# 面向服务的自律系统的自律可信性评估

张海涛 王慧强
(哈尔滨工程大学计算机科学与技术学院 哈尔滨 150001)

**摘 要** 从面向服务的角度分析自律系统的自律可信性,提出基于服务请求-服务过程-服务响应的层次化评估体系,该指标体系具有可拓展性和规范性。并采用层次分析法和模糊综合评估法,实现自律评估的定量分析和计算。最后通过实例分析证实了指标体系的有效性和可行性。

**关键词** 自律评估,自律可信性,服务,层次分析法
**中图法分类号** TP393 **文献标识码** A

## Autonomic Dependability Evaluation of Autonomic System for Service-oriented

ZHANG Hai-tao WANG Hui-qiang
(Department of Computer Science and Technology, Harbin Engineering University, Harbin 150001, China)

**Abstract** Based on service-oriented aspect, a hierarchy evaluation system of service request-service process-service responsibility was proposed, which analyses autonomic dependability of autonomic system. This index system has extensibility and standardization. According to the result of test, the index system is valid and possible.

**Keywords** Autonomic evaluation, Autonomic dependability, Service, Analytic hierarchy process

自律计算具有自我管理功能,能够降低人工干预的频率和复杂度,增强系统的可靠性和安全性,解决系统管理危机和安全危机。但目前自律计算研究还处于初级阶段,由于研究者的领域差别,实现的手段各异,使用的技术不同,取得的各项研究成果不能集成、共享、分析和比较,这些都制约着自律计算的可持续性发展。当前明确自律评估标准和规范,建立可度量的评估体系是自律计算研究亟待解决的问题之一。

本课题针对自律计算缺乏评估标准和度量工具,不能进行有效的评估和分析,在研究已有成果的基础上,将可信性应用到自律评估中,从服务的角度对系统的自律可信性进行研究,建立可度量的指标体系,实现自律系统的定量描述和分析。本文首先对已有的自律评估研究进行总结;其次将自律服务分解为服务请求、服务过程和服务响应3个过程,建立自律可信性评估指标。最后经过实例验证了指标体系的有效性和可信性。实现自律系统评估的量化分析和计算可为优化自律系统的设计提供科学的依据。

## 1 已有的自律评估方法

目前自律系统的评估标准不统一,主要归纳为以下3类:

(1)利用已有的评估体系分析法

典型研究成果主要包括:(1)采用ISO9126质量模型定义的6个特征,可信性、高效性、可维护性、可用性、功能性和便携性与自律特征的关系进行评估[1]。(2)建立包括服务质量、开销、适应和反应时间、粒度、错误避免、健壮性、自动化程度、适应性、灵敏度和稳定性[2]的10个属性集合,作为评估系统自律性指标集合。评价强调服务的可靠性和可用性,是理论性分析方法,而且指标没有明确定义和约束,不能实现量化分析。不适合当前自律等级较低的自律系统进行评估。

(2)传统测评与自律属性结合分析法

在传统质量测评属性的基础上,增加某些自律特性进行分析。其中质量属性主要包括可修改性、可扩展性、错误容忍和健壮性等。新增的自律质量特性包括可观察性、自我感知性、耦合性、复杂性、可测试性等。当自律计算系统进化后,可以重新分析系统对质量属性的影响并对评估属性进行改变,使其具有可扩展性,如文献[3]是采用该方法对自恢复系统进行评估,但该方法受限于应用领域,不具有通用性。

(3)其它软件或工具评价分析法

研究者构造特定领域的自律计算系统,并进行自我评价,评价标准将系统的自律性与任务目标结合,强调系统面临攻击、故障、意外事故等情况下,要保证系统任务目标的实现。系统任务和目标体现在向用户提供的服务上,关注系统服务的可持续性和可用性。服务的观点是自律性的基本观点,也是目前系统自律性常用的测定标准。如文献[4]通过异常状态的服务响应速度和服务恢复效益两个指标分析系统性能,评估其自律性。文献[5,6]采用对运行状态的分析进行服务自律性评估。评估指标过于简单,缺乏全面性、科学性和准确性。

通过以上分析研究不难发现,自律性评估研究存在标准模糊、指标不规范和不能定量计算等缺陷。而有效评估是提高和优化自律系统设计的重要途径,是自律研究的重要组成

到稿日期:2010-02-05 返修日期:2010-05-31 本文受863项目(2007AA01Z401),国家自然科学基金重大研究计划(90718003),中央高校基本科研业务费专项资金(HEUCF100601)资助。

张海涛(1972—),女,副教授,主要研究方向为自律计算、软件工程等,E-mail:zhanghaitao@hrbeuc.edu.cn;王慧强(1962—),男,教授,主要研究方向为网络与信息安全等。

部分，具有重要的实际意义。

## 2 基于服务的指标体系

自律系统具有自我管理能力，其典型特征是自我配置、自我优化、自我保护和自我恢复。依据自律计算的理论体系，可以判断自律可信性是自律计算系统的服务关键属性之一。在面向服务的自律可信性分析中，评估不关注实现方式和技术，而关注服务执行过程和服务完成质量。因为面向服务的评估对不同的自律系统具有一致性，所以基于服务提取的评估指标具有通用性。本课题通过服务链上服务请求、服务过程、服务响应 3 个过程进行综合分析，服务请求记录系统的服务类型、次数和频率。服务过程记录请求执行过程中的全部状态指标，全面有效地反映服务的可靠性和可信性。服务响应记录服务响应速度、成功率等。从服务运行状态的不同层面，可以反映自律系统整体的自律可信性。

### 2.1 服务请求

按照自律系统的服务特性，将服务请求划分为服务配置请求、服务优化请求、服务保护请求和服务恢复请求 4 个方面，其范围比自我管理的内容更宽泛。

1）服务配置请求：系统接受用户请求，并根据系统的当前运行状况，动态地实现服务资源的重新配置等操作，包括自律组件的增加或减少、优先级的分配、网络流量的变化等。

2）服务优化请求：系统在资源重新调配的基础上，寻找最优的服务资源和通道，为用户提供高可靠可用服务质量的操作请求。

3）服务保护请求：针对各类异常、故障等意外事件，系统发现并保护所完成的操作请求。

4）服务恢复请求：系统在维持正常工作的前提下自动修复各类错误所完成的操作请求。

### 2.2 服务过程

自律可信系统的关键特征是面对资源重组、系统故障、入侵、意外等事故时能完成服务的能力。系统具有的 4 个关键自律属性是保证有效服务的基础，服务过程的量化指标主要是对服务配置、服务优化、服务保护和服务恢复等不同服务请求的运行状况的具体解析。表征其服务运行过程的参数很多，具有通用性的参数主要有：

1）CPU 占用率：CPU 线程的处理时间是影响系统服务速度和质量的重要指标，若系统的 CPU 占用率过高，则系统服务响应时间增加，服务质量下降。

2）感知能力：系统在单位时间内，感知风险次数和总风险数量的百分比。感知能力是影响系统服务及其质量的关键性指标。

3）服务可用性：单位时间内系统可以提供有效服务的数量与总服务数量的百分比。完备的自律系统的运行状态始终始是有效的和可用的，各项服务都具有可用性，但考虑当前处于自律计算的研究初期，大多数是不成熟的半自律系统，所以运行状态受病毒、网络攻击、意外事故等行为的影响会产生服务失效，记录服务可用性能够获得较全面的自律可信性分析。

4）自优化周期：在服务过程中，系统执行自优化的平均时间间隔。自律系统具有知我感知和环境感知能力，自动实现自优化保持完备服务。

5）失效时间：系统的各项服务在单位时间内的失效时间，用平均失效时间进行测量和表示。完备的自律系统不存在服务失效，但是当前自律系统等级较低，所以允许存在服务失效。

### 2.3 服务响应

自律系统通常具有大型的、异构的和复杂的应用环境，系统作为服务的提供者要具有可控性和灵活性。服务响应能够反映系统服务运行的最终结果，其中主要的量化指标有：

1）响应时间：系统完成各类操作所花费的时间，即系统或用户从提交服务请求到响应完成所需的时间。

2）响应速度：系统单位时间内响应系统或用户服务请求的速度。

3）服务恢复率：系统恢复完备服务或者是修复受损系统的百分比。

4）成功率：系统成功地完成服务请求的数量占服务请求总数量的百分比。

5）服务失效率：系统提供的有效服务数量占总服务数量的百分比。

基于服务请求、服务过程和服务响应的评估指标，构成了系统的自律可信性评估模型，如图 1 所示。

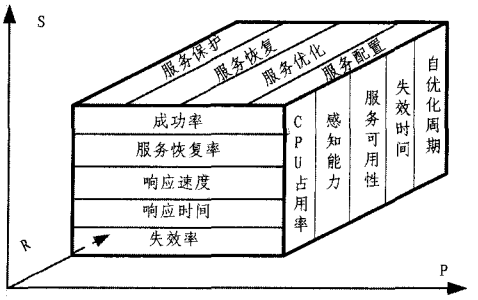


图 1 自律可信性评估模型

该评估体系模型具有可拓展性，每一维的指标要素根据自律系统的应用领域和进化程度的不同进行调整，适用于不同类型自律系统的自律可信性评估描述和分析，不涉及自律系统的具体实现过程，所以该指标体系具有规范性和通用性。

## 3 层次化指标体系和评估过程

### 3.1 层次化指标体系

自律可信性评估指标是一个层次空间模型，各指标属性之间具有一定的关联性，层次关系如图 2 所示。可以利用层次分析方法 AHP[7] 采取自底向上的计算方法：首先计算评估第一层指标因素，得到各指标的相对权重，获得上层的评估结果；利用所得结果进行综合评价，依次一直评到最高层次得出总的综合评价结果。

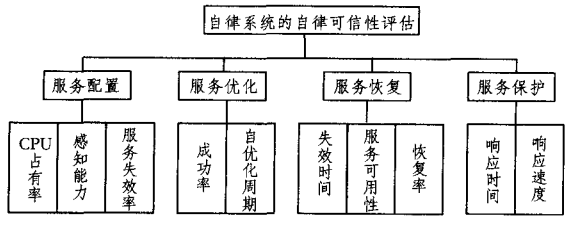


图 2 自律可信性指标层次图

模型由上至下具体描述如下：顶层（第一层）记录自律可信性评估结果，记为 $T$；中间层（第二层）为服务请求，记为 $S$；底层（第三层）为因素集，记为 $N$。每个服务请求由 $n$ 个相关因素决定。$S_i$ 表示第 $i$ 个服务请求组成的重要程度（即权值）；$S_{ij}$ 表示第 $i$ 个服务请求的值。系统的自律可信性 $T$ 由

计算公式得出：$T=\sum_{i=1}^{s}S_iS_{ij}$。

同理，设 $N_i$ 表示第 $i$ 个因素的值，$N_{im}$ 表示对于第 $m$ 个服务请求定义分组第 $i$ 个因素的重要程度，第 $m$ 个服务请求定义分组的值 $S_m$ 可表示为：$S_m=\sum_{i=1}^{n}N_iN_{im}$。

### 3.2 综合评判过程

自律系统具有不确定性和不稳定性，为了定量化分析不易定量的因素，采用了多级模糊综合评价的方法。该方法以模糊数学为基础，应用模糊关系合成原理进行综合评价。

多级模糊综合评价的实现过程如下：

1)首先建立影响评价对象的因素集，$U=\{u_1,u_2,\cdots,u_n\}$，并将 $u_i=\{i=1,2,\cdots,m\}$ 划分为子集 $U_i=\{u_{i1},u_{i1},\cdots,u_{i1}\}$，如果需要再细化分子集。

2)然后建立评价集，由评价结果组成 $V=\{v_1,v_2,\cdots,v_m\}$。

3)分配各因素权值，建立权重集，表示为权向量 $A=\{a,a_2,\cdots,a_n\}$，$a_i$ 为对第 $i$ 个因素的加权值，规定 $\sum_{i=1}^{n}a_i=1$，对第 $i$ 个因素的单因素模糊评价为 $V$ 上的模糊子集 $R_i=\{r_{i1},r_{i2},\cdots,r_{im}\}$。

该评判对象所有因素的模糊综合评判结果是 $V$ 上的模糊子集 $B=A\circ R$。根据权重集 $A$ 与单因素模糊评价矩阵 $R$ 合成，进行模糊综合评价求取评价模糊子集 $B$。算子 $M(\circ,+)$考虑所有因素，能够保留各因素评价的全部信息，是自律性评价的优化算子，计算 $b_j=\sum_{1}^{n}a_ir_{ij}\ (j=1,2,\cdots,m)$，其中 $\sum_{1}^{n}a_i=1$。

第 1)—第 3)步可循环多次，给出指标体系中最低层因素的模糊变换矩阵和权重值矩阵，就可以计算各层次的综合评判结果。

## 4 实例应用与分析

### 4.1 实验过程

采用仿真实验方式对指标体系进行验证，主要验证其有效性和可行性。实验环境选取基于 B/S 结构、UNIX 开发环境下的程序自动评测系统(具有自配置、自优化、自保护等功能)进行验证，实验环境配置如图 3 所示。实验室核心设备采用华为 S6500 路由器，接入层设备采用华为 E050 交换机，出口采用华为 NE40 路由器连接网通出口，并使用华为千兆(Eudemon 1000)防火墙实现对外安全防护。本次实验共有 30 台客户机向服务器发出服务连接请求，没有外来服务连接请求和攻击等其它故障，同时开启网络监控软件和系统监测模块，实验持续进行 30 分钟。

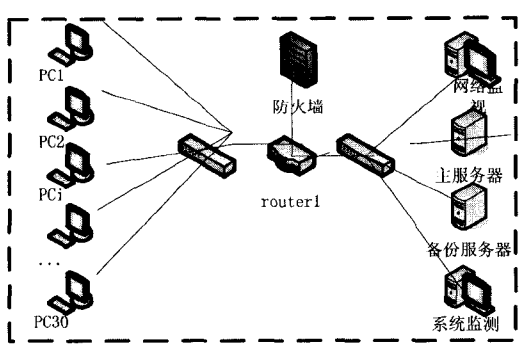


图 3 实验环境配置

### 4.2 算例分析

自律系统的自律可信性评估按照以下步骤进行：

1)通过仿真实验测得的数据，经过分析和计算，给出待评估系统的各性能指标的初始评估值，并应用层次分析法计算各级评估要素权重，如表 1 所列。

表 1 待评估系统的初始数据与指标权重表

| 二级评估指标 | | 一级评估指标 | | |
|---|---|---|---|---|
| 名称 | 权重 | 名称 | 权重 | 初始数据 |
| 服务配置 | 0.150 | CPU 占有率 | 0.401 | 0.6 |
| | | 感知能力 | 0.327 | 0.5 |
| | | 服务失效率 | 0.272 | 17% |
| 服务优化 | 0.150 | 成功率 | 0.500 | 78% |
| | | 自优化周期 | 0.500 | 0.5 |
| 服务恢复 | 0.350 | 失效时间 | 0.327 | 0.75 |
| | | 服务可用性 | 0.264 | 85% |
| | | 恢复率 | 0.409 | 100% |
| 服务保护 | 0.350 | 响应时间 | 0.520 | 0.4 |
| | | 响应速度 | 0.480 | 0.5 |

(2)建立评估集合 $V$ 和对应的决策值集合 $X$：

$V=\{V_1,V_2,V_3,V_4,V_5\}=\{$优 较好 一般 较差 很差$\}$；$X=\{90,80,60,40,20\}$。

(3)对指标体系的底层要素进行单个指标的评估，根据初始评估值分别计算其权重值并构造模糊化向量 $A$，计算并确定底层的模糊矩阵，计算过程数据如表 2 所列。

表 2 模糊层次评估计算过程

| 一级因素集 | 权重 | 模糊化向量 | 一级运算结果 |
|---|---|---|---|
| CPU 占有率 | 0.401 | (0.37,0.63,0,0,0) | |
| 内存占有率 | 0.327 | (0.58,0.42,0,0,0) | (0.43,0.57,0,0,0) |
| 服务失效率 | 0.272 | (0.34,0.66,0,0,0) | |
| 成功率 | 0.500 | (0.9,0.1,0,0,0) | (0.45,0.55,0,0,0) |
| 自优化周期 | 0.500 | (0,1,0,0,0) | |
| 失效时间 | 0.327 | (0,0.4,0.6,0,0) | |
| 服务可用性 | 0.264 | (0.5,0.5,0,0,0) | (0.54,0.26,0.2,0,0) |
| 服务恢复率 | 0.409 | (1,0,0,0,0) | |
| 响应时间 | 0.520 | (0.83,0.17,0,0,0) | (0.43,0.57,0,0,0) |
| 响应速度 | 0.480 | (0,1,0,0,0) | |

(4)计算二级模糊评估向量。已知二级因素权重向量和一级运算结果，则依据层次分析法模糊公式 $V=W\circ R$，二级模糊运算结果为：

$$V=W\circ R=\{0.15\quad 0.15\quad 0.35\quad 0.35\}\begin{Bmatrix}0.43 & 0.57 & 0 & 0 & 0\\ 0.45 & 0.55 & 0 & 0 & 0\\ 0.54 & 0.26 & 0.2 & 0 & 0\\ 0.30 & 0.70 & 0 & 0 & 0\end{Bmatrix}$$

$$=\{0.43\quad 0.50\quad 0.07\quad 0\quad 0\}$$

(5)应用合成运算公式 $T=A\circ X^{\mathrm{T}}$，系统自律可信性的综合评估结果为：

$$T=A\circ X^{\mathrm{T}}=(0.43\quad 0.50\quad 0.07\quad 0\quad 0)\begin{pmatrix}90\\80\\60\\30\\20\end{pmatrix}=82.9$$

由计算结果可知，系统的自律可信性综合评估值为 82.9，属于较好等级。

**结束语** 本文提出一种基于服务的自律可信性定量评估指标体系，它能够准确全面地反映系统的自律可信性，并结合

实例应用层次分析法说明其具体应用，计算过程简单。实验结果表明，评估方法是有效的和可行的，可用于不同应用领域、不同类型的自律系统的量化评估计算。在今后的研究中，需要对评估模型进行形式化描述，并通过实验数据，优化各指标的权值，提高自律评估结果的精确度。

## 参 考 文 献


[1] Kephart J, Chess D. The Vision of Autonomic Computing [J]. IEEE Computer Society, January 2003: 41-59

[2] De Wolf T, Holvoet T. Evaluation and Comparison of Decentralized Autonomic Computing Systemts[R]. CW 437. Leuven: K. U. Leuven, 2006

[3] Neti S, Müller H A. Quality Criteria and an Analysis Framework for Self-Healing Systems[C]// International Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS'07)

[4] Zhang Rui, Moyle S, McKeever S. Performance Problem Localization in Self Healing, Service Oriented Systems using Bayesian Networks[C]// SAC'2007

[5] 刘文洁，李战怀. 一种基于自律计算的性能监视工具[J]. 微电子学与计算机，2008，25(3)：130-133

[6] Kwon Ohbyung. Assessing Level of Ubiquitous Computing Services for Ubiquitous Business Design[J]. Journal of Electronic Science and Technology of China, 2004, 9

[7] 李发泽，胡钢墩. 基于层次分析和模糊数学的网络安全评价模型[J]. 宁夏工程技术，2006，5(3)：338-342


(上接第 155 页)

具 VALSOFT[16]，它也只能对中小规模程序计算程序 Chop，对于大程序图形可视化仍是一个非常复杂的问题。这些工具由于难以构造大程序的 SDG，进一步导致了难以计算大程序的程序 Chop 问题。

针对以上问题，本文基于文献[6]介绍的一种简洁的 Java 程序描述方法，提出一种 Java 程序 Chopping 方法。这种描述方法中，不同的程序方法通过参数之间的依赖关系相互影响；每个 PDG 是一个独立的图，不与外部连接，这有利于本文工具 JChoper 的开发实现，有效地解决了计算大程序的程序 Chop 问题，也成为本文工具的特色。与基于 SDG 的计算 Chop 的算法相比，本文方法不必建造复杂的 SDG，减少了出错的可能性。而且本文的算法改变 Chopping 准则时，能重用先前保存的结果；它还能够并发构造，从而提高了 Chopping 的效率。

**结束语** 本文基于 JavaSDG 的描述方法，首先分析了 Java 程序方法调用引起的参数之间的依赖关系，给出了参数 Chop、方法间程序 Choppig 算法。接着用实例和实验说明了如何计算一个给定 Chopping 准则的程序 Chop 和本文算法的有效性。然后比较本文 Chopping 方法与已有的 Chopping 方法。最后概述了与本文相关的研究工作。与基于 SDG 的计算 Chop 的方法相比，本文方法不必建造复杂的 SDG，减少了出错的可能性。而且本文的方法改变 Chopping 准则时，能重用先前保存的结果，因而大部分 PDG 不需要遍历；它还能通过参数之间的依赖关系，把方法间的依赖分析和 Chopping 转换到方法内分析和 Chopping，而且每个 PDG 是独立的，因此能够并发构造，从而提高了 Chopping 的效率。

下一步可进行的研究包括：(1)完善本文实现的 JChopper 工具，期望本文算法能在工业应用中得到推广；(2)进行动态 Chopping 算法的研究；(3)研究其它新的类型程序的 Chopping 算法，如 AspectJ。

## 参 考 文 献


[1] Horwitz S, Reps T, Binkley D. Interprocedural Slicing Using Dependence Graphs[J]. ACM Transactions on Programming Languages and System, 1990, 12(1): 26-60

[2] Larsen L, Harrold M. Slicing Object-oriented Software[C]// Proceedings of the International Conference on Software Engineering(ICSE-18). Berlin, 1996: 495-505

[3] Zhao J. Applying Program Dependence Analysis to Java Software[C]// Proceedings of Workshop on Software Engineering and Database Systems. Taiwan, 1998: 162-169

[4] Liang D, Harrold M J. Slicing Objects Using System Dependence Graphs[C]// Proceedings of the 1998 International Conference on Software Maintenance. Bethesda, 1998: 358-367

[5] Walkinshaw N, Roper M, Wood M. The Java System Dependence Graph[C]// Proceedings of the Third IEEE International Workshop on Source Code Analysis and Manipulation(SCAM'03). 2003: 55-64

[6] Chen Zhen-qiang, Xu Bao-wen. Slicing Object-oriented Java Programs[J]. ACM SIGPLAN Notices, 2001, 36(4): 33-40

[7] Jackson D, Rollins E J. A New Model of Program Dependences for Reverse Engineering[J]. ACM SIGSOFT Software Engineering Notes, 1994, 19(5): 2-10

[8] Reps T, Rosay G. Precise Interprocedural Chopping[C]// Proceedings of the Third ACM Symposium on the Foundations of Software Engineering. Washington, 1995: 41-52

[9] Krinke J. Evaluating Context-sensitive Slicing and Chopping[C]// International Conference on Software Maintenance. Montreal, 2002: 22-31

[10] Krinke J. Barrier Slicing and Chopping[C]. //Proceedings of the Third IEEE International Workshop on Source Code Analysis and Manipulation(SCAM'03). Amsterdam, Netherlands, 2003: 81-87

[11] Tip F, Palsberg J. Scalable Propagation-based Call Graph Construction Algorithms[J]. ACM SIGPLAN Notices, 2000, 35(10): 281-293

[12] 徐宝文，张挺，陈振强. 递归子程序的依赖性分析极其应用[J]. 计算机学报，2001，24(11)：1278-1285

[13] Krink J. Slicing, Chopping, and Path Conditions with Barriers [J]. Software Quality Journal, 2004, 12(4): 339-360

[14] 张迎周，徐宝文. 一种新型形式化程序切片方法[J]. 中国科学(E)，2008，38(2)：161-176

[15] Anderson P, Teitelbaum T. Software inspection using CodeSurfer[C]// Workshop on Inspection in Software Engineering (WISE'01). PARIS, 2001: 4-11

[16] Krinke J. Visualization of program dependence and slices[C]// International Conference on Software Maintenance. Chicago, USA, 2004: 168-177

[17] Giffhorn D. Chopping Concurrent Programs[C]// The Ninth. IEEE International Working Conference on Source Code Analysis and Manipulation(SCAM'09). Sept. 2009: 13-22