成本所构成的.

3.4 交付时间约束(Delivery time constraint)

$$T_i=(\sum_{j=1}^{J}t_{ij}x_{ij}+\sum_{j'=1}^{J'}\bar{t}_{ij'}\bar{x}_{ij'}+\tau z_{ij'})\leqslant T. \tag{3}$$

T为预期交付时间. $\bar{t}_{ij'}$从构件产品经销商处获取, τ为一次测试所需要花费的平均时长. 这里假设全部模块并行开发, 所消耗的总工时为所需开发时间最长模块的工时.

3.5 可靠性约束(Reliability constraint)

产品线的两阶段故障率均视为服从泊松分布[4,6]. 以下分别确定其失败强度.

3.5.1 领域工程失败强度(Reliability constraint)

基于构件技术的软件产品线模型, 可靠性具有如下特点:

1) 不能精确地预测构件(主要指COTS. 内部构件资产的失败强度如上章所述可直接获得, 并归入到应用工程阶段)的执行环境和用户使用模式. 故构件开发者不可能做到完全彻底地构件测试, 且很难确定何时结束测试;

2) 构件复用者和第3方测试人员通常无法得到构件的源代码及详细设计知识, 对构件只能进行黑盒测试. 易使构件执行过程中的一些错误被隐藏, 很难应对高可靠性需求;

3) 软件产品线中核心构件资产的可靠性保证要求高于传统开发方式, 因此在领域分析优化模型中需要考虑测试力度、测试时间等核心资产开发重要的影响因素.

引入易测试函数$\rho_{ij'}(z_{ij'})$来表示第i个模块使用第j'个COTS时, 在经过$z_{ij'}$次测试后能够无故障运行的概率. 它能为测试计划确定每个模块所需要的测试强度提供参考, 可以解决在产品开发不同模块策略选择时, 处于生命周期靠后的测试强度及所需付出代价的估计问题. 关于此函数的确定, 本文提出基于贝叶斯理论的建立方法在下一章详细介绍.

$1-\rho_{ij'}(z_{ij'})$即是第i个模块使用第j'个COTS的失败强度.

3.5.2 应用工程失败强度(Probability of failure on demand for application engineering)

假设完成各软件模块的技术条件与人员水平不变. 从时间角度分析应用工程阶段的可靠性通常从以下方面考虑: 平均故障间隔时间(mean time between failures, MTBF)、平均失效前时间(mean time to failure, MTTF)、系统平均不工作间隔时间(mean time between system downs, MTBD)、平均修复时

间(mean time to repair, MTTR). 应用工程的平均失败率本文选择广泛使用MTBF(即$\lambda_{ij}=1/\mathrm{MTBF}$)描述, 这样$\lambda_{ij}t_{ij}$即为第$i$个模块使用第$j$策略时的失败强度. 得到对于第$i$模块的平均失败强度及产品线可靠性的约束方程:

$$\begin{cases} X_i=\sum\limits_{j=1}^{J}\lambda_{ij}t_{ij}x_{ij}+\sum\limits_{j'=1}^{J'}(1-\rho_{ij'}(z_{ij'})\bar{x}_{ij'}). \\ 1-\prod\limits_{i=1}^{n}\exp[-\theta_i X_i]\leqslant R. \end{cases} \tag{4}$$

其中R为产品线的可容忍失败率. 为了简化模型, 考虑指数函数线性化的过程. 将上式变形为

$$1-\exp[\sum_{i=1}^{n}(-\theta_i X_i)]\leqslant R.$$

为了简化计算, 使用麦克劳林数列展开式的前两项对上不等式左边近似变换, 得到其线性化形式:

$$\sum_{i=1}^{n}(-\theta_i X_i)\leqslant R. \tag{5}$$

此外还需要考虑以下3个系统约束, 保证模型准确选择到复用策略及构件资产:

$$y_i\leqslant\sum_{j\in\dot{J}}x_{ij}, \tag{6}$$

$$\bar{y}_i\leqslant\sum_{j\in\ddot{J}}x_{ij}, \tag{7}$$

$$\bar{y}_i+y_i=1. \tag{8}$$

$\dot{J}$和$\ddot{J}$表示机会主义复用与系统化复用的策略集. 式(6)(7)保证当某一应用工程方法被选中作为产品线中模块的解决方案时, 相应的构件资产获取成本必须被考虑进产品总成本中. 式(8)保证对于任一模块无论黑盒还是白盒均只复用一次实现.

4　易测试函数及其线性化(Testability function and its linearization)

确定易测试函数, 并为了进一步揭示该函数意义且方便运算, 本部分将其转化线性形式.

假设A, B两事件分别表示:

A: 一定条件规则下, 模块进行$z'_{ij'}$次测试成功. 其中$z'_{ij'}=(1-\eta_{ij'})z_{ij'}$.

B: 模块无故障运行.

根据贝叶斯理论可得易测试函数为

$$\rho_{ij'}=\mathrm{P}(B|A)=\frac{\mathrm{P}(A|B)\mathrm{P}(B)}{\mathrm{P}(A|B)\mathrm{P}(B)+\mathrm{P}(A|\bar{B})\mathrm{P}(\bar{B})}.$$

通过分析得到如下相关事件的概率值

$$\begin{aligned} &\mathrm{P}(A|B)=1,\\ &\mathrm{P}(B)=1-\eta_{ij'},\\ &\mathrm{P}(A|\bar{B})=(1-\eta_{ij'})^{z'_{ij'}},\\ &\mathrm{P}(\bar{B})=\eta_{ij'}. \end{aligned}$$

代入易测试函数得:

$$\rho_{ij'}=\frac{1-\eta_{ij'}}{(1-\eta_{ij'})+\eta_{ij'}(1-\eta_{ij'})^{z'_{ij'}}}. \tag{9}$$

需要指明的是, 商业优化软件ILOG CPLEX (version 10.1)是世界领先并广泛使用的数学规划优化程序包, 但该软件仍未解决精确求解含指数函数优化模型的问题. 故本文通过数学推导对模型进行线性化处理, 使其操作与分析更具实际意义. 其次, 在实际基于构件的软件开发中, 参数$\eta_{ij'}$和$\rho_{ij'}$数量级很小, 往往处于千分位[6,13], 本文的线性化过程也是在这一客观条件下提出的, 具体推导证明过程如下.

将式(9)变形为

$$(1-\eta_{ij'})-(1-\eta_{ij'})\rho_{ij'}=\eta_{ij'}\rho_{ij'}(1-\eta_{ij'})^{z'_{ij'}}.$$

因为 $1-\eta_{ij'}\neq 0$, 所以

$$1-\rho_{ij'}=\eta_{ij'}\rho_{ij'}(1-\eta_{ij'})^{z'_{ij'}-1}.$$

对等式两边取自然对数:

$$\begin{aligned} &\ln(1-\rho_{ij'})=\\ &\ln\eta_{ij'}+\ln\rho_{ij'}+\ln(1-\eta_{ij'})^{z'_{ij'}}-\ln(1-\eta_{ij'}). \end{aligned}$$

即

$$\begin{aligned} z'_{ij'}\ln(1-\eta_{ij'})=&\ln(1-\rho_{ij'})-\ln\rho_{ij'}+\\ &\ln(1-\eta_{ij'})-\ln\eta_{ij'}. \end{aligned} \tag{10}$$

又因为

$$z'_{ij'}=(1-\eta_{ij'})z_{ij'}\geqslant 0,\ \ln(1-\eta_{ij'})<0,$$

所以等式右边恒为小于零, 即

$$\ln(1-\rho_{ij'})-\ln\rho_{ij'}+\ln(1-\eta_{ij'})-\ln\eta_{ij'}<0.$$

因为

$$\ln(1-\eta_{ij'})-\ln\eta_{ij'}>0,$$

所以$\rho_{ij'}$满足$1-\rho_{ij'}<\rho_{ij'}$, 可行域为$\rho_{ij'}\in(\frac{1}{2},1]$.

令$f(\rho_{ij'})=\ln(1-\rho_{ij'})-\ln\rho_{ij'}$, 可判断曲线$f(\rho_{ij'})$的拐点为$\rho_{ij'}=\frac{1}{2}$.

又因为$f(\rho_{ij'})$的1阶导数

$$f(\rho_{ij'})=\frac{1}{\rho_{ij'}-1}-\frac{1}{\rho_{ij'}}=\frac{1}{\rho_{ij'}(\rho_{ij'}-1)}<0.$$

所以$f'(\rho_{ij'})$趋近于$-\infty$　(由$\rho_{ij'}$的意义), 所以$f(\rho_{ij'})$在$\rho_{ij'}\in(\frac{1}{2},1]$上单调.

又由式(9)可知$\rho_{ij'}^{\min}=\lim\limits_{z_{ij'}^{\mathrm{suc}}\to 0^+}\rho_{ij'}=1-\eta_{ij'}$, 根据函数$f(\rho_{ij'})$上的点$(\rho_{ij'}^{\min},f(\rho_{ij'}^{\min}))$及$f'(\rho_{ij'}^{\min})$, 得函数$f(\rho_{ij'})$的近似直线估计方程$L(\rho_{ij'})$为:

$$L(\rho_{ij'})=$$
$$\ln\eta_{ij'}-\ln(1-\eta_{ij'})+\frac{1-\rho-\eta_{ij'}}{\eta_{ij'}(1-\eta_{ij'})}=\hat{f}(\rho_{ij'}).$$

由式(10)得

$$(1-\eta_{ij'})\ln(1-\eta_{ij'})z_{ij'}^{\rm tot}=$$
$$\hat{f}(\rho_{ij'})+\ln(1-\eta_{ij'})-\ln\eta_{ij'}=$$
$$\frac{1-\rho_{ij'}-\eta_{ij'}}{\eta_{ij'}(1-\eta_{ij'})}.$$

得到$(1-\rho_{ij'})$的线性变化形式, 如下:

$$1-\rho_{ij'}=\eta_{ij'}+\eta_{ij'}(1-\eta_{ij'})^2\ln(1-\eta_{ij'})z_{ij'}. \quad (11)$$

由式(11)知, 每增加一次测试对系统可靠性改善程度的函数值为$\eta_{ij'}(1-\eta_{ij'})^2\ln(1-\eta_{ij'})$. 该函数在定义域内单调递减, 说明$\eta_{ij'}$越小时, 对测试可靠性的改善程度越大.

综上, QOCRED优化模型即以(1)为目标函数, 式(2)∼(8)(11)分别为预算约束、预期约束、可靠性约束及系统约束. 有助于选择COTS/in-house, 还考虑了典型in-house开发策略的成本、可靠性、开发工时这3个主要影响因素, 解决在满足可获得成本预算与系统需求约束下的产品线质量最优问题.

5 实例计算(Case study)

本文给出一个运用此模型求解产品线优化过程的实际应用, 对于软件开发组件模块化过程研究已经相对成熟[25], 这里不再赘述. 案例为某大型电子邮箱服务企业开发4个产品. 通过领域分析, 可使用5个模块(mod)来实现相应需求, 关系图如图3所示. 并预设参数

$$\theta=\{0.2,0.3,0.3,0.15,0.05\},$$
$$B=4300,T=1000,R=0.005.$$

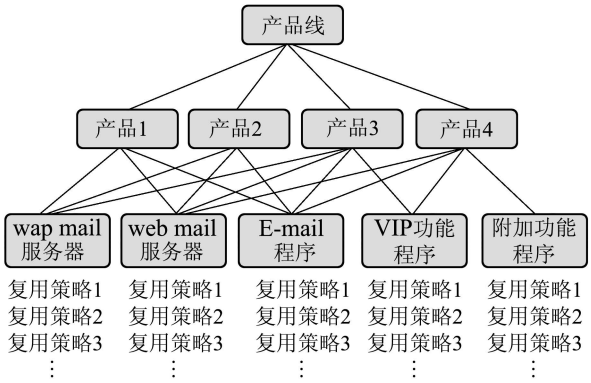

图 3 邮箱产品线层次结构图

Fig. 3 The hierarchy of E-mail sever system

以下4个表格列出了每一策略完成各模块的质量、成本、失败率及开发工时(单位为周). 限于篇幅, 在此未列出每项操作的具体参数.

表 1 各策略质量

Table 1 The quality of scenarios

	SRC	SRA	SRN	CR	OR	PD
Mod1	—	—	0.77	—	—	0.33
Mod2	—	0.84	0.64	0.64	0.43	0.54
Mod3	—	0.85	0.78	—	—	0.42
Mod4	—	0.72	0.88	0.57	0.58	0.34
Mod5	—	0.70	0.84	0.61	0.62	0.30

表 2 各策略成本

Table 2 The cost of scenarios

	SRC	SRA	SRN	CR	OR	PD
Mod1	100	—	1300	—	—	1300
Mod2	100	600	1000	500	300	1300
Mod3	100	600	1300	—	—	1300
Mod4	100	600	1200	500	300	1300
Mod5	100	600	800	600	500	1300

注 1 黑盒复用成本为200, 白盒复用成本为100.

表 3 各策略失败率

Table 3 The reliability of scenarios

	SRC	SRA	SRN	CR	OR	PD
Mod1/(10^{-5})	—	—	41	—	—	80
Mod2/(10^{-5})	—	35	41	57	77	80
Mod3/(10^{-5})	—	35	38	—	—	74
Mod4/(10^{-5})	—	25	38	43	71	74
Mod5/(10^{-5})	—	30	35	40	71	70

表 4 各策略工时

Table 4 The man-hour of scenarios

	SRC	SRA	SRN	CR	OR	PD
Mod1	10	—	90	—	—	140
Mod2	10	85	100	80	50	140
Mod3	10	85	100	—	—	140
Mod4	10	78	90	60	30	140
Mod5	10	78	90	60	30	140

表5, 6表示从构件提供商处得到的各构件质量、到货时间、故障率及价格.

表 5 构件质量

Table 5 The quality of COTS

	COTS					
	1	2	3	4	5	6
Mod1	0.88	—	0.23	—	—	—
Mod2	—	0.84	—	0.17	0.32	—
Mod3	—	—	—	—	0.38	—
Mod4	—	—	0.55	—	—	0.40
Mod5	—	—	—	0.45	—	0.40

表 6　构件的交货时间、价格、故障率

Table 6　The time, price, reliability of COTS

	COTS					
	1	2	3	4	5	6
交货时间	10	30	20	10	30	30
价格	800	600	100	800	300	300
出错概率/(10^{-4})	2	4	2	1	3	3

经过线性化处理后, 本模型为混合整数规划, 属ILOG CPLEX等数学规划程序求解范畴. 因此, 求解过程采用ILOG OPL-CPLEX语言编程实现(篇幅所限, 未列源码).

根据ILOG CPLEX输出的结果, 得到在开发预算、预期时间和可靠性保证的约束下, 最优质量目标为0.782, 领域工程使用的策略: $x_{11} = x_{22} = x_{32} = x_{42} = x_{55} = \bar{x}_{11} = 1$.

应用工程使用的策略: $y_4 = \bar{y}_1 = \bar{y}_2 = \bar{y}_3 = 1$, 另外对模块所需要的测试强度$z_{11}^{\mathrm{tot}} = 253$.

通过实例分析可以看出, 产品线开发较于传统单产品开发的成本和时间减少比率平均可达70%, 整个产品的可靠性增加了近1倍. 且使用优化模型可实现在开发预算、预期时间、可靠性约束下, 找到质量最优的软件产品线开发方案. 其中有3个模块使用(SRA), 表明内部复用构件资产的效果优于其他复用策略.

通过Mod2选择SRA方式开发可以看出, 尽管其与COTS2的质量与成本相同, 甚至COTS2的开发周期短于SRA, 但考虑到系统的可靠性和测试成本及开发工时问题, 最优解决方案仍选择了前者. Mod4也选择SRA是因为尽管原有资产的质量并不能保证高于新开发构件资产的质量, 但出于成本和开发时间的关系只能选择前者. Mod5选择OR是由于Mod5实现的功能相对较少, 直接白盒复用代码级资产可以缩短开发周期, 但对整个产品线周期的缩短是没有帮助的.

更进一步的分析可以发现, 尽管本问题中最优解仅使用了一个COTS, 但为保证系统高可靠性(0.05)所耗费的测试时间就已占用掉整个产品线开发时间的13%. 所以在规划中绝对不可忽略测试所占用的产品开发时间, 在产品线工程计划的前期阶段就必须将其考虑进来.

6　结论(Conclusions)

文章通过分析产品线—-构件复用的开发范型, 构造了基于产品线的软件开发两阶段模型, 归纳出6种涵盖不同粒度的典型复用策略及其实现方式. 进而提出QOCRED模型, 在考虑开发过程中成本、开发时间、故障率的情况下进行可复用构件的策略选择, 解决在满足可获得预算、开发周期、系统可靠性需求约束下产品线质量最优的问题. 提出在早期的领域分析优化模型中就考虑生命周期靠后的测试强度设计问题, 并基于贝叶斯理论构建了易测试函数, 提供了一种确定产品开发中测试强度的估算方法.

参考文献(References):

[1] 杨芙清, 王千祥, 梅宏, 等. 基于复用的软件生产技术[J]. 中国科学, 2001, 31(4): 363 – 371.
(YANG Fuqing, WANG Qianxiang, MEI Hong, et al. Production technologies based on software reuse[J]. *Science in China(E Volume)*, 2001, 31(4): 363 – 371.)

[2] 杨芙清, 梅宏, 李克勤. 软件复用与软件构件技术[J]. 电子学报, 1999, 27(2): 68 – 75.
(YANG Fuqing, MEI Hong, LI Keqin. Software reuse and software component technology[J]. *Chinese of Journal Electronics*, 1999, 27(2) : 68 – 75.)

[3] COLIN A, JOACHIM B, CHRISTIAN B, et al. *Component-based Product Line Engineering with UML*[M]. Boston : Pearson Education, 2002.

[4] JUNG H W. Optimizing value and cost in requirement analysis[J]. *IEEE Software*, 1998(7): 74 – 78.

[5] BERMAN O, CUTLER M. Optimal software implementation considering reliability and cost[J]. *Computers & Operations Research*, 1998, 25(10): 857 – 868.

[6] VITTORIO C, FABRIZIO M, PASQUALINA P. *Automated Selection of Software Components Based on Cost/Reliability Tradeoff*[M] //Lecture Notes in Computer Science. Heidelberg: Springer Press, 2006: 66 – 81.

[7] 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.

[8] LINDA M N. SEI's software product line tenets[J]. *IEEE Software*, 2002, 19(4): 32 – 40.

[9] USAF Material Command, Electronics Systems Center. Software technology for adaptable, reliable systems[R] *//Hanscom AFB.Air Force/STARS Demonstration Project Experience Report(Version 3.1)*, 1996.

[10] LIM W. Reuse economics: a comparison of seventeen models and directions for future research[C] *//Proceedings of the 4th Intentional Conference on Software Reuse*. Washington DC: IEEE Computer Society, 1996: 41 – 51.

[11] CATAL C, DIRI B. A conceptual framework to integrate fault prediction sub-process for software product lines[C] *//2nd IFIP/IEEE International Symposium on Theoretical Aspects of Software Engineering*. China Nanjing: IEEE Computer Society, 2008: 99 – 106.

[12] STEFAN L, RUDOLF R, CHRISTIAN F, et al. Testing high-reliability software for continuous casting steel plants-experiences and lessons learned from Siemens VAI[C] *//33rd EUROMICRO Conference on Software Engineering and Advanced Applications*. Germany Lübeck: IEEE Computer Society, 2007: 255 – 262.

[13] BOEHM B, BROWN A W, MADACHY R, et al. A software product line life cycle cost estimation model[C] // *International Symposium on Empirical Software Engineering*. Southern California: IEEE Computer Society, 2004: 156 – 164.

[14] BOEHM B. Managing software productivity and reuse[J]. *IEEE Computer*, 1999, 16(9): 111 – 113.

[15] CLEMENTS P, NORTHROP L M. *Software Product Lines: Practices and Patterns*[M]. Boston: Pearson Education Addison-Wesley, 2001.

[16] TOMER A, GOLDIN L, KUFLIK T, et al. Evaluating software reuse alternatives: a model and its application to an industrial case study[J]. *IEEE Transactions on Software Engineering*, 2004, 30(9): 601 – 612.

[17] WILL T. *Confessions of A Used Program Salesman-Institutionalizing Software Reuse*[M]. New York: Addison-Wesley Publishing Company, 1995.

[18] 刘瑜, 张世琨, 王立福, 等. 基于构件的软件框架与角色扩展形态研究[J]. 软件学报, 2003, 14(8): 1364 – 1370.
(LIU Yu, ZHANG Shikun, WANG Lifu, et al. Component-based software frameworks and role extension form[J]. *Journal of Software*, 2003, 14(8): 1364 – 1370.)

[19] ISO 9126, Information technology-software product evaluation-quality characteristics and guidelines for their use[S]. *International Organization for Standardization*, 1992.

[20] CARVALLO J P, FRANCH X. Extending the ISO/IEC 9126-1 quality model with non-technical factors for COTS components selection[C] //*Proceedings of the 4th ICSE Workshop of Software Quality*. Toronto: IEEE Computer Society, 2006: 354 – 370.

[21] ANDREOU A S, TZIAKOURIS M. A quality framework for developing and evaluating original software components[J]. *Information and Software Technology*, 2007, 49(2): 122 – 141.

[22] JIN S H, JI H K, SANG H O, et al. A framework for evaluating reusability of core asset in product line engineering[J]. *Information and Software Technology*, 2007, 49(7): 740 – 760.

[23] MILI A, CHMIEL S F, GOTTUMUKKALA R, et al. Managing software reuse economics: an integrated ROI-based model[J]. *Annals of Software Engineering*, 2001, 11(1): 175 – 218.

[24] SANA B A, LAMIA L J, HENDA H B. Cost estimation for product line engineering using COTS components[M] //*Lecture Notes in Computer Science*. Heidelberg: Springer Press, 2005: 113 – 123.

[25] 董苗波, 孙增圻. 学习系统的一个统一的组件化框架[J]. 控制理论与应用, 2004, 21(6): 1041 – 1044.
(DONG Miao bo, SUN Zengqi. Unified componentialized framework of learning system[J]. *Control Theory & Applications*, 2004, 21(6): 1041 – 1044.)

作者简介:

吴志樵 (1981—), 男, 博士研究生, 主要从事质量系统工程方面的研究, E-mail: wuzhiqiao@sina.com;

唐加福 (1965—), 男, 博士, 教授, 博士生导师, 主要从事生产与物流运作优化、质量系统工程等方面的研究;

周进刚 (1979—), 男, 博士研究生, 主要从事软件复用与软件构件技术方面的研究.