一种自适应软件体系结构框架模型的建立

苏世雄 齐金平

(兰州交通大学机电技术研究所 兰州 730070)

摘 要 针对网络动态变化的环境和用户需求的变化,提出了一种动态自适应软件体系结构模型,然后在此基础上给出了自适应系统的适应过程,通过调整自身的行为,使得系统具有一定的自适应能力。最后,通过一个简单的实例验证该模型,结果表明该模型具有适应复杂多变的网络环境的能力。

关键词 自适应系统,软件体系结构,内置运行时体系结构

中图法分类号 TP311 **文献标识码** A

Adaptive Software Architecture Framework Model

SU Shi-xiong QI Jin-ping

(Mechatronic T&R Institute, Lanzhou Jiaotong University, Lanzhou 730070, China)

Abstract For dynamic change of network environment and user requirements, this paper proposed a dynamic adaptive software architecture model, and then on the basis of the adaptation process, the adaptive system was given. The system gets certain self-adaptive ability by adjusting their own behavior. Finally, the model was validated by a simple example, and the results show that the model is able to adapt to the complex network environment.

Keywords Adaptive system, Software architecture, Built-in runtime architecture

随着互联网技术和应用形式的不断更新和快速发展,软件系统面临着用户需求的变化和运行环境的快速连续变化。目前,软件的运行环境具备以下特点:1)计算环境的多样性;2)用户需求的多变性;3)网络的开放性;4)资源的动态性[1]。传统的软件系统不能够对环境的变化做出反应,无法根据用户的需求动态地调整改变。因此,为了适应当前的运行环境和需求,要求软件系统能够快速感知外部运行环境的变化,并且系统能够随着这种变化按照功能指标、性能指标和可信性指标等进行一些动态的调整和演化,来支持适应复杂环境下的系统运行以满足用户的需求。所以,需要提出一种新的软件系统。新系统应该具有以下要求:1)在不同环境下系统应该具有保护系统逻辑功能正常运行的自适应模块;2)在复杂多变的环境下仍然具有高效稳定的性能[2],即系统应当具有一定的自适应能力。

当前,大量的自适应研究都集中在自适应机制和演化技术方面,但是模型驱动是系统演化的主要方法之一[3]。文献[4]提出了一种在运行时监视周围环境变化的自适应机制,动态地改变系统的行为。但其局限性是没有一种合适的系统模型来实现这种机制。基于此本文首先提出一个自适应系统的概念,然后提出自适应体系结构的模型,并且对其结构模型进行了说明分析,最后通过一个简单的实例验证来证明该模型的有效性。

1 自适应系统的定义与描述

自适应系统是一个在开放的、动态多变的环境中的一种软件体系结构模式,它与以往传统的系统的区别在于:自适应系统具有实时性,能够自动感知环境变化,并且根据运行环境的变化自动地做出相应的调整,使系统的性能稳定并且达到系统的最高效用。

自适应系统由文献[5]提出,是指软件系统可以在运行过程中实时地采集系统的各种信息,并根据预先设定的策略,在必要的时候对自身进行调整,以便更好地为用户提供服务。文献[6]提出,软件系统可以感知环境的变化,并根据环境的变化采取适应性动作改变自身行为,以适应资源、用户需求和系统错误。虽然以上都对自适应系统做出了定义,但是该定义都有一定的局限性——都是从满足用户的角度对自适应系统进行定义的。因此,本文在借鉴以上文献的基础上,从满足用户和供应商效益最大化的角度提出自适应系统的概念,即指软件系统根据它运行的环境以及用户需求的变化,实时地收集系统的各种变化信息,加以分析决策,当满足系统调整的条件时,系统会自动地调整系统结构以适应当前环境与需求的变化,稳定地发挥其最大性能,并且系统自身能够自动平衡环境变化和系统运行的矛盾,使得系统的投入最少,效益最大。

本文用一个五元组来表示自适应系统,$SAS=\langle E,M,A,C,I\rangle$,其中 SAS(Self-Adaptive System)表示自适应系统。

E(Environment)代表系统环境(系统的内部环境和外部环境),表示系统运行时,系统所处的内外环境的集合。

M(Monitor)代表监控器,负责系统监控的传感器的集

本文受"十二五"国家科技支撑计划项目(2012BAH20F05),甘肃省财政厅基本科研业务费(214153),兰州交通大学青年科学基金(2012012)资助。

苏世雄(1989—),硕士生,主要研究方向为软件体系结构、数据挖掘,E-mail:398933559@qq.com;**齐金平**(1978—),男,博士,副教授,硕士生导师,主要研究方向为载运工具运用工程、铁路质量管理体系。

合，用于收集系统运行时的信息。

A(Analyzer)代表分析器(决策器)，根据 Monitor 所提供的信息按照指定的算法或者策略计算环境的性能，以达到系统适应变化的环境。

C(Constraint)代表约束规则集，当系统提供服务时，系统所设置的约束条件集合。

I(Implementation)代表实现，根据 Analyzer 分析的结果及约束的内容对系统进行调整的集合。

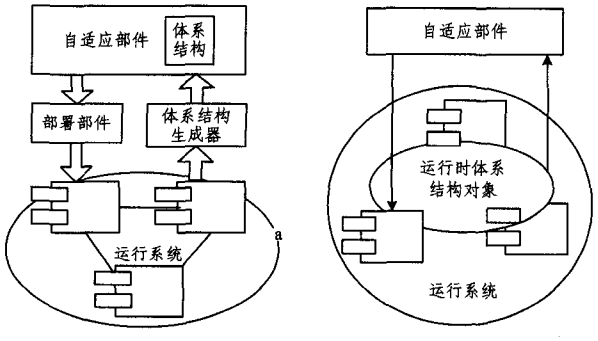
用函数表示自适应的过程如下：

$E(S)\times C_E \rightarrow E'(S)\times M\times A\times SA \rightarrow E'(S')$，where consistent$(S')$

$E(S)$表示系统 S 在环境 E 中运行，由于环境发生了一个变化，C_E 变成一个新的环境 E'，系统 S 需要适应环境的变化 $E'(S)$，此时系统 S 监测到环境的变化，发送给分析器(A)，由分析器作出分析与决策，找到最适合的调整动作，然后做出自适应的调整 SA(Software Architecture)，使得系统自身发生调整 S'，适应新的环境 $E'(S')$。此时需要保证系统调整后前后的一致性 $consistent(S')$。

2 自适应体系结构模型

目前，大多数系统采用外部自适应，即自适应控制部件处于系统外部，通过监控部件感知外部变化，进而采取适应性动作，作用到正在运行的系统。然而，近年来，运行时体系结构在系统动态演化中的作用已经备受关注：系统运行过程中体系结构约束的破坏(可能)是自适应的驱动因素，而运行时体系结构的信息又是如何进行演化的重要依据，应用这些体系结构进行系统演化的方法通常称为"基于体系结构"的自适应[7-9]。但是这种方法需要将体系结构作为运行系统的抽象对象，并且与系统分离，同时需要特定的方式收集系统的运行信息，这点比较难。文献[10]提出了内置的运行时刻体系结构的思想，该思想将运行时刻体系结构以分布共享对象的方式融入到运行系统中，作为运行系统的一部分，从而无需采用特定的措施收集体系结构信息。同时将该运行时体系结构作为自适应行为的载体，无需通过作用器部署到真实系统，解决了"基于体系结构"自适应的自适应动作难部署的问题。图 1 为两种典型的软件体系结构。



(a)外置运行时体系结构　　(b)内置运行时体系结构

图 1　两种典型的软件体系结构

图 1(a)采用外置的 SA，SA 层检查系统状态是否违反了结构约束，当系统需要调整时，需要专门的效用器来解释和翻译。这种外置的 SA 的优势在于将自适应层彻底分离出来，能更好地发挥其自适应能力，但是由于需要特定的效应器支持以及部署部件，因此将会限制软件的适应领域范围。相比较而言，采用图 1(b)内置的 SA 更容易实现。SA 作为元层，是一个运行目标系统内部的可操作的实体，可以通过反射技术将应用系统信息状态与结构反映出来，而不需要特定的效用器和部署部件，使得该结构具有更广泛的应用范围。因此，本文采用内置运行时软件体系结构。

图 2 给出了传统软件体系结构框架模型。

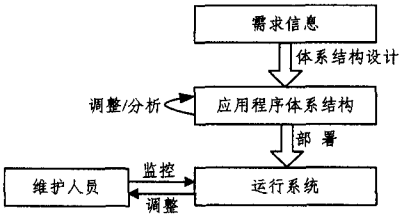


图 2　传统软件体系结构框架模型

图 2 的系统框架模型说明传统的软件设计方法没有考虑自适应。设计人员直接根据需求信息设计应用软件的体系结构，然后根据系统整体的体系结构效果进行分析和调整，最后应用到现实的系统中运行，由维护人员对系统进行监控，并且分析监控得到结果，然后再反馈给设计人员(维护人员)对系统作适当的调整。这样设计出来的软件不但费时而且还必须通过人工参与来识别解决问题。这种方式开发出来的软件属于静态软件，不具有动态自动调整的能力。

针对以上的不足，本文提出自适应软件系统的框架模型，如图 3 所示。

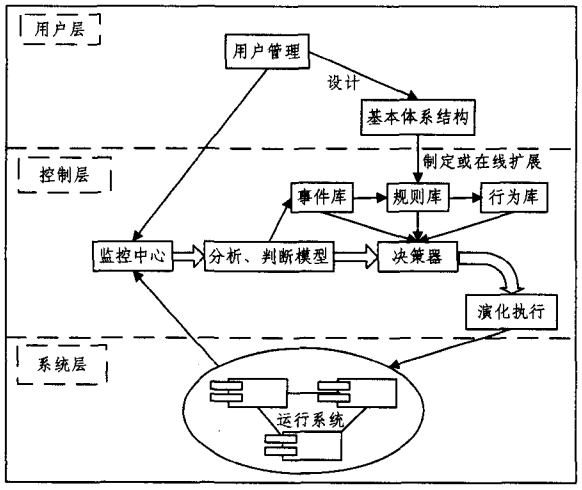


图 3　自适应软件系统的框架模型

从图中可以看出，系统框架模型由 3 层组成：用户层、控制层和系统层。

用户层：用户层是系统对用户提供的接口。包括用户的需求、与需求相关的自适应规则、将动作添加到控制层相应的库中。在开发阶段，系统分析师先确定系统的质量属性以及功能属性，根据需求完成体系结构的设计，此时的体系结构是基本的体系结构拓扑结构的元模型，通过组装部署来完成系统实现。为了增添自适应能力，开发人员使用基于质量属性场景来分析体系结构的自适应性。然后将生成的逻辑规则和适应动作添加到相应库中。在运行阶段，自适应体现为体系结构的动态调整。因此，用户层为系统提供自适应策略以及相应库的接口。

控制层：控制层是系统的核心，由其完成系统自适应的过程。包括系统的监控中心(Monitor)、分析判断模块(Analyzer)、决策器、演化执行模块以及事件库、规则库和行为库。

监控中心是用户层和系统层之间的桥梁，由其监控运行环境和运行系统，将测控到的信息反映给分析器，分析器检查信息是否违反了相关约束(Constraint)，如果违反则抛出相应事件，决策器对抛出的事件查询规则库中能够响应的动作集，并通过策略选择算法选择最佳的动作。然后决策器对演化执行模块(Implementation)发出演化命令，驱动执行系统层的演化。

系统层：系统层包括软件体系结构元层、基层以及适应性体系结构模型，是自适应系统调整最底层的核心部件，由其完成自适应动作的最底层的演化过程。SA 模型作为一种元层实体存在于系统层，与基层保持因果关联，元层的变化导致基层的结构变化，这种变化通过反射的机制来实现。SA 采用内置的运行时刻体系结构，该模型作为一种分布式共享的对象存在于运行系统中。

3 自适应过程

自适应系统能够在线地对不满足 SA 约束的部件进行自主的调整，图 4 刻画了一个完整的自适应过程。

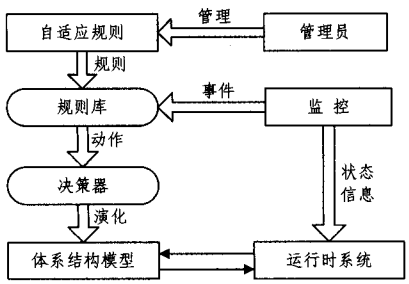


图 4 自适应过程

该自适应过程如下：

(1)根据环境信息(Environment)和需求信息进行系统体系结构的设计，得到基本的应用程序的体系结构，然后添加到规则库中，由管理员进行管理；

(2)监控中心(Monitor)收集环境信息，将各部件状态、属性交给分析模块处理，当感知的信息违反了体系结构的约束(Constraint)时，抛出相应的事件，事件与规则库中的策略匹配，找出可执行的动作集；

(3)分析决策器(Analyzer)根据预设的决策算法(决策算法在后续文章中介绍，采用基于复合优先级决策算法)选择最合适的动作并生成演化策略；

(4)对于(3)中得到的演化策略，对其进行效用理论和构件调整数目优先级影响及质量属性的分析，实施(Implementation)具体调整；

(5)将重新适应的体系结构应用到支撑平台上，并且控制层的监控中心继续监控，使系统达到一个闭环的控制系统。

自适应过程通过监控到的变化事件(外部环境和运行环境)分析其原因，匹配相应的调整策略，自动调整及执行相关的部件，从而达到调整运行系统的功能，完成运行系统的自适应，整个过程不需要人为参与。

4 实验与评估

为了展示提出的自适应模型系统的效果，我们开发了一个简单的模拟实验，对该系统在自适应动态调整的情况下和一般情况下，进行了系统的性能测量和比较。本实验效仿文献[7]中 Rainbow 系统上设计的类似实验。

模拟实验主要模拟了一个提供某种功能服务的网站系统，实现方式为 Request-Response 结构，Request 主要接受请求的任务，将各个请求分别交给 Response，由它进行任务的完成。当 Request 收到多个任务的处理时，每个 Response 就需要同时处理多个请求，因此系统的响应时间会随着并发任务的增多而延长。对于一些对时间比较敏感的用户来说，其响应时间不允许超过某一个临界值，否则将会造成很大的损失。因此，系统监控其响应时间，当响应时间大于临界值时，系统就会采取相应的措施(例如增加一个 Response，以加强计算能力)，降低响应时间，从而提高服务质量。具体采用何种措施，后文会给出基于复合优先级选择策略设计的选择措施的条件。本文重点是验证该模型框架，所以本文对此问题拟采取的动作为增加一个 Slave，旨在说明模型的有效性。

这个系统中的各个计算服务分别分布在一个由 5 台通过 100Mb/s 快速以太网连接的双 Xeon2.8G CPU 工作站构成的局域网上，设置系统约束条件响应时间的临界值为 10s。实验在传统的软件系统和具有自适应系统的软件上进行，实验的结果如图 5 和图 6 所示。在图 5 中，系统的响应时间随着并发用户的数目增加而延长；而图 6 中，当响应时间超过 10s，说明系统的约束条件被破坏，系统就自动触发相应的部件，分析部件对系统的影响，然后生成相应的调整策略，最后系统根据运行的时机自动地选择合适的时机进行调整，这样系统的响应时间就可以保持在一个相对稳定的水平上。实验结果表明，本文提出的自适应体系结构模型可以适应动态变化的环境，能够对系统进行自动的调整。

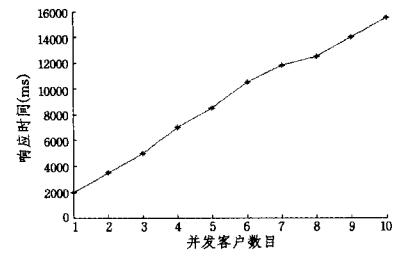


图 5 无动态自适应调整的性能曲线

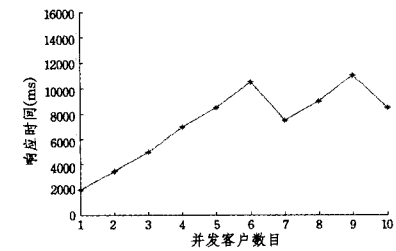


图 6 有动态自适应调整的性能曲线

结束语 本文提供了一个自适应系统的体系结构框架模型，该模型采用内置的运行时体系结构，当环境需求变化时，系统能够根据变化做出相应的处理，使得系统保持在一个相对稳定的状态。通过实例验证了该结构模型的可行性。与一般的传统方法相比，本文提出的自适应体系结构一方面使得调整能够更加精确，另一方面也可以得到调整后的策略是否对系统有效，同时维护人员可以根据系统反馈的信息对系统进行快速有效的调整修改。本系统自动完成监控、分析、解决和调整，在一定程度上达到了“自适应”。下一步工作将针对自适应决策算法和决策机制以及体系结构动态配置一致性问题进行深入的研究与分析，以便使其更好地服务于自适应系统。

参 考 文 献

[1] 杨芙清,梅宏,吕建,等.浅谈软件技术发展[J].电子学报,2002,30(12A):1901-1906

[2] 李阳,吴朝晖.一种动态自适应体系结构的研究[J].浙江大学学报:工学版,2005,39(2):216-210,237

[3] 陈湘萍,黄罡,孙艳春,等.支持自适应过程中分析与规划的软件体系结构模型[J].计算机研究与发展,2010,47:1156-1164

[4] Cheng S W. Rainbow: Architecture-based Self-adaptation with Reusable Infrastructure [J]. Computer,2004,37(10):46-54

[5] 冯耀东,黄罡,梅宏.一种自适应软件系统结构建模及其实施方法[J].北京大学学报(自然科学版),2008,44(1):67-76

[6] 俞春,马骞,马晓星,等.一种面向体系结构的软件系统自适应机制[J].南京大学学报(自然科学版),2006,42(2):120-129

[7] Krammer J, Magee J. Analyzing dynamic change in distributed software architectures [J]. IEEE Proceedings-Software, 1998, 145(5):146-154

[8] Garlan D, Cheng S W, Huang A C, et al. Rainbow: Architecture-basedself-adaptation with reusable infrastructure [J]. IEEE Computer, 2004, 37(10):46-54

[9] 黄罡.反射式软件中间件原理与技术研究[D].北京:北京大学,2003

[10] 马晓星.一种面向服务的动态协同架构及其支撑平台[J].计算机学报,2005(4):467-477

(上接第 466 页)

图 5 示出按照表 2 的参数设置运行 LDA 程序后,所得的所有文件。文件内的信息已在表 3 中阐述。

图 5 结果文件集

表 4 列举了最后生成的文件"model-final. twords"中的部分信息,该信息为一个词以相应的概率值属于某个主题,具体可解释为:单词"search"以概率值0. 014580406400986875属于一个特定的主题。

表 4 "model-final. twords"中部分信息

词	概率值
search	0. 014580406400986875
class	0. 013209745399719013
actionperformed	0. 008069766644964534
actionhandler	0. 006699105643696673
match	0. 006185107768221225
constructor	0. 006013775143062742
bufferset	0. 006013775143062742
file	0. 005328444642428811
replace	0. 005328444642428811

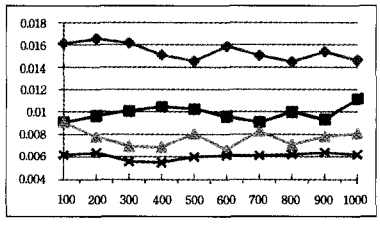


图 6 概率变化统计

图 6 是"model-final. twords"文件中的信息。从该文件中选取了 4 个词"search"(◆)、"save"(■)、"actionperformed"(▲)和"match"(×)以及在不同迭代次数下该词属于某个主题的概率,用这些数据进行绘图。图 6 中的横轴代表迭代次数,纵轴代表词属于某个主题的概率。从图 6 中可以看出,除了"save"(■)外,其他词在迭代次数较大时,其概率趋于平稳,这说明 LDA 方法在一定的迭代次数后,会呈现明显的收敛趋势。

结束语 本文基于 LDA 主题模型的方法,提出了一种对软件演化进行确认的模型方法。首先从源代码中提取能代表软件行为的特征(注释、方法名和变量名),然后调整其格式,使用 LDA 进行处理,得出文档、主题和词的概率模型。

下载两个版本的同一软件源代码,对其进行归并处理,然后对归并后的代码利用本文方法进行建模,在产生结果的主题中,找出词或词的概率差异,即计算每个主题与各个文档的相似度,用这些计算出来的相似度与软件的更新报告中的实际被修改代码进行对比,就能从代码层面确认该软件的演化是否与用户要求或更新报告相一致。

在 LDA 主题模型中,有两个极其重要的参数 α 和 β,其意义已在文中做了阐述。本文在使用 LDA 的过程中,根据人工的经验对这两个参数进行赋值从而失去了一般性,使得参数设置的好坏直接影响最后结果收敛与否,且针对不用的输入数据,设置的值差异较大。同时本文没有对输入数据进行词根还原、去除停词等处理,因此产生结果中会有停词和一词多式(英语中的时态,如过去式)的情况。找到一种可靠且有效的方法来解决上述两个问题,将是今后研究的重点。

参 考 文 献

[1] 郁湧,王丽霞,赵娜.基于构件的软件演化波及效应分析[J].计算机科学,2013,40(11):164-168

[2] Deerwester S, Dumais S T, Furnas G W, et al. Indexing By Latent Semantic Analysis[J]. Journal of the American Society For Information Science, 1990, 41(10):391-407

[3] Blei D M, Ng A Y, Jordan M I. Latent Dirichlet Allocation[J]. Journal of Machine Learning Research, 2003, 3:993-1022

[4] Griffiths T L, Steyvers M. Finding scientific topics[J]. PNAS, 2004, 101(1):5228-5235

[5] Brooks S P. Markov chain Monte Carlo method and its application[J]. The Statistician, 1998, 47(1):69-100

[6] Shimura G. 初等 Dirichlet 级数和模形式(影印版)[M].北京:科学出版社,2011

[7] 刘忠,茹诗松.分组数据的 Bayes 分析-Gibbs 抽样方法[J].应用概率统计,1997,13(2):211-216

[8] 马海云.基于 Gibbs 抽样的测试用例生成技术研究[J].自动化与仪器仪表,2011(3):11-15

[9] 何云,王炜,李彤,等.面向行为主题的软件特征定位方法[J].计算机科学与探索,2014,8(12):1452-1462