文章编号：0253-374X(2016)10-1624-12 DOI:10.11908/j.issn.0253-374x.2016.10.022

网络化软件自适应动态演化技术的趋势分析

陆超泽[1,2]，曾国荪[1,2]

(1.同济大学 电子与信息工程学院，上海 200092；
2.同济大学 嵌入式系统与服务计算教育部重点实验室，上海 200092)

摘要：综述了网络化软件自适应动态演化技术，以软件演化基本概念为基础，深刻分析了网络化软件自适应动态演化的动因，重点评述了网络软件自适应动态演化技术在需求捕捉、情境建模、感知操作、决策技术、演化操作五个方面的发展现状，由此指出网络化软件自适应动态演化面临的挑战和发展趋势.

关键词：网络化软件；软件演化；自适应技术；研究综述
中图分类号：TP311 **文献标志码**：A

Trend Analysis of Adaptive Dynamic Evolution Technology for Networked Software

LU Chaoze[1,2], *ZENG Guosun*[1,2]

(1. College of Electronics and Information Engineering, Shanghai 201804, China; 2. Embedded System and Service Computing Key Laboratory of the Ministry of Education, Tongji University, Shanghai 201804, China)

Abstract: The paper presents a review of networked software adaptive dynamic evolution technology. First, on the basis of the concept of software evolution, an analysis was made of the reason of networked software adaptive dynamic evolution. Then, comment was made of development state of networked software adaptive dynamic evolution in the requirements capturing, context modeling, perception operation, decision-making technology, evolution operation. In the end, the challenge and trend for the networked software adaptive dynamic evolution was pointed out.

Key words: networked software; software evolution; self-adaptive technology; research review

演化最早被生物学家达尔文用来表示变化的继承，揭示不同世代的物种之间存在着差异现象. 后来，随着软件技术的发展，人们把“演化”概念引进了软件工程领域，出现了“软件演化”这一术语. 近年来，不少学者从多种角度对软件演化的发展进行分析和预测，综述软件演化的发展现状和发展趋势，虽然不能形成统一的见解，却充分展现了各自的认识和观点.

早期，软件演化并不被人们所认识，在第二届软件工程国际大会(ICSE)上，Swanson[1]发表了《维护的维度》一文，软件演化雏形初步形成. Lehman[2]根据早期对软件操作系统版本的追踪，从生命周期和演化定律的角度对软件程序级的演化进行综述，软件演化开始被人们所认识. Perry[3]从认知的角度对软件演化进行总结，认为软件系统演化速度和相关人员的认知水平有重要关系. 我国学者杨芙清等人[4]从软件技术发展的角度，总结了函数、对象级的演化已经不能满足 Internet 软件的发展，指出了网构软件演化模型是软件演化的重要方向. 随着互联网的发展，软件演化的大量的工作集中在网络化软件演化中，南京大学的吕建等人[5]分别从软件演化的原因、软件演化的定律、自演化的要素进行了重要阐述. 北京大学的梅宏等人[6]以体系结构为中心，将软件演化技术围绕着生命周期作了详细技术分析. 湖南大学的陈洪龙等人[7]在总结自适应演化软件特征的基础上，从语言、体系结构和框架应用层面对已有工作进行概括和分析. 浙江大学李长云等人[8]从动态演化的语言、模型和平台三者之间的关系综述已有的研究成果，提出自适应演化是动态演化未来的方向. 最近几年，很多研究学者从软件服务的角度对软件在线自适应演化进行综述. 例如，中科院王青

收稿日期：2016-03-05
基金项目：上海市优秀学科带头人计划(10XD1404400)；华为创新研究计划(IRP-2013-12-03)；高效能服务器和存储技术国家重点实验室开放基金(2014HSSA10)
第一作者：陆超泽(1988—)，男，博士生，主要研究方向为软件演化和可信软件. E-mail:luchaoze@126.com
通讯作者：曾国荪(1964—)，男，教授，博士生导师，工学博士，主要研究方向为并行计算、可信软件和信息安全. E-mail:gszeng@tongji.edu.cn

等人[9]从互联网的角度去总结软件演化，认为互联网软件将成为“永远的测试版”. 国防科技大学的王怀民等人[10-11]从软件服务的在线演化角度提出了“成长构造性”和“适应性演化”两条法则. Aversano 等人[12]从 ERP 开源软件在企业中的适应的角度去总结软件演化的发展. Liu 等人[13]从模型驱动的需求工程探索网构软件的自适应演化.

虽然很多学者对软件演化以及软件自适应演化的研究进展进行了综述性总结，主要以网构软件为对象，分别从软件体系结构、Web 服务、互联网等方面或角度描述自适应演化，但是由于考虑角度不一样，对其理解也不一样. 因此，软件自适应演化的概念不清晰、产生动因不明确、发展所涉及技术阐述不全面、发展趋势不明确，以上综述工作都有“盲人摸象”之嫌. 当前，以互联网技术为中心的网构软件、web 服务软件等成为了软件的一种新形态，这类软件与传统软件本质的区别在于其赖以生存的环境为互联网、组成的基本单位为互联网上的服务资源、其结构和行为可以动态演化，为此统称具有这类特点的软件为网络化软件. 网络化软件已成为软件发展的主流方向，为此以网络化软件为对象，拟从“需求捕捉、情境建模、感知方式、决策方式、演化方式”五个方面，对自适应动态演化技术的研究现状进行全方位的分析，力求将自适应演化技术概念清晰化、自适应演化产生的动因阐述明确，试图将软件自适应演化发展所涉及技术阐述更全面、软件自适应演化发展趋势更明确.

1　软件演化及动因

定义 1（软件维护）：指在软件产品发布运行后，为了确保实现软件需求规约要求，对软件进行的修正错误、适应环境、增加功能、提升性能、预防问题发生的修改活动.

定义 2（软件演化）：指在软件的生命周期内，对已有软件系统不断进行修改、补充和完善，以便适应外界环境变化，满足用户不断变化需求和目标的维护和更新活动.

软件维护和软件演化这两个概念既有关联又有区别[9]：①软件维护通常是系统为了实现软件预先的需求规约而进行的一组有计划的活动；软件演化除了有计划活动外还包括软件随时遇到的突发情况，如事先没有预料到体系结构的改变、全新计算模式的出现等. ②软件维护没有“本质变化”含义，注重的是保持和修复；软件演化更加强调“本质变化”，表示从旧的设计变化到新的设计的过程. ③软件维护通常是发生在软件运行期间的修改活动；而软件演化着眼于软件整个生命周期，全面观察和推进软件系统向前演化. ④软件维护更加偏向于实际工程领域，强调下一步该做什么、风险期望等；软件演化则是范围更加广、更加偏向理论化，强调的是出现了某种现象，该采取何种方法应对.

纵观软件演化的发展历程，软件演化的研究最早在 20 世纪 60 年代，Lehman 对 IBM OS/360 操作系统版本进行软件演化研究，发现持续演化是软件固有的基本属性. 在 20 世纪 80～90 年代后期相继总结出了著名的 Lehman 软件演化八大定律[14]，具有演化过程的 Bennet 分段演化模型、Bohem 螺旋演化模型等相继被提出. 20 世纪 90 年代后期，研究者开始从程序语言、体系结构等层面以软件复用的角度对软件演化进行研究，出现各种支持软件演化的技术：构件技术、软件架构、软件逆向工程、软件再工程、领域工程等. 然而在这些支持软件演化的各项技术中，通常将软件演化划分为静态演化和动态演化.

定义 3（静态演化）：指软件在停机状态下，对软件系统的功能进行完善与扩充、性能的提高与加强、体系结构的改变与重构等维护和更新活动，并且这种活动必须要重新启动，或者重新编译软件系统才能得以实现.

定义 4（动态演化）：指软件在运行期间，不需要重启或者重新编译软件系统，就能动态或在线地实现对软件进行功能完善与扩充、性能的提高与加强、体系结构改变与重构等行为的维护和更新活动.

静态演化和动态演化虽然都是为了实现软件维护和更新这一共同目标而进行的，但是两者却具有自己的优点和缺点. 静态演化主要优点在于不用考虑软件状态迁移和活动进程处理等问题，缺点是会造成软件服务的中断、暂时失效、重启时间过长等问题. 动态演化的优点在于演化期间服务持续不断地可用，适合开放多变的分布式网络环境，缺点是需要考虑状态迁移、触发时机不确定、演化范围难控等. 除了静态演化和动态演化概念之外，人们还从不同的角度对软件演化进行了分类，具体如表 1 所示.

近年来，随着网构软件、普适计算软件、云软件、信息物理系统软件等新一代软件范型的出现，软件的形态正在发生深刻地变革，网络化软件成为了常态. 针对网络化软件，人们对它的期盼更加强调的是自适应的效果，于是出现了网络化软件自适应动态

表1 软件演化分类

Tab.1 Software evolution classification

序号	演化分类依据角度	演化类型
1	触发时机	静态演化、动态演化
2	演化粒度	函数演化、对象演化、构件演化、网构软件演化、服务演化
3	功能	功能性演化、非功能性演化
4	演化的预期性	预期演化、非预期演化
5	演化的主动性	反应型演化、前摄型演化
6	演化层次	单节点演化、跨节点演化
7	自动化程度	人参与交互演化、自动型演化

演化的新概念.

定义5(网络化软件自适应动态演化):指网络化软件可以从外部环境中,自动地收集自身行为信息,根据某些评价机制来评判自身行为,并依据评判结果主动决策是否调整自身行为和结构,以便更好地达到预期目标.

网络化软件自适应演化现象普遍存在,例如:①苹果手机的官方经过一段时间后就会发布最新的IOS,要求IOS具有自动收集手机的硬件资源、并根据评判机制的评判结果自动决策是否进行升级活动.②移动云计算环境下,移动终端可以根据当前终端电量剩余量、任务数量多少等信息,自动地将任务在云端和移动终端之间的自由地迁移,以最小的开销来满足用户的体验.可见,网络化自适应软件体现出了以下特征:感知性、自查性、自主性、动态演化性、智能适应性、协同性、多目标性.总之,网络化软件自适应动态演化的目标是软件实现失效时可以实现自我动态配置、自动恢复且检测自身状态、自我保证可靠性、不断优化自身功能和性能,即实现自配置、自治愈、自保护和自优化.

2 网络化软件自适应动态演化的动因

计算机的几十年发展中,计算模式由个人计算到网络计算的转变,分布式计算软件得到了广泛地部署.网络化软件在需求变化、运行环境、应用模式等方面都表现出与传统应用软件诸多不同,自适应动态演化成为网络化软件应具备的基本能力,下面将分析网络软件自适应动态演化的动力和原因.

2.1 用户需求的变化

"变化"是现实世界的永恒主题,世界只有"变化"才能发展.软件是对真实世界里问题空间以及问题解空间的实在描述,用户需求是问题空间的一种具体反映,因此用户需求也具有"变化性".然而当前用户需求更加强调时效性地变化,由此驱动了软件必须具有自适应演化的能力.例如:在当前最流行的电子商务平台中,原有的基本业务流程为选择商品、在线支付、下订单、接收订单、检查库存、发货、收货,然而现在用户把在线支付改为货到付款,这就驱动了软件必须具有可以自适应的为用户重新规划业务流程的能力.对于大家使用的手机,原先只有简单的彩色拍照功能,然而用户希望拍照可以根据当前的环境自适应的调节光圈和快门,使拍照的效果更好,这就驱动了手机必须增强相机模块自适应调节功能.数值求解时,响应时间要求由5 s变为3 s,计算结果精度要求小数点后2位,变为小数点后5位,这就驱动了软件必须具有自适应地选择更高速度与精度云节点的能力.当前很多电子商务平台都要求在保护顾客隐私的前提下具有自动捕捉顾客所在地能力,以此方便提供不同的语言和商品的购买页面,这就驱动了软件具有依据不同国家和地区的政策和法律自适应地提供合法购买页面的能力等等.由此可见,由于网络化软件的快捷、轻便、易用等特点使得用户尝到了甜头,促使用户在互联网环境下对服务的要求更高、个性化的追求更强、安全隐私保护更渴望、性能要求更高、功能要求更加齐全、业务流程更加便捷、智能程度要求更高、实时性要求更突出等,然而用户的这些需求是无止境的、无规律的、无确定目标的,因此用户需求不确定性的变化是驱动网络化软件自适应演化的动因之一.

2.2 软件运行环境的变迁

软件系统通常在交付后持续运行多年,在此期间,CPU的处理速度按摩尔定律每18个月加速1倍,计算机内存从1MB变成4GB,硬盘变换成了存储阵列,运行终端由台式机变换成笔记本电脑和iPad,服务器换成了集群,新设备的不断使用,导致硬件环境变化.由此必然驱动软件必须具有自动识别终端、服务器和各种设备的能力,从而自适应地选择不同的软件版本等提供相应的支持.软件系统的基础支持平台是操作系统和工具软件.操作系统从早期的DOS,发展到现在Windows,Unix和Linux.数据库从早期的Foxbase,发展到Mysql,Oracle,再发展到现在的NoSQL和NewSQL.这些必然导致软件环境变化.由此必然驱动软件必须具有自动识别操作系统和数据库类型和版本的能力,从而自适应地选择不同的软件版本等提供相应的支持.近年来,网络经过改造,由局域网变成广域网,由广域网变成WWW全球互联网,由有线网变成无线网,接入方式更加灵活方便,软件的运行环境已从单机环

境上过渡到了网络环境，然而网络环境具有动态性不稳定性，由此驱动软件必须具有自适应选择较高质量网络环境的能力. 当前物联网和云计算环境都体现出了弹性构造，也就是说硬件资源和软件资源可以按需选取，由此驱动了软件必须具有自适应地快速匹配软件环境和硬件环境的能力. 由此可见，网络化软件与传统软件的运行环境存在本质上不同，表现为：硬件资源规模化、操作系统多样化、数据库系统非结构化、应用环境多元化、硬软件资源按需而取化等. 上述运行环境特征使得网络化软件的应用、数据、计算等资源随时处于可以变化的状态，且难以对这些变化资源的边界给予明确地界定，体现环境的开放、动态、难控等特点，因此软件运行环境的变迁也是驱动网络化软件自适应演化的动因之一.

2.3　软件应用模式的变更

早期，软件的应用模式很单一，主要是针对计算机本身而设计一款软件，应用范围很窄，例如：引导操作系统启动、命令形式的简单算术问题等，软件基本上属于自编自用应用模式. 后来随着计算机开始商用化，软件通常被刻在软盘、光盘上进行买卖，例如，Windowxp、瑞星杀毒等，那个时期软件属于我编他用的应用模式. 随着开源世界的到来，软件作为一种资源开始被人们所共享，例如，在开源社区你会找到计算器软件很多版本，可以对其修改后还可以上传供大家使用，这个时期的软件开始走向了共享使用的应用模式. 近年来，智能终端的不断普及，人们可以通过网上在线购买 App 软件，这标志着软件在线购买应用模式. 随着普适计算、网格计算、服务计算、云计算的发展，人们更希望将互联网当作一个大型的软件，只需要通过点击浏览器，就可以获取自己所需的软件服务应用模式，例如：网上租用存储空间，科学实验网上租用高性能计算服务等，软件的应用发展到了租用模式. 现在随着大数据和挖掘技术的发展，将来有可能促使软件根据挖掘到的用户偏好规律、行为规律、作息规律等发展到软件私人订制应用模式. 然而以上的一系列应用模式改变促使了网络化软件应该具备自适应演化的能力，例如：开源社区的 Linux 操作系统在进行剪裁时是否可以符合硬件环境的各项配置，需要软件具有自我检测能力等. 因此，应用模式的改变也是促使网络化软件应具备自适应动态演化能力的原因之一.

综上所述，正是由于需求的变化、运行环境的变迁、应用模式的变更使得网络化软件面临着诸多与传统软件不同的技术挑战. 然而自适应动态演化技术为系统化地解决这些基本问题提供了一条可行的技术路径.

3　网络化软件自适应动态演化的技术进展

网络化软件促使了软件演化由人为参与的被动式演化转换成主动式无人参与的演化. 主动性的自适应演化通常包括五个方面要素：需求、情境、感知、决策、演化，这五个方面要素相互关联，通过前馈和反馈技术形成一个闭环机制相互影响[5]，如图 1 所示. 但是由于各要素的目标追求不一，采用的描述技术方法不一，大多数研究成果都是交错在一起没有特别明显的区别. 针对需求捕捉、情境建模、感知操作、决策方式、演化操作这几个方面，学者们开展了大量研究工作.

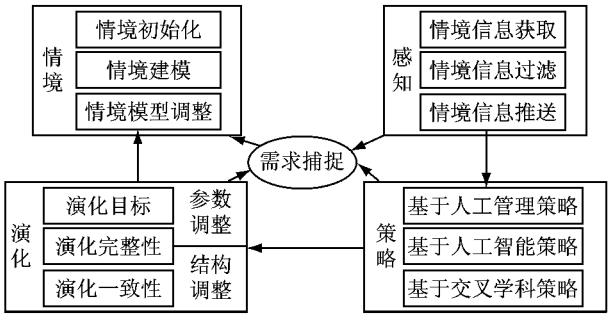

图 1　自适应动态演化五大要素的内容和关系

Fig.1　The content and relation to five key elements of adaptive dynamic evolution

3.1　自适应动态演化的需求捕捉

传统的需求捕捉方法主要有访谈、调查问卷、观察和业务档案研究等. 这些方法具有动态地与访谈者进行交互、调查用户的多元化、亲身体会用户的工作业务过程的优点，但是具有反馈延时很长、准确度不高、触犯一些单位或者个人隐私等缺点. 现在的需求捕捉方法是软件原型法，群体头脑风暴，联合应用开发等. 这些方法具有可视化环境便于沟通、需求一致性达成、群体协作重点突破一些重大问题的优点，但是具有成本高、费时、不易召集相关人员的缺点.

然而这些需求捕捉方法都存在一些客观的不足，导致了一些问题的产生：用户参与度低、协商程度不够全面、需求不一致和不完整、用户与开发人员背景知识存在鸿沟等. 针对这些问题，研究者又相应的提出了各种解决方案，用于补充需求捕捉技术. 针对用户参与度低，主要有使用模式的方法诱导用户的需求，使用情景协同的方式诱导用户需求，使用问

题驱动场景的协同诱导需求方法等.针对协商程度不够全面,主要有使用基于个性领域知识场景协同的需求捕捉方法,使用四维及多维的交流协商框架方法,以及 WinWin 协商模型方法等.针对需求不一致和不完整问题,主要有使用面向领域知识的需求不一致验证及管理框架方法,多视点集成的方法,以及本质用例引导用户逐步完善和精化需求方法等.针对用户与开发人员背景知识存在鸿沟问题,主要有通过领域知识的相互渗透来消除用户与开发人员的知识鸿沟到达获取需求的目的方法,通过上下文感知来消除两者之间的知识鸿沟来捕捉需求的方法,以及本体自动捕捉需求来消除两者之间的鸿沟方法等[15].然而网络化软件更加强调的是自主性,因此这些需求捕捉技术也不能很好地满足网络化软件自适应的要求.

因此,面对网络化软件用户呈现出扁平性、广泛性,研究人员又提出了各种技术组合的需求捕捉方式.例如,将目标分类技术和建模技术结合起来捕捉需求;将目标和本体技术结合自动捕捉需求;将本体、机器学习技术、数据挖掘引入智能需求捕捉技术中;将社会网络技术和需求工程技术结合用于广泛用户需求捕捉;将用户评论和特征挖掘技术用于高时效性的需求捕捉;将云计算、服务计算、社会计算等与控制论结合起来用于自动控制需求捕捉等[13,15-16].各种组合技术不断出现,但是由于网络环境中软件需求本身的不确定性成分太多,这些需求捕捉技术只能是片面的或者是某个时间某个领域对网络化软件自适应演化技术适用而已,通用的自适应网络化软件需求捕捉技术还有待进一步探索.

3.2 自适应动态演化的情境建模

当前对情境的定义并没有统一的认识,然而 Dey 等人[17]所描述的情境定义被广为借鉴:"情境是用于表示实体状态特征的一切信息,其中,实体指的是人、机、以及人机交互相关的所有客体".如今,情境信息已经在各类应用系统得到广泛的应用,但是关注的内容不一样.从整体上来看,情境信息主要包括时间、位置、设备类型、设备状态、周围环境(包括天气、季节等)、情绪、意图等,还有些系统可能包括更广泛的情境信息,如计算平台、社会网络、通信平台、存储平台等.然而网络化软件自适应动态演化主要关注的情境信息是什么,如何对情境信息建模是本文所重点关注的问题?其实情境对于网络化软件来说是软件实体进行演化的环境或平台的一种抽象表示,它不仅包括系统本身的一些静态信息,还包括运行时刻的动态信息以及一些非功能性方面的需求信息,至今在网络化软件自适应动态演化方面并没有系统的情境模型,但是很多研究人员在一些应用领域已经使用了不同方法对情境建立了模型值得借鉴,如下描述.

情境建模是情境信息的数据表达和组织形式,是获取网络软件动态运行状态的关键技术,其发展经历了几个重要过程:理解模型、交互模型、层次模型、推理模型和本体模型.理解模型主要是通过符号、向量和图等形式表述情境信息,如 Adomavicius 描述的基于图的情境建模语言(CML)[18]等,这种模型优点简洁、直观、易于展开数学计算,但是人为操作因素过多、对情境交互缺少统一的格式.交互模型主要通过标记语言表示,如 Knappmeyer[19]提出的 ContextML 等,这个模型的优点简单、易理解,但是缺乏严格的逻辑验证.层次模型主要通过树结构实现,如 Stefanidis 等人[20]用树模型来表示对象情境和实例情境之间的逻辑从属关系,这个模型的优点层次从属关系明确、范围大小清晰,缺点是随着树深度增加复杂度不断增加.推理模型通过将情境信息用逻辑语言或推理机表示,如 Riahi[21]等人将 XML 和 Petri 结合表示情境信息,通过 XML 的语义和 Petri 的过程进行语义推理.这个模型的优点具有严格的推理过程,缺点就是不易于理解.本体模型主要通过知识系统进行情境信息描述,具有代表性的研究是 CoBrA 项目[22],具有易推理、易共享、易重用的优点,这也是目前情境建模的热点之一.

当前已经有支持软件的自适应动态演化系统,但是对情境建模方面有明显的缺陷.例如:CMU 的 Garlan 等人提出的 Rainbow[23]是一种层次模型建立的自适应动态演化软件系统,但是情境信息是隐式描述的.K-Component 的框架[24]由都柏林大学的 Dowling 等人提出,情境信息由感知部件直接获取,但是没有专门管理历史情境信息的部件.ArchStudio 系统[25]由加州大学欧文分校的 Taylor 等人开发,情境信息被体系结构语言隐式的描述,在高层的体系结构映射到底层的运行对象导致情境信息不明确.Arch-Ware 项目[26]是由欧盟资助的,情境信息没有显示刻画,情境信息分散,难于管理情境信息,导致后期的扩展相当困难.MADAM[27]情境信息被中间件封装在实例对象中,提取情境信息增加了难度.Artemis-MAC[28]由南京大学马晓星等人开发,主要通过 RDF 和 OWL 技术对情境信息给予建模,但是该平台在情境信息的冲突检测方面并没有

提及. 由上分析可知，对于网络化软件自适应动态演化技术来说，情境建模的好与坏决定了自适应动态演化的扩展性和通用性.

3.3　自适应动态演化的感知操作

网络化软件情境信息的感知操作主要包括三个环节：情境信息的获取、情境信息的过滤推荐、情境信息的管理.

情境信息的获取是网络化软件自适应动态演化关键的一步，因为能否准确获取情境信息对决策方式的影响很大. 当前对情境信息的获取主要有三种方式[29]：①显示获取. 主要是通过传统的物理设备以及现在的各类传感器感知、人工的用户主动问询、机器轮询等方式，可以直接获得各种关注的显示情境信息. ②隐式获取. 主要有时间匹配法、位置匹配法、归纳法、相似法、分类法等，通过已有的情境信息间接地获取部分更详细的情境信息，例如可以通过直接获取方式获取系统中的日志文档情境信息，然后通过日志文档间接归纳法获取更详细的某用户的时间情境信息. ③推理获取. 主要是对一些显示或隐式获取的情境信息，需要用挖掘技术、统计学、机器学习等方法进行推理才能获得更高层次的情境信息. 这三种方式相铺相成的，因为显示获取方式虽然能够得到很精确的情境信息，但是这些情境信息有可能并没有实用价值，需要隐式获取或者推理获取才能获得更加有意义的情境信息. 因此将这些情境信息的获取技术进行组合或者改进对网络化软件的自适应演化技术可以提供有效的支持.

情境信息的过滤推荐是为了决策阶段获取更加精确有效的信息，同时减少无用信息的干扰，经过主动推荐提高精度. 当前从信息的角度来看，通常使用的过滤推荐方法有三种：协同过滤法、基于内容过滤法、混合式过滤法. ①协同过滤法. 主要是利用“群体智慧”的思想，根据当前事件/实体或者其他事件/实体对部分情境信息的已知偏好来预测当前事件/实体对其他情境信息的偏好；或者利用部分事件/实体对当前情境信息或者其他情境信息的已知偏好来预测其他事件/实体对当前情境信息的潜在偏好，这一技术已经得到了广泛的应用. 例如：Shin 等人[30]根据余弦相似度理论来度量某实体当前情境信息与历史情境信息的相似度，然后联合历史情境实体的偏好来预测当前情境的实体偏好完成协同过滤推荐. 现在还有一些利用机器学习（如支持向量机）或者统计相关模型（张量分解）来改进协同过滤技术. 这种方法的优点在于充分利用“群体智慧”思想，知识面丰富；缺点就是冷启动、高维数据处理难度大、稀疏性等. ②基于内容的过滤. 主要通过挖掘实体在不同的情境下对不同具体实例属性的偏好，然后根据每个具体实例的属性描述计算实体、实例、情境之间的匹配度（或相似度、概率），从而预测实体偏好，完成情境信息的过滤推荐. 该技术的关键在于实例属性特征的描述和计算匹配度的方法，实例属性特征的描述当前已有的方法为：如 TF-IDF 法、聚类、贝叶斯网络等分类法；计算匹配度的方法主要有：杰拉德（Jaccard）相似系数、余弦相似度、皮尔逊（Pearson）相关系数、曼哈顿（Manhattan）距离等. 这种方法的优点在于可以利用成熟的分类方法，过滤推荐的结果简单、直观、易理解；缺点是内容情境信息受限、情境信息的有效性受限于分类方法的好坏等. ③混合式过滤法. 这种方法主要是通过各种不同的混合策略（如串联、加权、扩充、层次叠加、特征组合等）将各种不同的过滤方法组合完成过滤推荐. 例如：Wang 等人[31]将协同过滤原型和基于情绪特征的协同过滤方法混合，各取优点完成情境信息的过滤推荐. 这个方法的优点在于可以借鉴集成学习理论，用共同学习的方法提供推荐精度；缺点在于会引入很多组合参数，增加了计算复杂度. 因此这些情境信息的过滤推荐技术各有利弊，都在一定程度上对网络化软件的自适应演化技术提供支持，但是并不完备.

情境信息的管理在当前的网络化软件自适应演化中并没有得到过多的重视，但其也是情境信息方面的一个难点. 情境信息的管理主要包括情境信息的冲突、历史情境信息的管理、情境信息的缓存管理三个方面. ①情境信息的冲突. 情境冲突主要是来源于情境信息获取并非严格的进行分类、或者是情境信息定义的范围划界不明确、或者是获取过程中出现偏差造成的，通常现在情境信息的冲突解决方法有两个方面. 一种就是利用非形式地方法通过编写冲突检测的逻辑程序进行检测；另一种方法就是在情境建模过程中用严格的逻辑推理进行规避. ②历史情境信息的管理. 历史情境信息通常作为当前的情境信息的一个参照、或者模仿信息，在进行偏好相似度方面有着很重要的地位，但是往往由于设备的存储空间有限，历史情境信息并不是能够完全保留，制作一套高效的历史情境信息管理机制显得相当重要. ③情境信息的缓存管理. 在网络化软件的分布式环境中，情境信息的缓存是为了提高响应、支持实时性，如何从高层态势的情境信息中选取至关重要的

情境信息构成缓存机制也显得很重要.

当前网络化软件自适应演化在感知操作方面相当薄弱,很多已有的自适应演化系统在感知操作所涉及的三个环节并没有考虑完全,只是注重其中的一个或两个环节. 如 Rainbow[23] 系统、K-Component[24] 的框架、ArchStudio 系统[25]、Arch-Ware 项目[26] 和 MADAM[27] 都只是分别通过 Probe-Gauge、Monitor、Bus-sensor、Monitor、Distribute-Sensor 进行情境信息的获取,而 Artemis-MAC[28] 和李青山等人[32] 描述的情境信息都是由 Agent 进行获取的,但是在情境信息过滤和情境信息管理方面并没有提及.

3.4 自适应动态演化的决策方式

策略是以指导方式对软件的动态行为进行管理和控制,人们将管理逻辑通过策略的形式独立出来就是为了更好地利用自适应演化过程的知识. 决策作为自适应动态演化技术中比较重要的一个环节,软件实体可以根据预先制定的策略进行决策,也可以通过人工智能技术和交叉学科相关技术实现决策[33]. 当前对策略的研究已经有很多研究成果,但是对网络化软件自适应动态演化技术提供支持的主要有基于策略管理技术、基于人工智能的策略技术、基于交叉学科的策略技术等.

在基于策略管理方面:主要是通过策略语言和策略冲突两个角度进行管理. 策略语言角度,通常以比高级语言更加简单的形式对策略进行描述,且以动态运行解释为主. 例如,IBM 的自主计算平台 PMAC 使用基于 XML 语言的 ACPL (Automatic Computing Policy Language)来描述策略,以此对具有标签的策略进行管理[34]. 策略冲突角度,由于多条策略同时执行可能会导致不可预期的结果,因此要对策略进行冲突管理,如 Rainbow 中使用事件匹配机制来对策略的优先级区分[23].

在基于人工智能的策略方面:软件自适应动态演化研究的人工智能策略主要是从三种技术进行开展:Agent 技术、基于用例推理和强化学习技术. Agent 技术由于具有自适应的内部组织结构和自配置、自优化、自保护、自修复等特点,较适合自适应动态演化机制,例如李青山等人[32] 就是通过改进合同网的 Agent 协作策略完成基于 Agent 的自适应动态演化机制. Rani 等人[35] 将 Agent 和本体技术结合用于完成 web 软件的自适应动态演化的云存储机制. 基于用例推理主要思想是从过往成功的策略中获取经验,在近年来也得到了广泛的发展,例如 ArchStudio 系统[25] 就是使用基于用例推理的一个成功例子. 强化学习主要是将学习作为一种试探与评价,通过不间断地与情境信息进行交互获取相关知识从而更好的改进策略,例如 *K*-Component[24] 的框架通过合作强化学习实现在分布式环境下的协同策略进行自适应演化. 其他的一些自适应人工智能策略主要有基于规划技术、基于生物编程技术和基于模糊逻辑等技术.

在基于交叉学科的策略方面:网络化软件自适应动态演化策略相关的交叉学科主要有仿生学、控制论、决策论、博弈论等. 仿生学主要通过对生物的个体行为和群体行为特征进行策略的模仿,例如 Mckinley 等人[36] 通过模仿生物进化行为的过程来构造自适应进化的优胜劣汰策略. 在控制论中,主要是运用已有控制论的已有知识构造策略,例如 Chen 等人[37] 将模型检测器作为控制器,软件体系结构模型作为控制目标,构造一个实时性的知识决策器,不断地优化动态软件体系结构模型. 决策论主要是运用概率理论和效用理论这两方面的知识,例如,POISED[38] 是通过概率论来在线评估策略;博弈论方面,主要是通过经济领域的博弈思想引入软件自适应动态演化中,例如 Ye 等人[39] 将博弈论思想的引入自适应分布式网络交互操作演化模型上,利用囚徒困境模型构造互操作的自适应优化策略.

3.5 自适应动态演化的演化操作

网络化软件的自适应演化的操作主要体现在参数调整和结构调整[33]. 参数调整通常表现为不对软件具体实现进行修改,只是对变量、属性等进行重配置,从而完成软件行为的改变;结构调整主要体现在需要对软件的具体实现进行修改,如替换代码和模块等,从而实现软件行为的改变.

参数调整的自适应演化操作主要的方法就是通过参数的动态配置完成. 例如,Boyapati 等人[40] 提出了一种中断模型的参数动态配置方法,主要在某个点进行中断,等待这个点之后的所有代码都进入了不活跃状态,参数更新配置才开始,完成后返回且恢复系统正常运行,该方法缺点是不支持多线程. Kang 等人[41] 提出一种调用模型的参数动态配置方法,首先在某个点先进行通知,过了一段时间运行到更新点才开始进行参数的更新配置,完成后返回且恢复系统正常运行,该方法支持多线程. Chen 等人[42] 提出一种松一致模型的参数动态配置方法,它是中断模型的一个扩展,它不同之处在于更新完毕后,新老进程可以并存,等待所有的新老进程都达到某个点

时，才完全执行新进程等.

结构调整的自适应演化操作涉及的方法有：基于构件的动态配置，动态体系结构，服务动态组合，代码动态迁移，动态方面编程. ①基于构件的动态配置. 构件的动态配置支持比函数、对象更大粒度的软件重用，基本的演化操作是构件的替换、增加、删除. 随着容器和构件的接口不断地标准化，基于构件开发的软件可以通过构件的演化操作完成对用户需求和硬件环境等因素的自适应. 例如，微软公司的开发的 COM/DCOM/COM+系列构件模型，Sun 公司提出的 EJB 系列，OMG 提出的 CORBA 构件模型，是通过容器管理构件的替换、增加、删除操作，通常是用恢复性方法、避免性方法、通用动态配置框架法完成系统一致性. 恢复性方法，记录多个检查点，当演化操作发生时破坏了一致性，就回滚到特定的检查点. 避免性方法，首先由配置算法方案检测可能影响的范围，然后冻结响应范围的构件行为，完成演化操作后恢复正常运行. 通用动态配置框架方法，基本思想是每个构件都配置一个控制器，且存储有构件当前的状态和交互约束等. ②动态体系结构. 动态软件体系结构的演化操作依赖于构件、连接件和体系结构配置，连接件是用于构件间的交互和交互规则的设定，体系结构配置是用于确保构件和连接件的交互和拓扑关系. 例如，南京大学的马晓星等[43]使用类型化图文法给出了基于“Master/Slaver”风格的实例系统的增加、删除演化操作规则，由体系结构配置完成结构调整. 我国学者赵会群等人[44]利用代数模型给出了网构体系结构的激发、使用、协同、并行、重复、选择演化操作. Xu[45]等人利用超图文法描述动态体系结构，给出了顺序、并行、拆分、重组等组合演化操作的超图文法演化规则. ③服务动态组合. 服务动态组合的演化操作主要是依赖于服务组合的模型. 例如，曾晋等[46]从服务组合的角度给出了服务的替换、增加、删除和结构调整 4 种演化操作，给出了一个组合服务流程结构的动态在线迁移机制. 曾国荪等人[47]基于条件 Pi 演算扩展给出了插入、删除、迁移、替换、并行、交换、增加依赖、去除依赖、更新条件、嵌入选择分支、嵌入循环的服务演化操作，形成了一个组合服务柔性演化模型. ④代码的动态迁移. 代码的动态迁移主要是根据周围运行环境变化时，软件通过迁移部分代码或者模块以适应环境，完成软件演化. 当前这种技术主要运用在云分布式环境中，例如 Zhang 等人[48]将单个移动云计算的任务自动地划分为多个 Weblet 构件，然后根据当前的运行环境状况自适应动态的分配 Weblet 构件是在云端还是在移动端执行等. ⑤动态方面编程. 动态方面编程技术是将业务逻辑的公共代码(即横切关注点)通过预定义截获点、代理设计等手段动态的植入正在运行的软件系统中，也属于自适应演化的范畴，且已经成为了自适应领域的一种重要的发展方向. 例如 Machta 等人[49]利用动态方面编程与 UML 结合为遗产软件系统增加实时、安全等非功能方面的自适应机制.

在完成各类演化操作后，如何保证系统的一致性和完整性使其不偏离原有系统也是软件演化发生后必须关心的问题. 首先，一致性在这里是指系统演化前后结构和状态的一致性. 为了保证这类一致性，已经有很多学者从建模的角度提出了各种方法，图文法、Petri-Net、自动机、可靠性度量模型等，例如，徐洪珍等人[50]构造了一种基于条件超图文法的系统一致性判定方法. 周宇等人[51]从行为的角度采用层次式时间自动机以及模型检测工具对演化约束规则进行一致性验证. 其次，完整性指的是系统演化后不能破坏原有系统规约中的约束. 当前对完整性检测的相关技术，主要集中在反射技术、一阶逻辑、高阶逻辑等. 例如，ArchStudio 系统[25]利用反射技术和推理逻辑保证系统的完整性. Artemis-MAC[28]是基于利用反射技术和一阶逻辑的保证系统的完整性等.

综上所述，将所提及的若干经典自适应动态演化系统在建模语言、情境建模、感知、决策和演化五个方面的使能技术进行比较分析，结果如表 2 所示：

4　网络化软件自适应动态演化技术的挑战和发展趋势

随着网络软件规模不断地增大、组成结构日益复杂，运行环境呈现出开放、动态和难控的局面，致使软件需要自适应动态演化来满足环境和用户需求. 从宏观的角度来看，当前自适应动态演化技术存在诸多挑战，本文试图指出一些发展趋势.

4.1　网络化软件自适应动态演化技术的挑战

早期的软件工程领域开发模型大多数都是基于确定性假设进行开发的，随着网络化软件的出现，这种假设难以成立，不确定性成为了常态，成为了自适应演化的一大挑战. 在软件的自适应动态演化中不确定性主要来源于需求的不确定和环境的不确定. 此外，由于演化数据和资源的分散、多样、异构、泛在

表2 若干经典的自适应动态演化系统比较

Tab.2 Compare to several classic adaptive dynamic evolution system

系统/项目	建模语言	情境建模	感知操作	决策方式	演化操作
Rainbow	形式化	隐式建模	探针感知	策略管理	结构调整为主,参数调整为辅
K-Component	半形式化	隐式建模	监视器	合作强化学习	参数调整为主,结构调整为辅
ArchStudio	形式化	隐式建模	总线传感器	基于用例推理	参数调整为主,结构调整为辅
Arch-Ware	形式化	隐式建模	监视器	策略管理	参数调整为主,结构调整为辅
Artemis-MAC	半形式化	显示建模	Agent 感知	一阶谓词推理	结构调整为主,参数调整为辅
MADAM	半形式化	隐式建模	上下文感知	效用理论	参数调整为主,结构调整为辅

等原因,导致了演化数据和资源的安全性不能得到保证,这也是自适应演化的又一挑战.

(1)网络化软件超常复杂

美国卡耐基梅隆大学软件工程研究所在2006年发表题为《超大规模系统:软件未来的挑战》[52]中指出未来软件在代码行数、节点数目、数据量等都是千万级甚至更高量级的,其超大规模性是现在软件系统无法比拟的.网络化软件作为未来软件的一种形态,规模激增已经在各种维度的复杂性上引起内在变化[53]:例如,用户需求和目标的复杂性;互联网环境多元素的复杂性;元素之间的交互复杂性;元素组成员边界划分的复杂性;去中心化的非集中式管理复杂性等.因此,面对规模剧烈增长、复杂性不断增加、持续不断演化的网络化软件,需要寻找新的观念、新的思想、新的技术来应对这些问题.

(2)需求的不确定性

需求的不确定性指的是用户需求的不确定性,导致需求不确定有几个原因.首先,认识带来的需求不确定.需求是用户对客观世界的描述,但是人们认识客观世界的能力受限于知识和经验,即对客观世界得认识是不完全的,因此导致了用户需求的不确定性.其次,变化带来的不确定性.客观世界是变化的,因此需求规格也会随着客观世界的变化而变化,随着普适计算、网格计算、服务计算、云计算新型计算的出现,用户需求变化程度愈加强烈,表现出来的不确定性更加广泛.

(3)外部环境信息的不确定性

在开放、动态的背景下的系统所处的运行的环境是具有极其高的不确定性和动态性.由于在开发阶段,网络软件很难精确地预测可能要面对什么样的环境,特别是面对分布式环境的影响因素过多,且因素的动态性太强,当前网络化软件面对着环境不确定性,主要的方法就是对环境进行实时地监控,然而基于监控的技术往往只是针对某些固定因素进行监控,对那些瞬时出现的环境信息无法捕捉,该技术并不是能够很好地支持环境信息不确定性,急需新的理论、方法和技术手段叠加或者创建给予支持.

(4)演化数据和资源的不可信

当前在开发环境下参与自适应动态演化的构造单元或者数据种类繁多、来源不一,特别是开源软件库主导的分布式软件的可信性面临着巨大的挑战.我国学者吕建等人[5]指出建立可信管理和可信评估是演化数据和资源得以重要前提.我国学者王怀民等人[54]指出了参与演化的数据和资源的多元性、分散性等原因是演化安全性隐患的重要因素.网络化软件部署在泛化环境中,参与的演化资源和数据的来源更是多样化、不可预测等特点,因此演化数据和资源的不可信性是网络化软件自适应演化的一个重大挑战.

(5)自适应演化理论方法不完善

由于软件自适应演化本身是一个外延很广泛的技术,它的实现涉及软件工程、复杂系统、计算机网络、人工智能、生物学、社会组织学等多个学科,传统的软件演化技术理论方法已经无法胜任.如今已有的动态演化、在线演化的理论方法对自适应技术的支撑只是针对某个特定领域,并没有形成系统化的理论、方法、工具和基本的支撑设施.自适应演化作为网络化软件生命周期内持续演化的重要技术,没有完整的理论方法支撑将会严重阻碍网络化软件的发展.

4.2 网络化软件自适应动态演化技术的发展趋势

针对网络化软件自适应动态演化技术面临的各方面挑战,网络化软件自适应动态演化技术的发展趋势有如下几个方面:

(1)弹性的网络化软件体系结构模型.随着网络化软件规模和复杂性不断地增加,开发可自适应演化的网络化软件系统遇到了严重的阻碍.最大的阻碍之一就是无法将不同种类的原系统和软件模型清晰地区分开,然而软件体系结构在开发过程中扮演着全局视图的重要角色,创建弹性的网络化软件体系结构模型是解决该问题的方法之一.

(2)基于大数据挖掘的演化诱因捕捉.当前引起网络化软件演化的动因主要为用户需求和环境变化,然而这些变化由人为的观察只能分析表面的原

因并不能正确地把握这些变化的规律，大数据技术的发展却可以为挖掘事件变化深层的规律提供了支持，因此为了延长软件生命周期，发展基于大数据的网络化软件演化动因的捕捉势在必行.

(3)基于深度学习的多目标演化决策技术. 在网络化软件中，由于用户多需求的特点，导致软件系统多种不同的功能、性能属性演化时发生冲突，缓解这种冲突的方式显然为多目标演化决策的过程，由于认知技术具有记忆功能，可以清楚地掌握不同功能和性能在不同需求情况下的权重，可以为这种多目标演化决策的过程提供强有力的保证.

(4)基于行为和内容可信的演化管理. 网络化软件演化呈现出演化资源的泛在性和异构性、演化数据的分散性与多样性、演化目标的多元性、演化过程的交错性等给软件的演化安全管理提出重大的挑战，拥有一套可信演化管理的机制才是保证演化正确实施的关键.

(5)基于群体智慧的自适应动态演化优化. 由于自适应动态演化被激发的原因主要来源于用户，当前的大多数自适应演化框架都只是停留在开发人员对用户需求的被动了解，因此让用户主动参与软件演化才是自适应演化的关键，因此基于群体智慧技术对自适应演化框架的优化将成为一种趋势.

5　结束语

随着计算机环境从封闭、静态、可控转向开放、动态、难控的局面，给网络化软件演化提出了新的挑战，从而驱动了建立网络化软件自适应动态演化机制势在必行. 本文从构造自适应动态演化机制这一角度出发，首先介绍软件演化的基本概念；在此基础上归纳出研究网络化软件自适应动态演化的动因；接着就从需求捕捉、情境建模、感知操作、决策技术、演化操作五个方面归纳出网络化软件自适应动态演化的主要研究工作和进展；最后总结分析出网络化软件自适应动态演化技术的挑战和发展趋势，为自适应动态演化的推进提供了方向性指导.

参考文献：

[1] Swanson E B. The Dimensions of Maintenance [C]// Proceedings of the 2nd international conference on Software engineering. Los Alamitos: IEEE Press, 1976:492-497.

[2] Lehman M M. Programs, life cycles, and laws of software evolution[J]. Proceedings of the IEEE, 1980, 68(9):1060.

[3] Perry D E. Dimensions of software evolution [C]// Proceedings of International Conference on Software Maintenance, 1994. Victoria: IEEE Press, 1994:296-303.

[4] 杨芙清，梅宏，吕建，等. 浅论软件技术发展[J]. 电子学报，2002,30(12A):1901.
YANG Fuqing, MEI Hong, LÜ Jian, *et al*. Some discussion on the development of software technology[J]. *Acta Electronica Sinica*, 2002,30(12A):1901.

[5] 吕建，马晓星，陶先平，等. 网构软件的研究与进展[J]. 中国科学 E 辑：信息科学，2006,36(10)：1037.
LÜ Jian, MA Xiaoxing, TAO Xianping, *et al*. Research and development of internetware[J]. Science in China Series E: Information Sciences, 2006, 36(10): 1037.

[6] 梅宏，黄罡，兰灵，等. 基于体系结构的网构软件自适应方法[J]. 中国科学 E 辑：信息科学，2008,38(6):901.
MEI Hong, HUANG Gang, LAN Ling, *et al*. An adaptive method of internetware based on architecture[J]. Science in China Series E: Information Sciences, 2008,38(6): 901.

[7] 陈洪龙，李仁发. 自适应演化软件研究进展[J]. 计算机应用研究，2010,27(10):3612.
CHEN Honglong, LI Renfa. Survey on self-adaptive evolution software[J]. Application Research of Computers, 2010, 27(10):3612.

[8] 李玉龙，李长云. 软件动态演化技术[J]. 计算机技术与发展，2008,18 (9):83.
LI Yulong, LI Changyun. Dynamic evolution technology of software[J]. Computer Technology and Development, 2008,18 (9):83.

[9] 王青，李娟. 互联网对软件演化的挑战[J]. 中国计算机学会通讯，2009,5(12):27.
WANG Qing, LI Juan. The challenge of Internet to Software Evolution[J]. Communications of the CCF, 2009, 5(12):27.

[10] 王怀民，史佩昌，丁博等. 软件服务的在线演化[J]. 计算机学报，2011,34(2):318.
WANG Huaimin, SHI Peichang, DING Bo, *et al*. Online evolution of software services [J]. Chinese Journal of Computers, 2011,34(2):318.

[11] 王怀民，吴文峻，毛新军等. 复杂软件系统的成长性构造于适应性演化[J]. 中国科学 E 辑：信息科学，2014,44(6):743.
WANG Huaimin, Wu Wenjun, Mao Xinjun, *et al*. Growth Construct and adaptive evolution of complex software systems [J]. Science in China Series E: Information Sciences, 2014,44 (6):743.

[12] Aversano L, Brino M D, Guardabascio D, *et al*. Understanding enterprise open source software evolution[J]. Procedia Computer Science, 2015, 64:924.

[13] Lin L, Chen Y, Wang J M, *et al*. Requirements model driven adaption and evolution of Internetware[J]. Science in China Information Sciences, 2014, 57(6):1.

[14] Lehman M M, Ramil J F, Wernick P D, *et al*. Metrics and Laws of Software Evolution - The Nineties View[C]// IEEE International Software Metrics Symposium. Albuquerque: IEEE Press, 1997:20-32.

[15] 湛浩旻. 软件需求获取过程关键技术研究[D]. 哈尔滨：哈尔滨工程大学计算机科学与技术学院，2013.
ZHAN Haomin. Research on key technologies of software requirements elicitation process [D]. Harbin: Harbin Engineering University: School of Computer Science and

Technology,2013.

[16] Jiang W, Ruan H, Zhang L, *et al*. For User-Driven Software Evolution: Requirements Elicitation Derived from Mining Online Reviews[C]// Proceedings of the 18th Pacific-Asia Conference, PAKDD 2014. Switzerland:Springer Press, 2014: 584-595.

[17] Dey A K. Understanding and Using Context[J]. Personal & Ubiquitous Computing, 2001, 5(1):4.

[18] Adomavicius G, Ricci F. RecSys'09 workshop 3: workshop on context-aware recommender systems (CARS-2009)[C]// Proceedings of the 2009 ACM Conference on Recommender Systems, RecSys 2009. New York: ACM Press, 2009: 423-424.

[19] Knappmeyer M, Kiani S L, Fra C, *et al*. ContextML: a light-weight context representation and context management schema [C]// Proceedings of the 5th IEEE international conference on Wireless pervasive computing. Modena: IEEE Press, 2010: 367-372.

[20] Stefanidis K, Pitoura E. Fast contextual preference scoring of database tuples[C]// Proceedings of the 11th International Conference on Extending Database Technology: Advances in Database Technology. New York:ACM Press,2008:344-355.

[21] Riahi I, Moussa F. A formal approach for modeling context-aware human-computer system [J]. Computers & Electrical Engineering, 2015, 44(0):241-261.

[22] Chen H, Finin T, Joshi A. Semantic web in the context broker architecture[C]// IEEE International Conference on Pervasive Computing and Communications. Orlando: IEEE Press, 2004: 277-286.

[23] Huang A C, Garlan D, Schmerl B. Rainbow: Architecture-Based Self-Adaptation with Reusable Infrastructure [J]. Computer, 2004, 37(10):276.

[24] Dowling J, Cahill V. The *K*-component architecture meta-model for self-adaptive software[C]// Proceedings of the Third International Conference on Metalevel Architectures and Separation of Crosscutting Concerns. Berlin: Springer-Verlag, 2002:81-88.

[25] Georgas J C, Taylor R N. Towards a knowledge-based approach to architectural adaptation management [C]// Proceedings of the 1st ACM SIGSOFT Workshop on Self-managed Systems. New York:ACM Press, 2004:59-63.

[26] Oquendo F, Warboys B, Morrison R, *et al*. ArchWare: architecting evolvable software[C]// Proceedings of the 1st European Workshop on Software Architecture. Berlin Heidelberg:Springer-Verlag, 2004:257-271.

[27] Floch J, Hallsteinsen S, Stav E, *et al*. Using architecture models for runtime adaptability[J]. IEEE Software, 2006, 23 (2):62.

[28] 马晓星,余萍,吕建等.一种面向服务的动态协同架构及其支撑平台[J]. 计算机学报,2005,28(4):467.
MA Xiaoxing, YU Ping, TAO Xianping, *et al*. A service-oriented dynamic coordination architecture and its supporting system[J]. Chinese Journal of Computers, 2005,28(4): 467.

[29] 王立才, 孟祥武, 张玉洁. 上下文感知推荐系统[J].软件学报, 2012,23(1):1.
WANG Licai, MENG Xiangwu, ZHANG Yujie. Context-aware recommender systems[J]. Journal of Software, 2012,23(1): 1.

[30] Shin D, Lee J W, Yeon J, *et al*. Context-aware recommendation by aggregating user context[C]// IEEE Conference on Commerce & Enterprise Computing. Vienna: IEEE Press, 2009:423-430.

[31] Wang L, Meng X, Zhang Y, *et al*. New approaches to mood-based hybrid collaborative filtering[C]// The Workshop on Context-Aware Movie Recommendation. New York: ACM Press, 2010:28-33.

[32] 李青山,王璐,褚华,等.一种基于智能体技术的软件自适应动态演化机制[J].软件学报,2015,26(4):760.
LI Qingshan, WANG Lu, CHU Hua, *et al*. Agent-based software adaptive dynamic evolution mechanism [J]. Journal of Software, 2015,26(4):760.

[33] 丁博,王怀民,史殿习.构造具备自适应能力的软件[J].软件学报,2013,24(9):1981.
DING Bo, WANG Huaimin, SHI Dianxi. Constructing software with Self-adaptability[J]. Journal of Software, 2013,24(9): 1981.

[34] Agrawal D, Lee K W, Lobo J. Policy-based management of networked computing systems [J]. IEEE Communications Magazine, 2005, 43(10):69.

[35] Rani M, Nayak R, Vyas O P. An ontology-based adaptive personalized e-learning system, assisted by software agents on cloud storage[J]. Knowledge-Based Systems, 2015, 90(C): 33.

[36] Mckinley P K, Cheng B H C, Ofria C . Applying digital evolution to the development of self-adaptive ULS systems [C]// International Workshop on Software Technologies for Ultra-Large-Scale Systems. Minneapolis: IEEE Press, 2007: 3.

[36] Chen L, Huang L, Li C, *et al*. Design and safety analysis for system architecture: a breeze/ADL-based approach [C]// Computer Software and Applications Conference. Vasteras: IEEE Press, 2014:261-266.

[38] Esfahani N. Management of uncertainty in self-adaptive software[J]. Dissertations & Theses—Gradworks, 2014, 14 (3):234.

[39] Ye D, Zhang M. A self-adaptive strategy for evolution of cooperation in distributed networks[J]. IEEE Transactions on Computers, 2015, 64(4):899.

[40] Boyapati C, Liskov B, Shrira L, *et al*. Lazy modular upgrades in persistent object stores[J]. Acm Sigplan Notices, 2003, 38 (11):403.

[41] Kang L, Cao D. An extension to computing elements in erlang for actor based concurrent programming[C]// 2012 IEEE 15th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops. Shenzhen: IEEE Press, 2012:99-105.

[42] Chen H, Yu J, Hang C, *et al*. Dynamic software updating using a relaxed consistency model[J]. IEEE Transactions on Software Engineering, 2010, 37(5):679.

[43] 马晓星, 曹春, 余萍,等. 基于图文法的动态软件体系结构支撑环境[J]. 软件学报, 2008, 19(8):1881.
MA Xiaoxing, CAO Chun, YU Ping, *et al*. A supporting environment based on graph grammar for dynamic software architecture[J]. Journal of Software, 2008, 19(8):1881.

[44] 赵会群，孙晶. 网构软件体系结构代数模型[J]. 中国科学 E 辑：信息科学，2013，43(1)：161.
ZHAO Huiqun，SUN Jing. Algebraic model of internetware architecture[J]. Science in China：Series E：Information Sciences，2013，43(1)：161.

[45] Xu H Z，Zeng G S. Modeling and verifying composite dynamic evolution of software architectures using hypergraph grammars[J]. International Journal of Software Engineering and Knowledge Engineering，2013，23(6)：775.

[46] 曾晋，孙海龙，刘旭东等. 基于服务组合的可信软件动态演化机制[J]. 软件学报，2010，21(2)：261.
ZENG Jun，SUN Hailong，Liu Xudong，*et al*. Dynamic evolution mechanism for trustworthy software based on service composition[J]. Journal of Software，2010，21(2)：261.

[47] 刘涛，曾国荪. 一种基于条件 *Pi* 演算的组合服务柔性演化模型[J]. 计算机科学，2011，38(10A)：230.
LIU Tao，ZENG Guosun. Model for flexible evolution of composite services based on conditional *Pi* calculus[J]. Computer Science，2011，38(10A)：230.

[48] Zhang X，Kunjithapatham A，Jeong S，*et al*. Towards an elastic application model for augmenting the computing capabilities of mobile devices with cloud computing[J]. Mobile Networks & Applications，2011，16(3)：270.

[49] Machta N，Bennani M T，Benahmed S. NOLE：an AOM weaver for aspect oriented modeling of real-time system[J]. Procedia Computer Science，2015，46：742.

[50] 徐洪珍，曾国荪，陈波. 软件体系结构动态演化的条件超图文法及分析[J]. 软件学报，2011，22(6)：1210.
XU Hongzeng，ZENG Guosun，CHEN Bo. Conditional hypergraph grammars and its analysis of dynamic evolution of software architectures[J]. Journal of Software，2011，22(6)：1210.

[51] 周宇，黄延凯，黄志球等. 一种开放环境下软件在线演化一致性验证方法[J]. 软件学报，2015，26(4)：747.
ZHOU Yu，HUANG Yankai，HUANG Zhiqiu，*et al*. Towards an approach of consistency verification for online software evolution in open environments[J]. Journal of Software，2015，26(4)：747.

[52] Northrop L，Feiler P，Gabriel R P，*et al*. Ultra-large-scale systems—the software challenge of the future[J]. Software Engineering Institute，2006，5(2)：297.

[53] 马于涛，何克清，李 兵等. 网络化软件的复杂网络特性实证[J]. 软件学报，2011，22(3)：381.
MA Yutao，HE Keqing，LI Bin，*et al*. Empirical study on the characteristics of complex networks in networked software[J]. Journal of Software，2011，22(3)：381.

[54] 王怀民，尹 刚. 网络时代的软件可信演化[J]. 中国计算机学会通讯，2010，6(2)：28.
WANG Huaimin，YUN Gang. Software credible evolution in the network era[J]. Communications of the CCF，2010，6(2)：28.

·下期文章摘要预报·

高强度 U 肋加劲钢板残余应力测试及模拟分析

肖维思，王　佳，刘玉擎，黄李骥

为研究 Q420 级高强度 U 肋加劲钢板纵向焊接残余应力分布特点及影响因素，利用切割法对 U 肋加劲钢板进行了纵向残余应力测试，通过三维实体热弹塑性有限元模型和单元生死技术模拟了焊缝填充和焊接过程，比较分析了高强度钢和普通强度钢的残余应力分布特点，探讨了母板厚度及 U 肋的厚度、间距、宽度、高度对加劲板焊接残余应力的影响. 研究结果表明，U 肋两侧的焊接先后顺序并不影响加劲板的残余应力分布；非焊接区域残余压应力峰值和分布特点与板件材料的屈服强度基本不相关；板件厚度、U 肋顶宽和 U 肋高度是影响高强度 U 肋加劲钢板焊接残余应力的主要因素.