软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2015,26(4):778−789 [doi: 10.13328/j.cnki.jos.004759]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

流程片段自适应重用策略研究*

杨 荣[1,2,3], 李 兵[4]

[1](武汉大学 计算机学院,湖北 武汉 430072)
[2](软件工程国家重点实验室(武汉大学),湖北 武汉 430072)
[3](湖北科技学院 计算机科学与技术学院,湖北 咸宁 437100)
[4](武汉大学 国际软件学院,湖北 武汉 430072)
通讯作者: 李兵, E-mail: bingli@whu.edu.cn, http://libingteam.com/

摘 要: 近年来,自适应软件是软件工程领域的研究热点.研究者们从不同角度对如何促进和提高软件系统的自适应进行了大量研究,有的以体系结构为中心研究软件的自适应,有的则从需求的角度进行研究.但是,从软件系统的流程片段自适应重用的角度来研究软件自适应,类似的研究工作还很少.借鉴软件控制论中的思想来研究流程片段的自适应重用,基于受控的 Markov 链模型来探讨流程片段的最优查询策略.提出了针对流程片段查询特殊应用环境下的 CMC(controlled Markov chain)模型,并对该模型进行了优化处理.基于逐次最小二乘法,进一步提出了流程自适应查询策略,该策略充分利用流程片段的历史查询信息,通过在线参数调整,能够帮助查询人员及时调整和优化查询策略.Matlab 环境下的仿真实验和真实流程数据集下的实验,共同验证了该模型和算法的有效性和可行性.

关键词: 自适应软件;流程片段重用;Markov 链
中图法分类号: TP311

中文引用格式: 杨荣,李兵.流程片段自适应重用策略研究.软件学报,2015,26(4):778−789. http://www.jos.org.cn/1000-9825/4759.htm
英文引用格式: Yang R, Li B. Study on strategy of adaptive process fragments reusing. Ruan Jian Xue Bao/Journal of Software, 2015,26(4):778−789 (in Chinese). http://www.jos.org.cn/1000-9825/4759.htm

Study on Strategy of Adaptive Process Fragments Reusing

YANG Rong[1,2,3], LI Bing[4]

[1](School of Computer, Wuhan University, Wuhan 430072, China)
[2](State Key Laboratory of Software Engineering (Wuhan University), Wuhan 430072, China)
[3](College of Computer Science and Technology, Hubei University of Science and Technology, Xianning 437100, China)
[4](International School of Software, Wuhan University, Wuhan 430072, China)

Abstract: Recent research on the self-adaptive software is one of the new focuses in the field of software engineering. The researchers pay more attention on how to improve the adaptation of software from different angles. While some focus on the architecture information, others pay more attention to the requirement. However as of now, there is little work about the process fragments reuse in self-adaptive software. This paper employs the idea of software cybernetics to study the process fragments reuse, and searches the optimal query method based on the model of controlled Markov chain. Firstly, a CMC model in the context of process fragments query is proposed, followed by subsequent optimizations. Then, a self-adaptive query strategy is addressed based on the iterative least square method. With the on-line parameter adjustment, this strategy utilizes the history of process fragment query to help people adjust strategies. The

* 基金项目: 国家重点基础研究发展计划(973)(2014CB340401); 国家自然科学基金(61273216, 61202031); 湖北省重大科技创新计划(2013AAA020)

收稿时间: 2014-08-01; 修改时间: 2014-10-14; 定稿时间: 2014-11-14

experiments in the context of Matlab and real process dataset validate the efficiency and feasibility of the model and algorithm presented in this paper.

Key words: self-adaptive software; process fragments reuse; Markov chain

自 20 世纪 90 年代起,自适应软件得到广泛的研究.如今,随着 Web 服务越来越普及,普适计算、网格计算等新型应用不断出现,使得软件系统的规模不断扩大、复杂性不断增强.越来越多的软件系统运行和部署在 Internet 上,应用系统从封闭性、静态性和可控制性逐步走向开放性、动态性和难控制性.伴随着软件技术的发展,如何适应外部环境和用户需求的动态变化,一直是一个难题.人们也一直在追求更具表达能力、更符合人类思维模式、更具演化性的软件模型[1].因此,在当今软件生态系统环境下,必须使软件系统能够在运行过程中对外部环境和用户需求的变化做出适当反应,从而获得一个令人满意的整体服务水平.

有不少学者从不同角度对自适应系统进行了定义:有学者提出,自适应软件是一类能够在运行过程中实时收集系统的各种变化信息,并根据预定义的策略,在适当时候能够自我调整的特殊软件[2];也有学者将自适应当作系统的一种能力,即,软件系统可以动态响应环境变化的能力,并自我调节以提高系统的性能[3];有的学者认为,自适应软件能够评估自身的行为,当评估结果表明系统目前的行为不能达到期望的目标,或者需要实现更好的功能或性能时,软件必须自我调整自身的行为[4].

关于自适应软件系统的研究,主要包括支撑自适应系统的基础理论和开发技术两个方面的研究工作.为了方便进行自适应研究,很多学者借鉴控制理论[5,6]中的概念和方法[7−10].Kokar 等学者将控制理论作为分析和设计自适应软件系统的数学基础[7].根据传统控制理论的思想,我们可以将自适应系统分为两部分,即,系统自身和环境.软件所处的环境是一个动态的生态系统,它的行为可看成是它以前的状态、作用于它的动作和时间的函数.系统自身通过不断地选择和调整动作来满足用户或系统设定的目标.Cheng 等人提出,可以学习控制工程,特别是反馈控制中的思想来研究自适应系统[8],他们将反馈循环分为 4 个阶段:收集、分析、决策和动作,并深入分析了每个阶段必须考虑的问题和可能面临的挑战.总结现有以控制论为理论基础的自适应系统研究,我们可以将软件系统当作一个“运行-收集-反馈-决策-调整-再运行”的循环过程.虽然基于“收集-反馈-控制”可以构成一个完整的反馈和控制回路,但是,目前的自适应系统研究还不够成熟,要形成稳定的框架体系结构还有很长的路要走[2].同时,关于自适应系统的开发技术方面,学者们提出了基于模型、反射式中间件、体系结构和 Agent 等技术.但是现有的研究工作还存在很多局限性,比如软件实体抽象、自适应逻辑表达等方面的局限性.

本文从另外一个角度来探讨软件的自适应问题,即,关于软件系统中的流程片段重用问题.现代网络等技术的发展,为不同组织的人们共同解决同一问题提供了契机.服务计算(service-oriented computing,简称 SOC)广泛采用 Web 服务来解决企业或组织之间的互操作性和动态协作性问题.伴随着云计算和物联网的兴起,我们正进入一个“万事皆服务”的信息时代.但在实际应用中,单个服务通常不能满足要求,必须将多个服务个性化地组合起来,形成满足复杂功能的服务流程.服务流程不仅是协调服务和资源的手段,它也是一种可共享、可复用的解决问题的知识.在科学、商业等领域,服务流程的成功应用实践,都证明了共享和复用服务流程的重要性.比如,英国的 e-Science 计划致力于人类疾病基因科学研究,该计划通过服务流程实现了科学家群体之间的共享并促进了科学的发展[11].使用已有的流程知识,并考虑自身需要快速构建服务流程,已成为增强企业敏捷性和竞争力的重要途径.很多大的企业集成软件提供商(如 SAP)将一些常用的流程片段抽取出来,并保存在流程库中,这样可以提高定制组合业务流程的效率[12].本文研究的流程片段重用,其应用场景类似于文献[12],用以解决如何从流程库中进行流程片段的最优重用查询问题.

本文采用受控 Markov 链(controlled Markov chain,简称 CMC)来建模流程片段查询策略,并设计出一个自适应流程片段重用模型,研究如何以最小的期望成本检测并找到满足要求的可重用服务流程片段.本文借鉴软件控制论的思想对制约条件进行一系列转换,提出一种资源约束的受控 Markov 链模型,该模型具有很好的性能.根据该模型,设计人员能够高效且低成本地找出可重用流程片段,该片段能够满足指定的功能要求.

为了利用流程片段的历史使用记录信息,再通过在线参数估计对重用策略进行在线调整,以构造流程片段的自适应重用策略.

本文第 1 节通过一个流程模型引出我们的研究问题,即,关于流程片段任意粒度的重用查询问题.第 2 节建模 CMC 流程片段重用模型.第 3 节构造 CMC 自适应流程片段查询策略.第 4 节为实验分析.第 5 节为相关工作介绍.第 6 节总结全文并提出未来的工作.

1 启发性流程实例

我们首先通过一个流程片段实例,引出本文的研究动机.图 1 为一个“一日游”服务流程,通过该服务过程,能够帮助旅行者合理地、划算地安排行程.该服务流程包括天气查询服务、景点搜索、自行车租赁、汽车租赁和旅行代理这 5 个功能服务.为了简洁起见,分别用 S_1,S_2,S_3,S_4 和 S_5 表示.不失一般性,仅仅考虑每个功能服务的输入和输出.根据 5 个服务之间的输入/输出依赖关系,可以构建该服务流程的依赖图.

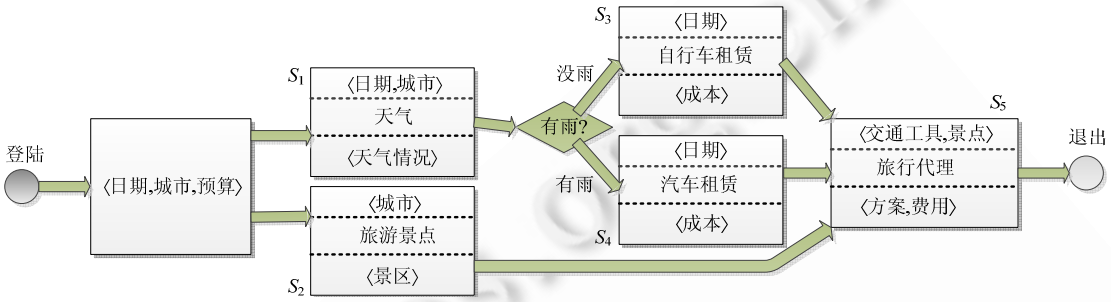

Fig.1 An example of process

图 1 一个流程实例

在某一时刻,如果 S_1 崩溃或失败,按照传统服务组合流程的方法,必须为用户重新构建整个服务流程,尽管可以重用其他子流程片段.比如,我们可以重用 S_3~S_5,S_4~S_5 或 S_2~S_5.事实上,可以重用任意粒度的子流程片段.当用户不断创造服务流程后,服务流程片段的共享和重用便成为一个重要的问题.如果缺乏一个好的策略来进行指导,用户将很难从流程库中找到合适的流程片段.因此,流程用户迫切需要一个科学的策略来协作寻找、理解和运用现存的服务流程资源.

本文所解决的流程片段重用问题与传统的服务组合流程既有联系,但同时也存在很大的区别.

- 首先,两者都是由一组子服务组成,比如由一些独立的个体 Web 功能服务组成,只有这些子服务一起执行才形成价值增值的服务;
- 其次,本文中的流程片段和传统服务组合流程都考虑了子服务 QoS(quality of service)属性,并都可以进一步分析整个流程的属性;
- 最后,本文工作与传统服务组合的不同之处在于:传统服务组合流程是在用户给出请求时,从 Web 服务库里发现或找出满足需求的一组独立的 Web 服务,然后组合起来去匹配用户的需求;但是本文所指的流程片段假设都是已经存在的,或者已经提前保存在流程库中[12],同样对于用户提出的请求,我们只需从流程库中查询满足需求的流程片段集合,而不是像传统服务组合流程那样从零开始构造.

接下来,本文将运用受控 Markov 链模型来解决这一问题.

2 CMC 流程片段重用模型

根据软件控制论思想,将流程片段重用当作一个控制问题,满足指定功能要求的抽象组合流程(abstract composite process,简称 ACP,$ACP=\{C_1,C_2,\ldots,C_m\}$,其中,每个组件服务 C_i 都有一批相应的候选原子服务实现)作为受控对象,重用策略作为相应的控制器.这样,二者形成一个闭环反馈系统,根据已知重用流程的约束目标,不断地优化重用策略.

2.1 流程片段模型描述

在一个服务流程系统中,对于每个组件服务 C_i,假设有 n_i 个候选服务去实现它,表示为 $C_i=(c_1^i,c_2^i,...,c_{n_i}^i)$. 假设 $S_P=(c_{j_1}^1,c_{j_2}^2,...,c_{j_m}^m)$ 代表一个具体的组合流程,它能够满足给定的抽象流程 ACP 的功能要求,其中,每个 $c_{j_i}^i$ 都是 C_i 中的一个元素.所有的组合方案形成一个组合流程空间 $S=(C_1\times C_2\times\ldots\times C_m)$.在本文的研究中,不失一般性,假设在流程库中包含任意长度的流程片段.对于每个服务流程,都具有相同类型的 QoS 属性,比如响应时间、吞吐量等.用一个向量 $A=(a_1,a_2,\ldots,a_L)$描述流程的 L 个属性. $F:(c_{j_1}^1,c_{j_2}^2,...,c_{j_m}^m)\to S_P$ 用来定义关于流程中候选服务属性的汇聚函数,通过该函数可以获得关于流程的属性值,并通过这些值可以定义条件约束.

用户在提出流程查询请求时,都会附带一些 QoS 约束.假设用 R 定义由 p 个属性组成的 QoS 属性约束向量, R 中的每个元素由一个三元组$\langle a_r,op_r,bound_r\rangle$构成,其中,$a_r$ 表示第 r 维 QoS 属性,op_r 是比较操作符(比如>,<,≤,≥,=),$bound_r$ 是 QoS 约束的边界值.定义一组约束函数向量, $F_R=\langle f_1^R,f_2^R,...,f_i^R,...,f_p^R\rangle$,其中, f_i^R 表示第 i 个约束属性的汇聚函数.用 R_i 定义由 f_i^R 计算出的值,则对于一个流程 $S_P\in S$,它的约束值向量定义为 $V_R=V_R(S_P)=(R_1,R_2,\ldots,R_i,\ldots,R_p)$.因此,通过比较流程片段 S_P 的每项 QoS 值 R_i 及其对应的边界约束值 $bound_i$,可以验证 S_P 是否满足给定的属性约束条件.

2.2 CMC流程片段查询模型

关于服务流程片段重用,为了后面建模简单,通过以下模型描述可以构建 CMC 流程片段重用模型.

1) 在搜索流程片段时,要用到一些资源,比如 CPU、内存等,假设总共有 X_0 单位数目的可用资源.

2) 假设所有的流程片段都存放在一个库中,即流程库.对于每一个具体的流程片段查询请求,由于功能已经确定,因此总共的满足该功能要求的流程片段总个数一定,假设为 N(N 可以任意大).很显然,查询人员开始不清楚满足要求的流程片段数量,不可能在一次查询中搜索到所有的流程片段,而且这样做也极大地浪费了资源.只有随着不断地查询,在拥有了一定规模的使用历史记录时,满足特定要求的流程片段数目才大概可以确定.假设每次搜索到的流程数目为一个期望值 $n(n\leqslant N)$.

3) 对于库中的每个流程片段 spf,它或者没有被搜索到,或者被查询出但未被使用过,或者被查询出且存在使用历史记录,则 spf 的 3 个状态可如下表示:

$$spf_t^i=\begin{cases}0, & t\text{时刻第}i\text{个流程片段未被搜到}\\ 1, & t\text{时刻第}i\text{个流程片段被找出,但未被使用}\\ 2, & t\text{时刻第}i\text{个流程片段被找出,且被使用}\end{cases}\tag{1}$$

其中,$i=1,2,\ldots,N$ 且 $spf_0^1=spf_0^2=...=spf_0^n=...=spf_0^N=0$.

4) 在任何时间点 t,所有库中满足指定功能要求的流程片段各自的状态,一起构成了整体流程重用状态,表示为 $\zeta_t=(spf_t^1,spf_t^2,...,spf_t^n,spf_t^{n+1},...,spf_t^N,X_t)$,其中,X_t 表示 t 时刻剩下的查询资源.根据前面第 2)条的假设,每次查询不可能搜索到所有满足功能约束的流程片段.每次搜索到的流程片段数目期望值为 n,剩下的 $N-n$ 个流程片段在这次查询中被查询出的概率很小,因此可当作一个小概率事件而不予考虑,则 t 时刻的整体流程重用状态可重新描述为 $\zeta_t=(spf_t^1,spf_t^2,...,spf_t^n,X_t)$.

5) 为了查询出可被重用的流程片段,需要很多的决策.假设总共有 $q+2$ 个不同的决策,定义决策集为 $D=\{1,2,\ldots,q,q+1,q+2\}$,其中,决策 1~$q$ 用于查询流程,即,有 q 个查询用例;决策 $q+1$ 用以初始化,并开始查询;当所有查询资源用完,或查询出的流程片段集合已经完全达到功能要求并满足属性约束时,决策 $q+2$ 用于退出查询过程.在任何时刻,调用任何一个决策 D_t 都会消耗 $Cost_{\zeta_t}(D_t)$ 代价的资源,有如下定义:

$$Cost_{\zeta_t}(D_t=i)=\begin{cases}Cost_{\zeta_t}(i)>0, & \text{如果}X_t>0\\ \infty, & \text{其他}\end{cases}\tag{2}$$

其中,$1\leqslant i\leqslant q$.

$$Cost_{\zeta_t}(D_t=q+1)=\begin{cases}Cost_1, & \text{若查询出的流程达到了要求或者}\min\{Cost_{\zeta_t}(1),Cost_{\zeta_t}(2),...,Cost_{\zeta_t}(q)\}>X_t\\ \infty, & \text{其他}\end{cases} \tag{3}$$

$$Cost_{\zeta_t}(D_t=q+2)=\begin{cases}\infty, & \text{若查询出的流程未达到要求且}\min\{Cost_{\zeta_t}(1),Cost_{\zeta_t}(2),...,Cost_{\zeta_t}(q)\}\leqslant X_t\\ Cost_2, & \text{其他}\end{cases} \tag{4}$$

公式(3)表明:如果已经搜索出满足给定要求的流程片段,或者从第 1 个决策到第 q 个决策它们所花费的成本的最小值大于此时剩下的资源,则此时花费常量 $Cost_1$ 的成本执行决策 q+1,重新初始化并启动新的搜索过程.

公式(4)说明:如果还没有查询出满足要求的流程片段或流程片段集,并且从第 1 个决策到第 q 个决策它们所花费的成本的最小值小于此时剩下的资源,则此时不能执行决策 q+2;否则以常量成本 $Cost_2$ 退出.换句话说,若此时不能执行决策 q+2,则只能执行决策 1,2,…,q,q+1,然而,执行决策 q+1 的条件是找到满足要求的流程片段,或者从第 1 个决策到第 q 个决策它们所花费的成本的最小值大于此时剩下的资源,但是此时条件恰恰相反,因此只能执行从 1 到 q 的查询决策.

当搜索到流程片段 S_P 的时候,根据前面第 2.1 节的定义,用它的每项 QoS 值 R_i 和其对应的边界约束值 $bound_i$ 比较,可以验证 S_P 是否满足给定的属性约束条件.再加上前面几个公式定义,可以得出属性约束和剩下的资源共同决定了查询过程的终止.

6) 对于 1 到 q 中的每个决策,都以一定概率查询出每个满足功能约束和属性约束的流程片段,q 个概率查询向量构成了下面的概率矩阵:

$$Q=\begin{bmatrix}\rho_1^1 & \rho_1^2 & \cdots & \rho_1^n\\ \rho_2^1 & \rho_2^2 & \cdots & \rho_2^n\\ \vdots & \vdots & \cdots & \vdots\\ \rho_q^1 & \rho_q^2 & \cdots & \rho_q^n\end{bmatrix}.$$

上面概率矩阵中的每一行对应一个决策的查询概率向量,即,ρ_i^j 表示第 i 个决策查询到第 j 个流程片段的概率.用 f_t 表示 t 时刻是否搜索到一个满足功能要求和约束的流程片段,对于 f_t,可以有下面的状态概率计算公式:

$$P\{f_t=1\mid D_i=i\}=\sum_{k=1}^{n}\rho_i^k,P\{f_t=0\mid D_i=i\}=1-\sum_{k=1}^{n}\rho_i^k \tag{5}$$

检测到 n 个流程片段中第 r 个流程的概率为

$$P(spf=r)=\sum\nolimits_{k=1}^{q}\rho_k^r \tag{6}$$

7) 为了加快搜索速度,除了原始的流程库以外,另外构建一个流程库,用来存放新近被使用或使用频率比较高的流程片段.因此对每个查询请求,优先在该新建库中查询.采用该策略能够加快流程查询速度,因为在一个公司内部,其核心业务流程或操作最频繁的工作流程基本上固定了,类似于经典的二八定律.在状态 ζ_t 下,采用的查询决策如果找出了一个以前未被搜索出的流程片段,因为该流程会被存入新建库中,则下次搜索速度就提高了.虽然本次查询会花费一定的代价,但却节省了以后查询的成本.从这个意义上说,可以认为,本次查询采用决策 D_t 查询出流程 spf 会产生一定的代价折扣,或者说本次查询减少了一定的代价,表示为 $\psi_{spf}(D_t|\zeta_t)>0$.决策 q+1 和 q+2 不会产生代价折扣.

通过前面的假设和定义,当用户提出某个流程查询请求,采用不同的查询决策进行搜索,经多次反复,最终,查询系统或者成功返回(找到满足功能约束和属性约束的流程片段),或者查询失败,即,没有满足要求的流程片段.我们将查询时间离散化,即,每次查询作为一次独立的个体,令 Γ 为查询系统返回的时间,则有如下等式成立:

$$X_0=n_1\times Cost_1+Cost_2+X_\Gamma+\sum\nolimits_{t=0}^{\Gamma}[Cost_{\zeta_t}(D_t)-\chi_{D_t}\psi_{spf}(D_t\mid\zeta_t)] \tag{7}$$

令

$$\chi_{D_t}(D_t=i)=\begin{cases}\sum_{k=1}^{n}\rho_i^k sign(spf_t^k-2)sign(-spf_t^k), & i=1,2,...,q\\ 0, & i=q+1,q+2\end{cases} \tag{8}$$

其中,$sign(x)$ 为三段符号函数,定义为

$$sign(x)=\begin{cases}1, & x>0\\ 0, & x=0\\ -1, & x<0\end{cases} \tag{9}$$

等式(7)中,前面两项都是常量.X_I虽然不是一个固定的值,但它是到某个时间点为止,所有采用的查询策略确定之后剩下的查询资源,因此也可以当作一个常值.对于每个查询策略,消耗的资源也是固定的,因此,令等式(7)中变化的部分为

$$\phi_D(X_0)=\sum_{t=0}^{\Gamma}\chi\psi_{spf}(D_t\mid\zeta_t) \tag{10}$$

其中,D 表示从查询系统启动到查询结束,所采用的所有查询策略,这些策略一起构成了针对某个用户请求的查询方案.$\phi_D(X_0)$表示查询过程结束时,D产生的代价总折扣.在前面定义的基础上,公式(10)就构建成了流程片段查询的 CMC 模型.本文的研究目的是要找到一个整体查询方案 D,使代价总折扣 $\phi_D(X_0)$ 最大,这样的查询方案就能以最低的、最划算的成本代价查询到既满足功能要求,也满足非功能属性要求的流程片段.

3 CMC 自适应重用流程片段查询策略

在第 2 节我们建模了一个 CMC 流程片段查询策略模型,本节将通过对模型的优化,进一步提出一个自适应流程查询策略模型.

3.1 查询决策优化

公式(10)对采用某个整体查询方案后所产生的代价总折扣进行了定义.根据 CMC 理论[13]可知:存在一个确定性静态查询策略,使得 $\phi_D(X_0)$ 最大.令 $H(\zeta)$是状态ζ的价值函数,本文的最优查询方案就是在此价值函数基础上进行选择的.

定理 1. 根据前面的模型 1)~模型 7),有:

$$H(\zeta_t)=\begin{cases}\min\limits_{1\leqslant i\leqslant q}\{Cost^*_{\zeta_t}(i)+\chi_i^*H(\zeta_{t+1}^j)+(1-\chi_i^*)H(\zeta_{t+1}^{j'})\}, & \min\{Cost_{\zeta_t}(1),Cost_{\zeta_t}(2),\ldots,Cost_{\zeta_t}(q)\}\leqslant X_i\\ 0, & \text{其他}\end{cases} \tag{11}$$

其中,$Cost^*_{\zeta_t}(i)=-\sum\limits_{\eta}p_{\zeta\eta}(i)\psi_{spf}(i\mid\delta)$,$p_{\zeta\eta}(i)=P\{\eta\mid\zeta,D_t=i\}=\sum_{k=1}^{n}\rho_i^k$ 为从状态ζ转换到η的概率,ζ_{t+1}^j 表示采用查询决策 D_t 查询到某个流程片段 spf 时状态ζ_t的下一状态,而 $\zeta_{t+1}^{j'}$ 表示 D_t 没有查询到流程片段时ζ_t的下一状态.

通过定理 1,可以得出:在任何时刻的查询决策选择,都应满足定理 1 中的价值约束;对每个流程查询请求,最后获得的整体查询方案,都是对查询决策集的一个选择和排列过程.

定理 2. 当 $\min\{Cost_{\zeta_t}(1),Cost_{\zeta_t}(2),\ldots,Cost_{\zeta_t}(q)\}\leqslant X_t$ 时,则有如下查询策略:

$$D^*_{\zeta_t}=\arg\min_{1\leqslant i\leqslant q}\{Cost^*_{\zeta_t}(i)+\chi_i^*H(\zeta_{t+1}^j)+(1-\chi_i^*)H(\zeta_{t+1}^{j'})\} \tag{12}$$

该策略即为针对查询过程中一系列约束条件的限制而得出的优化流程片段查询策略,它能够使得代价总折扣 $\phi_D(X_0)$ 最大.但在具体实施过程中,必须明确每个查询决策搜索到每个流程片段的概率,即要已知 ρ_i^k 的值,也就是要确定第 6)条中定义的矩阵 Q.很显然,要求出 Q,必须依赖于历史查询记录信息.因此,在公式(12)的基础上引入一个参数估计和调整模块,充分利用历史信息来对查询概率进行在线估计和调整,从而构造出一个流程片段自适应查询策略.

3.2 基于逐次最小二乘法的自适应流程查询策略

利用最小二乘法可以方便地求得未知的数据,并使得这些求得的数据与实际数据之间误差的平方和为最小[14].本文采用逐次最小二乘法来不断地获得未知流程片段的查询概率,该方法充分利用历史查询记录信息.将所有查询用例的查询情况构造成一个向量$\Omega=(\rho_1,\rho_2,\ldots,\rho_q)$.其中,$\rho_i$表示采用查询决策 i 时,它能查询到满足用户需求的某个流程片段或集合的概率.对于一个流程片段 spf,本文在考虑查询概率时只考虑两种状态,即:spf 处于原始流程库中时被查询到的概率(spf 在某个时间点还未被查询到),或者 spf 处于新建库中被查询到的概率(spf

在以前搜索中被查询到,且存入了新建库中).对于原始库中的流程片段,假设初始时候它们的查询概率一样,这个等概率估计值会随着后面的查询不断被更新,直至趋向于各自的真实值.假设在某个时刻ζ_i,对于用户提出的某个功能需求$Я$,查询系统从新建库中找出 n_0 个(假设总共有 N_0 个与$Я$匹配的流程片段)与$Я$匹配的流程片段,则对ρ_i有下面的等式:

$$\rho_i=\sum_{k=1}^{n}\rho_i^k=(n-n_0)\times\rho_i^{\circ}+\sum_{k=1}^{n_0}\rho_i^k \tag{13}$$

其中,$1\leqslant i\leqslant q$, ρ_i° 为存储于原始库中的流程片段的等概率值.为了计算方便,假设已将新建库中 n_0 个流程片段通过排序移到编号从 1 到第 n_0 的位置.根据逐次最小二乘法,可以对Ω进行在线估计.在公式(13)中,有两组未知变量,即 ρ_i^k 和 ρ_i°,有:

$$\left.\begin{aligned}\rho_i^k&=\frac{n_0}{N_0},k=1,2,\ldots,n_0\\ \hat{\rho}_i^{\circ}&=\frac{\hat{\rho}_i-\left(\sum_{k=1}^{n_0}\hat{\rho}_i^k\right)}{n-n_0}\end{aligned}\right\} \tag{14}$$

其中,$\hat{\rho}_i$ 和 $\sum_{k=1}^{n_0}\hat{\rho}_i^k$ 分别表示ρ_i和 $\sum_{k=1}^{n_0}\rho_i^k$ 的估计值.

这样,在最优流程片段查询策略(10)的基础上,再融入基于逐次最小二乘法的参数估计调整模块,便得到了本文的自适应流程查询策略.该流程查询策略方案很好地利用了历史流程片段查询信息来推断未知流程片段的查询信息.在软件系统中,该算法思想能够对设计者重用流程片段提供方法论的指导作用,并对流程片段的重用质量提供理论保障.自适应流程片段查询整体算法思想如图 2 所示.

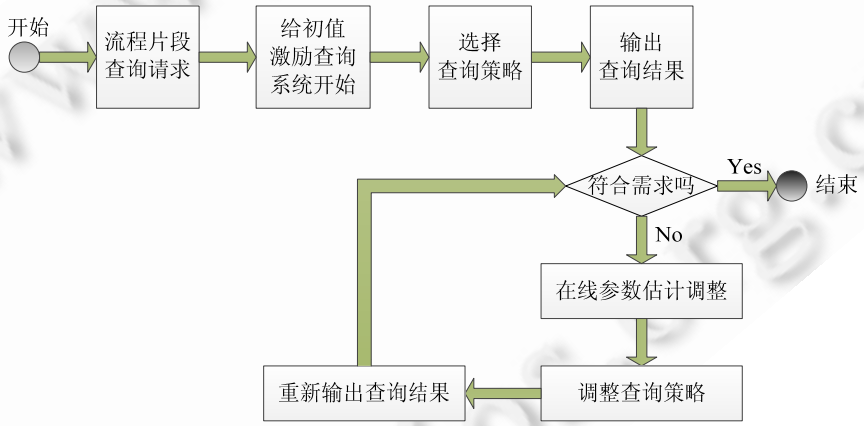

Fig.2 Flow chart of self-adaptive process fragments query

图 2 自适应流程片段查询整体流程图

4 实验分析

本节将通过两组实验来验证本文提出的自适应流程查询策略的有效性和可行性:首先,通过一组仿真实验验证其性能;然后,通过在真实流程数据环境下的查询实验,检验融入了本文提出的自适应查询策略以后对流程片段的查询会有怎样的性能影响.

4.1 仿真实验

本文首先用 Matlab 对流程片段自适应查询策略进行仿真.设开始有 40 单位数目的可用资源,即 X_0=40;有 20 个流程查询策略用例,即 D={1,2,…,18,19,20};设每次查询能够搜索到的流程片段数目期望值为 8,即 n=8,表示为一个集合$\{a_1,a_2,\ldots,a_8\}$,令查询它们所花费成本的均值分别为 $\mu_{a_1}=3,\mu_{a_2}=5,\mu_{a_3}=4,\mu_{a_4}=6,\mu_{a_5}=8,\mu_{a_6}=2,\mu_{a_7}=11,\mu_{a_8}=4$;假设前 18 个查询用例产生的代价相等,即,$Cost_{\zeta_i}(1)=Cost_{\zeta_i}(2)=\ldots=Cost_{\zeta_i}(18)=5$.

本文采用一个随机流程片段查询算法与上面提出的自适应流程查询算法比较,即,随机查询算法每次从查询用例集 D 中随机地选择查询用例.根据图 2,在自适应查询算法中,采用逐次最小二乘法,根据查询返回的结果对流程片段的查询概率进行在线估计.在实验中,对两种查询算法独立地各自仿真了 100 次,表 1 列出了前 15 次的仿真结果以及所有 100 次得出的平均值和标准差(表 1 中最后两列).

在表 1 中:$Re(i)$和 $Ae(i)$分别为第 i 次仿真实验中的随机查询算法和自适应查询算法所用的查询用例数;$Rp(i)$和 $Ap(i)$分别为第 i 次仿真实验中的随机查询算法和自适应查询算法搜索到的流程片段数;$Rd(i)$和 $Ad(i)$分别为第 i 次仿真实验中的随机查询算法和自适应查询算法产生的代价总折扣;$Mean$ 和σ分别代表平均值和标准差.从表 1 可以得出:整体上来看,自适应查询算法比随机算法要多用一些查询用例,但是自适应算法能够搜索出更多的满足需求的流程片段.而且非常有意义的是,自适应算法能够获得更大的代价折扣.从表 1 还可看出,自适应算法每项指标的标准差都小于随机算法的标准差,这说明自适应算法的稳定性更好.

Table 1 Results of front 15 times

表 1 前 15 次仿真结果

序号	i															$Mean$	σ
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15		
$Re(i)$	9	12	10	7	12	5	11	14	10	9	13	11	9	12	8	10.6	2.36
$Ae(i)$	13	15	9	13	14	10	11	11	13	12	10	14	11	10	11	12.9	1.82
$Rp(i)$	3	6	2	6	2	3	6	5	1	5	6	4	4	4	3	4.3	1.67
$Ap(i)$	8	8	5	4	2	6	7	6	5	7	6	6	6	4	5	6.2	1.53
$Rd(i)$	16	19	14	27	14	14	13	14	19	16	29	19	13	29	30	19.6	6.52
$Ad(i)$	29	22	25	28	32	25	32	34	24	22	26	26	28	30	21	28.5	3.88

4.2 真实数据集下流程查询实验

在上一节,我们通过仿真实验验证了本文提出的自适应流程查询算法的可行性和有效性,这一节通过真实流程数据下的实验进一步补充验证.首先,利用 2009 年的 Web 服务竞赛数据产生工具(Web service challenge testset generator)CTG[1](http://ws-challenge.georgetown.edu/wsc09/software.html)生成了一个包含 400 000 个流程片段的数据集,其中每个流程片段所包含的每个服务都拥有 5~10 个输入和输出参数.我们抽取出每个服务的两个 QoS 属性,即 response time 和 throughput(产生出的每个数据中已经包含这两个属性值).在实验中,将用到几个变量:用 AP 表示流程片段数目;用 AS 表示一个流程片段查询要求返回的流程片段中所包含的服务数目,AR 表示这些服务之间的关系.本节所有的实验用 Java 实现,且硬件为一台配置为 Intel(R) Core(TM) i5 CPU 760,2.80 GHz,4 GB RAM running Windows 7 (64-bit)的机器.

据我们所知,目前关于流程片段重用的研究工作还很少,尤其是关于任意粒度的流程片段重用的研究更少.本文的研究工作(用 SCKY-A 表示)建立在我们以前的研究工作(用 SCKY 表示)[15]基础之上(即,在 SCKY 框架之上融入本文的自适应模块),因此,实验中结合考虑以前的研究,同时与跟我们研究最类似的 VGI[16]进行比较.在实现时,我们规定:SCKY,SCKY-A 和 VGI 能搜索出的流程片段数目期望值为 20;当 SCKY-A 搜索到某个从未被查询到的片段时,以代价补偿变量随机产生的值进行补偿,其均值为 4、标准差为 1;总共有 60 个单位数目的可用资源;对于 SCKY-A,每次实验时,以均值为 5、标注差为 2 仿真出 20 个查询用例,仿真出的每个值表示查询用例的成本消耗;每次查询,随机产生出用户的 QoS 约束值.

第 1 组实验为有效性评估实验.在该组实验中,考虑两种问题规模下的查询串,即:AS=5 且 AR=4 和 AS=12 且 AR=11 两种规模的查询串.该组实验我们考虑流程片段的精确查找,即:搜索出的满足条件的流程片段,不仅

在长度上与查询串相等,而且结果串中每个位置上服务的功能必须与查询串中对应位置上要求的功能完全一致.我们分析每个流程片段查询在不同流程数量规模下(即 *AP* 变化),3 种算法各自需要的查询时间(即响应时间, response time).图 3 为该组实验的实验结果.图 3(a)和图 3(b)都得出一致的结论:给定一个用户的查询请求,随着 *AP* 的不断增长,即,从 100K 增加到 400K,3 种算法查询所需时间都不断增长;但是我们提出的两种算法(SCKY 和 SCKY-A)所花费的时间要远远小于 VGI 所花费的时间,我们的这两种算法都能在 80ms 以内返回结果;SCKY-A 算法由于加入了在线参数调整,因此比 SCKY 要多花费一点时间,但从图中可以看出,这个多花费的成本非常小,由我们的实验数据发现,它们之间的差距仅在 5ms 左右.另外,从图 3 也看出:问题规模越复杂(即,*AS* 的值从 5 变到 12),所需查询时间也会更长一点.

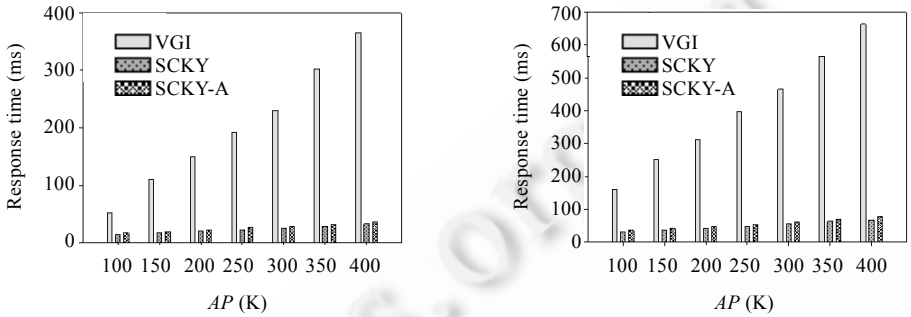

Fig.3 Efficiency evaluation with variable *AP*

图 3 效率评价实验结果

第 2 组实验为精确度评估实验.在这组实验中,我们仍然考虑流程片段的精确匹配,实验的数据集规模为 400K,即,我们取上面第 1 组实验中数据最大的那个流程片段库.当查询串的复杂度不同时(即,*AS* 的值以步长 1 从 6 增加到 12),我们分析 SCKY 和 SCKY-A 两种算法的精确度性能.对于 *AS* 的每个取值,我们变换查询需求,并对两种算法对应地各做 100 次实验,然后计算出各自满足用户需求的次数,最后与总次数 100 的比值就是我们要求的实验值.图 4 为两种算法的实验结果对比图,从图中可以看出:虽然两种算法的精确度都比较高,都在 80%以上,但在 SCKY 基础上,加入了本文提出的自适应调整模块而构造出的 SCKY-A 有明显更高的精确度,且 SCKY-A 比 SCKY 要相对地优 4%.不过,查询串的复杂度会对这两种算法的精度有一定的影响,从图中也可看出:随着 *AS* 的增大,两种算法的精确度会相对降低;但整体上,SCKY-A 的性能要稳定些.

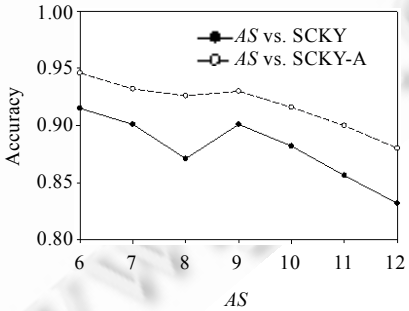

Fig.4 Accuracy evaluation with variable *AS*

图 4 随着 *AS* 变化的精度评估

5 相关工作

本文借鉴软件控制论中的思想[17]来解决流程片段的重用问题,即,采用受控 Markov 链 CMC 方法来建模流程片段的自适应重用模型.软件测试在整个软件开发周期中的地位非常重要,在学术界也有大量关于软件测试的研究,其中,利用 CMC 模型来研究软件的自适应测试取得了很好的成果.在文献[18]中,详细分析了软件控制论在软件测试中的可行性和有效性,同时,它也对测试中的部分条件进行了特殊化处理,即:这里所说的受控特性,比如被测软件包含的缺陷数一定等.而文献[19]则更具体地提出了一个测试资源约束下的模型,并促进了自适应软件测试的研究发展.文献[20]则提出:对不同的缺陷,检测到的概率不一样.本文正是在综合这些研究工作精华的基础上,提出用 CMC 方法来分析流程片段重用中的最优查询策略问题.而且,我们的流程片段重用问题与软件测试问题也有很多相似的地方,比如:一个软件版本中所含有的缺陷数是一定的,同样,对于流程库中满足某个流程片段查询请求的流程片段数目也是一定的;每个缺陷被发现的概率是不一样的,同样,每个流程片段被搜索到的概率也是不同的;每次软件测试不可能找出所有的缺陷,同样,我们也没有必要对每个流程查询请求,去在流程库中找出所有匹配的片段,而且这样做查询效率将非常低下.我们采用查询策略搜索出最优的片段或片段集合;随着软件版本的更新,将会引入新的缺陷,同样,随着网上 Web 服务的不断加入,我们也将获得新的满足特定应用的流程片段;对于软件测试,测试人员要考虑到测试成本和资源,同样,我们重用流程片段也要考虑代价.正是这些软件测试和流程片段重用之间的共同点,激发了我们借鉴软件测试中的思想来研究流程重用中的问题,同时也说明我们用 CMC 模型来解决流程重用是可行的.

本文要解决的问题是寻找最优流程片段重用策略,但是关于任意粒度的流程片段重用研究却很少.将一个大的流程碎片化是重用流程片段中一个非常关键的技术,有大量研究关于如何劈开和描述流程片段.例如:Schumm[21]研究了使用流程片段库在跨企业合作和应用集成中的潜在影响;Vanhatalo[22]将流程模型分解成单入单出的片段(single-entry-single-exit,简称 SESE),并用遗传算法对每个片段独立分析;而在文献[23]中,作者用形式化的方法将流程片段表示成不同的片段知识,并能对这些片段模型进行自由组合.但是这些工作都没有考虑如何搜索流程片段和如何提高流程查询的精度,他们主要关注过程中的复杂控制逻辑问题.本文不仅解决了流程片段的高速搜索问题,而且通过一个自适应的参数调整模块,能够充分利用历史查询记录信息,从而能够经济地对流程片段进行搜索并重用.

同时,关于任意粒度流程片段重用研究,我们与文献[16]中的工作最相近,即 VGI.文献[16]对服务过程片段建立可变粒度索引的 SSM-Tree,并在此基础上进行搜索查询.但该工作存在节点度大小瓶颈的缺陷,而且也未考虑流程片段的 QoS 属性以及概率查询.在我们以前的工作[15]中,即 SCKY,我们借鉴自然语言中的 Cocke-Kasami-Younger(CKY)算法思想来进行任意粒度大小的流程片段重用查询.SCKY 考虑了精确查询、模糊查询以及概率查询,文献[15]中的实验也证明了 SCKY 性能比文献[16]中的 VGI 更优.本文在 SCKY 的基础上,从另外一个视角,即,借鉴软件控制论中的思想来研究流程片段的重用问题(SCKY-A).该工作充分考虑了用户的历史查询记录信息,并结合逐次最小二乘法来进行流程片段的自适应查询.在 SCKY 框架之上融入本文前面阐述的自适应模块,便构成了本文的基本框架.

本文的研究工作是对自适应软件系统研究的一个很好的补充,在一个业务复杂的软件系统中,如果我们能够很好地对业务流程进行重用,并能有效地重用任意粒度大小的流程片段,这将对整个软件的自适应性有重要的促进意义.关于自适应软件系统的研究,本文开始部分已有介绍,这里不再赘述.

6 结束语

本文借鉴软件控制论中的思想来研究流程片段的自适应重用问题,即,用受控的 Markov 链模型(CMC 模型)来寻找流程片段的最优查询策略.经过一系列制约条件的约束和假设,提出了针对流程片段查询特殊应用环境下的 CMC 模型,并对该模型进行了优化处理.基于逐次最小二乘法,本文进一步提出了一个流程自适应查询策略,该策略充分利用流程历史查询记录信息,通过在线参数调整,能够帮助查询人员及时调整和优化查询策略.

实验部分,首先通过Matlab环境下的仿真实验验证了本文模型的有效性和可行性;然后,真实流程数据环境下的实验更进一步验证了该自适应流程查询策略模型的有效性和高精确性.通过实验数据也可得出:基于在线参数调整的该自适应流程查询策略所需的额外时间成本很低,但却能够获得一个可观的精确度性能提高.

下一步的工作是继续优化该自适应流程查询策略算法,同时对本文的模型和算法进行扩展,使其能够更好地融入到流程片段任意粒度大小的重用环境里.

致谢 作者衷心感谢审稿专家提出的宝贵意见,感谢NASAC 2014大会上专家们的意见和建议.

References:

[1] Yang FQ, Mei H. Software technology of the Internet era. In: Proc. of the Technology Ministry 10th Science, Chinese Academy of Sciences Academic Report Workshop on BBS Information Technology Science. 2004. 11−30 (in Chinese). http://www.miit.gov.cn/n11293472/n11295227/n11312239/11646373.html

[2] Wang QX, Shen JR, Mei H. An introduction to self-adaptive software. Computer Science, 2004,31(10):168−171 (in Chinese with English abstract).

[3] AI-Jumeily D, Al-Zawi M, Hussain AJ, Dobre C. Adaptive pipelined neural network structure in self-aware Internet of things. Big Data and Internet of Things: A Roadmap for Smart Environments, 2014,546:111−136.

[4] Laddaga R. Creating robust software through self-adaptation. IEEE Intelligent Systems, 1999,14(3):26−29. [doi: 10.1109/MIS.1999.769879]

[5] Wiener N. Cybernetics: Or Control and Communication in the Animal and the Machine. Cambridge: MIT Press, 1948.

[6] Zhong QH, Fu MY. Modern Control Theory and Application. Beijing: Mechanical Industry Press, 1997 (in Chinese).

[7] Kokar M, Baclawski K, Eracar Y. Control theory-based foundations of self-controlling software. IEEE Intelligent Systems, 1999, 14(3):37−45. [doi: 10.1109/5254.769883]

[8] Cheng BHC, de Lemos R, Giese H, Inverardi P, Magee J. Software engineering for self-adaptive systems: A research roadmap. In: Proc. of the Software Engineering for Self-Adaptive Systems. 2009. 1−13. [doi: 10.1007/978-3-642-02161-9_1]

[9] Salehie M, Tahvildari L. Self-Adaptive software: Landscape and research challenges. ACM Trans. on Autonomous and Adaptive Systems, 2009,4(2):1−42. [doi: 10.1145/1516533.1516538]

[10] Brun Y, Serugendo GDM, Gacek C, Giese H, Kienle H, Litoiu M, Müller H, Pezzè M, Shaw M. Engineering self-adaptive systems through feedback loops. In: Proc. of the Software Engineering for Self-Adaptive Systems. LNCS 5525, 2009. 48−70. [doi: 10.1007/978-3-642-02161-9_3]

[11] Hey T, Trefethen AE. The UK e-science core programme and the grid. Future Generation Computer Systems, 2002,18(8): 1017−1031. [doi: 10.1016/S0167-739X(02)00082-1]

[12] Al-Mashari M, Zairi M. The effective application of SAP R/3: A proposed model of best practice. Logistics Information Management, 2000,13(3):156−166. [doi: 10.1108/09576050010326556]

[13] Derman C. Finite State Markovian Decision Processes. New York: ACM Press, 1970.

[14] Hu H, Wong WE, Jiang CH, Cai KY. A case study of the recursive least squares estimation approach to adaptive testing for software components. In: Proc. of the 5th Int'l Conf. on Quality Software. Los Alamitos: IEEE Computer Society, 2005. 135−141. [doi: 10.1109/QSIC.2005.1]

[15] Yang R, Li B, Wang J, He LL, Cui XH. SCKY: A method for reusing service process fragments. In: Proc. of the 21th IEEE Int'l Conf. on Web Services (ICWS 2014). [doi: 10.1109/ICWS.2014.40]

[16] Zeng C, Lu Z, Wang J, Hung PC, Tian J. Variable granularity index on massive service processes. In: Proc. of the 20th IEEE Int'l Conf. on Web Services. 2013. 18−25. [doi: 10.1109/ICWS.2013.13]

[17] Cai KY, Li YC, Jing T, Bai CG. Software testing in the context of software cybernetics. Acta Aeronautics ET Astronautica Sinica, 2002,23(5):448−454 (in Chinese with English abstract).

[18] Cai KY, Gu B, Hu H, Li YC. Adaptive software testing with fixed-memory feedback. Journal of Systems and Software, 2007,80(8): 1328−1348. [doi: 10.1016/j.jss.2006.11.008]

[19] Cai KY, Li YC, Ning WY, Wong WE, Hu H. Optimal and adaptive testing with cost constraints. In: Zhu H, ed. Proc. of the 2006 Int'l Workshop on Automation of Software Test. Los Alamitos: IEEE Computer Society, 2006. 71−77. [doi: 10.1145/1138929.1138944]

[20] Hu H, Jiang CH, Cai KY. Adaptive software testing in the context of an improved controlled Markov chain model. In: Proc. of the Annual IEEE Int Computer Software and Applications Conf. Los Alamitos: IEEE Computer Society, 2008. 853−858. [doi: 10.1109/COMPSAC.2008.186]

[21] Schumm D, Karastoyanova D, Kopp O, Leymann F, Sonntag M, Strauch S. Process fragment libraries for easier and faster development of process-based applications. Journal of Systems Integration, 2011,2(1):39−55.

[22] Vanhatalo J, Volzer H, Leymann F. Faster and more focused control-flow analysis for business process models through SESE decomposition. In: Krämer BJ, Lin KJ, Narasimhan P, eds. Proc. of the 5th Int'l Conf. on Service-Oriented Computing. Vienna: Springer-Verlag, 2007. 43−55. [doi: 10.1007/978-3-540-74974-5_4]

[23] Eberle H, Unger T, Leymann F. Process fragments. In: Proc. of the 17th Int'l Conf. on Cooperative Information Systems. 2009. 398−405. [doi: 10.1007/978-3-642-05148-7_29]

附中文参考文献:

[1] 杨芙清,梅宏.Internet 时代的软件技术.见:中国科学院技术科学部第 10 次技术科学论坛信息技术科学专题学术报告会论文集.2004.11−30. http://www.miit.gov.cn/n11293472/n11295227/n11312239/11646373.html

[2] 王千祥,申峻嵘,梅宏.自适应软件初探.计算机科学,2004,31(10):168−171.

[6] 钟秋海,付梦印.现代控制理论与应用.北京:机械工业出版社,1997.

[17] 蔡开元,李勇超.景涛,白成刚.软件测试的控制论方法.航空学报,2002,23(5):448−454.

杨荣(1980－),男,湖北巴东人,博士生,讲师,CCF 学生会员,主要研究领域为服务计算,软件工程.

李兵(1969－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为复杂网络,服务计算,软件工程,人工智能.