ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.19, No.4, April 2008, pp.779−802
DOI: 10.3724/SP.J.1001.2008.00779
© 2008 by *Journal of Software*. All rights reserved.
E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

自主计算概念模型与实现方法*

廖备水[1+], 李石坚[2], 姚 远[3], 高 济[2]

[1](浙江大学 语言与认知研究中心,浙江 杭州 310028)
[2](浙江大学 计算机科学与技术学院,浙江 杭州 310027)
[3](上海大学 快速制造工程中心,上海 200436)

Conceptual Model and Realization Methods of Autonomic Computing

LIAO Bei-Shui[1+], LI Shi-Jian[2], YAO Yuan[3], GAO Ji[2]

[1](Center for the Study of Language and Cognition, Zhejiang University, Hangzhou 310028, China)
[2](College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China)
[3](Rapid Manufacturing Engineer Center, Shanghai University, Shanghai 200436, China)
+ Corresponding author: Phn: +86-571-88273089, E-mail: baiseliao@zju.edu.cn

Liao BS, Li SJ, Yao Y, Gao J. Conceptual model and realization methods of autonomic computing. *Journal of Software*, 2008,19(4):779−802. http://www.jos.org.cn/1000-9825/19/779.htm

Abstract: Autonomic computing is an emerging research hotspot, which aims at hiding system management complexity from human users by means of "technologies managing technologies", to establish guidable, state-aware, and self-adaptive computer systems. Currently, the research of autonomic computing is still in its infancy, without systematic and mature theories. After clarifying the definition of autonomic computing, this paper proposes a conceptual model, which describes the basic working mechanisms and principles of autonomic elements and autonomic computing systems. On the basis of this model, two categories of autonomic computing systems based on knowledge model and mathematical model respectively are summarized and analyzed, with their advantages and disadvantages. Finally, the future research directions of autonomic computing are proposed.
Key words: autonomic computing; policy; agent; Web service; control theory

摘 要: 自主计算是一个新兴的热点研究领域,旨在通过"技术管理技术"的手段隐藏系统管理复杂性,建立用户可指导的、状态觉察的和自适应的计算机系统.目前,自主计算的研究仍处于起步阶段,尚缺乏系统而成熟的理论体系.在阐明自主计算概念的基础上,提出一个自主计算概念模型.该模型刻画了自主元素和自主计算系统的基本工作机制和原理.以该模型为依据,概括性地提出了两类分别基于知识模型和数学模型的自主计算系统,分析它们的优点和不足.最后,给出了自主计算研究展望.
关键词: 自主计算;策略;Agent;Web 服务;控制理论

* Supported by the National Natural Science Foundation of China under Grant No.60773177 (国家自然科学基金); the China Postdoctoral Science Foundation under Grant Nos.20060400316, 20060401039 (中国博士后科学基金); the National Basic Research Program of China under Grant No.2006CB303000 (国家重点基础研究发展计划(973)); the Research Program in the Center for the Study of Language and Cognition, Zhejiang University of China under Grant No.C04 (浙江大学语言与认知研究中心科研课题资助项目)

Received 2007-05-30; Accepted 2007-09-30

© 中国科学院软件研究所 http://www.jos.org.cn

中图法分类号: TP18 文献标识码: A

随着计算机技术和网络技术的迅猛发展,大规模、开放、异构、动态的信息系统不断涌现,如动态企业联盟、网格计算系统和普适计算系统等.在这样的系统中,高层商业目标、系统结构和环境等均会不断发生变化,因此,客观上要求系统能够动态地适应这些变化,以实现特定的目标,包括资源的动态配置、服务的动态合成、系统参数的动态校正等.为了达到该要求,目前的IT管理者需要依据系统状态和高层的管理策略,直接对系统进行配置、诊断、修复和重配置等,其结果是IT企业通常要花费高出设备成本 4~20 倍的管理费用[1],而且,仅靠IT专家和技术人员的努力越来越难以驾驭开放、动态和异构的信息系统.在这样的背景下,以解决日益突出的分布式系统管理复杂性问题和提高系统可靠性、可用性和容错能力为目标的自主计算[2,3]已成为当前的研究热点.其基本思想是通过“技术管理技术”的手段来隐藏系统复杂性,使得系统能够在IT管理者制定的管理策略的指导下自己管理自己,实现系统的自配置、自优化、自修复和自保护等.作为一个重要的研究领域,自主计算的思想一经提出,就吸引了众多的研究者,取得了一系列研究成果,如Tianfield2004[4],Sterritt2005[5],Hariri2006[6]等.同时,国内在这方面的研究也有了起步,典型的包括曹建(2004)[7]、刘涛(2006)[8]、张海俊(2006)[9]、王千祥(2004)[10]等人的工作.不过,从现有研究成果来看,自主计算的概念尚不明确,对自主计算的研究还未形成系统而成熟的理论体系,各种应用也大多是领域相关的.因此,继续澄清自主计算的概念和定义,从更高的视角深入研究和剖析自主计算理论和方法,将有助于促进自主计算研究领域的发展.

1 自主计算的定义和概念模型

1.1 自主计算的概念和定义

“自主计算(autonomic computing)”又译为“自治计算”,鉴于该术语来自于自主神经系统中“自主”的概念,本文将其译为“自主计算”.这一新的研究领域于 2001 年由IBM公司发起[2],旨在参照自主神经系统的自我调节机制,以现有的理论和技术(包括面向服务的计算、自适应控制理论、优化理论、基于策略的管理、多主体技术等)为基础构建自主计算系统,使得信息系统在“整体上(holistic)”实现自我管理[3].

根据神经科学理论[11],自主神经系统是广泛分布于内脏器官的相互连接的神经元网络.“自主”来源于希腊语autonomia,大意是“独立”.自主神经系统管理人体的内脏机能,包括对内部平滑肌、心肌和腺体的管理.描述其基本原理的一个典型例子是:当遇到紧急情况时,自主神经系统的交感神经活动增强,激发一系列生理应答,如加快心律、升高血压、抑制消化功能和动员葡萄糖储备等,表现为 4 个F,即fight(格斗)、flight(逃避)、fright(惊恐)和sex(性,性欲);另一方面,在放松的情况下,交感神经活动减弱,副交感神经活动增强,使得心律变慢、血压降低、消化功能加强、出汗停止,表现为非 4F.同时,自主神经系统通过交感和副交感神经的协调作用,还可“自主地”调节人体内的各种参数(体温、心律、血压、血糖水平等),并使之保持在一定的范围内.依据上述原理,自主神经系统与内分泌系统、免疫系统及中枢神经系统相互配合,能够觉察身体的内外状态,自主地调动体内各器官和谐工作,以适应环境变化、维持身体各参数的动态平衡.它具有 3 个主要特点:(1) 自主性:自主神经系统尽管需要接受脊髓、延髓以及下丘脑各中枢所发出的冲动,受到中枢神经系统的管制,但它不受人的意志的直接指挥,可以自动地发挥作用,有一定的自主性.因此,内脏中发生的各种调节过程不需要人在意识上的努力;(2) 状态觉察性:自主神经系统与脊髓、延髓及下丘脑等功能调节中枢相配合,可感知身体内外的状况,实现状态觉察.一方面,自主神经系统的传入神经传递人体器官内部感受器所获得的信息,向上传递到中枢.中枢对传来的关于体内各器官工作状态的信息进行综合分析后,获得内脏器官的当前状态;另一方面,下丘脑与大脑皮层相结合,可以觉察到反映外部环境状态的各种情绪(如爱、恨、高兴、恐惧、焦虑等),然后激活自主神经系统的交感神经系统.例如,恐惧一般伴随着心律加快、消化抑制和出汗增加等现象;(3) 自稳态性(homeostasis):这是指机体能够适应内外状态的变化,把内部环境维持在一个狭小的生理范围的过程,实现“整体上”的自适应.首先,从自适应结构上看,由肠神经部、延髓(孤束核)和下丘脑等组成的层次式自我调节体系依据身体内外的状态变化,对各个器官进行协调一致的调整,达到局部自适应和全局自适应的有机统一;其次,从自适应特性上看,自主神经系统

© 中国科学院软件研究所 http://www.jos.org.cn

与内分泌系统以及免疫系统等相结合,从多个层面出发,对机体进行综合的调节,满足其“整体上”的动态平衡.例如,当遇到寒冷时,自主神经系统会自动调节各个器官,进行合理的资源分配,以维持体内平衡,如减少体表血流量、抑制尿液生成、动用体脂储备等(自配置、自优化);当受到惊吓时,自主神经系统会使心律上升、骨骼肌肉供血增加,甚至头发竖起,以防护机体、免受伤害(自保护);当体内的某一参数超出正常范围时,自主神经系统会协调各个器官,试图使其恢复到正常状态(自修复),等等.

由上述分析可知,自主神经系统能够维持身体的内环境稳定,而不需要人主观意识的支配和调控.这对于大脑来说,可以不必关心内脏系统的调节过程,因此,它对人体这个复杂系统的管理复杂性将大幅度降低.受此思想的启发,自主计算的思想应运而生[2].在自主计算系统中,可以将IT管理者比作“大脑”,他无须也不能“意识”到信息系统如何调节其内部行为,而只需指定高层的管理策略(相当于大脑产生的各种情绪,如爱、恨、高兴、恐惧等);自主的信息系统则可以比作“自主神经系统管理下的人体内脏系统”,它在高层策略的指导下实现状态觉察,并自主地保持系统的动态平衡.值得注意的是,在这里,自主计算系统不是要完全模拟人体自主神经系统的基本结构和具体工作机制,而是要通过软件工程、人工智能等方法和技术,构建具有自主神经系统主要特点(自主性、状态觉察性和自稳态性)的信息系统,即通过技术的手段,把那些原本应由人类完成的系统管理工作交由机器进行管理.该思想可以通过一个例子来获得更直观的解释:假设有一个未来的自主数据中心(Autonomic Data Center,简称ADC)[12],它可以为多个客户提供各种应用或服务.每一次,来自开放环境中的各种客户通过服务级协定(service level agreement,简称SLA,用于说明期望的性能、可用性和安全等级等)征订新的服务,ADC则依据可用的资源情况以及来自IT管理者的管理策略,自主地选择和配置相关资源,如服务器、数据库、存储器和网络资源等,并在这些资源上安装和配置各种应用模块,以满足SLA所规定的服务需求——实现系统的自配置.然后,在服务实例化之后,ADC将监视服务的执行过程,觉察运行中发生的错误、薄弱环节和各种攻击.当异常情况发生时,它自主地把问题定位到具体的数据库、服务器、路由器、应用模块、Web服务或虚拟机器.然后,它将围绕这些问题进行必要的处理,如诊断问题并把它固定在原来的位置,重启一个失败的组件,添加一个合适的补丁,快速切换到备份的组件,等等——实现系统的自修复.同时,它通过连续的检查和升级内部组件来保护自己,通过检测入侵并自动采取措施来遏制它们,最小化它们的影响——实现系统的自保护;当目标改变、服务添加或删除、工作负载波动时,它将以各种方式动态地调整自己,并使用各种能够掌控的参数来优化系统的性能,以最好地满足SLA的规定以及IT管理者的要求——实现系统的自优化.容易看出,ADC也具有自主性、状态觉察性和自稳态性等类似于自主神经系统的特点:首先,“自主性”表现在ADC可以自主地完成那些原来需要人工进行的管理工作,包括资源的选择和配置,系统运行情况的监视、诊断和处理,安全问题的检测和处理,以及系统状态发生偏移时的调整和优化等,即实现系统的自配置、自修复、自保护和自优化.这样,对于IT管理者来说,他只需定义管理目标和策略(如“允许金卡客户访问A类服务”、“如果对金卡客户的反应时间大于 100 毫秒,那么分配给它的CPU比例增加 5%”、“对金卡用户的反应时间不大于 100 毫秒”[13]、“当A类组件异常时,用B类组件替换”等),不必也无法直接指挥底层IT资源的行为;其次,“状态觉察”表现为ADC对内部被管资源状态和外部环境状态的觉察,前者涉及到资源的可用性、利用率和健康情况等,后者则与用户需求、负载波动和安全入侵等相关;第三,“自稳态性”体现为ADC能够在IT管理者制定的目标和策略以及用户需求的约束下,进行自我配置、监视、异常诊断、修复、优化和调整,使系统实现“整体上的”自适应.

除了上述 3 个主要特点以外,我们认为,自主计算的另一个重要特点是它必须能够接受IT管理者的动态“指导”.在自主神经系统中,“指导”系统行为的约束信息体现为大脑皮层产生的各种情绪状态以及体内的各种参数范围(如心律、血压、血糖水平等).与之相对应,在自主计算系统中,指导系统行为的约束信息是来自IT管理者制定的管理策略.在这里,“管理策略”的概念与传统基于策略的管理系统中的“策略(policy)”类似.所谓“基于策略的管理”是指一种用于管理网络和分布式系统的方法,它把系统的管理逻辑和应用逻辑相分离,并将管理逻辑表示为控制系统行为选择的规则,即策略.策略可以动态部署、更新或删除,因此,可以在不改变软件编码或停止系统运行的前提下,通过改变策略来支持系统行为的动态适应,这意味着可以通过动态更新由分布式实体解释的策略规则来改变它们的行为.由此可知,传统的基于策略管理的思想恰好可以用于自主计算系统.依据典型的基

© 中国科学院软件研究所 http://www.jos.org.cn

于策略的管理方法(Ponder[14],Rei[15]和KAoS[16]),策略是一种用于指导系统决策和行动的规则,一般分为两类:义务(obligation)型策略和授权(authorization)型策略,分别规定策略执行者“应该/不应该做什么”和“允许/不允许做什么”.不过,在自主计算背景下,应该允许策略有更宽松的定义,Kephart等人从AI的角度把自主计算系统中的策略分为 3 种类型:动作策略、目标策略和效用函数策略[13,17].其中,动作策略规定系统处于给定状态时应该采取的动作;目标策略只规定期望的状态,具体条件下的动作由系统规划产生;效用函数策略不直接指定期望的状态,只给出一个目标函数,这个函数用于表达每个可能状态的标量值.依据效用函数,系统计算最大效用值来确定具体条件下的期望状态.于是,以传统的基于策略的管理方法[14−16]以及Kephart等人的工作[13,17]为基础,自主计算系统中的“策略”可以进一步定义为:

定义 1(策略). 在自主计算背景下,策略是表征高层管理目标的任何形式化的规范,用于指导自主元素的行为.

在该定义中,我们采用了 Kephart 等人的思想,从更宽泛的角度来定义策略,即策略可以采用各种形式,如动作策略、目标策略、效用函数策略等.不过,与此同时,我们强调策略应该是一种形式化的规范,必须以一定的形式加以严格定义,使得机器可以准确理解策略的语义并对其进行推理.

另一方面,关于自主计算的定义,目前的看法尚不统一,但现有的文献一般认为自主计算系统应该至少具有如下 4 个主要特性:自配置(self-configuration)、自修复(self-healing)、自保护(self-protection)和自优化(self-optimization).其中,自配置是指系统能够根据高层策略自动配置自己,以适应环境的变化;自修复是指当软/硬件发生故障或异常时,系统能够自动地发现、诊断和修复故障;自保护是指当系统遇到恶意攻击或者由于自修复措施无效而发生连串失败时,能够从整体上保护自己,同时,它也可以根据来自传感器的相关报告预测问题,并采取措施加以预防;自优化是指系统能够不断寻找方法来改善性能、降低消耗[3].关于这 4 个特性的概念,我们还可以从上述自主数据中心的例子中得到比较直观的理解,不过,对于它们的具体内涵,由于应用领域的不同,各种解释之间也存在差别(限于篇幅,在这里不作进一步的讨论).除了上述 4 个主要特性以外,一些学者还指出自主计算系统应该具有一些其他特性,如Sterritt提出的自管制(self-governing)、自适应(self-adapting)、自恢复(self-recovery)和自诊断(self-diagnosis)[18];Tianfield的自规划(self-planning)、自学习(self-learning)、自调度(self-scheduling)和自演化(self-evolution)[19],等等.然而,经过仔细分析可以看出,这些特性有些是上述 4 个主要特性的细化,如自诊断、自规划、自学习等.有些采用了不同的术语,如自恢复在概念上与自修复相似.因此我们认为,自主计算除了 4 个主要特性外,依据具体应用背景进行适当的细化和延伸,不仅是合理的,也是有益的.但问题的关键是如何通过运用各种必要特性,使得系统在整体上实现自适应.鉴于此,我们把“整体自适应”作为自主计算各种特性的更高一层的抽象,并把它作为自主计算定义中的主要部分之一.与此同时,依据上述对自主神经系统和自主计算系统的分析,自主计算的定义中还应包括另外两个部分:状态觉察和可指导.这是因为:一方面,状态觉察是系统实现整体自适应的基础和前提,在自主计算系统中占有非常重要的位置,所以我们把它单独抽取出来,作为自主计算定义的另一部分;另一方面,可指导性是确保系统能够反映高层管理目标的关键,也是区别于传统信息系统的主要特征——在传统的信息系统中,系统的管理策略与系统的应用逻辑捆绑在一起,在系统设计和开发时就已经确定,因此在系统运行的过程中不能动态变更,即系统不能接受IT管理者动态策略的“指导”.于是,综合上述分析,我们可以给出如下定义:

定义 2(自主计算系统). 理想地,自主计算系统是一种可指导的、状态觉察的、整体自适应的计算机系统.

值得注意的是,在定义 2 中,要求系统实现状态觉察和整体上的自适应是一种理想的情况.在实际的系统中,自主计算各种特性和功能的实现往往是阶段性的、部分的.例如在文献[20]中,把自主计算的成熟度分为 5 个级别(basic,managed,predictive,adaptive,autonomic),只有到了最后的“自主级别”,才能实现真正意义上的“自主计算”.

1.2 自主计算概念模型

自主计算系统(autonomic computing systems,简称 ACS)由若干相互联系的自主元素组成.在策略的指导下,这些自主元素在实现内部自适应的同时(局部自主管理),通过交互和协作实现整个系统的自适应(全局自主

© 中国科学院软件研究所 http://www.jos.org.cn

管理).

1.2.1 自主元素

自主元素(autonomic element,简称AE)是自主计算系统的基本构造块,它由一个自主管理者(autonomic manager,简称AM)和一到多个被管资源(或其他被管的自主元素)组成.首先,我们分析被管资源和被管的自主元素.以前面自主数据中心为例,AM既可以管理各种内部资源,包括数据库、服务器、路由器、应用模块、Web服务或虚拟机器等,也可以管理其他自主元素.例如,负责全局编排的AM可以管理多个下级的AMs(详见第 1.2.2 节).为了管理这些内部资源或自主元素,一个主要的前提是它们必须是可管理的.因此,对于内部资源,它们需要提供标准化的接口(touchpoint)[21],每个Touchpoint对应于一个传感器/效应器组.对于自主元素,它一方面通过AM管理内部资源,另一方面,它向外提供标准接口(传感器/效应器组)接受管理,包括IT管理者指定的策略以及与其他自主元素的协作信息等;其次,对于自主管理者,它们代替了那些原本需要人工进行的管理工作,包括状态觉察和系统配置、修复、优化和保护等.AM通过一个“监视-分析-规划-执行”控制循环来实现这种管理任务.其中,重要的环节是AM如何依据当前的状态决定该采取什么动作(或目标),而这在许多情况下依赖于高层管理策略.例如,假设自主数据中心的IT管理者原先指定策略“p_1:当A类组件异常时,用B类组件替换”,那么当负责管理内部资源的AM通过分析相关信息并确定A类组件存在异常时,可以依据策略p_1,决定应该采取动作“用B类组件替换A类组件”.后来,由于上层管理策略的变更,策略p_1被改变为“p_2:当A类组件异常时,用C类组件替换”.这时,如果遇到相同的情景,AM将依据p_2进行决策,采取与前面不同的动作.依据上述分析,我们可以得出自主元素的概念体系结构如图 1 所示,该图参考了文献[3,21]的相关思想.不过,我们依据定义 2 关于自主计算系统的概念,强调了策略对自主元素的指导作用.另外,关于AM与外界AMs的合作关系未在图中表示出来.最后,我们给出自主元素的形式定义:

定义 3(自主元素). 自主元素表示为 3 元组:$AE=(AM,MR,Sensors/Effectors)$,式中,$AM$ 是自主管理者,提供对被管资源的自主管理功能;MR 是被管资源(或其他被管的自主元素);$Sensors/Effectors$ 是传感器/效应器组(关于传感器和效应器的具体内涵,详见定义 8 和定义 9).

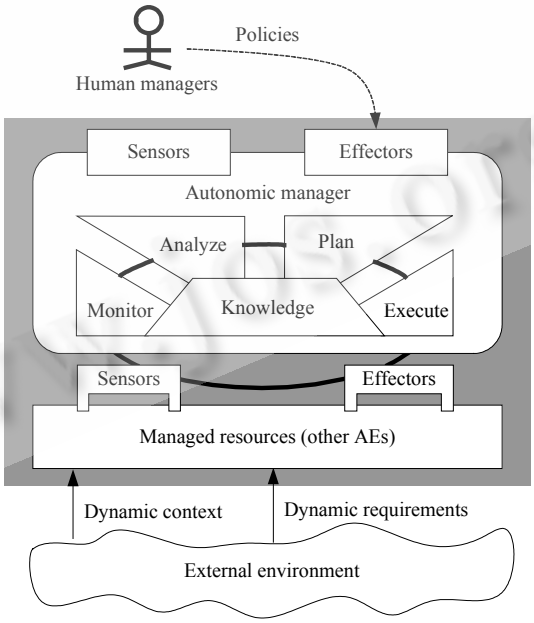


Fig.1 The conceptual architecture of autonomic element

图 1 自主元素概念体系结构

接下来,从“自主管理者及其工作机制”和“被管资源及其与自主管理者的关系”两方面来对自主元素作进

© 中国科学院软件研究所 http://www.jos.org.cn

一步的剖析.

- 自主管理者及其工作机制

如图 1 所示,自主管理者由监视部件、分析部件、规划部件、执行部件和知识库部件组成[3],形成了MAPE(监视-分析-规划-执行)控制环.其中,监视和分析部件提供自我觉察和外部环境觉察的能力,并以此为基础进行自主决策,确定系统的自适应目标;规划和执行部件实现系统状态偏离期望目标时的自适应功能.上述 4 个功能部件均在知识库的支持下运作.知识库中的知识一般可分为 3 类:状态判定知识、策略知识和问题求解知识(分别记作K_D,K_P,K_S),即:AM的知识$K=K_D+K_P+K_S$.状态判定知识K_D包括获得的监测数据和症状等,用于觉察被管资源和外部环境的状态;策略知识K_P定义从状态到动作(或目标)的映射,包括IT管理者定义的策略和通过机器学习获得的策略;问题求解知识K_S包括规划知识、安装和配置等知识,用于在系统状态偏离期望目标时的问题求解.依据AM所实现的功能的不同,AM可分为自配置型、自修复型、自优化型和自保护型,分别记作AM_{sC},AM_{sH},AM_{sO}和AM_{sP}.在一个自主管理系统中,它们既可以管理同一资源,也可以管理不同的资源.下面,我们首先给出自主管理者工作机制的定义,然后分析工作机制中的各个步骤.

定义 4(自主管理者工作机制). 一般来说,AM的基本工作机制可表示为如图 2 所示的情况(图中P_A,P_G和P_U分别表示动作策略、目标策略和效用函数策略)[13].它包含 4 个主要步骤:① 自我觉察/上下文觉察;② 基于策略的决策;③ 目标导向的规划(可选);④ 动作/规划执行.根据不同的问题域和自主管理的目标,每个控制循环有下列不同的工作机制:

i) 1→2a→4;

ii) 1→2b→3→4;

iii) 1→2c→3→4;

iv) 1→2c→4.

在上述 4 个机制中,第 1 个步骤是自我觉察/上下文觉察,它是 AM 进行决策的依据,具体含义详见定义 5.第 2 个步骤是基于策略的决策,它是 AM 的核心部分,依据策略的不同可以分为 3 种不同的决策过程:(2a) Action Selection(动作选择);(2b) Goal Generation(目标产生);(2c) Optimization/Adaptive Control(优化/自适应控制).它们 3 者的具体含义详见定义 6.第 3 和第 4 个步骤分别是规划和执行.另外,在这些机制中,机制 i)和机制 ii)基于知识模型,机制 iv)基于数学模型,机制 iii)采用数学模型和知识模型相结合的方法.关于知识模型方法和数学模型方法详见第 2 节.

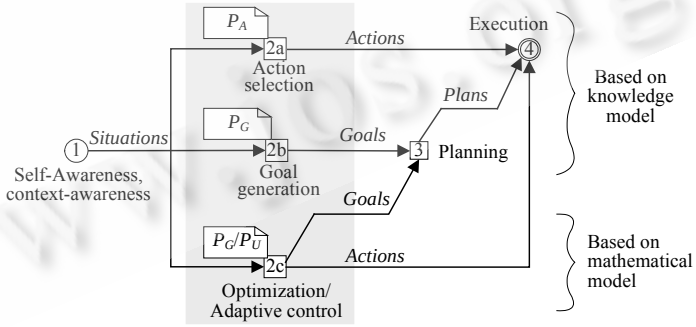


Fig.2 The working mechanisms of autonomic manager

图 2 自主管理者的工作机制

(1) 自我觉察/上下文觉察

自我觉察/上下文觉察是AM实现自主管理的基础,它把系统在某时刻的现象(症状)映射到特定的状态.其中,自我觉察是AM对被管资源状态的觉察;上下文觉察是指AM对外部环境状态的觉察.首先,我们通过一个例子来说明自我觉察的概念.例如,在上述自主数据中心的例子中,被管资源的状态涉及到数据库、存储器、服务

© 中国科学院软件研究所 http://www.jos.org.cn

器、路由器、应用模块和Web服务等资源的可用性、利用率和健康情况等.为了获得这些资源的状态,AM必须掌握两种知识:被管资源模型(M_R)和状态判定知识(K_D),其中M_R用于形式化地说明资源的标识、属性集和属性约束等;K_D表示从传感器获得的(经过适当处理的)数据,包括状态数据和症状等.以存储器为例,其模型可以表示为 (*M-ID*,*WholeSize*,*OccupiedSize*,*HasExceptionInfo*,*Constraints*), 其中,*M-ID* 是存储器标识,*WholeSize* 和 *OccupiedSize*分别是该存储器的全部空间和已占用的空间,*HasExceptionInfo*表示AM是否接收到异常信息,*Constraints*是部分属性的约束值(如*WholeSize*的值被指定为 100GB);相对应地,状态判定知识可以包括与模型属性相关的数据,如已占用的存储空间和异常信息等.基于上述两种知识,通过一个映射函数就可以获得被管资源的当前状态.假设存储器的状态空间表示为$S_{I(memory)}$={*BUSY*,*NORMAL*,*EXCEPTION*},映射函数表示为

{If (*HasExceptionInfo*=TRUE) Then (*state*:=*EXCEPTION*) Else
IF (*OccupiedSize*/*WholeSize*≥80%) THEN (*state*:=*BUSY*) Else
(*state*:=*NORMAL*)}

如果在某一时刻,依据来自传感器的数据得知 *OccupiedSize*=60GB,*HasExceptionInfo*=FALSE,则可以得到存储器的当前状态(NORMAL).同样地,我们可以通过自主数据中心的例子来说明上下文觉察的概念,考虑到篇幅所限,这里从略.以上述例子为参照(具体的映射函数更加复杂和多样),自我觉察/上下文觉察可以形式化地定义如下:

定义 5(自我觉察/上下文觉察). 设S_I是自主元素的内部状态空间,K_D是状态判定知识集合,M_R是被管资源模型,则自我觉察定义为 4 元组:(S_I,K_D,M_R,f_I),式中,f_I: $\wp(K_D)\times M_R\rightarrow S_I$,其中,$\wp$表示幂集;另一方面,设$S_C$是自主元素的上下文状态空间,$K_D$是状态判定知识集合,$M_C$是上下文模型(描述上下文信息提供者的身份、位置和所提供的信息特性和指标等),则上下文觉察定义为 4 元组:(S_C,K_D,M_C,f_C),式中,f_C: $\wp(K_D)\times M_C\rightarrow S_C$.

在定义 5 中,我们假定系统模型和映射函数都是可以确定的(包括通过机器学习进行更新后确定的),而对于一些非常规的意外情景(不包含于系统模型的),在该定义中没有得到体现.不过从目前的情况看,要处理非常规的情景,仍然存在重大挑战,本文暂未进行深入讨论.另外,对于上述定义还有下列相关考虑:首先,由于被管资源和环境的分布性和开放性,K_D,M_R和M_C中的各类知识普遍存在语义上的异构性问题.为了克服该问题,基于本体语言的知识表示和推理被认为是行之有效的方法,相关工作如Cebulla 2005[22],Lehtihet 2006[23]和Stojanovic 2004[24]等;第二,对于K_D中的知识,包括监测数据和症状等,经常是大量动态的、甚至不精确的数据,因此,通常需要通过日志过滤[25]、适配和事件关联[26]等方法获得标准形式的有用知识;第三,根据M_R和M_C规定的各类属性和性能规格等“期望值”和K_D中的状态数据(症状),通过映射f_I和f_C获得当前状态.视应用领域的不同,这两种映射可以有多种方式,包括症状-问题直接映射法[27]、贝叶斯网络[28]、决策树[29]以及数据挖掘[30]等.

(2) 基于策略的决策

定义 6(基于策略的决策). 设S_I和S_C分别是自主元素的内部状态空间和上下文状态空间,K_P是策略知识,Δ为反应方案集(包括个体动作、动作序列或目标),则基于策略的决策定义为 5 元组:(S_I,S_C,K_P,Δ,f_D),式中,f_D:$S_I\times S_C\times K_P\rightarrow\Delta$.

在定义 6 中,策略知识集合K_P包含 3 类策略:动作策略(P_A)、目标策略(P_G)和效用函数策略(P_U)[13].首先,动作策略和目标策略表示为

$$\text{IF } v(s_I,s_C) \text{ THEN } \delta \tag{1}$$

在式(1)中,$s_I\in S_I$,$s_C\in S_C$分别是当前的内部状态和上下文状态;评估函数$v(s_I,s_C)$依据当前状态决定策略是否被执行,$v(s_I,s_C)$的取值包括True和False(运行时判断)或“⊤”(永真);$\delta\in\Delta$是该策略的反应方案.例如,若我们假设$S_{I(memory)}$={*BUSY*,*NORMAL*,*EXCEPTION*}$\subseteq S_I$;$S_{C(memory)}$={*Gold-Request*,*Silver-Request*}$\subseteq S_I$,其中,*Gold-Request*和*Silver-Request*分别表示AM接收到金卡客户和银卡客户的请求,那么,动作策略“如果存储器忙且接收到银卡客户的请求,则拒绝这个请求”可表示为:IF (*state*=*BUSY*) AND (*request*=*Silver-Request*) THEN *Reject*(*request*).同理,也可以采用类似的方法表示目标策略“对金卡用户的反应时间不大于 100 毫秒”(具体表示略).不过,在该策略中评估函数为空,即$v(s_I,s_C)$的取值为永真.

© 中国科学院软件研究所 http://www.jos.org.cn

其次,效用函数策略可以表示为 $U(x)$,其中,x 是一个性能指标向量(如反应时间、可用性等).$U(x)$是一个函数,它把系统的每一个可能状态映射为一个标量值.例如,如果系统的效用(用货币单位表示)只与反应时间有关,则可以依据系统模型和优化算法建立效用函数 $U(RT)$,该函数的横坐标为 RT(反应时间),纵坐标为效用 $U(RT)$.

AM依据上述 3 种策略进行决策,采用不同的决策过程:(2a) Action Selection(动作选择):使用的策略为动作策略,这时,δ是可执行动作,因此,当$v(s_I,s_C)$为真时,如果不存在冲突,则δ被选择,直接输出结果(原子动作);(2b) Goal Generation目标产生:使用的策略是目标策略,这时,δ是抽象目标,因此,当$v(s_I,s_C)$为真时,如果不存在冲突,则δ被选择为目标,并被送往规划部件;上述冲突可能包括策略之间的冲突和因资源约束引起的冲突等.因此,策略的冲突处理成为动作策略和目标策略决策的主要内容[16,31];(2c) Optimization/Adaptive Control(优化/自适应控制):使用的策略为效用函数策略或目标策略,此时的目标策略一般是期望的系统性能指标.对于效用函数策略,AM通过计算系统的最大效用值来确定最优的反应方案(δ):设$C(s_I,s_C)$是当前状态下的约束条件(如可用的资源集),$U_i(x_i)$是与第i个被管资源对应的局部效用函数,则对于一个管理n个($n\geq1$)被管资源的AM,基于效用函数的决策表示为

$$\delta=\arg\max_{\delta\in\Delta}\sum_{i=1}^{n}U_i(x_i)\text{,满足约束条件}C(s_I,s_C) \tag{2}$$

式(2)是指在当前约束条件下,依据系统模型(反映性能指标和系统控制参数之间的关系)计算反应方案集Δ中使得系统效用值最大的反应方案δ,即最优的系统控制参数配置.

(3) 目标导向的规划

如果上述的决策结果δ是抽象目标,那么,AM 的规划部件基于当前状态启动一个规划过程,为相应的规划问题寻找满足目标δ的动作步骤.

定义 7(规划问题). 一个规划问题(PP)是 7 元组:$PP=\langle O,P,S_I,S_C,I,G,A\rangle$.其中,$O=(C,R)$是领域本体(领域的概念化),$C$是领域概念集合,$R$是$C$上的关系;$P$是谓词集合;$S_I$和$S_C$分别是自主元素的内部状态空间和上下文状态空间,$I(\subseteq S_I\times S_C)$是初始状态的完全描述;$G(\subseteq S_I\times S_C)$是目标状态的部分描述;$A$是可执行(原子)动作集合.

依据定义 7,目标导向的规划基于内部资源模型和上下文模型,模型中的各类实体以及内部状态和上下文状态表示为谓词集合.各类谓词及其属性表示为本体.本体的引入使得分布、异构环境中规划的推理有了语义上的支持,方便于谓词及其属性的解释与推理.

(4) 动作/规划执行与反馈

在动作或动作序列的执行过程中,通过效应器监视动作的执行结果,然后检查动作的后置条件是否满足.如果动作失败,则重试该动作或者重新规划以获得另外的路径达到成功.

- 被管资源及其与自主管理者的关系

被管资源(MR)是指各种物理资源和虚拟资源,它们必须满足可管理性,包括状态可观测和状态可调整,前者支持状态觉察,后者支持动作/规划执行.资源的状态数据是指各种能够反映资源当前状态的数据(事件),包括各种日志事件和实时事件,如资源的操作状态、性能状态(吞吐量、资源利用率等)和异常事件等.由于分布式系统的硬件和软件来自不同的服务提供商,因此需要标准的接口来屏蔽资源内部的异构性问题.解决该问题的根本途径是通过标准化和语义化技术来建立传感器和效应器.以文献[21]关于传感器/效应器的相关理论为基础,可以得到如下形式化定义:

定义 8(传感器). 设$T=\{t_1,\ldots,t_n\}$是一组可以反映MR当前状态的特性集;$V=\{v_1,\ldots,v_m\}$是一组反映MR状态变更的事件集;$O=(C,R)$是领域本体,C是领域概念集合,R是C上的关系;$\xi=\{get,report\}$是一组操作集,则传感器(sensor)定义为 4 元组:$Sensor=(T,V,O,\xi)$.其中,对于$\forall t_i,v_j(1\leq i\leq n,1\leq j\leq m)$,有$t_i\in C,v_j\in C$.

定义 8 表明,MR的特性集和事件集均遵从特定的领域本体,通过具有清晰语义的本体语言来表示领域本体,为机器的自动解释和推理、实现自我觉察提供支持[22,23];操作get用于获取MR的状态特性,$get(t_i)$表示AM通过传感器向MR取得特性t_i;操作$report$用于报告MR的状态变更事件,$report(v_j)$表示MR向AM报告事件v_j.

定义 9(效应器). 设$A=\{a_1,\ldots,a_n\}$是一组可以操纵MR状态的可执行动作集;$Q=\{q_1,\ldots,q_m\}$是一组由AM实现

© 中国科学院软件研究所 http://www.jos.org.cn

的、供MR请求的动作集;$O=(C,R)$是领域本体,C是领域概念集合,R是C上的关系;$\psi=\{set,request\}$是一组操作集,则效应器(effector)定义为 4 元组:$Effector=(A,Q,O,\psi)$.其中,对于$\forall a_i,q_j(1\le i\le n,1\le j\le m)$,有$a_i\in C,q_j\in C$.

在定义 9 中,操作set用于执行动作,$set(a_i)$表示AM通过效应器执行动作a_i;操作$request$用于让MR向AM发送请求(如帮助、咨询等),$request(q_j)$表示MR向AM执行请求动作q_j.

基于上述的效应器和传感器,AM以动态绑定或静态绑定的方式与MR联系起来.目前,运用Web服务(网格服务)和语义Web技术实现的(自动)动态绑定成为主流方向[32,33].

- 自主元素自适应特性的实现

在一个自主元素中,AM 在策略的指导下,实现各种局部的自主特性,包括自配置、自优化、自修复和自保护等.对于这些不同的特性,AM都可采取如图2所示的工作机制.不过,基于知识模型的方法一般适合于自配置、自修复和自保护;而基于数学模型的方法则适合于自优化.关于这些机制的具体实现技术及其基本工作原理详见第 2 节.

1.2.2 自主计算系统

自主计算系统由自主元素组成,它主要研究如何把担任各种功能的自主元素联合起来,形成合作的或协作的群体,来实现系统的全局自适应.这与传统的多Agent系统(multi-agent systems,简称MAS)的研究重点类似.在MAS中,现有的研究主要集中在多Agent协调、协商和合作求解以及Agent组织和MAS社会性等问题上[34].这些理论成果为建立自主计算系统理论体系奠定了坚实的基础.不过,ACS有着自己的特点:(1) 需要接受动态的IT策略的指导;(2) 重点研究如何利用个体自主元素的自我管理功能来实现整个系统的动态平衡.因此,把ACS特点和MAS理论结合起来,可望建立满足要求的自主计算系统.参照MAS理论,ACS的研究内容将非常丰富.鉴于篇幅所限,本文仅讨论自主元素组织结构、AM协作关系形成和AM行为编排等 3 个主要方面,其中,AM组织结构和AM协作关系形成主要研究ACS的静态特征;AM行为编排则研究ACS的动态特征.

首先,从系统体系结构上看,ACS与MAS相比增加了一层策略管理框架(policy framework)(如图 3 所示),用于提供人机接口功能,把IT管理者制定的抽象管理策略细化[35,36](转化、映射)为自主管理者可理解和操纵的可实施策略;自主管理层(对应于多Agent系统)由上述 4 种类型的自主管理者(AM_{sC},AM_{sH},AM_{sO}和AM_{sP})组成,分别称为自配置型、自修复型、自优化型和自保护型自主管理者.这些自主管理者通过一定的方式参与交互,实现系统级的(宏观的)自主管理行为.

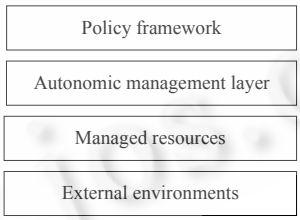


Fig.3 The conceptual architecture of ACS

图 3 ACS 概念体系结构

下面从 AM 组织结构、AM 协作关系形成、AM 行为编排这 3 个方面来阐述 ACS 的主要协作机制.

- AM 组织结构

在一个ACS中,参与协作的AM以一定的方式形成组织,以便提供系统级的自主管理功能.与分布式AI系统类似[37],AM组织结构研究个体AM之间的信息和控制关系模式.一般来说,由AM组成的组织结构包括 3 种基本形式:层次(分级)结构、对等(P2P)结构和混合结构.如图 4 所示.

定义 10(AM组织结构). 在AM组织结构中,包含 3 个基本要素:自主管理者、AM之间的状态信息传递关系(state information,简称SI)和控制信息传递关系(control information,简称CI).AM组织结构定义为 3 元组:$G_C=(M,E,F)$,式中,M是所有AM的集合,E是SI的集合,F是CI的集合.

(1) 层次结构满足条件:$\forall AM_i,AM_j\in M$,如果AM_j是与AM_i相邻的下层AM,且存在协作关系,则(AM_j,

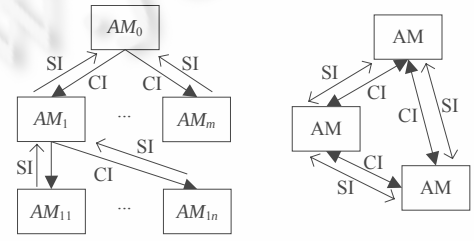
© 中国科学院软件研究所 http://www.jos.org.cn

$AM_i)\in E,(AM_i,AM_j)\in F,(AM_i,AM_j)\notin E,(AM_j,AM_i)\notin F$;否则,如果$AM_j$与$AM_i$在同一层或不在相邻层,则它们之间不存在SI和CI.

(2) 对等结构满足条件:$\forall AM_i,AM_j\in M$,允许$(AM_i,AM_j)\in E,(AM_j,AM_i)\in E,(AM_i,AM_j)\in F,(AM_j,AM_i)\in F$同时成立.

(3) 混合结构满足条件:$\forall AM_i,AM_j\in M$,如果AM_j是与AM_i相邻的下层AM,且存在协作关系,则$(AM_j,AM_i)\in E,(AM_i,AM_j)\in F,(AM_i,AM_j)\notin E,(AM_j,AM_i)\notin F$;否则,如果$AM_j$与$AM_i$在同一层,则允许$(AM_i,AM_j)\in E,(AM_j,AM_i)\in E,(AM_i,AM_j)\in F,(AM_j,AM_i)\in F$同时成立.

依据定义10,在层次结构中,上层AM可以向其下层AM传递控制信息,下层AM则向其上层AM传递状态信息;上层AM控制系统的宏观(或中观)自主特性,CI型出度为0的AM为底层自主管理者,实现微观控制.例如,在文献[38,39]中分别描述了基于控制论和效用函数优化的两层自主计算系统.在对等结构中,参与协作的AM不存在等级关系,控制信息和状态信息的传递是双向的,系统的全局自主特性通常是在个体的局部交互中"涌现"出来的.例如,在文献[40,41]中描述了基于自组织涌现理论的体系结构.在这种体系结构中,AM的关系是对等的,不存在管理全局自治行为的AM,即系统的宏观自主特性是在AM的局部交互中产生的.在混合结构中,上层AM可以向其下层AM传递控制信息,下层AM向其上层AM传递状态信息;上层AM控制系统的宏观(或中观)自主特性,下层AM则基于上层AM提供的约束,通过交互实现该层的宏观特性.



(a) Hierarchical architecture (b) P2P architecture
(a) 层次结构 (b) 对等结构

Fig.4 The organizational structures of AMs

图4 AM组织结构

- AM协作关系形成

担任各种功能的、分布的AM之间通过形成协作关系,建立上述组织结构.协作关系的形成包括静态和动态两类.静态的协作关系在系统设计时确定,运行时不能动态变更;动态的协作关系则在系统运行时动态形成,可以随着系统的状态而变更,包括协作关系的建立、解除和更新等.例如,在上述自主数据中心的例子中,对于负责管理中心内部资源(不同的数据库、存储器等)的AMs,如果它们之间的协作关系在系统设计时就能确定,则可以形成静态的关系;而如果这些AMs属于处在开放环境中的实体,可以动态地加入或退出一个组织(虚拟组织),则需要通过服务发现、协商等手段建立动态的协作关系.由上述分析可知,AM协作关系的形成机制类似于开放、动态的多Agent系统中的对应机制[42].因此,基于文献[42]的工作,我们可以得到如下定义:

定义 11(AM协作关系动态形成). 处在分布式环境中的AM之间协作关系形成建模为7元组:(M_P,M_C,S,O,SLA,f,g),式中,M_P,M_C分别表示服务提供方AM集合和消费方AM集合;S表示需求方所需的服务集合;O是领域本体,用于支持服务和服务级协定的表征;SLA表示服务级协定集合;$f:M_C\times S\rightarrow \wp(M_P)$表示服务提供者发现操作;$g:M_C\times S\times M_P\rightarrow \wp(SLA)$表示服务协商操作.

依据定义11,AM之间通过发现和协商操作形成服务级协定,确立协作关系.其中,服务发现操作的实现通常建立在注册表(或服务中介)的基础上:服务提供方AM把服务能力发布在注册表上,服务使用方AM通过查询和匹配找到候选AM集合.

- AM行为编排

上述的AM组织结构和协作关系形成给ACS的全局自适应建立了一个基本框架,但为了真正实现系统全局

© 中国科学院软件研究所 http://www.jos.org.cn

自适应这个目标,一个关键的环节是如何合理地编排(orchestrate)ACS中各个AM的行为,使它们协调一致地工作.这就好比一群舞蹈演员,它们各自可以进行各种舞蹈动作,但在没有组织起来之前,从整体上看却是一盘散沙.然而,仅仅把其中的一些演员组织在一起并规定他们的结构关系,仍然只形成了舞蹈节目的静态特征.因此,为了完成各种独具特色的舞蹈节目,需要恰当地编排他们的行为.一般来说,系统级的目标(特性)可以分解为子系统(或个体)的子目标(子特性),而通过适当的映射,可以把一组子目标(子特性)集成为总体目标(特性).事实上,这种集成的思想并不是很新的,在传统的MAS系统中也经常采用这种方法[43].不过,对于ACS来说,其目标是把实现不同自主特性(自配置、自修复、自优化和自保护等)的AM有机地集成起来,实现全局的自主特性.例如,在上述自主数据中心的例子中,当系统受到一个恶意攻击而发生异常时(如"A类组件A_1异常"),担任自修复功能的AM使用策略"当A类组件异常时,用B类组件替换",启动了修复过程;同时,担任自我保护功能的AM发现了攻击的原因,它依据特定的策略修改了相关配置;然后,在B类组件加入系统后,担任自配置和自优化功能的AMs共同把系统调整到最优状态.在这个过程中,担任不同功能的AM协调一致地工作,实现了系统的全局自适应.另外,也存在同类AM之间的行为编排.例如,在系统初始化时,负责配置本地资源的AMs结合起来,实现系统的全局自配置目标.所以,AM的编排模式可以分为两类:同类AM的编排(如图 5 上方所示)和不同类型AM的编排(如图 5 下方所示)[21],前者把同类AM以特定的方式组织形成某种体系结构,并在此基础上进行行为编排,如文献[39,44];后者则由不同类的AM(AM_{sC},AM_{sH},AM_{sO}和AM_{sP})编排起来,如文献[45,46].依据上述分析,可以得出AM行为编排的定义:

定义 12(AM行为编排). 设$M=\{AM_1,\ldots,AM_n\}$是一组参加协作的AM集合,γ_0为全局目标,$\Gamma=\{\gamma_1,\ldots,\gamma_n\}$为局部目标集合,则AM行为编排定义为 4 元组:$(M,\gamma_0,\Gamma,h)$,式中,$h$: $\wp(M)\times\wp(\Gamma)\rightarrow\gamma_0$,即存在$M$的一个子集和$\Gamma$的一个子集,通过合理的编排,可以实现目标$\gamma_0$.

在定义 12 中,编排方法h与领域有关,典型的方法包括基于动态规划的方法[47,48]、基于工作流的方法[32]和基于自组织涌现理论的方法[40,41]等.

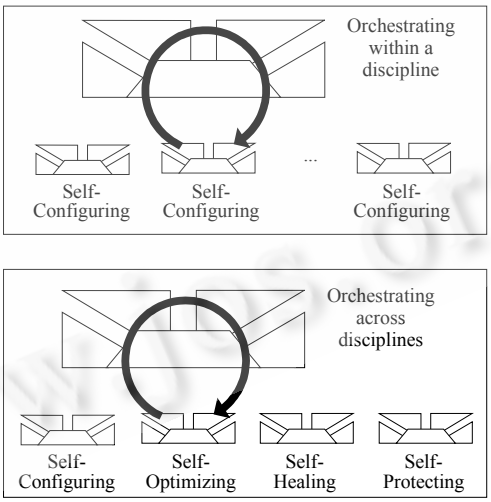


Fig.5 Orchestrating patterns of AMs

图 5 AM 编排模式

- ACS 协作机制

定义 13(ACS 协作机制). 粗略地看,ACS 的上述 3 个方面构成了 12 种可能的组合:{层次,对等,混合}×{动态,静态}×{同类,异类},分别对应于 ACS 各种可能的协作机制.

例如,文献[32]阐述的Web服务自主合成系统可以表示为:(混合,动态,异类),它把系统分为两层:协同管理者和服务管理者.协同管理者控制全局的自主管理:一方面,它把应用工作流分解为交互规则,并把这些规则分配

© 中国科学院软件研究所 http://www.jos.org.cn

给负责局部自我管理的服务管理者.服务管理者通过执行交互规则构成动态的协作关系,实现自配置功能;另一方面,协同管理者把自优化规则传递给适当的服务管理者,后者执行这些规则来调整各自的性能,实现自优化.

1.2.3 自主计算系统概念模型的特点及其与多 Agent 系统的区别

在上文中,我们依据自主计算的概念和定义,在现有技术的基础上,提出了自主元素和自主计算系统的基本工作原理和机制.

首先,AE作为ACS的基本构造块,可以在IT管理者制定的策略的指导下,通过感知内外环境实现自我觉察/上下文觉察;通过自主决策决定自适应动作(目标);通过对目标的自动规划,动态形成动作序列;通过与其他AEs的协作形成自主计算系统等.理论上,这些特点与传统软件Agent的特点很相似. Jennings和Wooldridge 1995 年指出,Agent具有自主性(autonomy)、反应性(responsiveness)、预动性(proactiveness)和社会能力(social ability)等特点[34].其中,自主性是指Agent能够在没有人或其他Agents的直接干预下解决大部分问题,并能够控制自己的动作和内部状态;反应性表示Agent能够感知环境并能对所发生的改变做出即时的反应,因此,它具有类似于自主元素的状态觉察能力;预动性是指Agent能够自发地产生目标导向的行为;社会性是指Agent可以与其他Agents或人进行交互和合作.不难看出,这些特点与AE的特点有着一一对应的关系.不过,AE与Agent也存在重要区别.在传统的Agent模型中,Agent的推理知识(规则)在系统的设计和开发时就已经确定,在系统的运行过程中不能变更,因此,在本质上它的推理过程是单调和线性的[49].这样,传统的Agent就无法处理动态的管理策略(即它是“不可动态指导的”).例如,若要将Agent知识库中的规则“当*A*类组件异常时,用*B*类组件替换”调整为规则“当*A*类组件异常时,用*C*类组件替换”,则需要对系统重新编码.反观AE的工作机制(如图 2 所示),它是在策略指导下工作的,运行策略可以动态变更,因此是“动态可指导的”.

其次,从系统的层面看,ACS与多Agent系统有着许多类似的特点.事实上,上述的AM组织结构、AM协作关系形成和AM行为编排均是建立在多Agent协作理论的基础上.不过,它们二者之间也存在重要区别: MAS不存在整体上的自适应机制,即系统没有综合利用自配置、自优化、自修复和自保护在内的自我管理机制来实现系统全局的“动态平衡”.以上述自主数据中心为例,现有基于Agent的技术(例如多Agent协调、协商和合作求解以及Agent组织形成机制等[42,50])虽然能够支持各种资源(服务)的集成和联合的问题求解,但它们没有从支持系统整体自适应的角度进行研究,即没有考虑如何把实现各种自主特性的Agent联合起来,通过合理编排来实现系统级别的动态平衡(在第 1.2.2 节的“AM行为编排”段落中,举例说明了自主数据中心的全局自适应机制,这里从略).

最后,Agent 和多 Agent 系统的研究出发点不是面向系统自我管理的(包括自配置、自修复、自优化和自保护等),它们所采用的理论和技术都相对单纯;而自主元素和自主系统以系统自我管理为目标,因此需要结合各种不同的理论和技术,包括自适应控制、面向服务的计算、软件体系结构和工程、人-系统接口、策略、建模、优化以及人工智能的众多分支.

综上所述,自主计算系统与多 Agent 系统的相同点和差异可以表示如下(见表 1).

Table 1 Comparison result between autonomic computing system and multi-agent system

表 1 自主计算系统与多 Agent 系统的比较结果

Entities \ Charateristics	Autonomous	Self-Aware context-aware	Self-Planning	Social ability	Guidable at run time	Homeostasis as a whole	Oriented to self-management
Autonomic computing system						Yes	Yes
Multi-Agent system						No	No
Autonomic element	Yes	Yes	Yes	Yes	Yes		Yes
Individual agent	Yes	Yes	Yes	Yes	No		No

2 两类主要的自主计算实现方法

依据自主计算的定义和概念模型,一方面要求自主元素能够在自我觉察/上下文觉察(回答“为什么管理”的问题)的基础上,通过基于策略的决策确定反应方案并规划执行(回答“怎么管理”的问题);另一方面要求自主元素之间以一定的方式和机制形成组织,并通过有效的协作实现系统级的自我管理功能.现有的许多理论和技术

© 中国科学院软件研究所 http://www.jos.org.cn

为上述两个要求奠定了良好的基础,它们包括基于策略的管理(policy-based management,简称 PBM)、面向服务的技术、智能主体技术、自适应控制理论、机器学习、优化理论等.以这些技术为基础的自主计算系统可以分为两种基本类型:知识模型方法和数学模型方法.

2.1 知识模型方法

依据文献[51],“知识模型是运用人工智能或知识工程的方法和技术,例如:知识表达方法(产生式规则、语义网络、框架等)、知识获取技术(人工移植、机器感知、机器学习等)所建立的知识模型”.用知识模型方法建立的自主计算系统的特点是:知识库中的 3 类知识(状态判定知识、策略知识和问题求解知识)用知识加以表示;同时,AM 的决策是基于知识的分析和逻辑推理,采用“1→2a→4”和“1→2b→3→4”的工作机制(定义 4).

目前,从知识模型的角度建立的实际自主计算系统主要基于下列技术:Agent技术、Web服务和语义Web技术等.其中,Agent又称为智能主体或智能代理,具有反应性、自治性和社会性等特点,能够感知环境、做出反应(反应Agent)或通过慎思和规划实现目标导向的行为(慎思Agent),已被普遍认为是支持大规模、开放和分布的信息系统实现动态服务集成和协同工作的关键技术[43,52,53].在Agent技术的基础上,引入基于策略的管理方法(即把动态策略或目标策略作为Agent决策知识的全部或一部分),结合Web服务和语义Web技术,可以建立各种自主计算系统.下面,我们介绍两种主要的方法.

2.1.1 方法 1:Web 服务+动作策略+反应 Agent

Web服务是一种分布式计算方法,它通过标准的服务接口和协议规范(WSDL,SOAP和UDDI等)[54]支持异构环境中各类资源的动态绑定和互操作.利用Web服务体系结构、接口和协议规范,通过适当的扩展,可以建立自主元素和自主元素之间的协作关系.

首先,在自主元素层面上,主要的扩展包括:(1) 扩展现有Web服务的端点,使之具备可管理性:增加管理端点和相应的实现,以支持Web服务状态信息的获取以及Web服务行为的控制.这样,用WSDL(Web services description language)表示的管理端点可以被自主管理者访问[55,56];(2) 在现有Web服务基础设施之上增加一个自主管理层[7,32,57−59],用于实现自我觉察/上下文觉察、自主决策和动作执行.其中,用于自主决策的知识K_P为动作策略,表示为IF Condition THEN Action的形式(对应于公式(1)中δ是可执行动作的情况);用于决策的自主管理者是反应Agent(规则引擎). AM的控制循环采用图 2 中第i)种方案,即“1→2a→4”.通过上述扩展的基于Web服务的自主元素如图 6 所示.在该控制循环中,用于决策的知识是动作策略,策略可以动态添加、删除或更新,因此可能引起策略冲突,需要引入适当的机制进行冲突处理[16,31].

其次,在系统层面上,多数系统采用了层次[7,58]或混合体系结构[32];上层AM起着全局协同的作用,通过工作流对下层AM的行为进行动态编排[7,32].

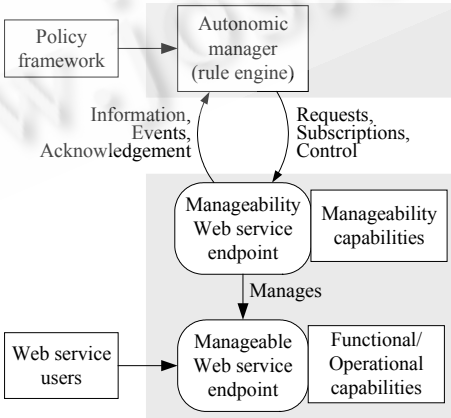


Fig.6 A reactive agent based autonomic element

图 6 一种基于反应 Agent 的自主元素

© 中国科学院软件研究所 http://www.jos.org.cn

2.1.2 方法 2:语义 Web 服务+目标策略+慎思 Agent

在方法 1 中,基于动作策略的反应Agent缺乏动态的目标产生机制,无法进行目标导向的自我规划,不能通过发现与协商自主地建立动态的协作关系;同时,Web服务的标准和规范缺乏语义互操作的能力.因此,用方法 1 建立的自主计算系统仍需要较多的人工管理,包括制定底层的策略规范、建立静态的协作关系等.鉴于慎思Agent可以依据环境和自身状态进行逻辑推理、产生目标、进行目标导向的规划,同时,能够进行自动的服务发现和协商[42,43,53],所以,用慎思Agent代替上述反应Agent,用本体语言(如DAML+OIL,OWL等)描述管理Web服务端点和被管Web服务端点,可以建立更加强大的自主计算系统.在这类系统中,AM的控制循环采用图 2 中第ii)种方案“1→2b→3→4”.

首先,在自主元素层面上,由Agent担任自主管理者.依据经典的Agent理论,即BDI(信念-愿望-意图)模型,Agent能够依据当前信念(通过自身觉察/上下文觉察)产生愿望和意图,并自动规划执行,因此,人类用户可以不必关注底层的技术细节,只需制定抽象的目标策略即可调控系统的行为,可以进一步降低人工管理负担.目前,在基于慎思Agent的自主计算研究方面已经有了初步进展.例如,Tianfield等人提出了基于多Agent的自主体系结构,把Agent作为自主元素,通过它们的协同工作,实现全局的自主管理功能[19,60];De Wolf等人提出了一种分散式自主计算,通过自组织涌现来实现宏观的自主特性[61];张海俊和史忠植提出了一种多Agent系统设计方法DPMAS(design process for multi-agent system)用于自主计算系统的建模[9].在这样的系统中,对于个体Agent而言,由于IT策略的引入,使得Agent的动机来源发生了根本改变:当引入动态策略后,Agent动机除了其内部愿望以外,还包括由策略产生的义务(obligation).这些不同的动机之间可能产生冲突.因此,需要从理论和体系结构上对传统Agent进行扩展,使其不仅对外提供管理接口,而且可以接受动态策略的指导、处理可能的冲突、形成一致的目标.廖备水等人在这一方向进行了研究,并取得了初步成果[49,62,63].图 7 是基于该思想的一种由慎思Agent担任自主管理者的自主元素[49].这种新型的慎思Agent能够接受动态变更的IT管理策略(表示为策略规则)的指导,并结合协作合同进行非单调推理,具有可指导性和自主性等特点.

其次,在系统层面上,自主管理系统就是多Agent系统.在策略指导下,Agent之间通过各种形式的交互与协作实现全局的自我管理:(1) Agent和语义Web技术相结合,可以自动解释和处理服务端点信息和通信信息,进行服务的自动注册、发现、协商和合成;(2) Agent之间形成层次结构或对等结构,通过适当的方式进行行为编排,实现系统级的自主管理.对于层次体系结构,Agent行为编排可以使用基于角色的动态规划或动态工作流;对于对等式体系结构,多Agent“涌现(emergence)”机制可望发挥重要作用[64].

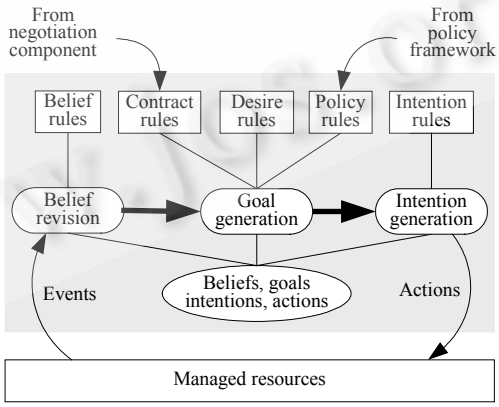


Fig.7 A deliberative agent based autonomic element

图 7 一种基于慎思 Agent 的自主元素

2.2 数学模型方法

上述知识模型方法只能进行定性分析和逻辑推理,因此不适合于那些需要定量描述系统的有关过程和特性的场合,如对系统响应速度、吞吐量、CPU和存储器的利用率等系统性能的管理.数学模型方法可以弥补这一

© 中国科学院软件研究所 http://www.jos.org.cn

不足,运用控制论或运筹学的理论、方法所建立的数学模型[51]能够依据不断改变的资源和环境状态自主决定系统参数的调整,使得系统性能保持在期望的范围内.用数学模型方法建立的自主计算系统的特点是:知识库中的策略知识是期望的性能指标或效用函数策略;问题求解知识是系统的性能模型;同时,AM的决策是基于性能模型的优化,采用“1→2c→4”的工作机制(定义 4).

2.2.1 方法 3:基于(自适应)控制理论的自主计算

控制理论是一门理论严谨、分支众多的学科领域,其中,自适应控制(adaptive control)是现代控制中最具活力的分支之一[65].自适应控制系统能够依据动态的被控对象和环境,通过测量输入/输出信息实时地获得被管对象和系统误差的动态特性,并依据其变化情况按一定的设计方法自动地做出控制决策、修改控制器参数,使其控制信号适应对象和扰动的变化,使得系统的控制性能维持最优或者满足要求.因此,通过适当的扩展,自适应控制理论可用于建立自主计算系统,尤其是自优化系统.图 8 是一种基于自适应控制理论的自主元素工作原理图,图中y是系统输出,u是输入的控制量,z是系统扰动.自主管理者由在线参数估计器、优化器和控制器组成,它们构成了两个控制环(i)和(ii),分别表示一般反馈控制环和参数自适应控制环.其中,参数估计器通过测量被管资源的输入/输出信息,估计出系统参数;优化器则依据当前的系统参数计算出控制器的控制参数;最后,由控制器给出具体的控制量并施加给系统.

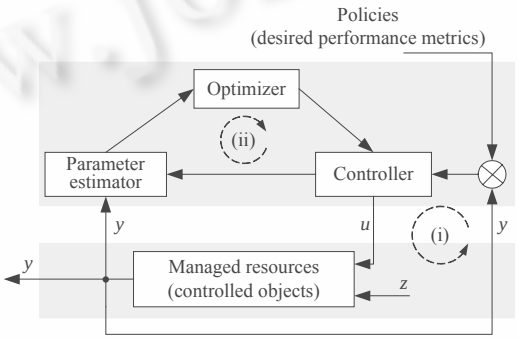


Fig.8 An adaptive control theory based autonomic element

图 8 一种基于自适应控制理论的自主元素

在基于控制理论的自主管理方法中,通常把系统的性能保证问题看作是反馈控制问题,即通过建立一个反馈控制系统来实现对系统性能的自主管理.如Abdelwahed等人提出的能够对计算需求和环境条件变化做出反应并进行连续性能优化的在线控制框架[38]、Abdelzaher等人提出的使用控制理论框架来实现软件系统QoS保证的方法[66]、Diao等人阐述的控制理论方法[67]、Wang等人提出的运用最优控制理论来管理运行于动态和不确定环境中的计算机集群的性能的方法[68],等等.在这些方法中,主要的系统构造任务包括:分析软件体系结构并把它建模为一个反馈控制系统;把特定QoS控制问题映射为一个系统的控制循环;选择适当的传感器来可靠地测量当前性能和一个效应器来实现系统行为的调整;为系统设计一个控制器.

另外,在系统层面上,基于控制理论的方法一般采取层次体系结构[38,69,70].在层次体系结构中,控制器分为两类:局部控制器和全局控制器.一组局部控制器之间的交互由一个全局控制器管理;全局控制器所使用的系统模型是抽象模型,它包含与全局目标相关的信息,如用于刻画系统组件之间的交互细节的局部变量.全局控制器管理系统的宏观特性,它通过命令的方式对每个局部控制器施加操作约束;每个局部控制器则基于操作约束优化本地的软件性能.

2.2.2 方法 4:基于效用函数的自主计算

在经济学中,效用是指当一个消费者消费一件商品或服务时他所获得的福利,反映的是商品或服务所带来的满足程度.在自主计算的背景下,效用函数把实体(自主元素)的每个可能状态(系统性能)映射为一个实数值[17,71],用于指示与系统性能(如反应时间、延迟、吞吐量等)对应的价值.基于效用函数,通过下列方式可以实现

© 中国科学院软件研究所 http://www.jos.org.cn

系统的自优化:用效用函数来表示管理策略;依据当前的系统性能模型,通过效用的最大化,得到期望的系统状态和相应的可调系统参数的取值情况(参见公式(2));最后调整系统参数,使系统达到期望的状态.基于该原理的一种自主元素如图 9 所示.其中,自主管理者由系统建模和基于效用的优化两部分组成:系统建模部分用于建立系统性能模型,即系统的性能指标与控制参数之间的关系;效用的优化部分依据系统性能模型、效用函数策略和当前的约束(如可用的资源),计算出一组使得效用最大化的控制参数集.

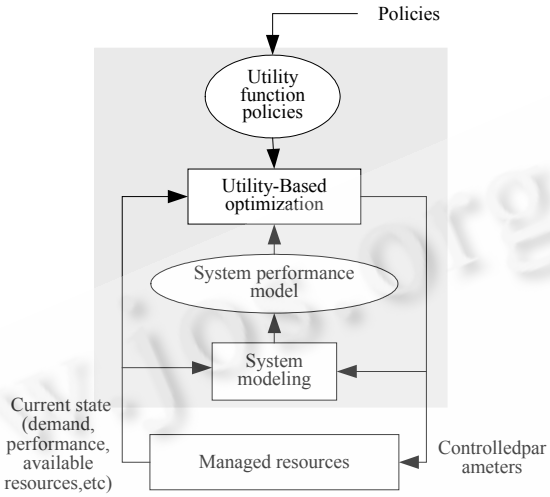


Fig.9 A utility function based autonomic element

图 9 一种基于效用函数的自主元素

另一方面,在基于效用函数的自主计算系统中,一组AM通常组成层次体系结构[17,72,73].例如,在文献[73]中,把系统分为两层:上层是资源仲裁者,负责全局的资源分配,实现全局效用的最大化;下层是应用管理者,对于给定的资源,应用管理者通过调整局部参数,实现本地效用的最大化.应用管理者之间的行为编排通过效用的转化和计算来实现:应用管理者把本地服务级效用函数转化为资源仲裁者使用的资源级效用函数,后者通过计算系统级的效用得到全局的资源分配方案,并以此来调整下层应用管理者的行为.

3 比较与分析

上述基于不同技术的自主计算有着各自的特点.下面从功能特性、AM决策机制和ACS协作机制 3 个方面进行分析和比较(见表 2).表 2 中,sC,sH,sO和sP分别表示自配置、自修复、自优化和自保护;P_A,P_G和P_U分别表示动作策略、目标策略和效用函数策略;H,P和Y分别表示层次、对等和混合体系结构.

Table 2 Comparison result of autonomic systems based on different technologies

表 2 基于各种不同技术的自主系统比较结果

Methods \ Compared items		Mainly supported properties				AM decision-making mechanism			Coordination mechanisms		
		sC	sH	sO	sP	Policy type	Policy conflict	Decision-Making approach	Architecture	Relation forming	Behavior orchestrating
Knowledge model based methods	Method 1	√	√		√	P_A	Yes	Rule engine	H/Y	Static	Workflow, plan
	Method 2	√	√		√	P_G	Yes	Agent deliberating	$H/P/Y$	Static dynamic	Workflow, plan, self-organization
Mathematical model based methods	Method 3	√		√		P_G	No	Control optimization	H	Static	Variable constraints
	Method 4	√		√		P_U	No	Utility optimization	H	Static	Utility transformation

首先,在功能特性上,知识模型方法主要支持系统的自配置、自修复和自保护,但无法很好地支持系统的自

© 中国科学院软件研究所 http://www.jos.org.cn

优化,其主要原因是:在知识模型方法中,AM 的行为选择依赖于专家知识(动作策略或目标策略),只能对系统进行定性的判断与调整;同时,当系统的状态发生定量改变时,无法对系统的性能参数作出连续的调整,以便使得系统性能保持在期望的范围内;相反地,数学模型方法通过优化算法,能够对系统作出定量的判断和调整,可以较好地满足系统的自优化要求.不过,在自修复和自保护方面,数学模型方法就显得无能为力,这是因为系统的问题判定和处理以及系统的授权与认证等均依赖于知识表达、获取和推理技术,而这些恰好是知识模型方法所具备的.由此可以看出,上述两类方法具有互补性,因此,把二者相结合进行异类 AM 的编排,可以建立较完善的自主计算系统.

其次,在个体 AM 工作机制上,知识模型方法与数学模型方法所面临的技术挑战不同:

一方面,知识模型方法在状态觉察的基础上使用规则引擎(反应 Agent)或基于 BDI 逻辑的慎思 Agent,在动作策略或目标策略的指导下进行推理和决策.它们共同面临的主要技术挑战包括:

(1) 状态觉察和问题确定:(a) 在状态信息的获取和过滤方面,涉及到监测数据的标准化、数据来源和可信度确定、原始数据的处理和分析等问题,需要日志过滤[25]、适配和事件关联[26]、统计分析、数据挖掘和机器学习等技术的支持.不过,这些技术的使用可能带来附加的问题,包括计算复杂性问题和系统效率问题等;(b) 在问题确定方面,涉及到如何产生和维护症状库、如何判定和追溯处于动态环境中的由多个交互元素引起的问题根源以及如何产生自动关联规则等[27,30].

(2) 策略的细化和冲突处理:(a) 从高层IT策略到底层可实施策略的映射和细化是构建该类自主系统的关键环节,其技术难题是如何自动地对策略进行细化,并分析和验证细化结果的一致性和完整性[35,36,74,75];(b) 策略在实施的过程中,可能引起逻辑上或资源上的冲突,因此,策略冲突处理[13,16,31,61,76]和自适应[77]成为该类AM决策的主要问题.

另外,知识模型中的两种方法也有所区别.对于方法 1,反应 Agent 使用动作策略,推理机制简单、易于实现,但由于动作策略属于底层的控制规范,抽象级别低,需要系统管理者了解技术细节,管理开销大;同时,在设计和制定动作策略时,要求动作策略能够覆盖所有的状态空间,使得每个状态映射到唯一的动作,因此,当被管资源较复杂时,设计动作策略变得困难.对于方法 2,慎思 Agent 使用目标策略,能够自主决定如何采取动作满足目标,因此可以进一步降低系统管理者的管理负担.但基于慎思 Agent 的自主元素需要复杂的系统模型,计算复杂度高,系统实现和验证技术不够成熟;同时,解决策略冲突仍然困难,可以优化个体目标,但多个目标组合引起的冲突难以处理.

另一方面,知识模型方法使用效用函数策略或指定的性能指标进行优化决策,多目标冲突的折衷方案直接蕴涵于优化的过程之中,因此自然解决了冲突.这类方法的主要挑战是:(1) 效用函数的建立;(2) 系统性能模型的建立.

第三,ACS 协作机制方面,两类方法都可以采用定义 13 的各种协作机制,实现宏观的自主特性.对于知识模型方法,如果采用层次或混合体系结构,系统的宏观特性由上层 AM 协调控制,下层 AM 的行为编排由上层 AM 控制(如采用工作流或规划的方法),具有易于实现的特点,但这样的体系结构不适合于大规模分散式系统.反之,如果采用对等式体系结构,由于各 AM 的行为不受集中的控制,因此更适合于大规模分散式系统(例如,基于突现理论的动态自组织系统),但如何通过 AM 的局部交互实现全局的自主特性仍然是一个挑战.

另外,在协作关系形成方面,在方法 1 中,参与协作的 AM 需要静态确定,因此在自配置和自修复的功能上存在缺陷,即当需求变更或协作失败时,缺少自动获取替代资源的能力;在方法 2 中,慎思 Agent 具有的社会行为能力使得 AM 之间可以通过动态发现和协商建立关系,因此可以实现动态的组织形成.

另一方面,对于数学模型方法,一般采用层次式体系结构,协作关系静态确定.

4 研究展望

上一节介绍了运用知识模型方法和数学模型方法建立的自主计算系统以及各自面临的挑战.为了克服这些挑战,建立可指导的、状态觉察的、自适应的自主计算系统.我们认为,多种理论、方法和技术的融合将成为

© 中国科学院软件研究所 http://www.jos.org.cn

未来研究的总的发展趋势,具体包括:

(1) 知识模型方法和数学模型方法相结合:由表 1 可知,两类方法具有互补性,因此,把它们有机地结合起来进行异类 AM 的行为编排,可望建立更加完善的自主计算系统.该方向的重点将是研究异类 AM 行为编排的机制和方法;研究运用效用函数来指导慎思 Agent 的决策(即采用定义 4 的第 iii)种方案:"1→2c→3→4")的相关理论和方法,等等.

(2) 标准化和语义化相结合:现代分布式信息系统具有开放性和异构性的特点,使得AM的状态觉察和交互协作都需要克服被管资源和环境信息在语法和语义上的异构性问题.为了使各种异构的资源、软件、管理软件能够进行有效的交互和协同工作,需要各类Web标准和协议的支持,如XML,SOAP,WSDL[54],WSDM[55]等.不过,这些标准和协议缺乏语义上的互操作能力,无法支持智能软件(如慎思Agent)对信息内容的自动解释和推理,因此需要结合现有的语义Web技术,采用基于Web标准的本体语言,如RDF[78],DAML+OIL[79],OWL[80]等,在被管资源、环境和自主管理者之间建立标准化的、语义清晰的接口,以支持传感器、效应器和决策部件对各类信息的处理.

(3) 硬计算和软计算相结合[81]:在上述的AM状态觉察、AM决策和ACS协作等各个方面,硬计算和软计算展示出了不同的特点.首先,包括基于动作策略的决策、基于症状映射的问题判断、基于工作流的协作等在内的"硬计算"具有易于实现、算法计算复杂度低等优点,但却难以适应不确定、不精确、部分真的情况,因此需要诸如神经网络理论、模糊逻辑、概率推理、遗传算法、混沌理论、机器学习等"软计算"的支持.

5 讨论与结论

自主计算作为新兴的热点研究领域,近年来在概念、理论、技术和应用等层面都取得了重要进展.

(1) 概念.自主计算的概念最早由IBM公司高级副总裁Paul Horn于 2001 年 3 月在哈佛大学作主题报告时提出[2].之后,Kephart 于 2003 年系统地论述了自主计算的愿景[3],指出自主计算的本质是自我管理(self-management),其目的是使系统管理者可以从系统操作和维护的细节中解脱出来,为用户提供全天候的运行于性能高峰的机器.他详细论述了自我管理包含的 4 个主要方面:自配置、自优化、自修复和自保护.后来,一些学者在此基础上进一步发展了自主计算的概念.例如,Tianfield于 2003 年添加了自规划、自学习、自调度、自演化等特性[19];Sterritt于 2005 年增加了自管制、自适应、自恢复、自诊断等特性[18];Hinchey于 2006 年认为自主计算还应具有"自我毁灭(self-destruction)"的特性,它可以在自我保护中发挥重要作用[82].另外,Ganek于 2003 年[83],White于 2004 年[84],Tesauro于 2004 年[39],Parashar于 2005 年[85]和De Wolf于 2007 年[64]等也分别支持Paul Horn的观点,在更加具体的技术背景下进一步阐明了自主计算的概念.由此可知,尽管自主计算的概念尚处于发展过程中,但其基本内涵已较为清晰,即自主计算就是要建立类似于人体自主神经系统主要特点的信息系统,以降低IT管理者的管理负担,提高系统的可靠性、可用性和容错能力.本文第 1.1 节就是从自主计算的基本内涵出发,以上述工作为基础,通过分析、总结自主神经系统和自主计算系统的特点,进一步阐明了自主计算的概念,强调了自主计算的"可指导性"和"整体自适应"特征.

(2) 理论和技术.如果说自主计算的概念是回答"什么是自主计算"的问题,那么,自主计算的理论和技术就是回答"如何实现自主计算"的问题.目前,在该方向上的研究已经取得了重要进展,典型的包括:① Kephart等人的工作:Kephart于 2003 年提出了一种MAPE(监视-分析-规划-执行)控制环作为自主元素的自适应控制机制,并分析了自主计算在体系结构、工程和科学方面应有的考虑[3]——为自主计算的研究初步确立了方向和路线;Kephart于 2004 年提出了由 3 种不同策略(动作策略、目标策略和效用函数策略)组成的统一框架——为解决自主计算的人机接口问题以及实现自主计算系统的"可指导性"奠定了重要的理论基础[13];Kephart于 2005 年从自主元素、自主系统和人与系统的交互这 3 个方面出发,全面分析了实现自主计算所面临的主要挑战[86],并指出:为了实现期望的自我管理特性,需要结合多种科学和技术领域的研究进展(包括系统、软件体系结构和工程、人-系统接口、策略、建模、优化以及人工智能的一些分支,如规划、学习、知识表示和推理、多Agent系统、协商和突现行为等),通过适当的体系结构和方法把它们集成起来——为自主计算的进一步发展指明方

© 中国科学院软件研究所 http://www.jos.org.cn

向;Kephart于 2007 年以效用函数策略作为反映高层目标的手段,建立了一种用于原型数据中心的资源自主管理和分配模型——提供了一个实现自主计算系统的具体范例[17];② Parashar等人的工作:Parashar于 2005 年[87]和 2006 年[88]介绍了一种自主计算项目(AutoMate),它针对网格环境中所存在的复杂性、动态性、异构性和不确定性等挑战,给出了一种自主系统的概念模型和实现体系结构.AutoMate包括 3 个主要部分:Accord编程系统(阐述了自主元素的定义、执行和运行时管理,以及通过动态集成自主元素而形成自主应用的理论)[32]、Rudder分布式协同框架(提供了一个协同框架和基于Agent的推理引擎,用于服务的动态发现和选择,工作流的制定、配置和管理等)[89]和Meteor基于内容的中间件基础设施(用于基于内容的路由、发现和消息传递等).AutoMate针对网格计算这一应用背景,全面研究了自主计算的理论和技术实现,是自主计算研究领域的一个典型代表.目前,该项目正处于发展过程中,仍然存在一些不足.例如,在自主元素的推理和决策方面,尚未深入考虑策略(规则)的冲突以及对动态规则进行推理的非线性特征;在系统的整体自适应方面,尚未全面研究如何通过局部的自配置、自由化、自修复和自保护,实现系统全局的动态平衡;在策略方面,采用单一的动作策略,存在一定的局限性;等等;③ Sterritt等人的工作:Sterritt于 2003 年指出为了实现自主计算,需要把软计算和硬计算相结合,这样既可以利用软计算所具有的处理不精确、不确定和部分真的能力,又可以获得硬计算所具有的高度可预测的解决方法和较低的计算复杂性[81];Sterritt于 2003 年又给出了一种自主系统体系结构,提出了“心跳/脉搏监视(heartbeat/pulse monitor)”和“自主信号通道(autonomic signal channel)”的概念[90];Peña于 2006 年针对自主计算系统中策略的制定和部署问题,提出了一种基于AOSE(service oriented software engineering)(面向Agent的软件工程)的方法,以支持不同抽象层次的策略制定[91];Hinchey和Sterritt于 2006 年在自主计算的基础上提出了自我管理软件的概念,提供了一个软件开发和演化的整体观,有望把系统的自动化、自治和可靠性带到一个新的水平[82];Carsten Franke于 2007 年提出把自主计算、网格计算和虚拟化技术结合起来,建立自主的商务网格的思想[92];等等;④ 其他具有开创新性的工作还包括:De Wolf的“基于自组织突现理论的分散式自主计算”[40,61,64],Tianfield的“基于Agent的自主计算”[19,60],Tesauro的“实现自主计算的多Agent系统方法”[39],White的“实现自主计算的体系结构方法”[84]以及Johnston-Watt的“面向过程的执行管理系统(EMS)”[93]等等.另外在国内,包括吕建[94]、王千祥[10]、曹健[7]、刘涛[8]和廖备水[49,63]等人在内的学者也分别在自主计算的理论和技术方面开展了较为深入的研究.由于篇幅所限,对上述工作的具体讨论从略.同时,对自主计算的具体应用[95−101]的研究也不再一一加以讨论.

依据上述讨论可知,自主计算在理论和技术方面的发展很迅速.近几年来,有关自主计算的文献每年都以几百篇的速度增长.在本文中,我们仅对其中部分重要文献进行了讨论,因此并不全面.不过从整体上看,目前在有关自主计算模型和实现方面的研究还远未成熟,尚未建立起完整的自主计算理论体系和方法体系.鉴于这些考虑,我们尝试在这一方向上做些初步工作,包括:进一步澄清自主计算的概念和定义;提出一个自主元素和自主计算系统概念模型,该模型刻画了自主计算系统的一般工作机制和原理;然后以概念模型为依据,概括性地提出基于知识模型和数学模型的两类自主计算实现方法,并对这两类方法进行了比较和分析,指出各自的优点和不足.

References:

[1] Patterson D, Brown A, Broadwell P, Candea G, Chen M, Cutler J, Enriquez P, Fox A, Kıcıman E, Merzbacher M, Oppenheimer D, Sastry N, Tetzlaff W, Traupman J, Treuhaft N. Recovery oriented computing (ROC): Motivation, definition, techniques, and case studies. Technical Report, UCB//CSD-02-1175, Berkeley, 2002. 1−16.

[2] Horn P. Autonomic computing: IBM’s perspective on the state of information technology. IBM Corporation, 2001. http://www.research.ibm.com/autonomic/manifesto/autonomic_computing.pdf

[3] Kephart JO, Chess DM. The vision of autonomic computing. IEEE Computer, 2003,36(1):41−50.

[4] Tianfield H, Unland R. Towards autonomic computing systems. Engineering Applications of Artificial Intelligence, 2004,17(7):689−699.

[5] Sterritt R. Autonomic computing. Innovations in Systems and Software Engineering, 2005,1(1):79−88.

© 中国科学院软件研究所 http://www.jos.org.cn

[6] Hariri S, Khargharia B, Chen H, Yang J, Zhang Y, Parashar M, Liu H. The autonomic computing paradigm. Cluster Computing, 2006,9(1):5−17.

[7] Cao J, Wang J, Zhang SS, Li ML. A dynamically reconfigurable system based on workflow and service Agents. Engineering Applications of Artificial Intelligence, 2004,17(7):771−782.

[8] Liu T, Zeng GS, Wu CJ. Autonomic computing approach for task distribution in heterogeneous computational grid. Journal of Communications, 2006,27(11):139−147 (in Chinese with English abstract).

[9] Zhang HJ, Shi ZZ. Software engineering for autonomic computing. Journal of Chinese Computer System, 2006,27(6):1077−1082 (in Chinese with English abstract).

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

[11] Bear MF, Connors BW, Michael A. Wang JJ, Trans. Paradiso Neuroscience: Exploring the Brain. Beijing: Higher Education Press, 2004. 472−496 (in Chinese).

[12] Menascé DA, Kephart JO. Guest editors introduction: Autonomic computing. IEEE Internet Computing, 2007,11(1):18−21.

[13] Kephart JO, Walsh WE. An artificial intelligence perspective on autonomic computing policies. In: Verma D, Devarakonda M, Lupu E, Kohli M, eds. Proc. of the 5th IEEE Int'l Workshop on Policies for Distributed Systems and Networks. New York: IEEE Computer Society, 2004. 3−12.

[14] Damianou N, Dulay N, Lupu E, Sloman M. The ponder policy specification language. In: Sloman M, *et al.*, eds. Proc. of the Workshop on Policies for Distributed Systems and Networks. Berlin: Springer-Verlag, 2001. 18−38.

[15] Kagal L, Finin T, Joshi A. A policy language for pervasive computing environment. In: Kawada S, ed. Proc. of the IEEE 4th Int'l Workshop on Policy. Los Alamitos: IEEE Computer Society, 2003. 63−76.

[16] Uszok A, Bradshaw J, Jeffers R, Suri N, Hayes P, Breedy M, Bunch L, Johnson M, Kulkarni S, Lott J. KAoS policy and domain services: Toward a description-logic approach to policy representation, deconfliction, and enforcement. In: Kawada S, ed. Proc. of the IEEE 4th Int'l Workshop on Policy. Los Alamitos: IEEE Computer Society, 2003. 93−98.

[17] Kephart JO, Das R. Achieving self-management via utility functions. IEEE Internet Computing, 2007,11(1):40−47.

[18] Sterritt R, Parashar M, Tianfield H, Unland R. A concise introduction to autonomic computing. Advanced Engineering Informatics, 2005,19(3):181−187.

[19] Tianfield H. Multi-Agent autonomic architecture and its application in E-medicine. In: Liu JM, Faltings B, Zhong N, Lu RQ, Nishida T, eds. Proc. of the IEEE/WIC Int'l Conf. on Intelligent Agent Technolocy (IAT 2003). Los Alamitos: IEEE Computer Society, 2003. 601−604.

[20] Ganek AG, Corbi TA. The dawning of the autonomic computing era. IBM Systems Journal, 2003,42(1):5−18.

[21] An architectural blueprint for autonomic computing. 2006. http://www-03.ibm.com/autonomic/pdfs/AC_Blueprint_White_Paper_4th.pdf

[22] Cebulla M. Knowledge-Based assessment of behavior in dynamic environments. In: Schlenoff C, *et al.*, eds. Proc. of the 2005 ACM Workshop on Research in Knowledge Representation for Autonomous Systems. New York: ACM Press, 2005. 17−26.

[23] Lehtihet E, Strassner J, Agoulmine N, ó Foghlú M. Ontology-Based knowledge representation for self-governing systems. In: State R, *et al.*, eds. Proc. of the DSOM 2006. Berlin: Springer-Verlag, 2006. 74−85.

[24] Stojanovic L, Schneider J, Maedche A, Libischer S, Studer R, Lumpp Th, Abecker A, Breiter G, Dinger J. The role of ontologies in autonomic computing systems. IBM Systems Journal, 2004,43(3):598−616.

[25] Park J, Yoo G, Lee E. Proactive self-healing system based on multi-agent technologies. In: Malloy B, ed. Proc. of the 2005 3rd ACIS Int'l Conf. on Software Engineering Research, Management and Applications. Washington: IEEE Computer Society, 2005. 256−263.

[26] Sterritt R, Bustard D, McCrea A. Autonomic computing correlation for fault management system evolution. In: Unland R, Ulieru M, Weaver AC, eds. Proc. of the IEEE Int'l Conf. Industrial Informatics (INDIN 2003). Los Alamitos: IEEE Computer Society, 2003. 240−247.

[27] Topol B, Ogle D, Pierson D, Thoensen J, Sweitzer J, Chow M, Hoffmann MA, Durham P, Telford R, Sheth S, Studwell T. Automating problem determination: A first step toward self-healing computing systems. 2003. http://www-306.ibm.com/autonomic/pdfs/Problem_Determination_WP_Final_100703.pdf

© 中国科学院软件研究所 http://www.jos.org.cn

[28] Cohen I, Goldszmidt M, Kelly T, Symons J, Chase JS. Correlating instrumentation data to system states: A building block for automated diagnosis and control. In: Brewer E, Chen P, eds. Proc. of the 6th Symp. on Operating Systems Design & Implementation. San Francisco: USENIX, 2004. 231−244.

[29] Chen M, Zheng AX, Lloyd J, Jordan MI, Brewer E. Failure diagnosis using decision trees. In: Kephart J, Parashar M, Das R, Sunderam V, eds. Proc. of the 1st Int'l Conf. on Autonomic Computing (ICAC 2004). Washington: IEEE Computer Society Press, 2004. 36−43.

[30] Chen MY, Kıcıman E, Fratkin E, Fox A, Brewer E. Pinpoint: Problem determination in large, dynamic Internet services. In: Fabre J, Jahanian F, eds. Proc. of the Int'l Conf. on Dependable Systems and Networks. Washington: IEEE Computer Society Press, 2002. 595−604.

[31] Ananthanarayanan R, Mohania M, Gupta A. Management of conflicting obligations in self-protecting policy-based systems. In: Schwan K, Wang Y, eds. Proc. of the ICAC 2005. Washington: IEEE Computer Society Press, 2005. 274−285.

[32] Liu H, Bhat V, Parashar M, Klasky S. An autonomic service architecture for self-managing grid applications. In: Katz DS, ed. Proc. of the 6th IEEE/ACM Int'l Workshop on Grid Computing. Seattle: IEEE Computer Society Press, 2005. 132−139.

[33] Bonino D, Bosca A, Corno F. An agent based autonomic semantic platform. In: Sunderam V, *et al.*, eds. Proc. of the ICAC 2004. Washington: IEEE Computer Society Press, 2004. 189−196.

[34] Wooldridge M, Jennings NR. Intelligent agents: Theory and practice. The Knowledge Engineering Review, 1995,10(2):115−152.

[35] de Albuquerque JP, Krumm H, de Geus PL. Policy modeling and refinement for network security systems. In: Firozabadi B, Winsborough W, Sahai A, eds. Proc. of the 6th IEEE Int'l Workshop on Policies for Distributed Systems and Networks. Washington: IEEE Computer Society, 2005. 24−33.

[36] Bandara AK, Lupu EC, Moffett J, Russo A. A goal-based approach to policy refinement. In: Lupu E, Kohli M, eds. Proc. of the 5th IEEE Workshop on Policies for Distributed Systems and Networks. New York: IEEE Computer Society Press, 2004. 229−239.

[37] Jennings NR. Coordination techniques for distributed artificial intelligence. In: O'Hare GMP, Jennings NR, eds. Foundations of Distributed Artificial Intelligence. New York: Wiley, 1996. 187−210.

[38] Abdelwahed S, Kandasamy N, Neema S. A control-based framework for self-managing distributed computing systems. In: Garlan D, *et al.*, eds. Proc. of the 1st ACM SIGSOFT Workshop on Self-Managed Systems. New York: ACM Press, 2004. 3−7.

[39] Tesauro G, Chess DM, Walsh WE, Das R, Segal A, Whalley I, Kephart JO, White SR. A multi-agent systems approach to autonomic computing. In: Jennings NR, *et al.*, eds. Proc. of the 3rd Int'l Joint Conf. on Autonomous Agents and Multiagent Systems. New York: ACM Press, 2004. 464−471.

[40] De Wolf T, Holvoet T. Design patterns for decentralised coordination in self-organising emergent systems. In: Brueckner S, *et al.*, eds. Proc. of the 4th Int'l Workshop on Engineering Self-Organizing Applications. Berlin: Springer-Verlag, 2007. 28−49.

[41] Anthony R. Emergence: A paradigm for robust and scalable distributed applications. In: Sunderam V, Das R, eds. Proc. of the ICAC 2004. Washington: IEEE Computer Society Press, 2004. 132−139.

[42] Norman TJ, Preece A, Chalmers S, Jennings NR, Luck M, Dang VD, Nguyen TD, Deora V, Shao JH, Gray WA, Fiddian NJ. Agent-Based formation of virtual organisations. Knowledge-Based Systems, 2004,17:103−111.

[43] Jennings NR. On agent-based software engineering. Artificial Intelligence Journal, 2000,117(2):277−296.

[44] Liao BS, Gao J. Dynamic self-organizing system supported by PDC-agent. Journal of Computer-Aided Design & Computer Graphics, 2006,18(2):217−224 (in Chinese with English abstract).

[45] Chess DM, Palmer CC, White SR. Security in an autonomic computing environment. IBM Systems Journal, 2003,42(1):107−118.

[46] Messig M, Goscinski A. Self healing and self configuration in a WSRF grid environment. In: Hobbs M, Goscinski A, Zhou W, eds. Proc. of the 6th Int'l Conf. on Algorithms and Architectures (ICA3PP 2005). Berlin: Springer-Verlag, 2005. 149−158.

[47] Srivastava B, Kambhampati S. The case for automated planning in autonomic computing. In: Schwan K, Wang Y, eds. Proc. of the ICAC 2005. Washington: IEEE Computer Society Press, 2005. 331−332.

[48] Ranganathan A, Campbell RH. Autonomic pervasive computing based on planning. In: Sunderam V, Das R, eds. Proc. of the ICAC 2004. Washington: IEEE Computer Society Press, 2004. 80−87.

[49] Liao BS, Huang HX, Gao J. A defeasible logic-based flexible agent for autonomic computing. Journal of Software, 2008,19(3): 605−620 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/19/605.htm

[50] Shi CY, Zhang W. Agent-Based Computing. Beijing: Tsinghua University Press, 2007. 275−320 (in Chinese).

© 中国科学院软件研究所 http://www.jos.org.cn

[51] Tu XY, Wang C, Guo YH. Large Systems Cybernatics. Beijing: Beijing University of Posts and Telecommunications Press, 2005. 22−25 (in Chinese).

[52] Lü J, Tao XP, Ma XX, Hu H, Xu F, Cao C. A study of agent-based internetware model. Science in China (Series E), 2005,35(12): 1233−1253 (in Chinese with English abstract).

[53] Gao J, Yuan CX, Wang J. SASA5: A method system for supporting agent social activities. Chinese Journal of Computers, 2005, 28(5):838−848 (in Chinese with English abstract).

[54] Curbera F, Duftler M, Khalaf R, Nagy W, Mukhi N, Weerawarana S. Unraveling the Web services Web: An introduction to SOAP, WSDL, and UDDI. IEEE Internet Computing, 2002,6(2):86−93.

[55] Web services distributed management: Management using Web services (MUWS 1.1) Part 1. 2006. http://docs.oasis-open.org/wsdm/wsdm-muws1-1.1-spec-os-01.pdf

[56] Kreger H, Vambenepe W, Sedukhin I, Graham S, Murray B. Management using Web services: A proposed architecture and roadmap. 2005. http://devresource.hp.com/drc/resources/muwsarch/ManagementRoadmap.pdf

[57] Lewis D, O'Donnell T, Feeney K, Brady A, Wade V. Managing user-centric adaptive services for pervasive computing. In: Sunderam V, Das R, eds. Proc. of the ICAC 2004. Washington: IEEE Computer Society Press, 2004. 248−255.

[58] Casati F, Shan E, Dayal U, Shan M. Business-Oriented management of Web services. Communications of the ACM, 2003,46(10): 55−60.

[59] Gurguis SA, Zeid A. Towards autonomic Web services: Achieving self-healing using Web services. In: Garlan D, *et al.*, eds. Proc. of the 2005 Workshop on Design and Evolution of Autonomic Application Software. New York: ACM Press, 2005. 1−5.

[60] Tianfield H. Multi-Agent based autonomic architecture for network management. In: Tianfield H, Unland R, eds. Proc. of the Workshop on Autonomic Computing Principles and Architectures. 2003. 462−469. http://ieeexplore.ieee.org/iel5/9109/28887/01300380.pdf?tp=&arnumber=1300380&isnumber=28887

[61] De Wolf T, Holvoet T. Evaluation and comparison of decentralised autonomic computing systems. Report, CW 437, Leuven: K.U.Leuven, 2006.

[62] Liao BS, Huang HX, Gao J. An extended BDI agent with policies and contracts. In: Shi ZZ, Sadananda R, eds. Proc. of the 9th Pacific Rim Int'l Workshop on Multi-Agents. Berlin: Springer-Verlag, 2006. 94−104.

[63] Liao BS. Service-Oriented autonomic computing based on PDC-agent [Ph.D. Thesis]. Hangzhou: Zhejiang University, 2006 (in Chinese with English abstract).

[64] De Wolf T, Holvoet T. A taxonomy for self-* properties in decentralised autonomic computing. In: Parashar M, Hariri S. eds. Autonomic Computing: Concepts, Infrastructure, and Applications. Boca Raton: CRC Press, 2007. 101−120.

[65] Xu XY. Adaptive Control Theory and Applications. Beijing: Publishing House of Electronics Industry, 2007 (in Chinese).

[66] Abdelzaher TF, Stankovic JA, Lu C, Zhang R, Lu Y. Feedback performance control in software services. IEEE Control Systems Magazine, 2003,23(3):74−90.

[67] Diao Y, Hellerstein JL, Parekh S, Griffith R, Kaiser G, Phung D. Self-Managing systems: A control theory foundation. In: Rozenblit J, *et al.*, eds. Proc. of the 12th IEEE Int'l Conf. and Workshops on the Engineering of Computer-Based Systems. Washington: IEEE Computer Society, 2005. 441−448.

[68] Wang M, Kandasamy N, Guez A, Kam M. Distributed cooperative control for adaptive performance management. IEEE Internet Computing, 2007,11(1):31−39.

[69] Litoiu M, Woodside M, Zheng T. Hierarchical model-based autonomic control of software systems. In: Garlan D, *et al.*, eds. Proc. of the Design and Evolution of Autonomic Software. New York: ACM Press, 2005. 1−7.

[70] Kandasamy N, Abdelwahed S, Khandekar M. A hierarchical optimization framework for autonomic performance management of distributed computing systems. In: Ahamad M, *et al.*, eds. Proc. of the 26th IEEE Int'l Conf. on Distributed Computing Systems. Washington: IEEE Computer Society, 2006. 9−18.

[71] Kumar V, Cooper BF, Schwan K. Distributed stream management using utility-driven self-adaptive middleware. In: Schwan K, Wang Y, eds. Proc. of the ICAC 2005. IEEE Computer Society Press, 2005. 3−14.

[72] Das R, Whalley I, Kephart JO. Utility-Based collaboration among autonomous agents for resource allocation in data centers. In: Weiss G, *et al.*, eds. Proc. of the 5th Int'l Joint Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS). New York: ACM Press, 2006. 1572−1579.

© 中国科学院软件研究所 http://www.jos.org.cn

[73] Walsh WE, Tesauro G, Kephart JO, Das R. Utility functions in autonomic systems. In: Sunderam V, Das R, eds. Proc. of the ICAC 2004. Washington: IEEE Computer Society Press, 2004. 70−77.

[74] Bandara AK. A formal approach to analysis and refinement of policies [Ph.D. Thesis]. London: Imperial College London, 2005.

[75] Darimont R, van Lamsweerde A. Formal refinement patterns for goal-driven requirements elaboration. In: Garlan D, Morieoni M, eds. Proc. of the 4th ACM Symp. on the Foundations of Software Engineering (FSE4). New York: ACM Press, 1996. 179−190.

[76] Davy S, Jennings B, Strassner J. Conflict prevention via model-driven policy refinement. In: State R, *et al.*, eds. Proc. of the 17th IFIP/IEEE Int'l Workshop on Distributed Systems: Operations and Management. Berlin: Springer-Verlag, 2006. 209−220.

[77] Anthony RJ. A policy-definition language and prototype implementation library for policy-based autonomic systems. In: Rana O, Yousif M, *et al.*, eds. Proc. of the ICAC 2006. New York: IEEE Computer Society Press, 2006. 265−276.

[78] Beckett D, McBride B. RDF/XML syntax specification (Revised). World Wide Web Consortium, 2004. http://www.w3.org/TR/rdf-syntax-grammar/

[79] Horrocks I, Patel-Schneider PF, van Harmelen F. Reviewing the design of DAML+OIL: An ontology language for the semantic Web. In: Proc. of the 18th National Conf. on Artificial Intelligence (AAAI 2002). AAAI Press, 2002. 1−2. http://www.cs.man.ac.uk/~horrocks/Publications/download/2002/AAAI02IHorrocks.pdf

[80] Bechhofer S, van Harmelen F, Hendler J, Horrocks I, McGuinness DL, Patel-Schneider PF, Stein LA. OWL Web ontology language reference. World Wide Web Consortium, 2004. http://www.w3. org/TR/owl-ref/

[81] Sterritt R. Autonomic computing: The natural fusion of soft computing and hard computing. In: Proc. of the 2003 IEEE Int'l Conf. on Systems, Man & Cybernetics. Washington: IEEE Computer Society Press, 2003. 4754−4759. http://ieeexplore.ieee.org/iel5/8811/27913/01245735.pdf

[82] Hinchey MG, Sterritt R. Self-Managing software. IEEE Computer, 2006,39(2):107−109.

[83] Ganek AG, Corbi TA. The dawning of the autonomic computing era. IBM System Journal, 2003,42(1):5−18.

[84] White SR, Hanson JE, Whalley I, Chess DM, Kephart JO. An architectural approach to autonomic computing. In: Sunderam V, Das R, eds. Proc. of the ICAC 2004. Washington: IEEE Computer Society Press, 2004. 2−9.

[85] Parashar M, Hariri S. Autonomic computing: An overview. In: Banâtre JP, *et al.*, eds. Proc. of the Int'l Workshop on Unconventional Programming Paradigms (UPP 2004). Berlin: Springer-Verlag, 2005. 247−259.

[86] Kephart JO. Research challenges of autonomic computing. In: Griswold W, *et al.*, eds. Proc. of the 27th Int'l Conf. on Software Engineering. New York: ACM Press, 2005. 15−22.

[87] Parashar M, Li Z, Liu H, Matossian V, Schmidt C. Enabling autonomic grid applications: Requirements, models and infrastructures. In: Babaoglu O, *et al.*, eds. Proc. of the Int'l Workshop on Self-* Properties in Complex Information Systems. Berlin: Springer-Verlag, 2005. 273−290.

[88] Parashar M, Liu H, Li Z, Matossian V, Schmidt C, Zhang G, Hariri S. AutoMate: Enabling autonomic applications on the Grid. Cluster Comput, 2006,9(2):161−174.

[89] Li Z, Parashar M. An infrastructure for the dynamic composition of grid service. Technical Report, Rutgers University, 2007. http://www.caip.rutgers.edu/TASSL/Papers/rudder-grid-06.pdf

[90] Sterritt R, Bustard DW. Towards an autonomic computing environment. In: Ibrahim MT, *et al.*, eds. Proc. of the 1st Int'l Workshop on Autonomic Computing Systems at DEXA 2003. Washington: IEEE Computer Society Press, 2003. 699−703.

[91] Peña J, Hinchey MG, Sterritt R. Towards modeling, specifying and deploying policies in autonomous and autonomic systems using an AOSE methodology. In: Sterritt R, *et al.*, eds. Proc. of the 3rd IEEE Int'l Workshop on Engineering of Autonomic & Autonomous Systems (EASE 2006). Washington: IEEE Computer Society, 2006. 37−46.

[92] Franke C, Theilmann W, Zhang Y, Sterritt R. Towards the autonomic business grid. In: Sterritt R, *et al.*, eds. Proc. of the 4th IEEE Int'l Workshop on Engineering of Autonomic and Autonomous Systems (EASe 2007). Washington: IEEE Computer Society, 2007. 107−112.

[93] Johnston-Watt D. Under new management. ACM Queue, 2006,4(2):50−58.

[94] Lü J, Ma XX, Tao XP, Xu F, Hu H. Research and development of internetware. Science in China (Series E), 2006,36(10): 1037−1080 (in Chinese with English abstract).

[95] Greenwood D, Rimassa G. Autonomic goal-oriented business process management. In: de Souza JN, *et al.*, eds. Proc. of the 3th Int'l Conf. on Autonomic and Autonomous Systems. Washington: IEEE Computer Society, 2007. 43−48.

© 中国科学院软件研究所 http://www.jos.org.cn

[96] Rana OF, Kephart JO. Building effective multivendor autonomic computing systems. IEEE Distributed Systems Online, 2006,7(9): 3−7.

[97] Denaro G, Pezzè M, Tosi D. Designing self-adaptive service-oriented applications. In: Fortes J, *et al*, eds. Proc. of the ICAC 2007. Washington: IEEE Computer Society, 2007. 16−35.

[98] Hall D. Automatic parameter regulation of perceptual systems. Image and Vision Computing, 2006,24(8):870−881.

[99] Pautasso C, Heinis T, Alonso G. Autonomic resource provisioning for software business processes. Information and Software Technology, 2007,49(1):65−80.

[100] Nichols J, Demirkan H, Goul M. Autonomic workflow execution in the grid. IEEE Trans. on Systems, Man, and Cybernetics—Part C: Applications and Reviews, 2006,36(3):353−363.

[101] Arora H, Mishra BK, Raghu TS. Autonomic-Computing approach to secure knowledge management: A game-theoretic analysis. IEEE Trans. on Systems, Man, and Cybernetics—Part A: Systems and Humans, 2006,36(3):487−497.

附中文参考文献:

[8] 刘涛,曾国荪,昊长俊.异构网格环境下任务分配的自主计算方法.通信学报,2006,27(11):139−147.

[9] 张海俊,史忠植.自主计算软件工程方法.小型微型计算机系统,2006,27(6):1077−1082.

[10] 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.

[11] Bear MF,Connors BW,Paradiso MA,著;王建军,译.神经科学——探索脑.北京:高等教育出版社,2004.472−496.

[44] 廖备水,高济.PDC-Agent 支持的动态自组织系统.计算机辅助设计与图形学学报,2006,18(2):217−224.

[49] 廖备水,黄华新,高济.一种支持自治计算的基于可废止逻辑的柔性 Agent.软件学报,2008,19(3):605−620. http://www.jos.org.cn/1000-9825/19/605.htm

[50] 石纯一,张伟.基于 Agent 的计算.北京:清华大学出版社,2007.275−320.

[51] 涂序彦,王枞,郭燕慧.大系统控制论.北京:北京邮电大学出版社,2005.22−25.

[52] 吕建,陶先平,马晓星,胡昊,徐锋,曹春.基于 Agent 的网构软件模型研究.中国科学(E 辑),2005,35(12):1233−1253.

[53] 高济,袁成祥,王进.支持 Agent 社交活动的方法体系 SASA5.计算机学报,2005,28(5):838−848.

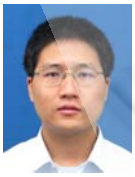
[63] 廖备水.基于 PDC-Agent 的面向服务的自治计算研究[博士学位论文].杭州:浙江大学,2006.

[65] 徐湘元.自适应控制理论与应用.北京:电子工业出版社,2007.

[94] 吕建,马晓星,陶先平,徐锋,胡昊.网构软件的研究与进展.中国科学(E 辑),2006,36(10):1037−1080.

廖备水(1971－),男,福建古田人,博士,主要研究领域为人工智能,Agent 与多 Agent 系统,自主计算,逻辑学.

姚远(1978－),男,博士,主要研究领域为面向服务的分布式制造系统理论与构架.



李石坚(1979－),男,博士,CCF 会员,主要研究领域为传感器网络,普适计算,人工智能.

高济(1946－),男,教授,博士生导师,主要研究领域为人工智能,网格计算,软件工程,信息智能.

© 中国科学院软件研究所 http://www.jos.org.cn