软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2013,24(9):1981−2000 [doi: 10.3724/SP.J.1001.2013.04432]

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

构造具备自适应能力的软件*

丁 博, 王怀民, 史殿习

(国防科学技术大学 计算机学院 并行与分布处理国家重点实验室,湖南 长沙 410073)

通讯作者: 丁博, E-mail: dingbo@nudt.edu.cn

摘 要: 随着应用场景的变迁和自身复杂性的增长,软件需要具备主动适应环境变化的能力,能够依据环境变化动态调整其行为.软件自适应的实现技术跨越了软件监控、上下文敏感计算、决策和控制理论、软件演化和维护等多个学科分支,如何系统化地构造此类软件,是软件工程领域所面临的巨大挑战.从自适应软件构造与实现这一角度出发,以"感知-决策-执行"软件自适应基本周期为主线,对已有的研究和实践进行综述.给出软件自适应的概念内涵,概述软件自适应活动在感知、决策、执行各环节上的特征分类,阐述面向自适应软件构造、关注程度较高的一系列使能技术,进而在分析典型研究项目现状的基础上,给出自适应软件构造领域的未来主要研究趋势.

关键词: 软件自适应;上下文感知;中间件;软件体系结构;群体适应

中图法分类号: TP311 **文献标识码**: A

中文引用格式: 丁博,王怀民,史殿习.构造具备自适应能力的软件.软件学报,2013,24(9):1981−2000. http://www.jos.org.cn/1000-9825/4432.htm

英文引用格式: Ding B, Wang HM, Shi DX. Constructing software with self-adaptability. Ruan Jian Xue Bao/Journal of Software, 2013,24(9):1981−2000 (in Chinese). http://www.jos.org.cn/1000-9825/4432.htm

Constructing Software with Self-Adaptability

DING Bo, WANG Huai-Min, SHI Dian-Xi

(National Key Laboratory for Parallel and Distributed Processing, College of Computer, National University of Defense Technology, Changsha 410073, China)

Corresponding author: DING Bo, E-mail: dingbo@nudt.edu.cn

Abstract: Along with the transition of application scenario and the increase of its own complexity, software needs the capability of actively adapting itself to the changes in its environment, which means that it needs to adjust its behavior in response to those changes. The realization of software self-adaptation is closely related to many academic fields such as software monitoring, context-aware computing, decision and control theory, software evolution and maintenance, and etc. It is a great challenge to construct this kind of software systematically in software engineering practice. From the perspective of the construction and implementation of self-adaptive software, this paper reviews the existing research and practices based on the "sensing-decision-execution" basic self-adaptation cycle. This study presents the concept of software self-adaptation and a taxonomy of the characteristics of sensing, decision and execution activities, summary those most focused enabling techniques for constructing adaptive software, analyze a set of typical research projects, and then describe the future research trend in the field of constructing adaptive software.

Key words: software self-adaptation; context-aware; middleware; software architecture; collective adaptation

在生物学领域,当环境(外部客观条件)改变时,机体的细胞、组织或器官通过自身代谢实现功能和结构的相应改变,这个过程被称为适应.20 世纪 90 年代初,研究者将"适应"一词引入软件工程领域,提出了软件自适应的概念,代表了人们对软件能力的一种理想期望——软件应当能够主动应对各种变化,在动态环境下持续提供符

* 基金项目: 国家自然科学基金(61202117, 91118008); 国家重点基础研究发展计划(973)(2011CB302600)
收稿时间: 2012-09-27; 定稿时间: 2013-05-28

合用户预期的服务.

软件自适应的概念提出以来,一直是学术界和工业界关注的热点.例如:美国国防部高级研究计划署(DARPA)在20世纪90年代资助了一系列与"自适应软件"[1]相关的项目;2006年,卡内基梅隆大学软件工程研究所在美国军方资助下发表了题为《超大规模系统:软件未来的挑战》的报告,指出持续的适应是军事领域大规模软件系统夺取制信息权的关键[2];欧盟FP7框架所资助的PerAda项目(2008~2011)在可演化和自适应的普适系统、自适应的安全和可靠性等方面开展了一系列研究[3].

软件自适应之所以成为研究热点,主要由如下两个因素所直接驱动:一方面,软件与现实世界的结合越来越紧密.例如:美军F-22飞机上有80%的设备由软件控制[4];近3年内,基于手机等新型终端的移动云计算市场预计将年均增长88%,2014年全球移动云计算市场将达到95亿美元[5].应用场景的变化使得软件不再局限于封闭静态的企业和桌面环境,动态、开放的现实世界已成为其运行环境的直接组成部分.软件只有具备了适应能力,才有可能持续提供符合用户预期的服务.另一方面,目前许多大型分布式应用的代码行数已经达到了千万甚至更高量级[6],软件自身变得越来越复杂,并呈现去中心化、故障常态化等特点,为软件管理、维护和可靠运行等带来了新挑战.软件迫切需要具备自适应能力,能够及早发现问题的诱因,并在出现问题时进行自动修复.

软件工程研究者已经在自适应领域开展了大量工作.例如:早期计算反射(computing reflection)研究的关注点即为如何在运行时感知软件状态及调整其行为[7,8];IBM所提出的自主计算概念强调软件的自配置、自优化、自修复和自保护等;Rainbow[9]、K-Component[10]、MADAM[11]、网构软件技术体系[12]等强调基于体系结构技术实现软件自适应;Hadas[13],Accord[14],PCOM[15],ACEEL[16],CompAA[17]等从构件模型角度研究如何构造自适应软件;OpenCom/OpenORB[18],Gaia[19],CASA[20]及早期的Quo[21],DynamicTAO[22]等则从中间件/软件框架角度支持或实现动态适应.此外,具备自管理能力的体系结构描述语言和框架[23]、软件自愈[24,25]、分布环境下负载均衡和容错等研究均可划归软件适应的范畴.

然而,软件自适应是一个外延十分广泛的概念,其实现技术涉及到软件工程、人工智能、复杂系统等多个学科,无论对于研究者还是实践者都是巨大的挑战.具体到软件工程领域,这种挑战主要表现为缺乏系统化的理论、方法、工具和运行支撑设施,人们距离系统化构造自适应软件这一目标尚有较远距离.本文以"感知-决策-执行"软件自适应基本周期为主线,对自适应软件构造的已有研究和实践进行综述,在分析现状基础上,指出未来研究趋势.

本文第1节给出软件自适应的概念内涵.第2节对软件自适应活动在感知、决策、执行等环节所表现出的特征进行分类.第3节概述上述各个环节的关键使能技术.第4节对若干具有代表性、以构造自适应软件为目标的典型项目进行分析.第5节给出本领域的未来研究趋势.第6节对全文进行总结.

1 概念内涵

考虑如下一些典型场景:

场景1(多链路通信中间件). 在服务器上配备了互为备份的多块网卡(多个物理链路).当正在使用的链路发生故障时,服务器上的中间件可以自动切换链路,使得上层网络应用能够以7×24小时提供不间断服务.

场景2(自适应服务器池). 某个云服务后端采用大量虚拟机组成服务器池来处理用户请求.为了使得运行费用最小化,系统能够依据负载来动态调整服务器池中的虚拟机数目.

场景3(任务的动态Offloading[26]). 在移动云计算环境下,移动终端(如智能手机)通过无线网络连接到云计算基础设施,计算服务由云计算基础设施和终端协同提供.在完成图像识别等复杂任务时,任务执行模块可以依据当前网络带宽、终端剩余电量等环境状态,在移动终端和云端之间迁移,从而最大限度地降低开销、提高用户体验.

直观上,我们称上述场景中的软件具备自适应能力.进一步分析则可以发现,这些场景的共同点在于:当外部运行环境发生变化时,软件能够通过内部的主动调整来应对变化.因此,综合"适应"一词的本意和实际需求,本文将软件自适应定义如下:

定义 1(软件自适应). 软件自适应是指为了保证持续、高质量地提供服务,软件在运行时检测环境变化和自身状态,据此对自身行为进行主动调整的活动.

能够实施自适应活动的软件被称为自适应软件.传统软件与自适应软件的区别在于:前者仅仅具有明确定义的输入和输出,而后者还可以根据输入之外的环境变化和自身状态信息来调整其行为(如图 1 所示).

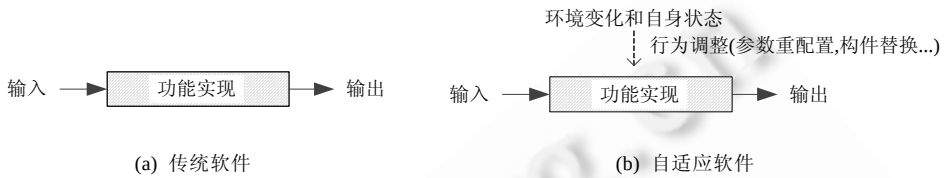

Fig.1 Differences between traditional software and self-adaptive software

图 1 传统软件与自适应软件的对比

从定义 1 可以看出,软件自适应包含如下 3 个阶段:

(1) 感知,即软件感知到环境变化和自身状态;

(2) 决策,即软件依据所感知到的内容做出行为调整决策;

(3) 执行,即软件实施在线调整动作.

例如,在场景 1 中:软件首先需要感知到外部通信链路可用性的变化;然后,根据环境和当前自身状态进行决策,决定是否切换以及切换到哪一条通信链路;最后,执行链路切换动作.这样的“感知-决策-执行”过程在本文中被称为软件自适应基本周期,它将是贯穿本文后续内容的主线.

2 特征分类

综合本领域已有研究和工程实践,本节将对软件自适应活动在感知、决策、执行各个环节所表现出的基本特征进行分类(如图 2 所示).

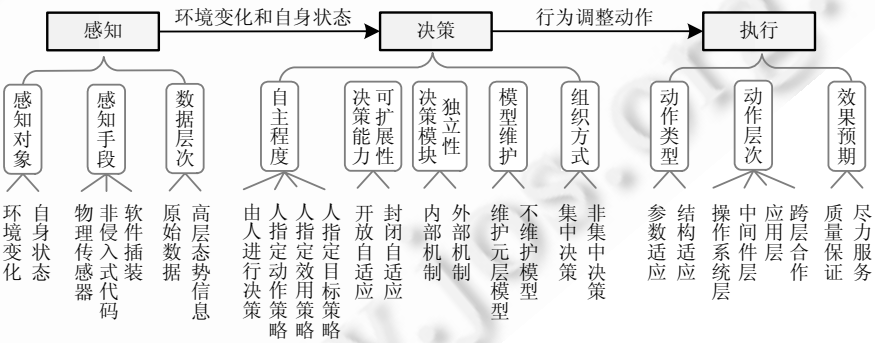

Fig.2 Basic software adaptation loop and its taxonomy

图 2 软件自适应基本周期及其特征分类

2.1 感知环节的特征分类

在感知环节,可以从感知对象、感知手段、数据层次等角度对软件自适应活动进行分类.

在感知对象方面,早期软件被假定在封闭、预设的环境中运行,因此并不强调对环境的直接感知,环境的变化主要通过作用于软件内部状态的方式间接表现出来.例如,DARPA 于 1997 年将自适应软件定义为可以评估其自身行为,并在评估结果指明其无法完成指定任务,或有可能实现更佳功能或性能时,改变自身的行为[1],这个定义中并未出现任何与环境相关的内容.然而,随着 Cyber-Physical 系统[27]、软件密集型系统[28]、网构软件[12]等新型计算范型的出现,软件运行环境可能发生频繁的动态变化,因此,近期研究更多强调环境信息的显式化,

实现对外部环境的直接感知并驱动适应动作[29].但是,软件对自身状态的感知仍是必须的,因为对自身状态的理解是实现行为调整的前提和基础(见第 3.1 节计算反射),在场景 1 中,即表现为软件需要知道自身正在使用的通信链路.

在感知手段方面,软件可以通过如下一些手段来感知环境变化和自身状态:

(1) 直接访问物理传感器;

(2) 监视计算空间状态的非侵入式代码,例如,通过硬件、操作系统、虚拟机、中间件或应用 API 获取数据[30];

(3) 通过硬编码、面向方面编程等各种形式植入到软件中,探测软件内部状态的程序插装[31]代码.

例如,在场景 1 中,可使用第 2 种手段来感知底层通信链路的可用性,使用第 3 种手段来确定中间件当前使用的通信链路.

在数据层次方面,感知环节得到的数据可能是原始的传感器或探针信息,也可能是经过去伪存真、去粗取精、从大量原始信息中汇聚得到的高层态势信息.后者往往才是后续决策环节真正需要的数据.从原始信息获得高层信息的过程,往往涉及到规则推理、机器学习、数据挖掘等技术.

2.2 决策环节的特征分类

在决策环节,可以从自主程度、决策模块可扩展性、决策能力独立性、模型维护、组织方式等角度对软件自适应活动进行分类.

在自主程度方面,需要回答的问题是软件适应过程具有何种程度的自主性.文献[32]提出了软件自主能力成熟度的 5 个分级,并且认为完全自主才是理想的,但在当前的技术条件下,这一目标显然不现实.只有在人或多或少干预软件适应过程的前提下,才能构造出具有实用价值的自适应软件[33,34].本文依据人参与的程度,将软件决策能力由低到高分为:

(1) 由人做出决策,此类软件通常被称为可适应软件(adaptable software)而非自适应软件(adaptive software)[35];

(2) 人预先指定动作策略,即明确的、形如“何时干什么”的策略,软件在运行时根据策略进行决策,最常见的策略形式是 If-Then 和 ECA(event-condition-action)[36]形式;

(3) 人预先指定效用(utility)策略,即指定各种场景下的收益,软件据此进行实时规划,进行决策;

(4) 人在某个层面上指定目标(goal)策略,如何达到这一目标,由软件基于其知识和内建的学习、规划等算法来决定.

第(2)种和第(3)种是当前软件自适应研究和实践中采用较多的方法.

在决策能力的可扩展性方面,文献[37]中提出了开放/封闭适应性的概念:前者是指软件在运行时可以引入新的自适应动作,从而应对开发阶段未预期的变化;后者是指软件的自适应能力在开发阶段即已确定.在开放适应性中,自适应能力(可细分为感知、决策、执行等能力)既可以由第三方来动态扩展[35],也可以是软件通过某些学习算法动态获得的[38].例如,场景 1 仅仅预期了链路故障的情况,而实际运行时,若能通过动态加载新的策略来考虑线路带宽差异,则自适应通信中间件的自适应决策能力是可扩展的.

在决策模块独立性方面,根据决策模块与软件功能模块分离的程度,文献[39]将自适应分为内部机制和外部机制:前者是指决策逻辑和软件业务逻辑混杂在一起,通常使用条件语句、异常等程序语言层面的机制来实现决策过程;后者是指将决策模块与软件功能模块分离,决策模块控制软件业务模块的行为,驱动适应动作.外部机制的优点包括:(1) 可以实现关注点分离,从而有利于软件维护、决策能力的扩展等;(2) 决策模块可以被重用,从而降低自适应软件的开发难度;(3) 有可能在决策模块掌握环境和软件状态的全局信息,从而改善决策的质量.

在模型维护方面,决策过程可能是 Ad Hoc 的,也可能基于所显式维护的模型.模型是软件对环境和(或)自身状态的理解,它的存在有助于使相关的知识显式化,从而提高自适应决策的质量.典型的模型包括实现自优化的队列模型[40]、对软件进行高层抽象的体系结构模型[37]、领域相关模型[41]等.

在组织方式方面,除了常见的集中决策方式外,分布式系统的许多场景下可能需要引入非集中式方式,通过软件实体之间的对等协商来实现群体自适应决策.典型适用场景包括:由于物理条件的限制,集中式协同无法实现(例如在某些无线传感器网络中);成员属于不同的管理域,无法集中控制;需要非集中决策的其他优点,如避免单点失效、提高性能等.

2.3 执行环节的特征分类

在执行环节,可以从动作类型、动作层次、效果预期等角度对软件自适应活动进行分类.

在动作类型方面,要对软件行为进行调整,所执行的自适应动作可以是参数适应或是结构适应[42]:前者是指不对软件实现进行修改的情况下改变软件行为,例如重新设定某些变量、修改软件的某些可配置属性等;后者则是指通过修改软件的实现来改变软件行为,如替换算法和模块等.参数适应最简单的例子是 TCP 协议中的拥塞控制[43]——通过调整窗口大小来改变协议栈行为,从而适应网络流量的变化.结构适应可以通过动态 AOP (aspect-oriented programming)、构件和服务动态组合等方式来实现,它往往能够带来更大的灵活性,但也会带来更大的风险,例如可能破坏软件体系结构的完整性和一致性等.

在动作层次方面,自适应动作可以发生在操作系统、中间件、应用本身等各个层次上.其中,内建于操作系统和中间件层的适应动作一般是共性的,甚至对上层应用是透明的,场景 1 的中间件链路切换就是典型实例.适应动作也可跨越多个层次.仍以场景 1 为例,若上层是对数据实时性要求较高的应用(如基于流媒体的视频),则在链路切换过程可能无法达到完全的透明,需要上层应用也进行适当处理(如暂停播放或降低视频质量).

在效果预期方面,对软件适应效果的预期可以是尽力服务(best-effort),也可以是不允许失败(mission-critical)的:前者是指软件在环境发生变化时尽量提供满足用户需求的服务;后者则是指软件自适应必须达到某个目标,通常会伴随着对动作结果正确性、有效性和开销等的评估和校验机制[38].在场景 1 中,如果中间件上层支撑的是关键应用,则要求其适应动作不允许失败,即使由于客观原因无法保证链路可用性也应当及时报警.

3 使能技术

本节将以感知-决策-执行基本周期为主线,对自适应软件构造与实现过程中当前关注度较高的一些使能技术进行概述(如图 3 所示).

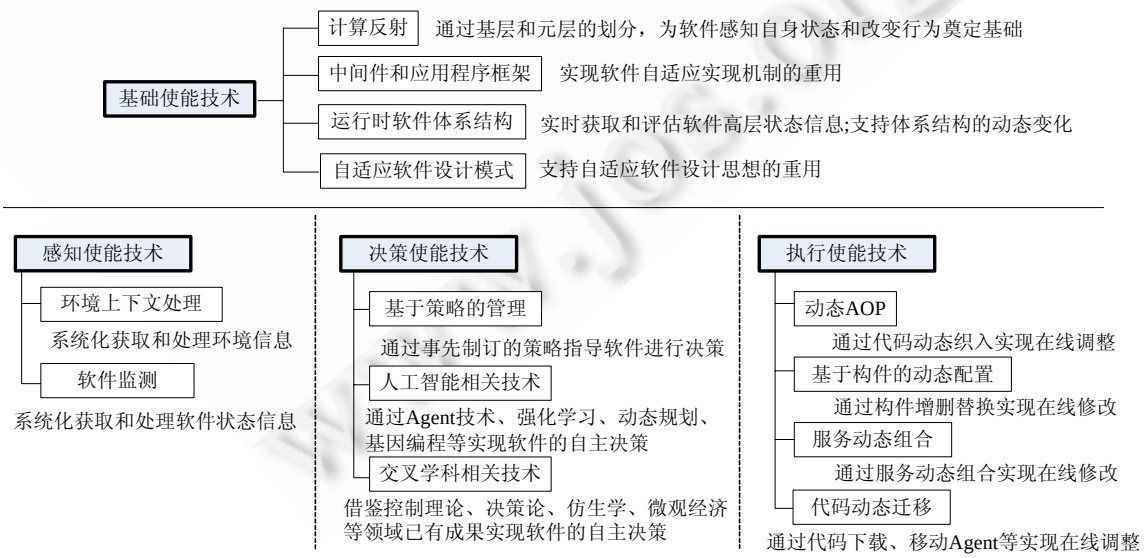

Fig.3 Software adaptation enabling techniques

图 3 软件适应使能技术

3.1 基础使能技术

基础使能技术是指跨越了感知-决策-执行各个阶段,为自适应软件构造与实现奠定基础的技术,包括反射、中间件和应用程序框架、运行时软件体系结构、自适应软件设计模式等.

(1) 计算反射

计算反射最早由 Smith 在 1982 年作为一种程序设计范型提出[7].他指出:既然软件可建立外部世界的机器描述,并借助这一描述来操纵外部世界,那么它也可建立对自身的描述,并借助这一描述来操纵自身.Maes 在此基础上系统地阐述了反射系统和反射的概念[8]:反射系统是指内建了自描述数据结构的系统,这一数据结构与其描述对象因果关联(causally connected);反射是指反射系统基于这一自描述所实施的访问、推理和修改自身的行为.反射在软件自适应的使能技术中具有基础性地位,文献[44]指出:"自适应软件本质上是一个通过对自身进行反射来改变其行为的系统".

(2) 中间件和应用程序框架

中间件是位于平台(硬件和操作系统)和应用之间的可重用软件服务,其基本思想是:抽取软件构造中的共性问题(如通信、异构、并发、事务等),封装其解决机制,对外提供简单统一的接口,从而减少开发人员在解决这些共性问题时的难度和工作量.虽然许多场合我们并不严格区分中间件和应用程序框架,但相对于中间件所提供的公共服务而言,应用程序框架与应用结合更紧密,它是应用的部分实现,提供了大量实现通用功能的公共代码,这些代码可以被有选择性地使用、复写或是通过用户代码来扩展,从而支持开发者来构造面向特定领域的应用程序[45].具备自适应能力的软件比传统软件更为复杂,而现有操作系统和语言尚未为此类软件的构造提供有效支撑.因此,Rainbow[9,39],K-Component[38],MADAM[11]等大多数软件自适应项目都提供了相应的中间件或应用程序框架——通过将感知-决策-执行各环节可重用部件沉淀到中间件或框架中,可以简化上层自适应软件的开发难度.

(3) 运行时软件体系结构

IEEE 将体系结构(architecture)定义为"一个系统的基本组织方式,包括其构件、构件间联系、构件与环境的联系以及指导其设计和演化的原则"[28].简单地说,软件体系结构将系统抽象为由并发构件(计算实体)和连接子(计算实体间的交互)组成的网络[46],从而为架构师提供一个系统级、不涉及软件代码细节的高层抽象.这一抽象可以刻画软件的基本架构,支持系统性质的评估、分析和验证.

早期的软件体系结构仅仅是开发阶段的静态实体[47],其主要研究内容包括体系结构描述语言[48]、体系结构风格[46]等.随着应用模式的变更,软件的需求、环境等都可能在运行时发生变化,体系结构层面上的动态变化不可避免,运行时软件体系结构的概念被引入[23].它至少可以在感知和执行两个阶段为软件自适应提供支持:在感知阶段获取体系结构信息;在执行阶段使得体系结构可动态调整.文献[49]认为,基于体系结构实现软件自管理具有如下一些优点:通用性、适当的抽象层次、潜在的可扩展性、可以利用体系结构领域已存在大量工作、有可能产生集大成的方法等.

(4) 自适应软件设计模式

设计模式是对软件设计过程中普遍存在的问题所提出的解决方案[50],是重用软件设计思想的一种有效途径.针对自适应软件的构造与实现:Ramirez 等人总结了包括传感器工厂(sensor factory)、基于用例的推理(case-based reasoning)、服务器重配置(server reconfiguration)等在内的 12 种软件设计模式[51];Schmidt 等人总结出一系列可用于自适应中间件的设计模式,并在中间件 TAO 和 ZEN 等中得到应用[52];Shackleton 针对普适计算环境,给出了包括本地规则、正反馈和负反馈等自主计算的 6 种设计启发式,并认为有必要将之形式化为设计模式[53];Gomma 等人提出了运行时对软件进行重配置的 Master/Slave、Client/Server、非集中控制等模式[54].

3.2 感知使能技术

如第 2.1 节所述,感知环节的目标是获得环境信息和软件自身状态信息.上下文感知计算(context-aware computing)[55]中的环境上下文处理技术对前者已经有了较为系统的研究,后者则是软件监测(software

monitoring)技术的关注点.

(1) 环境上下文处理

上下文(context)又被译为情境,是与软件输入/输出相对的一个概念.研究者从不同角度对其作了诠释:Schilit 等人早期将上下文定义为位置、周围人和物的标识集合、这些物体的变化情况等[55];Brown 等人将上下文定义为计算机所了解的用户环境中的元素[56];徐光祐等人将上下文信息的内涵定义为计算的上下文、用户的上下文、物理的上下文和上下文的历史[57];等等.一般而言,在软件自适应领域,上下文主要指可能影响应用行为、需要应用进行适应的外部环境和场景集合.

目前,上下文处理技术的研究主要围绕如下一些方面展开:

- 上下文建模,即如何表达上下文信息及其逻辑关系,例如键值对模型、标记模型、图形模型、对象模型、基于逻辑的模型、基于本体的模型等[58];
- 上下文获取,即如何在运行时获取环境上下文以供应用访问.一种较为常见的机制是:将传感器以插件等形式包装为具有统一接口的软件实体[59−61],从而使上层软件在设计时无需与具体硬件绑定.应用访问上下文的方式则可以分为传统 API 风格、数据库风格、回调/事件风格、文件接口风格等[62];
- 上下文聚合,即对原始的、未经处理的底层上下文信息进行汇集、筛选,判断和推导出应用所需要的高层上下文信息.在现有实践中,基于规则和内置知识库的聚合方法具备可定制、行为可预期性好等优点,被许多与上下文处理相关项目采用,例如 CoBrA[60],SOCAM[61],Gaia[19]等.

(2) 软件监测

软件监测技术的研究始于 20 世纪 60 年代,早期主要用于程序调试,支持程序员实时查看程序的内部状态[63,64].目前,软件监测技术已经广泛应用于软件调优、质量评估、实时容错、软件维护等领域[65].分布式软件系统是当前软件监测领域关注的重点,其所具备的缺乏集中控制、规模增长、传输延迟、异构性等特点,为软件监测技术带来了一系列挑战[66,67].具体而言,相关研究表现出如下一些趋势:

- 监测粒度从代码级扩展到构件、服务和体系结构层次.例如:REQMON 通过需求分析和运行时监测技术结合实现对 Web 服务行为的监控,保证电子商务的可用性[68];Garlan 等人使用探针(probe)和量规(gauge)实现对软件体系结构的监测,进而实现软件的自适应[69];
- 监测范围从单一层次向多个层次扩展,通过对多个层次数据进行监测和综合,全面刻画复杂分布式系统的运行状态.典型实例是德国奥尔登堡大学的 TruSoft 项目,其监测系统的设计跨越了硬件、操作系统、Java 虚拟机、中间件和应用多个层次[70];
- 监测实现从单一进程向多进程乃至多处理机形式过渡.例如,佐治亚理工大学提出的软件断层(software tomography)技术[71]将监测任务分解,把子任务分配给同一监测目标的多个实例,从而减少监测过程给单个实例所带来的性能损失.

3.3 决策使能技术

软件可以依据预先指定的策略实现决策,也可以结合人工智能技术、交叉学科相关技术等应对各种复杂情况,减少人的干预.

(1) 基于策略的管理

策略是指导软件行为的元层信息[72].对策略(policy)的研究是由软件日益复杂的趋势所推动的——大规模软件系统需要一种灵活、系统化、可指导的方式进行动态行为管理和控制,人们开始将管理逻辑以策略的形式从软件功能中独立出来.这种独立性可以实现自适应决策与软件业务功能的关注点分离,从而有利于软件自适应过程中知识和基础设施的重用,实现软件自适应这一维度上相对独立的开发和演化活动.

对策略的研究主要围绕以下几个方面展开:

- 策略语言,为策略的表达提供手段.为便于理解和修改,它们大多比高级语言简单,且以动态解释执行为主.例如,PDL(policy description language)使用 ECA(event-condition-actioon)形式的声明性语言来定义策略[73];Ponder 语言使用域、事件和约束等一系列抽象来定义安全管理策略[74];IBM 的 PMAC 自主计

算平台使用基于 XML 的 ACPL(automatic computing policy language)来表达策略[75];等等;

- 策略冲突.多条策略的同时执行,可能会导致不可预期的后果.策略冲突研究包括两个方面:冲突检测和冲突消解.后者是指当检测到冲突时,选择哪一条(些)策略继续执行.例如:文献[72]研究了 Ponder 策略的模态冲突(modality conflict)问题,将安全管理背景下的策略模态冲突划分为义务冲突、非授权的义务冲突、授权冲突等;Rainbow 使用预定义的效用来区分策略优先级[9];等等;
- 策略求精(refinement)关注如何从所给定的高层需求和目标中生成具体的、机器可理解的动作策略,其实现往往与人工智能方法密切相关,包括基于用例的推理方法、基于强化学习的方法、基于基因编程的方法等[76].

(2) 人工智能相关技术

前文已指出,人或多或少要干预软件的自适应过程.但是,计算机科学最根本的问题之一是"什么可以(有效地)自动化"[77],通过计算机将人从繁琐的劳动中解放出来的追求是永不终止的.Agent、基于用例的推理、强化学习等人工智能技术可以尽可能地减少自适应决策过程中人的干预.

- 软件自适应研究的重要分支之一是采用 Agent 技术[78],包括借鉴 Agent 的内部组织方式[79]、应用 Agent 的理想-信念-意图(BDI)模型[80]、使用面向 Agent 的软件工程方法[81]等.文献[82]指出,IBM 在自主计算中所提出的自配置、自优化、自修复和自保护等概念即是受 Agent 特性的启发;
- 类似的问题往往有类似的解法,基于用例的推理(case-based reasoning,简称 CBR)即基于这一假设.基于用例的推理可以从过去的成功决策中获取经验,在软件自适应领域已经有一些初步的探索.文献[83]将基于用例的推理列为自适应系统重要使能技术;
- 强化学习(reinforcement learning,简称 RL)从行为学、自适应控制等理论发展而来,它把学习看作试探和评价的过程,通过不断试错(trial and error),与环境交互获得知识,改进策略.文献[84]指出,强化学习可以较好地应用于自主计算软件系统的动作选择之中,满足此类系统中动作规划过程对多目标决策、兼顾反应式决策和慎思决策、动态性(不确定性)、高效性等的要求.典型实例是 K-Component 项目中的合作强化学习(collaborative reinforcement learning)算法,它对传统强化学习算法中反馈模型进行了分布式扩展,以实现软件自适应动作在分布式环境下的协同[38].

其他一些可以辅助实现自适应决策的人工智能技术和方法包括基因编程、规划、模糊逻辑等[30,37].

(3) 交叉学科相关技术

与软件自适应决策环节密切相关的交叉学科包括控制理论、决策论、仿生学、社会和经济理论等.

- 控制理论是软件自适应的系统科学基础之一.许多研究者尝试直接应用控制理论的已有成果:Abdelwahed 将模型预测控制(model predicative control)应用到分布式系统中,实现性能自优化[85];Karsai 基于监督控制(supervisory control)理论,将自适应软件系统划分为 Supervisory 层和 Ground 层,前者负责优化、健壮性和灵活性,后者负责基本功能和性能[86];Kokar 基于控制理论提出了 4 种软件自控制模型[87];软件控制论(software cybernetic)试图将控制论映射到软件领域[88];等等;
- 决策论(decision theory)主要研究在环境可能无法被完全预知的情况下,决策者如何决策以及如何达到最优决策.效用理论(utility theory)和概率理论(probabilistic theory)是决策论的两个重要基石[89]:前者可以形式化定义用户偏好,后者则可以用来表达环境和其他因素的不确定性.二者在软件自适应中都得到了较为广泛的应用:Walsh 等人将效用函数用于自主计算,实现动态和异构环境下自主元素对计算资源使用的持续优化[90];MADAM 使用效用来支持基于软件体系结构的自适应,决定何时执行何种体系结构维护动作[11];基于概率理论的马尔可夫过程和贝叶斯网络被应用于软件的自诊断、自恢复和容错中[30];等等;
- 生物体是典型的自适应系统,对生物个体和群体行为的模仿,是实现软件自适应的有效途径之一.文献[91]基于平板扩散、复制、趋化(chemotaxies)、Stigmergy 等生物学现象,提出了若干适用于分布式系统的设计模式;文献[92]指出,生物学所发现的某些规律可以应用到软件自组织算法中;Bionet 中间件

上应用的架构是对蜜蜂、蚂蚁等社会化昆虫组织方式的模仿[93];文献[94]通过对生物进化过程的模仿来实现自适应决策引擎的自动构造和优胜劣汰;等等;

- 社会和经济领域的其他一些研究成果也可以被软件自适应所借鉴,例如,使用微观经济学中的博弈论、机制设计等实现非集中环境下的群体协同,在多 Agent 领域已经存在许多对此类工作的讨论[95].

3.4 执行使能技术

对软件进行在线修改的研究由来以久.在早期的 EDVAC(离散变量自动电子计算机)中,程序指令被放置在可以修改的存储器中,程序员即可使用所谓的自修改代码(self-modifying code)来达到程序动态优化、减少内存占用等目的[96].除了较为简单的参数动态配置外,在当前工程实践中,动态 AOP、基于构件的动态配置、服务动态组合、代码动态迁移等技术可以在不同维度上支持软件的在线修改.

(1) 动态 AOP

动态 AOP 通过运行时的方面(aspect)编织技术实现软件的在线修改,常见的实现手段包括使用预定义的截获点、使用代理(proxy)设计模式、二进制代码动态植入等.文献[97]指出,由于自主系统中许多与适应有关的内容都是横切关注点(如缓存控制、安全、持久性支持等),因此,动态 AOP 能够满足自主系统中适应能力的动态性、易修改、可封装、支持细粒度改动等特性.软件自适应领域一些典型的实践包括:文献[98]给出了策略和方面动态织入相结合实现软件自适应的实例;在 CASA(contract-based adaptive software architecture)[20]中,方面的动态织入和撤消(unweaving)是实现软件自适应的主要途径之一;文献[99]使用面向方面的组装(aspect-oriented compostion)技术实现软件适应;文献[100]面向构件化系统服务质量保证的需要,提出了一个基于规划和动态 AOP 的自适应中间件;等等.

(2) 基于构件的动态配置

相对于面向过程、面向对象等软件开发技术,构件技术至少有如下两个优点:

- 可以支持大粒度、二进制级别的重用;
- 由于构件和容器间接口的标准化,使得构件可被动态组装、配置和管理.

后者是软件自适应的重要使能技术[96]:为了适应不断变化的环境和需求,构件化系统可以通过构件粒度的动态配置进行在线演化,如构件升级、新功能构件的加入、过时构件的淘汰、必要时的构件迁移等.例如:SOFA 构件模型及其扩展 DCUP(dynamic component upgrade)通过定义一系列正交、可缩放的抽象来支持构件在线升级[101];文献[102]给出了反射式构件动态配置模型 RDRM 和相应参考实现 StarDRP;Accord 面向自主计算领域,通过构件系统的动态组装来实现软件自适应[14]等.

(3) 动态服务组合

所谓服务,是指对资源进行封装的自治、平台独立的实体,它们可以被描述、发布、发现和松散绑定.面向服务的计算(service-oriented computing)使用服务来支持快速、廉价、可互操作、可演化和大规模的分布式应用的开发[103].在面向服务的计算中,应用可以动态定位和使用服务,服务间组合关系可以被动态调整,从而为软件在线修改提供系统化的支持[104,105].

(4) 代码动态迁移

代码动态迁移是指在分布计算环境的不同结点之间动态迁移程序实例、代码或对象.这种迁移可以分为强迁移和弱迁移[106],前者是指代码、数据和执行状态一起迁移;后者是指仅仅迁移代码(和数据),执行状态并不迁移,在迁移之后程序可能要重新开始运行.

代码动态迁移是由分布环境下多个结点配合实现软件自适应的重要手段.典型实例是本文第 1 节场景 3 中的任务动态 Offloading 机制——在由“云”和“端”共同组成的移动云计算环境中,可以允许部分任务从移动端动态迁移到云端执行,从而适应能耗状态、网络带宽等环境变化.例如:CloneCloud 项目能够利用虚拟机迁移技术,无缝地将移动设备的应用程序执行块(execution block)迁移至云端执行[107];在文献[108]中,单个移动云计算应用被划分成多个称作 Weblet 的构件,允许根据实际运行时环境执行配置策略,从而动态地决定 Weblet 是在移动设备端执行还是在云端执行;等等.

4 典型实践

本节选取了若干具有典型性、以系统化地构造具备自适应能力软件为目标的软件工程研究项目,对这些项目的概况及其感知、决策、执行各个环节的实现技术进行分析.

4.1 典型项目概况

(1) Rainbow

Rainbow[9,39,109]是卡内基梅隆大学软件工程研究所的自适应软件项目,其目标是构建一个高可重用、基于软件体系结构技术的自适应软件框架.Rainbow 的总体架构由系统层、翻译设施和体系结构层组成.其中:

- 系统层由目标系统、获取监控信息的探针、对目标系统进行在线调整的效用器等组成;
- 体系结构层负责在运行时根据探针所收集的信息维护运行时体系结构模型,进而基于该模型状态和预定义策略触发自适应动作;
- 翻译层负责将体系结构层所做出的动作决策映射到各种类型的目标系统上,通过效用器对目标系统进行在线调整.

上述每一层又包括若干可定制实体(如策略和映射规则等),从而提高了软件框架的可重用性.

(2) K-Component

在 K-Component[10,38]项目中,运行时体系结构模型不仅可以为软件自适应决策提供依据,也可以作为软件体系结构在线调整的入口,通过操纵该模型来对目标系统进行在线修改.在 K-Component 中:构件是基层实体,负责完成计算任务;容器是元层实体,负责依据策略触发自适应动作.自适应动作可以是构件所提供的(特定形式)方法,也可以是对软件体系结构模型的修改,后者将会通过体系结构反射(architectural reflection)机制作用到实际运行系统上.K-Component 的另一特点是构件的每一个自适应动作均会返回一个回馈(reward)值,代表了自适应的效果.K-Component 可以基于该值来实现强化学习、确定当前环境下的最优动作,并进一步通过对强化学习算法的分布式扩展来实现多个结点之间的协同.

(3) MADAM & MUSIC

MADAM(mobility- and adaptation-enableingmiddleware)[11,110]是支持移动环境下自适应软件运行的中间件.在 MADAM 的设计中,同一构件可以有多个适合不同环境的实现体.自适应主要体现为实现体的动态切换,例如当应用迁移到不同设备上时自动加载不同界面.MADAM 为此维护了两类运行时体系结构模型:

- 框架体系结构模型指明了需要哪些构件;
- 实例体系结构模型则指明了对应于这些构件,加载了哪些实现体.

MADAM 中间件可以依据当前环境状态和效用函数进行规划,将框架体系结构模型实时映射到实例体系结构模型,并据此进行实现体切换动作.MUSIC[111]将上述自适应机制进一步扩展到面向服务的体系结构中,其实例体系结构模型中的成员既可以是构件实现体,也可以是动态发现的服务.

(4) Auxo

Auxo[112]是以体系结构为中心、支持自适应软件开发和运行的应用程序框架,其特点在于不仅能够实现自适应,也可以支持第三方(如运维人员)对“感知-决策-执行”软件自适应基本周期的灵活在线调整,从而在线扩展其自适应能力.Auxo 框架由两部分组成:Auxo 体系结构风格和 Auxo 运行基础设施.前者定义一系列用于支持 Auxo 软件开发的体系结构元素,包括感知构件、行为构件、策略连接子、约束等,其特点在于不仅可以封装业务逻辑,也能够将自适应周期各个环节封装为体系结构元素;后者维护了一个与实际系统因果关联的运行时体系结构模型,一方面可以基于该模型实现自适应,另一方面,由于基本周期各环节均已被封装为体系结构元素,因此第三方可以在必要时通过在线修改体系结构来扩展软件的自适应能力.

(5) 3PC & PCOM

3PC(peer-to-peer pervasive computing)[113]是以普适计算为背景、可以在多个层次上为软件适应提供支持的软件基础设施.3PC 由如下一些组件组成:具备网络环境适应能力的底层通信中间件 BASE,支持服务提供者动

态定位的 SANDMAN,面向资源受限设备的自适应构件模型 PCOM[15]以及强调多个应用协同对物理环境进行调整的 COMITY.其中,PCOM 构件具有显式定义的契约(contract),描述了该构件所需计算资源等运行环境、构件所提供或依赖的功能属性和非功能属性等,适应主要体现在 PCOM 容器依据需求和构件契约,自主、动态地调整构件之间的链接,以适应如下两种变化:某一构件所提供服务失效;有新的、能够提供更佳非功能属性的构件动态加入.

(6) Fractal & SAFRAN

Fractal 是一种支持反射的构件模型[114].Fractal 构件由内容(content)和表层(membrane)两部分组成:前者实现业务逻辑,对外提供功能接口;后者对内容进行封装和管理,实现构件暂停、产生检查点和恢复运行等控制接口.控制接口的存在,使得构件的行为和内部结构可以在运行时自省和调整,从而允许 Fractal 应用的体系结构被动态访问和修改.SAFRAN(self-adaptive FRActalcompoNents)[115]是对 Fractal 构件模型的扩展,它使用 Fractal 构件来构造业务逻辑,使用领域相关语言(domain specific language,简称 DSL)来表达自适应逻辑.自适应逻辑通过动态 AOP 技术织入到 Fractal 构件中,使得基于 Fractal 构件的应用具备自适应能力.

(7) Accord

Accord[14,116]是以异构、动态和开放网格计算环境为背景的构件编程模型和框架,以支持自主构件和由这些构件所动态组装而成的分布式应用.应用的自适应能力主要体现在构件可以依据预定义的高层规则管理自身的行为和交互.每一个 Accord 自主构件内建规则 Agent,可以解释和执行预定义的规则,实现构件的自主管理.自主构件同时也对外暴露控制端口,包含对构件状态进行感知的传感器、对构件状态进行修改的效应器等.外部实体可以通过该端口对构件和构件化系统进行运行时调整,例如,通过专门的组装 Agent 来实施的构件增删替换等.

(8) OpenCOM & OpenORB

英国兰卡斯特大学的 OpenCOM[18]是面向操作系统、中间件等系统软件的构件模型,在设计上强调基于反射的开放性、可适应性和可扩展性.OpenCOM 的突出特点是:引入了构件框架机制,构件框架本质上可以被视为一组接口和规则的集合,用以指导“插入”框架的构件如何交互[117].OpenCOM 的运行环境由最小化的内核和基于构件框架的扩展层组成,扩展层可以实现对运行时系统结构和行为的感知和调整.OpenORB[118,119]是 OpenCOM 构件模型在中间件领域的应用.OpenORB 将中间件内部的协议管理、缓冲管理、传输管理、线程管理等均设计为构件框架,使得中间件可以依据需求和资源的变化动态改变自身的配置.

(9) CASA

CASA(contract-based adaptive software architecture)[20,120]是面向动态环境下自适应软件的应用程序框架.CASA 可以通过资源管理等设施监视应用执行环境的变化,然后依据被称为契约(contract)的自适应策略实施自适应动作.在软件自适应的执行环节,CASA 支持如下 4 类动作:调用底层自适应中间件接口实现底层服务的动态改变,借助 PROSE 动态编织系统实现方面的动态编织和撤消,调用应用特定的回调函数实现应用属性的动态改变,构件的动态重组装.

除了上述项目外,国内学者在网构软件技术体系[12]下也针对软件自适应这一目标也开展了一系列实践.例如:文献[121]提出了一种基于反射式中间件 PKUAS[122]的网构软件运行平台,可以基于体系结构反射来实现网构软件的结构自适应,通过自主构件来实现网构软件的实体自适应;文献[123]提出了面向网构软件的环境驱动模型和相应支撑技术,其特点在于除了维护显式的运行时体系结构模型外,还强调环境的显式化,通过环境上下文的建模、管理与使用等来驱动软件的自适应行为;等等.

4.2 研究现状分析

如前所述,“感知-决策-执行”周期是现有软件自适应研究和实践的基础.表 1 给出了第 4.1 节所给典型项目在软件自适应基本周期各环节上的实现机理.

Table 1 Realization mechanisms of typical projects

表 1 典型项目实现机理

项目名称	实现机理		
	感知	决策	执行
Rainbow	使用探针和量规进行感知、聚合和维护体系结构模型	基于 Stitch 语言描述的动作策略	使用效应器实施自适应动作
K-Component	使用构件状态回馈端口获知构件状态	基于 ACDL 语言描述的策略	对体系结构模型进行操作或调用构件的适应动作端口
MADAM & MUSIC	使用上下文服务获取环境信息	基于预定义效用函数进行决策	对构件实例或服务的提供者进行替换
Auxo	通过感知构件获取和聚合环境信息,通过访问体系结构模型获取状态信息	基于封装为连接子、ECA 形式的策略进行决策	对构件运行参数进行调整,对构件和连接子进行增删替换等操作
Accord	使用构件控制端口中的传感器进行构件状态感知	基于行为和交互规则进行决策	改变构件状态或实施动态组装
Fractal & SAFRAN	使用事件机制监听内部和外部事件	基于 ECA 形式的策略进行决策	通过 FScript 语言实施构件动态配置动作
PCOM	通过信号监听器监听构件状态和发现新构件	容器依据内置的缺省策略或用户定义的策略作出决策	容器调整构件间连接关系
OpenCOM & OpenORB	访问 OpenCOM 各类元对象的状态信息,或通过构件框架获取构件状态	由专门的管理构件依据当前状态触发相应动作	操纵 OpenCOM 元对象来实现重配置
CASA	通过资源管理等设施监视应用执行环境中的变化	基于被称为契约的动作策略	实施服务动态改变、方面动态编织和撤消、应用属性动态改变、构件动态重组装

根据本文第 2 节的特征分类方法,前述典型项目所支持软件自适应活动的特征见表 2.

Table 2 Characteristics of adaptation actions supported by typical projects

表 2 典型项目所支持自适应活动的特征

项目名称	所支持自适应活动的特征							
	感知对象	自主程度	决策能力可扩展性	决策模块独立性	模型维护	组织方式	动作类型	动作层次
Rainbow	内部状态/外部资源	动作策略	封闭	外部	是	集中决策	参数/结构适应	应用层
K-Component	内部状态	动作策略	封闭	外部	是	集中/非集中决策	参数/结构适应	应用层
MADAM & MUSIC	外部环境	效用策略	封闭	外部	是	集中决策	结构适应	应用层
Auxo	内部状态/外部环境	动作策略	开放	外部	是	集中决策	参数/结构适应	应用层
3PC/PCOM	内部状态/外部环境	动作策略	封闭	外部	否	集中决策	结构适应	应用层/中间件层
Accord	内部状态	动作策略	开放	外部	否	集中决策	参数/结构适应	应用层
Fractal & SAFRAN	内部状态	动作策略	封闭	外部	否	集中决策	参数/结构适应	应用层
OpenORB	内部状态	动作策略	封闭	外部	是	集中决策	参数/结构适应	中间件层
CASA	内部状态/外部环境	动作策略	封闭	外部	否	集中决策	参数/结构适应	应用层/中间件层

通过表 2 可以看出:

(1) 在感知环节,具有移动或普适计算等背景的项目(如 MADAM)往往强调对物理世界环境变化的直接感知,其他一些项目、特别是早期项目不一定强调对外部环境变化的显式支持(如 K-Component),或者将环境的概念仅仅局限于 CPU 利用率、内存大小等信息空间资源状态(如 Rainbow);

(2) 在决策环节,绝大多数项目采用了实现相对简单的基于动作策略的方法,少数项目(如 MADAM)采用

了基于效用策略的方法,以提高决策自动化程度.大多数项目仅支持封闭自适应,未考虑决策能力的在线扩展问题.此外,除了 K-Component 项目中的合作强化学习机制外,现有的项目基本都未考虑非集中决策的软件群体自适应机制;

(3) 在执行环节,现有项目大多数均支持参数适应和结构适应.由于这些项目的物化成果一般都表现为支持自适应软件运行的软件基础设施,因此适应动作层次包括了中间件层和应用层.

5 研究趋势

近年来,软件系统规模持续增长、内部结构日趋复杂,同时,运行环境表现出了开放性和变化频繁等特征,使得软件自适应技术面临巨大挑战.结合实际应用需求和当前研究热点,我们认为,自适应软件构造和实现技术中,目前最为迫切的挑战在如下两个方面:

- 如何有效应对各种不确定性;
- 如何实现复杂分布式系统中的群体自适应.

本节将在概述相关探索工作的基础上给出未来研究趋势.

5.1 应对不确定性

在软件工程领域,早期的瀑布模型完全基于确定性假设,即假定用户需求、软件运行环境等都是可以事先精确预知.然而,大量已有实践表明,这一假设很难真正成立.特别是在具有长生命周期、规模巨大、与现实世界紧密结合的软件系统中,不确定性已经成为了常态[124,125].软件自适应活动中的不确定性主要来自于两个方面:环境的不确定性和软件自身的复杂性.前者是指开发阶段很难对软件究竟要适应什么样的环境变化做出精确预测,这也就使得动作策略等目前普遍采用的决策技术面临巨大挑战;软件自身的复杂性,则使得这一问题变得更为严峻.例如在基于动作策略的决策过程中,由于组合爆炸问题,很难对多条动作策略同时执行的后果进行完整预测[83],软件在运行时的自适应效果可能无法保证.

有效应对不确定性,方能凸显软件自适应的实际价值,但同时也是其实现技术所面临的重大挑战.结合已有研究和实际需求,我们认为未来研究趋势主要包括:

(1) 针对不确定性的自适应决策技术

在现有的动作策略和效用策略等方法基础上,结合强化学习、统计分析、概率理论等人工智能技术来扩展软件的自主决策能力,是自适应软件应对不确定性的可行途径之一.研究者在这一方面已经有一些初步实践,例如:Rainbow 会对软件自适应的结果进行评估,以确定是否达到了预期的目标,并据此决定下一步的动作;FUSION 项目基于机器学习的方法来实现决策逻辑的在线调整[126];POISED 使用概率理论来评估不确定性的积极和消极影响,据此进行决策[127].受限于人工智能技术现状,此类方法往往与领域和实际应用场景密切相关,其有效性和通用性尚有待进一步研究.

(2) 支持自适应能力在线调整和扩展的实现机制

从软件维护和演化的角度而言,当不确定性出现时,需要能够对“感知-决策-执行”自适应基本周期实施(在线)调整.例如,当非预期的环境出现时,需要动态为软件增加新的环境感知手段和决策逻辑.研究者在这一方面已经展开了一系列初步研究.例如:ACT 可在中间件上动态注册规则截获器(rule interceptor),引入新的自适应行为[128];Chisel 允许用户对基于 Iguna/J 语言的策略进行动态升级[35];Auxo 则进一步基于动态软件体系结构技术,将这种调整动作扩展到感知、决策、执行各个环节.

此外,研究者针对自适应软件需求描述过程中的不确定性也展开了研究,例如,RELAX 需求描述语言可以支持自适应软件环境不确定性的显式表达和推理[129].这些工作对于形成完整的、应对不确定性的软件工程方法体系具有重要意义,但离实用化的程度尚有较大距离,均有待进一步的深入探索.

5.2 实现群体自适应

群体自适应是指涉及多个计算结点,并由这些结点相互配合实现的自适应活动.随着越来越多与现实世界

紧密结合的大型分布式系统的实际部署,此类系统中的群体自适应机制已经引起研究者和实践者的关注.由于分布性和规模的扩展,软件自适应技术面临新的挑战.

(1) 大规模群体感知技术

分布性为环境感知带来了机遇,例如,软件实体间可以共享环境数据,从而更为准确和全面的把握环境状态.但这也同时为环境感知技术带来了新的挑战,特别是在大规模系统中.如何汇聚多个结点收集到的环境和状态数据、如何动态持续地得到全局层面上的态势信息、如何实现信息的有效共享等问题,在近两年引起了广泛关注.我国学者所提出的社群智能概念[130]以及国外学者提出的 Urban Sensing(urban sensing),People/Human-Centric Sensing[131],Mobile CrowdSensing[132]等概念都以此为核心,加州大学的 Common Sense 项目[133]、Dartmouth 大学的 MetroSense 项目[134]、麻省理工大学的 VTrack 项目[135]、苏黎世联邦理工学院的 Ikarus 项目[136]等,都在这一方面展开了初步实践.通过与基于数据流的推理(data stream reasoning)等领域研究成果交叉,大规模环境感知可望在未来为群体自适应的决策环节提供更为准确的依据.

(2) 面向群体自适应的协同决策技术

在决策环节,群体自适应的已有研究可以分为 3 类:

- 集中/分层决策,即通过在多个结点之间建立严格的控制-被控制关系来实现自适应动作,典型实例是 Rainbow 中感知-决策-执行周期的层次化实现[9]和 System-S 中实现大规模分布式系统的自愈的 JMN 协调者(JMN orchestrator)[137];
- 同构非集中决策,即允许多个决策点同时存在,但假定个体意图相同、不会产生相互冲突的决策结果,挑战主要来源于个体视图的一致性保证.例如,文献[138]提出一种面向分布式系统的自组织软件体系结构,通过广播和锁机制保证自适应过程中体系结构视图的一致性;
- 异构非集中决策.在大规模复杂软件系统中,软件个体的意图、管理策略等存在异构性甚至内在冲突,很难建立前述集中/分层决策和同构非集中决策机制,此时需要通过去中心化的方法(例如基于协商的方法)来实现软件自适应决策.文献[139]提出了此类非集中决策自适应的参考模型,但在实现技术方面亟待进一步的研究.

上述 3 类中,异构非集中决策是最具有挑战性的一种,如何构造具有此类决策能力的自适应软件,尚缺乏有效的理论、方法和工具.欧盟 FP7 框架计划已经于 2012 年启动资助额达 2 300 万欧元的群体适应系统基础理论项目,开展软件群体适应的研究,并指出,其难点主要来源于构件和目标的多样性、内部冲突常见性、巨大且不断变化的尺度、环境的复杂性和动态性等[140].

6 结束语

随着与现实世界的紧密结合和软件自身复杂性的增加,软件需要具备灵活主动适应环境变化的能力.这一目标无论对于研究者还是实践者都是巨大的挑战.本文从构造具备自适应能力的软件这一角度出发,对软件自适应的概念内涵、特征分类、使能技术、典型实践依次进行了分析和概述,对应对不确定性和实现群体自适应等方面的最新研究进展进行了阐述,在此基础上,给出了软件自适应领域未来研究趋势.

References:

[1] Laddaga R. Self-Adaptive software. Technical Report, 98-12, DARPA, 1997.

[2] Northrop L, Feiler PH, Pollak B, *et al*. Ultra-Large-Scale systems: The software challenge of the future. Technical Report, Pittsburgh: Carnegie Mellon University, 2006.

[3] PerAda (pervasive adaptation research). 2012. http://www.perada.eu/

[4] Ferguson J. Crouching dragon, hidden software: Software in DoD weapon systems. IEEE Software, 2001,18(4):105−107.

[5] Mobile cloud applications & services. Technical Report, Juniper Research, 2010.

[6] Horn P. Autonomic computing: IBM's perspective on the state of information technology. Technical Report, IBM, 2001.

[7] Smith BC. Reflections and semantics in a procedural language [Ph.D. Thesis]. Massachusetts Institute of Technology, 1982.

[8] Maes P. Concepts and experiments in computational reflection. ACM Sigplan Notices, 1987,22(12):147−155. [doi: 10.1145/38807.38821]

[9] Cheng SW. Rainbow: Cost-effective software architecture-based self-adaptation [Ph.D. Thesis]. Carnegie Mellon University, 2008.

[10] Dowling J, Cahill V. The K-component architecture meta-model for self-adaptive software. In: Proc. of the Int'l Conf. on Metalevel Architectures and Separation of Crosscutting Concerns. 2001. [doi: 10.1007/3-540-45429-2_6]

[11] Floch J, Hallsteinsen S, Stav E, Eliassen F, Lund K, Gjorven E. Using architecture models for runtime adaptability. IEEE Software, 2006,23(2):62−70. [doi: 10.1109/MS.2006.61]

[12] Yang FQ, Lü J, Mei H. Technical framework for internetware: An architecture centric approach. Science in China Series F: Information Sciences, 2008,51(6):610−622. [doi: 10.1007/s11432-008-0051-z]

[13] Ben-Shaul I, Holder O, Lavva B. Dynamic adaptation and deployment of distributed components in hadas. IEEE Trans. on Software Engineering, 2001,27(9):769−787. [doi: 10.1109/32.950315]

[14] Liu H, Parashar M, Hariri S. A component-based programming model for autonomic applications. In: Proc. of the Int'l Conf. on Autonomic Computing. 2004. 10−17. [doi: 10.1109/ICAC.2004.1301341]

[15] Becker C, Handte M, Schiele G, Rothermel K. PCOM—A component system for pervasive computing. In: Proc. of the Int'l Conf. on Pervasive Computing and Communications. 2004. [doi: 10.1109/PERCOM.2004.1276846]

[16] Chefrour D. Developing component based adaptive applications in mobile environments. In: Proc. of the ACM Symp. on Applied Computing. 2005. [doi: 10.1145/1066677.1066935]

[17] Lacouture J, Aniorté P. CompAA: A self-adaptable component model for open systems. In: Proc. of the Int'l Conf. and Workshop on the Engineering of Computer Based Systems. 2008. [doi: 10.1109/ECBS.2008.36]

[18] Coulson G, Blair G, Grace P, Taiani F, Joolia A, Lee K, Ueyama J, Sivaharan T. A generic component model for building systems software. ACM Trans. on Computer Systems, 2008,26(1):1−42. [doi: 10.1145/1328671.1328672]

[19] Román M, Hess C, Cerqueira R, Ranganathan A, Campbell RH, Nahrstedt K. Gaia: A middleware infrastructure to enable active spaces. IEEE Pervasive Computing, 2002,1(4):74−83. [doi: 10.1109/MPRV.2002.1158281]

[20] Mukhija A, Glinz M. Runtime adaptation of applications through dynamic recomposition of components. In: Proc. of the Int'l Conf. on Architecture of Computing Systems. 2005. [doi: 10.1007/978-3-540-31967-2_9]

[21] Zinky JA, Bakken DE, Schantz RD. Architectural support for quality of service for CORBA objects. Theory and Practice of Object Systems, 1997,3(1):55−73. [doi: 10.1002/(SICI)1096-9942(1997)3:1<55::AID-TAPO6>3.0.CO;2-6]

[22] Kon F, Román M, Liu P, Mao J, Yamane T, Magalhães LC, Campbell RH. Monitoring, security, and dynamic configuration with the DynamicTAO reflective ORB. In: Proc. of the Middleware Conf. 2000. [doi: 10.1007/3-540-45559-0_7]

[23] Bradbury JS, Cordy JR, Dingel J, Wermelinger M. A survey of self-management in dynamic software architecture specifications. In: Proc. of the ACM SIGSOFT Workshop on Self-managed Systems. 2004. [doi: 10.1145/1075405.1075411]

[24] Garlan D, Schmerl B. Model-Based adaptation for self-healing systems. In: Proc. of the Workshop on Self-Healing Systems. ACM Press, 2002. [doi: 10.1145/582128.582134]

[25] Sidiroglou S, Laadan O, Perez CR, Viennot N, Nieh J, Keromytis AD. Assure: Automatic software self-healing using rescue points. ACM Sigplan Notices, 2009, 44(3):37−48. [doi: 10.1145/1508284.1508250]

[26] Kumar K, Lu YH. Cloud computing for mobile users: Can offloading computation save energy? IEEE Computer, 2010,43(4):51−56. [doi: 10.1109/MC.2010.98]

[27] Lee EA, Austin TX. Cyber-Physical systems-are computing foundations adequate? In: Proc. of the NSF Workshop on Cyber-Physical Systems: Research Motivation, Techniques and Roadmap. 2006.

[28] IEEE. Recommended practice for architectural description of software-intensive systems. IEEEStandard 1471-2000, 2000. [doi: 10.1109/IEEESTD.2000.91944]

[29] Baresi L. Toward open-world software: Issue and challenges. IEEE Computer, 2006,39(10):36−43. [doi: 10.1109/MC.2006.362]

[30] 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]

[31] Huang JC. Program instrumentation and software testing. IEEE Computer, 1978,11(4):25−32. [doi: 10.1109/C-M.1978.218134]

[32] Murch R. Autonomic Computing. New Jersey: IBM Press and Prentice Hall, 2004.

[33] Wang QX, Shen JR, Mei H. An introduction toself-adaptivesoftware. Computer Science, 2004,31(10):168−172 (in Chinese with English abstract).

[34] Liao BS, Li SJ, Yao Y, Gao J. Conceptual model and realization methods of autonomic computing. Ruan Jian Xue Bao/Journal of Software, 2008,19(4):779−802 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/19/779.htm [doi: 10.3724/SP.J.1001.2008.00779]

[35] Keeney J. Completely unanticipated dynamic adaptation of software [Ph.D. Thesis]. Dublin: University of Dublin, 2004.

[36] Mccarthy D, Dayal U. The architecture of an active database management system. ACM SIGMOD Record, 1989,18(2):215−224. [doi: 10.1145/66926.66946]

[37] Oreizy P, Gorlick MM, Taylor RN, Heimbigner D, Johnson G, Medvidovic N, Quilici A, Rosenblum DS, Wolf AL. An architecture-based approach to self-adaptive software. IEEE Intelligent Systems, 1999,14(3):54−62. [doi: 10.1109/5254.769885]

[38] Dowling J. The decentralised coordination of self-adaptive components for autonomic distributed systems [Ph.D. Thesis]. Dublin: University of Dublin, 2004.

[39] Garlan D, Cheng SW, Huang AC, Schmerl B, Steenkiste P. Rainbow: Architecture-based self-adaptation with reusable infrastructure. IEEE Computer, 2004,37(10):46−54. [doi: 10.1109/MC.2004.175]

[40] Litoiu M, Woodside M, Zheng T. Hierarchical model-based autonomic control of software systems. ACM SIGSOFT Software Engineering Notes, 2005,30(4):1−7. [doi: 10.1145/1082983.1083071]

[41] Karsai G, Sztipanovits J. A model-based approach to self-adaptive software. IEEE Intelligent Systems and Their Applications, 1999, 14(3):46−53. [doi: 10.1109/5254.769884]

[42] Mckinley PK, Sadjadi SM, Kasten EP, Cheng BHC. Composing adaptive software. Computer, 2004,37(7):56−64. [doi: 10.1109/MC.2004.48]

[43] IETF. TCP congestion control. IETF RFC 2581, 1999.

[44] Andersson J, de Lemos R, Malek S, Weyns D. Reflecting on self-adaptive software systems. In: Proc. of the Int'l Workshop on Software Engineering for Adaptive and Self-Managing Systems. 2009. [doi: 10.1109/SEAMS.2009.5069072]

[45] Fayad M, Schmidt DC. Object-Oriented application frameworks. Communications of the ACM, 1997,40(10):32−38. [doi: 10.1145/262793.262798]

[46] Perry DE, Wolf AL. Foundations for the study of software architecture. ACM SIGSOFT Software Engineering Notes, 1992,17(4):40. [doi: 10.1145/141874.141884]

[47] Mei H, Shen JR. Progress of research on software architecture. Ruan Jian Xue Bao/Journal of Software, 2006,17(6):1257−1275 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/17/1257.htm [doi: 10.1360/jos171257]

[48] Medvidovic N, Taylor RN. A classification and comparison framework for software architecture description languages. IEEE Trans. on Software Engineering, 2000,26(1):70−93. [doi: 10.1109/32.825767]

[49] Kramer J, Magee J. Self-Managed systems: An architectural challenge. In: Proc. of the Conf. on the Future of Software Engineering. 2007. [doi: 10.1109/FOSE.2007.19]

[50] Gamma E, Helm R, Johnson R, Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Boston: Addison-Wesley, 1995.

[51] Ramirez AJ. Design patterns for developing dynamically adaptive systems [MS. Thesis]. Michigan State University, 2008.

[52] Schmidt D, Stal M, Rohnert H, Buschmann F. Pattern-Oriented Software Architecture, Patterns for Concurrent and Networked Objects, Vol.2. New York: John Wiley & Sons, 2001.

[53] Shackleton M, Saffre F, Tateson R, Bonsma E, Roadknight C. Autonomic computing for pervasive ICT—A whole-system perspective. BT Technology Journal, 2004,22(3):191−199. [doi: 10.1023/B:BTTJ.0000047132.31406.7f]

[54] Gomaa H, Hussein M. Software reconfiguration patterns for dynamic evolution of software architectures. In: Proc. of the Working IEEE/IFIP Conf. on Software Architecture. 2004. [doi: 10.1109/WICSA.2004.1310692]

[55] Schilit B, Adams N, Want R. Context-Aware computing applications. In: Proc. of the Workshop on Mobile Computing Systems and Applications. 1994. [doi: 10.1109/WMCSA.1994.16]

[56] Brown PJ. The stick-e document: A framework for creating context-aware applications. In: Proc. of the Int'l Conf. on Electronic Publishing, Document Manipulation and Typography. 1995.

[57] Xu GY, Shi YC, Xie WK. Pervasive/Ubiquitous computing. Journal of Computer, 2003,26(9):1042−1050 (in Chinese with English abstract).

[58] Strang T, Linnhoff-Popien C. A context modeling survey. In: Proc. of the Workshop on Advanced Context Modelling, Reasoning and Management. 2004.

[59] Salber D, Dey AK, Abowd GD. The context toolkit: Aiding the development of context-enabled applications. In: Proc. of the SIGCHI Conf. on Human Factors in Computing Systems. 1999. [doi: 10.1145/302979.303126]

[60] Chen HL. An intelligent broker architecture for pervasive context-aware systems. University of Maryland, 2004.

[61] Gu T, Pung HK, Zhang DQ. A middleware for building context-aware mobile services. In: Proc. of the IEEE Vehicular Technology Conf. 2004. [doi: 10.1109/VETECS.2004.1391402]

[62] Ding B, Wang HM, Shi DX. Pervasive middleware technology. Journal of Frontiers of Computer Science and Technology, 2007, 1(3):241−254 (in Chinese with English abstract).

[63] Ferguson HE. Debugging systems at the source language level. Communications of the ACM, 1963,6(8):430−432. [doi: 10.1145/366707.367526]

[64] Robert Balzer E. Exdams: EXtensible debugging and monitoring system. In: Proc. of the Spring Joint Computer Conf. 1969. [doi: 10.1145/1476793.1476881]

[65] Delgado N, Gates AQ, Roach S. A taxonomy and catalog of runtime software-fault monitoring tools. IEEE Trans. on Software Engineering, 2004,30(12):859−872. [doi: 10.1109/TSE.2004.91]

[66] Snodgrass R. A relational approach to monitoring complex systems. ACM Trans. on Computer Systems, 1988,6(2):157−195. [doi: 10.1145/42186.42323]

[67] Beth AS. On-Line monitoring: A tutorial. IEEE Computer, 1995,28(6):72−78. [doi: 10.1109/2.386988]

[68] Robinson WN. Monitoring Web service requirements. In: Proc. of the Int'l Conf. on Requirements Engineering. 2003.

[69] Garlan D, Schmerl B, Chang J. Using gauges for architecture-based monitoring and adaptatio. In: Proc. of the Working Conf. on Complex and Dynamic Systems Architecture. 2001.

[70] Hasselbring W, Reussner R. Toward trustworthy software systems. IEEE Computer, 2006,39(4):91−92. [doi: 10.1109/MC.2006.142]

[71] Bowring J, Orso A, Harrold MJ. Monitoring deployed software using software tomography. In: Proc. of the ACM SIGPLAN-SIGSOFT Workshop on Program Analysis for Software Tools and Engineering. 2002. [doi: 10.1145/586094.586099]

[72] Lupu EC, Sloman M. Conflicts in policy-based distributed systems management. IEEE Trans. on Software Engineering, 1999,25(6):852−869. [doi: 10.1109/32.824414]

[73] Lobo J, Bhatia R, Naqvi S. A policy description language. In: Proc. of the National Conf. on Artificial Intelligence. 1999.

[74] Damianou N, Dulay N, Lupu E, Sloman M. The ponder policy specification language. In: Proc. of the IEEE/IFIP Network Operations and Management Symp. 2001. [doi: 10.1007/3-540-44569-2_2]

[75] Agrawal D, Lee KW, Lobo J. Policy-Based management of networked computing systems. IEEE Communications Magazine, 2005, 43(10):69−75. [doi: 10.1109/MCOM.2005.1522127]

[76] Boutaba R, Aib I. Policy-Based management: A historical perspective. Journal of Network and Systems Management, 2007,15(4):447−480. [doi: 10.1007/s10922-007-9083-8]

[77] Ralston A, Reilly ED, Hemmendinger D. Encyclopedia of Computer Science. New York: John Wiley & Sons, 2000.

[78] Wooldridge M, Jennings NR. Intelligent agents: Theory and practice. Knowledge Engineering Review, 1995,10(2):115−152. [doi: 10.1017/S0269888900008122]

[79] Bratman ME, Israel D, Pollack ME. Plans and resource-bounded practical reasoning. Computational Intelligence, 1988,4(4):349−355.

[80] Rao AS, Georgeff MP. Modeling rational agents within a BDI-architecture. In: Proc. of the Int'l Conf. on Principles of Knowledge Representation and Reasoning. 1991.

[81] Wooldridge M, Ciancarini P. Agent-Oriented software engineering: The state of the art. In: Proc. of the Int'l Workshop on Agent-Oriented Software Engineering. LNCS 1957. 2001. 1−28. [doi: 10.1007/3-540-44564-1_1]

[82] Huebscher MC, Mccann JA. A survey of autonomic computing—Degrees, models, and applications. ACM Computing Surveys, 2008,40(3):7. [doi: 10.1145/1380584.1380585]

[83] Kakousis K, Paspallis N, Papadopoulos GA. A survey of software adaptation in mobile and ubiquitous computing. Enterprise Information Systems, 2010,4(4):355−389. [doi: 10.1080/17517575.2010.509814]

[84] Amoui M, Salehie M, Mirarab S, Tahvildari L. Adaptive action selection in autonomic software using reinforcement learning. In: Proc. of the Int'l Conf. on Autonomic and Autonomous Systems. 2008. [doi: 10.1109/ICAS.2008.35]

[85] Abdelwahed S, Kandasamy N, Neema S. Online control for self-management in computing systems. In: Proc. of the IEEE Real-Time and Embedded Technology and Applications Symp. 2004. [doi: 10.1109/RTTAS.2004.1317283]

[86] Karsai G, Ledeczi A, Sztipanovits J, Peceli G, Simon G, Kovacshazy T. An approach to self-adaptive software based on supervisory control. In: Proc. of the Int'l Workshop in Self-adaptive Software. 2001. [doi: 10.1007/3-540-36554-0_3]

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

[88] Cai KY, Cangussu JW, Decarlo RA, Mathur AP. An overview of software cybernetics. In: Proc. of the Int'l Workshop on Software Technology and Engineering Practice. 2003. [doi: 10.1109/STEP.2003.4]

[89] North DW. A tutorial introduction to decision theory. IEEE Trans. on Systems Science and Cybernetics, 1968,4(3):200−210. [doi: 10.1109/TSSC.1968.300114]

[90] Walsh WE, Tesauro G, Kephart JO, Das R. Utility functions in autonomic systems. In: Proc. of the Int'l Conf. on Autonomic Computing. 2004. [doi: 10.1109/ICAC.2004.1301349]

[91] Babaoglu O, Canright G, Deutsch A, Caro GD, Ducatelle F, Gambardella L, Ganguly N, Jelasity M, Montemanni R, Montresor A. Design patterns from biology for distributed computing. ACM Trans. on Autonomous and Adaptive Systems, 2006,1(1):26−66. [doi: 10.1145/1152934.1152937]

[92] Di Nitto E, Dubois DJ, Mirandola R. On exploiting decentralized bio-inspired self-organization algorithms to develop real systems. In: Proc. of the Int'l Workshop on Software Engineering for Adaptive and Self-Managing Systems. 2007. [doi: 10.1109/SEAMS.2009.5069075]

[93] Suzuki J, Suda T. A middleware platform for a biologically inspired network architecture supporting autonomous and adaptive applications. IEEE Journal on Selected Areas in Communications, 2005,23(2):249−260. [doi: 10.1109/JSAC.2004.839388]

[94] Mckinley PK, Cheng B, Ofria CA. Applying digital evolution to the development of self-adaptive ULS systems. In: Proc. of the Int'l Workshop on Software Technologies for Ultra-Large-Scale Systems. 2007. [doi: 10.1109/ULS.2007.1]

[95] Parsons S, Wooldridge M. Game theory and decision theory in multi-agent systems. Autonomous Agents and Multi-Agent Systems, 2002,5(3):243−254. [doi: 10.1023/A:1015575522401]

[96] Mckinley PK, Sadjadi SM, Kasten EP, Cheng BHC. A taxonomy of compositional adaptation. Technical Report, Michigan State University, 2004.

[97] Greenwood P, Blair L. Using dynamic aspect-oriented programming to implement an autonomic system. In: Proc. of the Dynamic Aspects Workshop. 2004.

[98] Greenwood P, Blair L. A framework for policy driven auto-adaptive systems using dynamic framed aspects. Trans. on Aspect-Oriented Software Development II, 2006. [doi: 10.1007/11922827_2]

[99] Grace P, Lagaisse B, Truyen E, Joosen W. A reflective framework for fine-grained adaptation of aspect-oriented compositions. In: Proc. of the Int'l Conf. on Software Composition. 2008. [doi: 10.1007/978-3-540-78789-1_17]

[100] Rouvoy R, Beauvois M, Eliassen F. Dynamic aspect weaving using a planning-based adaptation middleware. In: Proc. of the ACM Symp. on Applied Computing. 2008. [doi: 10.1145/1394272.1394280]

[101] Pláll F, Bálek D, Janecek R. SOFA/DCUP: Architecture for component trading and dynamic updating. In: Proc. of the Int'l Conf. on Configurable Distributed Systems. 1998. [doi: 10.1109/CDS.1998.675757]

[102] Dou L. Research on dynamic reconfiguration technology in component-oriented complex software system [Ph.D. Thesis], Changsha: National University of Defense Technology, 2005 (in Chinese with English abstract).

[103] Papazoglou MP, Traverso P, Dustdar S, Leymann F. Service-Oriented computing: state of the art and research challenge. IEEE Computer, 2007,40(11):38−45. [doi: 10.1109/MC.2007.400]

[104] Verma K, Sheth AP. Autonomic Web processes. In: Proc. of the Int'l Conf. on Service Oriented Computing. 2005. [doi: 10.1007/11596141_1]

[105] Casati F, Shan MC. Dynamic and adaptive composition of e-services. Information Systems, 2001,26(3):143−163. [doi: 10.1016/S0306-4379(01)00014-X]

[106] Fuggetta A, Picco GP, Vigna G. Understanding code mobility. IEEE Trans. on Software Engineering, 1998,24(5):342−361. [doi: 10.1109/32.685258]

[107] Chun BG, Ihm S, Maniatis P, Naik M, Patti A. Clonecloud: Elastic execution between mobile device and cloud. In: Proc. of the European Conf. on Computer Systems. 2011. [doi: 10.1145/1966445.1966473]

[108] Zhang XW, Jeong S, Kunjithapatham A, Gibbs S. Towards an elastic application model for augmenting the computing capabilities of mobile devices with cloud computing. Mobile Networks and Applications, 2011,16(3):270−284. [doi: 10.1007/s11036-011-0305-7]

[109] Garlan D, Schmerl B, Cheng SW. Software architecture-based self-adaptation. In: Denko MK, *et al*., eds. Proc. of the Autonomic Computing and Networking. 2009. 31−55. [doi: 10.1007/978-0-387-89828-5_2]

[110] Paspallis N, Papadopoulos GA. An approach for developing adaptive, mobile applications with separation of concerns. In: Proc. of the Int'l Computer Software and Applications Conf. 2006. [doi: 10.1109/COMPSAC.2006.22]

[111] Rouvoy R, Barone P, Ding Y, Eliassen F, Hallsteinsen S, Lorge J, Mamelli A, Scholz U. MUSIC: Middleware support for self-adaptation in ubiquitous and service-oriented environments. In: Betty H, Cheng C, *et al*., eds. Proc. of the Software Engineering for Self-Adaptive Software Systems. 2008. [doi: 10.1007/978-3-642-02161-9_9]

[112] Ding B, Wang HM, Shi DX, Cao JN. Taming software adaptability with architecture-centric framework. In: Proc. of the IEEE Int'l Conf. on Pervasive Computing and Communications. 2010. [doi: 10.1109/PERCOM.2010.5466983]

[113] Handte M, Schiele G, Matjuntke V, Becker C, Marrón PJ. 3PC: System support for adaptive peer-to-peer pervasive computing. ACM Trans. on Autonomous and Adaptive Systems, 2012,7(1):10. [doi: 10.1145/2168260.2168270]

[114] Bruneton E, Coupaye T, Leclercq M, Quema V, Stefani JB. An open component model and its support in Java. In: Proc. of the Int'l Symp. on Component-Based Software Engineering. 2004. [doi: 10.1007/978-3-540-24774-6_3]

[115] David PC, Ledoux T. An aspect-oriented approach for developing self-adaptive fractal components. In: Proc. of the Int'l Symp. on Software Composition. 2006. [doi: 10.1007/11821946_6]

[116] Liu H, Parashar M. Accord: A programming framework for autonomic applications. IEEE Trans. on Systems, Man, and Cybernetics, 2006,36(3):341−352. [doi: 10.1109/TSMCC.2006.871577]

[117] Syperski C. Component Software: Beyond Object-Oriented Programming. Boston: Addison-Wesley, 2002.

[118] Blair GS, Coulson G, Andersen A, Blair L, Clarke M, Costa F, Duran-Limon H, Fitzpatrick T, Johnston L, Moreira R, Parlavantzas N, Saikoski K. The design and implementation of open ORB 2. IEEE Distributed Systems Online, 2001,2(6):1−40.

[119] Parlavantzas N, Coulson G. Designing and constructing modifiable middleware using component frameworks. IET Software, 2007, 1(4):113−126. [doi: 10.1049/iet-sen:20060050]

[120] Mukhija A, Glinz M. A framework for dynamically adaptive applications in a self-organized mobile network environment. In: Proc. of the Workshop on Distributed Auto-Adaptive and Reconfigurable Systems. 2004. [doi: 10.1109/ICDCSW.2004.1284056]

[121] Mei H, Huang G, Zhao HY, Jiao WP. A software architecture centric engineering approach for internetware. Science in China Series F: Information Sciences, 2006,49(6):702−730.

[122] Mei H, Huang G. PKUAS: An architecture-based reflective component operating platform. In: Proc. of the Int'l Workshop on Future Trends of Distributed Computing Systems. 2004. [doi: 10.1109/FTDCS.2004.1316609]

[123] Lü J, Ma XX, Tao XP, Cao C, Huang Y, Yu P. On environment-driven software model for internetware. Science in China Series F: Information Sciences, 2008,51(6):683−721.

[124] Esfahani N, Malek S. Uncertainty in self-adaptive software systems. In: Proc. of the Software Engineering for Self-Adaptive Systems II. 2012. [doi: 10.1007/978-3-642-35813-5_9]

[125] Ramirez AJ, Jensen AC, Cheng BHC. A taxonomy of uncertainty for dynamically adaptive systems. In: Proc. of the ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems. 2012. [doi: 10.1109/SEAMS.2012.6224396]

[126] Elkhodary A, Esfahani N, Malek S. FUSION: A framework for engineering self-tuning self-adaptive software systems. In: Proc. of the ACM SIGSOFT Int'l Symp. on Foundations of Software Engineering. 2010. [doi: 10.1145/1882291.1882296]

[127] Esfahani N, Kouroshfar E, Malek S. Taming uncertainty in self-adaptive software. In: Proc. of the ACM SIGSOFT Symp. and European Conf. on Foundations of Software Engineering. 2011. [doi: 10.1145/2025113.2025147]

[128] Sadjadi SM, Mckinley PK. ACT: An adaptive CORBA template to support unanticipated adaptation. In: Proc. of the Int'l Conf. on Distributed Computing Systems. 2004. [doi: 10.1109/ICDCS.2004.1281570]

[129] Whittle J, Sawyer P, Bencomo N, Cheng BHC, Bruel JM. RELAX: Incorporating uncertainty into the specification of self-adaptive systems. In: Proc. of the Int'l Requirements Engineering Conf. 2009. [doi: 10.1109/RE.2009.36]

[130] Zhang DQ, Guo B, Yu ZW. The emergence of social and community intelligence. IEEE Computer, 2011,44(7):21−28. [doi: 10.1109/MC.2011.65]

[131] Campbell AT, Lane ND, Miluzzo E, Peterson RA, Lu H, Zheng X, Musolesi M, Fodor K. The rise of people-centric sensing. IEEE Internet Computing, 2008,12(4):12−21. [doi: 10.1109/MIC.2008.90]

[132] Ganti RK, Ye F, Lei H. Mobile crowdsensing: Current state and future challenges. IEEE Communication Magazine, 2011,49(11): 32−39. [doi: 10.1109/MCOM.2011.6069707]

[133] Dutta P, Aoki PM, Kumar N, Mainwaring A, Myers C, Willett W, Woodruff A. Common sense: Participatory urban sensing using a network of handheld air quality monitors. In: Proc. of the ACM Conf. on Embedded Networked Sensor Systems. 2009. [doi: 10.1145/1644038.1644095]

[134] Eisenman SB, Lane ND, Miluzzo E, Peterson RA, Ahn GS, Campbell AT. MetroSense project: People-centric sensing at scale. In: Proc. of the Workshop on World-Sensor-Web. 2006.

[135] Thiagarajan A, Ravindranath L, Lacurts K, Toledo S, Eriksson J. VTrack: Accurate, energy-aware road traffic delay estimation using mobile phones. In: Proc. of the ACM Conf. on Embedded Networked Sensor Systems. 2009. [doi: 10.1145/1644038.1644048]

[136] Von Kaenel M, Sommer P, Wattenhofer R. Ikarus: Large-Scale participatory sensing at high altitudes. In: Proc. of the Workshop on Mobile Computing Systems and Applications. 2011. [doi: 10.1145/2184489.2184503]

[137] Jacques-Silva G, Challenger J, Degenaro L, Giles J, Wagle R. Towards autonomic fault recovery in system-S. In: Proc. of the Int'l Conf. on Autonomic Computing. 2007. [doi: 10.1109/ICAC.2007.40]

[138] Georgiadis I, Magee J, Kramer J. Self-Organising software architectures for distributed systems. In: Proc. of the Workshop on Self-Healing Systems. 2002. [doi: 10.1145/582128.582135]

[139] Weyns D, Malek S, Andersson J. On decentralized self-adaptation: Lessons from the trenches and challenges for the future. In: Proc. of the ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems. 2010. [doi: 10.1145/1808984.1808994]

[140] FP7: FET proactive initiative: Fundamentals of collective adaptive systems. 2012. http://cordis.europa.eu/fp7/ict/fet-proactive/focas_en.html

附中文参考文献:

[33] 王千祥,申峻嵘,梅宏.自适应软件初探.计算机科学,2004,31(10):168−172.

[34] 廖备水,李石坚,姚远,等.自主计算概念模型与实现方法.软件学报,2008,19(4):779−802. http://www.jos.org.cn/1000-9825/19/779.htm [doi: 10.3724/SP.J.1001.2008.00779]

[47] 梅宏,申峻嵘.软件体系结构研究进展.软件学报,2006,17(6):1257−1275. http://www.jos.org.cn/1000-9825/17/1257.htm [doi: 10.1360/jos171257]

[57] 徐光祐,史元春,谢伟凯.普适计算.计算机学报,2003,26(9):1042−1050.

[62] 丁博,王怀民,史殿习.普适计算中间件技术.计算机科学与探索,2007,1(3):241−254.

[102] 窦蕾.面向构件的复杂软件系统中动态配置技术的研究[博士学位论文].长沙:国防科学技术大学,2005.

丁博(1978－),男,湖南攸县人,博士,助理研究员,CCF 会员,主要研究领域为分布计算,自适应软件.
E-mail: dingbo@nudt.edu.cn

史殿习(1966－),男,博士,研究员,CCF 会员,主要研究领域为分布计算.
E-mail: dxshi@nudt.edu.cn

王怀民(1962－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为分布计算,网络安全.
E-mail: whm_w@163.com