科学通报
评 述
《中国科学》杂志社
SCIENCE CHINA PRESS

网构软件研究: 回顾与展望

梅宏[1,2*], 黄罡[1,2], 刘譞哲[1,2], 张颖[2]

1. 北京大学计算机学院, 北京 100871;
2. 高可信软件技术教育部重点实验室(北京大学), 北京 100871
* 联系人, E-mail: meih@pku.edu.cn

2022-05-23 收稿, 2022-07-17 修回, 2022-07-20 接受, 2022-07-21 网络版发表

摘要 软件作为计算系统的“灵魂”, 是信息技术产业的核心组成部分, 也正在成为数字经济时代支撑社会运行的重要“基础设施”. 进入21世纪以来, 互联网及其延伸网络形成了开放、动态、难控的复杂计算环境, 对传统的软件范型与技术提出了巨大挑战, 也蕴含了技术体系重大变革的新机遇. 来自北京大学、南京大学和中国科学院的学者于2000年提出“网构软件”这一面向互联网的新型软件范型, 经过20余年的努力, 在其基本模型、开发方法、运行支撑和质量评估与保障等方面, 均取得一系列重要的研究进展, 形成了系统化的创新成果, 产生了广泛的学术和产业影响. 本文从软件范型入手, 以互联网及其延伸网络形成的复杂计算环境为线索, 回顾网构软件的研究进展, 并展望未来的研究工作.

关键词 互联网, 软件范型, 网构软件, 软件体系结构

软件是世界数字化的直接产物、自动化的计算途径、智能化的逻辑载体, 更是人类知识的固化、凝练和体现[1]. 软件技术研究主要涉及4个方面的内容[2](图1): 软件范型(paradigm)、软件构造方法、软件运行支撑及软件质量度量与评估. 其中, 软件范型是从软件工程师或程序员的视角看到的软件模型及其构造原理, 一直是软件技术体系的核心[3]. 软件范型总是不断追求更具表达能力、更符合人类思维模式、更具可构造性和演化性的软件结构, 例如, 结构化方法试图适应人分析问题和解决问题的思维模式, 面向对象方法围绕现实世界的结构来设计问题空间和解空间模型, 更符合人类看待世界的方式. 回顾软件技术发展历程, 软件范型大致经历了无结构、结构化、面向对象、面向构件、面向服务等的变迁. 每次软件范型的变迁, 都会引发软件开发方法和运行支撑机制的相应变化, 并导致新的软件质量度量评估方法和保障技术的出现[1].

在计算技术发展的相当长一段时期, 软件技术研究主要针对单机系统, 软件范型从无结构演变到结构化和面向对象, 为控制软件解空间和问题空间的复杂性提供了有效机制和手段; 软件开发技术则主要围绕相应的软件范型, 研究大型软件系统高质量高效率开发的方法、技术和工具; 在运行支撑方面, 面向单机的操作系统主要随着计算机硬件的发展而进步, 追求更高效地发挥底层硬件资源所提供的计算能力, 不断凝练应用共性并沉淀为操作系统的固有成分, 同时更好地满足用户对易用性的需求; 在软件质量度量、评估和保障方面, 则主要关注系统质量[2].

20世纪90年代中期到21世纪初, 互联网(internet)飞速发展和普及, 并逐渐成长为一台由数量巨大且持续增长的计算设备和资源所组成的“全球泛在计算机”(global ubiquitous computer). 与传统计算系统相比, 互联网为应用领域问题求解所能提供的支持, 无论在量还是质上均有飞跃, 呈现出如下基本特征: 无统一控制的“真”分布性, 节点的高度自治性, 节点链接的开放

引用格式: 梅宏, 黄罡, 刘譞哲, 等. 网构软件研究: 回顾与展望. 科学通报, 2022, 67: 3780–3792
Mei H, Huang G, Liu X Z, et al. Research on internetware: Review and prospect (in Chinese). Chin Sci Bull, 2022, 67: 3780–3792, doi: 10.1360/TB-2022-0585

© 2022《中国科学》杂志社

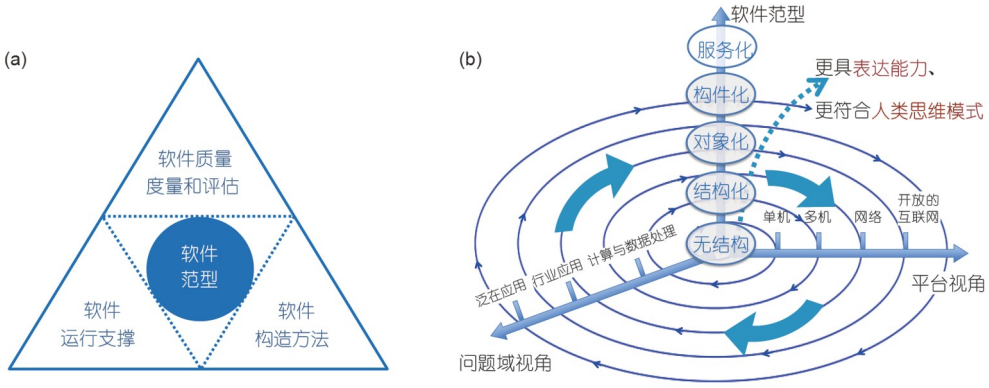


图 1 软件范型及其变迁. (a) 软件技术体系; (b) 软件范型的变迁
Figure 1 Software paradigm and its shift. (a) Software technology architecture; (b) software paradigm shift

性和动态性, 人、设备和软件的多重异构性, 实体行为的不可预测性, 运行环境的潜在不安全性, 使用方式的个性化和灵活性, 网络连接环境的多样性等. 互联网使得计算机软件开发、部署、运行和维护的环境开始从封闭、静态、可控逐步走向开放、动态、难控. 同时, 运行于互联网上的软件应用, 功能也已经从早期面向信息发布与共享的信息web(information web), 演化成为由分布在网络各个节点上的软件实体通过协同交互机制而形成的软件web(software web). 换言之, 这些软件实体通过跨网络的功能和数据互操作, 动态地构成新的系统, 呈现出柔性可演化、连续反应式、多目标自适应的新形态[1].

传统面向单机计算环境的软件范型及其相应的理论、方法和技术, 在本质上是一种静态和封闭的框架体系, 难以适应互联网计算环境开放、动态、难控, 以及应用领域全球化、个性化、持续成长等特点, 面临着一系列重大挑战. 为此, 来自北京大学、南京大学、中国科学院软件研究所等单位的研究人员, 于2000年提出了“网构软件”概念[4], 用以指称一种面向互联网计算环境的新型软件范型. 从形态上看, 网构软件包括一组分布于互联网环境下各个节点的、具有主体化特征的软件实体, 以及一组用于支撑这些软件实体以各种交互方式进行协同的连接子. 这些实体能够感知外部环境的变化, 通过体系结构演化的方法(主要包括软件实体与连接子的增加、减少与演化, 以及系统拓扑结构的变化等)来适应外部环境的变化, 展示上下文适应的行为, 从而使系统能够以足够满意度来满足用户的多样性目标. 一方面, 网构软件是传统软件在互联网环境下的自然延伸; 另一方面, 网构软件也有其独有的基本特征, 如自主性、演化性、协同性、情境性、涌现性和可信性等[2].

在国家重点基础研究发展计划(“973”计划)、国家高技术研究发展计划(“863”计划)、国家重点研发计划、国家自然科学基金、欧洲联盟第七框架等项目支持下, 经过20余年的努力, 网构软件在基本模型、开发方法、运行支撑和质量评估与保障等方面, 均取得了一系列重要的研究进展, 形成了系统化的创新成果. 作为我国学者提出的学术理念, 网构软件引领了我国软件技术研究的新局面, 较好地应对了开放、动态、难控的互联网环境下新型软件模型、开发方法、运行机理、工具与系统等一系列关键问题. 网构软件理念和技术已经得到国内外学术界和产业界的认可, 产生了积极广泛的国际影响. 网构软件的研究成果也已在国家、军队、省市的信息化建设以及国外的大中型信息系统中得到应用, 服务了国家大数据战略、“互联网+”等多个国家重大需求和重点工程, 先后获得了国家技术发明奖一等奖、国家技术发明奖二等奖、国家自然科学奖二等奖、国家科学技术进步奖二等奖, 产生了良好的社会经济效益, 为我国创新驱动发展战略和软件技术及产业发展作出了积极贡献.

本文将回顾网构软件研究, 介绍网构软件理念产生的背景, 围绕软件范型, 以互联网及其延伸网络形成的复杂计算环境为线索, 重点介绍网构软件在开发方法与运行支撑方面的研究进展, 并对未来研究进行展望.

1 网构软件技术体系

从2002年开始, 在国家重点基础研究发展计划连

续两期项目[a)]的支持下，项目组以软件Agent与开放协同模型、软件体系结构和软件中间件作为技术切入点，围绕网构软件在研究对象、构造方法、运行支撑、质量目标4个关注点的转变，即研究对象从产生于相对封闭、静态、可控环境下的传统软件转变为运行于开放、动态、难控网络环境下的网络化复杂软件，构造方法从满足功能需求并保障功能正确性转变为满足质量需求并保障可信度和服务质量，运行支撑从凝练共性应用功能并保证软件正确运行转变为凝练共性管理功能并保证软件可信、高服务质量运行，质量目标的重心从指标相对单一的系统内部和外部质量转变为指标比较综合的以可信度和服务质量为主的使用质量，建立了一套以体系结构为中心的网构软件技术体系[5]，为网构软件研究奠定了理论和方法基础.

1.1 基于Agent、以软件体系结构为中心的网构软件模型

软件模型是软件方法学的核心. 经典、主流的面向对象软件模型，由于其产生于静态、封闭、可控的环境，从而具有有限自主、固定封装、交互单调、结构紧耦、离线应变等特征. 20世纪末至21世纪初，尽管面向对象模型在分布对象、面向构件/服务等方面取得了一些进展，但其特征并没有发生根本性改变，无法直接支持网构软件.

基于Agent、以软件体系结构为中心的网构软件模型[6~9]致力于支撑网构软件自主性、协同性、演化性、情境性、涌现性和可信性等新型特性，提供了如下抽象[6]. 一是对基本实体的抽象，即网构软件基本实体应为独立的构件或服务，以确保单个实体可以独立部署、运行与演化. 与传统软件构件/服务类似，网构软件基本实体也具有业务接口，以提供计算、控制或连接等功能，与传统软件构件/服务不同，网构软件基本实体具有反射接口，以提供监测、分析、规划、调整其业务功能的反射功能，进而实现网构软件特性. 二是对协同能力的抽象，即在基本构件模型之上支持开放协同模型，其核心是给面向对象程序模型中的核心概念“(对象·消息)结构”赋予开放环境下的新含义，以此来适应开放网络环境的需要. 三是对外部环境的抽象，在开放协同模型的基础上，分析外部环境的特征及其多样的变化模式，将环境相关信息从业务功能中抽取出来并显式化，从而建立环境驱动模型的支撑框架，通过对网构软件系统与运行环境及使用者之间不断演进的互动模式进行感知和分析，归纳出环境驱动的行为模式，进而利用上述框架实现情境感知和主动适应. 四是对智能可信的抽象，即在环境驱动模型的基础上，引入了可信计算框架和人工智能技术，解决开放环境下软件的可信性、个性化、自发成长等方面的问题.

1.2 软件体系结构驱动的网构软件开发方法

互联网开放、动态、难控的特性以及用户使用方式的个性化要求，决定了网构软件的开发不同于传统软件开发的“一次成型”式. 网构软件开发面临着开发过程和开发活动两个方面的变化与挑战. 从开发过程看，网构软件在发布之后，需在运行时感知外部网络环境的动态变化，以使系统具有尽可能高的用户满意度，而且由于用户需求的多样化和个性化以及投资回收等因素，软件系统往往需要7天24小时不间断运行，新功能的开发部署也需要在线进行. 从开发活动看，网构软件往往需要对分布于网络上各个节点的基础软件资源(如对象、构件、服务、程序库、开源软件等)进行“组装式复用”，并根据环境和用户需求的变化，按照功能指标、性能指标和可信性指标等进行调整、适应和演化，基本上是自底向上、反复迭代的开发活动. 显然，相对于传统软件开发，网构软件的开发更加复杂和难控.

全生命周期软件体系驱动的网构软件开发方法——ABC(architecture-based component composition)方法[10]致力于理解和控制上述复杂性，其核心思想是将软件体系结构引入到软件开发的各个阶段，作为系统开发的蓝图，利用工具支持的自动转换机制缩小从高层设计到实现的距离，而后在构件运行平台(如软件中间件)的支持下实现自动的系统组装式生成. 该方法将软件生命周期各阶段的核心制品与活动，统一为软件体系结构模型及对其连续迭代的细化、映射和转换，实现对软件高层结构复杂性的一致、灵活、系统化的管理和控制，从而在原理层面确立了软件体系结构在网构软件系统整个生命周期中的核心地位与作用.

围绕生命周期主要阶段的不同制品及其特点，ABC方法分别采用特征模型和软件体系结构模型来建

a) Internet环境下基于Agent的软件中间件理论与方法(2002~2008年)与基于网络的复杂软件可信度和服务质量及其开发方法和运行机理的基础研究(2009~2013年)

模各个阶段的高层结构复杂性及其关联[10~18]，规约了不同类型需求依赖关系之间以及需求阶段与设计阶段体系结构之间的基本关系，建立了遗产系统的实现与设计及需求之间的逆向追踪关系，可以通过对源程序的分析与理解，辅助逆向建立遗产系统的全生命周期软件体系结构模型，如图2所示. 特别地，ABC方法提出运行时软件体系结构(runtime software architecture, RSA)概念[19]，从软件体系结构的角度系统地揭示了运行时系统维护与演化的机理与性质，阐述了在线维护与演化的基本目标、基本范围以及基本过程等机理. ABC方法揭示了遗产系统的实现与设计及需求之间的逆向追踪性质，如程序的结构信息以及标识符和注释中的文本信息，与软件特征及构件约束之间存在的紧密关联性，从而可以通过分析理解程序源码和文档甚至二进制执行代码和系统运行数据，辅助逆向建立遗产系统的运行时软件体系结构.

1.3 基于运行时软件体系结构的网构软件运行支撑

从能力角度看，网构软件运行支撑平台应具备两种主要能力[5]：其一是对网构软件模型的实现，即如何将所建立的网构软件基本实体变为可运行的实体，以及如何在软件体系结构的管控下实施这些运行实体之间的按需协同；其二是对网构软件系统的管理，即在开放、动态、难控的复杂网络计算环境中，如何保证网构软件系统的运行能够满足预期的功能性、性能、可靠性、安全性等可信指标.

基于RSA的网构软件运行支撑是一种重要的技术途径[19~29]，其基本原理是将RSA作为网构软件系统的一种运行态实体，建立与运行时系统的因果关联关系. 这种关系确保RSA的变化会实时转换为运行系统的变化，反之亦然. 具体地，每个网构软件基本实体均抽象并通过计算反射机制实现为一个元级构件，两个基本实体之间的交互则抽象并实现为一个元级连接子，元级构件和连接子就构成了可以实时监测与控制基本实体及其交互的RSA.

网构软件运行平台需要根据当前系统运行环境和用户需求的变化，动态按需分配并优化调度分布于网络上各个节点的计算、存储、网络资源，从而保障其可信度与服务质量，提高环境感知和适应能力. 运行平台建立了RSA驱动的网构软件资源自治管理回路[19]，基于RSA监测和调整运行系统，评估软件体系结构以确保自治的正确性和效用性. 对于自治管理最为关键的分析和决策，运行平台通过软件体系结构及决策的规范化描述将隐性、劣构的知识转变为显性、良构的知识，并进一步将其规约为体系结构风格、领域特定的体系结构、应用体系结构，制定了体系结构操作原语，以管理即服务(management-as-a-service)的方式将相关知识作用于实际运行系统，如企业计算环境的CORBA、J2EE、Web Services等中间件平台[20~29].

2 网构软件技术体系在移动互联网计算环境的延伸和发展

2005年起，随着第三代移动通信技术(3G)和第四代移动通信技术(4G)在全球范围内陆续投入大规模商用，互联网延伸到移动互联网；同时，云计算、大数据等新型应用模式加速创新，极大地推动了信息技术和应用的深度与广度. “云感知、端融合”(即“云-端”融合)的新型计算模式，为网构软件的研究带来新的机遇和挑战.

2.1 面向云-端融合的网构软件开发方法

在移动互联网环境下，软件的开发呈现了一些新

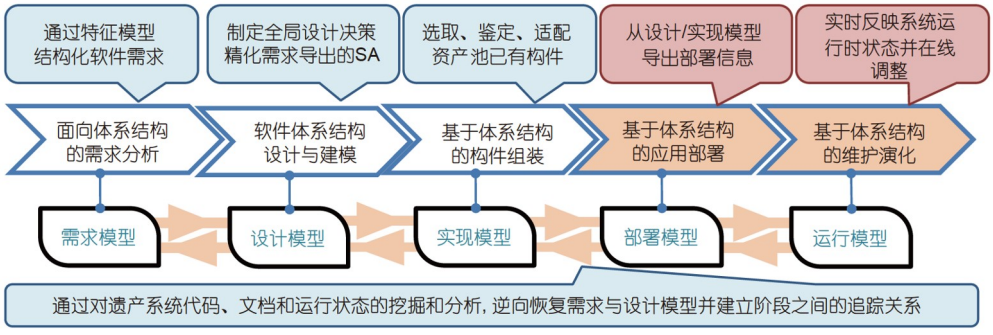


图 2 面向网构软件的全生命周期软件体系结构建模方法
Figure 2 Architecture-based component composition approach for internetware

的特点. 一方面, 智能手机逐渐成为用户接入互联网的主要客户端设备, 应用商店(App Store)开始成为一种主要的软件发布和交付模式, 涌现了数以千万计的App, 其面向最终用户的特征要求软件的更新频率更高, 开发迭代周期更短. 另一方面, 开发移动互联网环境下的软件时, 不仅需要利用云平台丰富的计算、存储和网络资源(通常以云服务的形式提供), 而且还应充分考虑如何灵活地利用终端设备本地的计算、存储、显示、电能、传感等资源. 为了适应上述需求, 融合模型-视图-控制器(model-view-controller, MVC)软件体系结构风格和面向服务架构(service oriented architecture, SOA)软件体系结构风格的面向服务的模型-视图-控制器(service oriented model-view-controller, SMVC)构件模型[30,31], 致力于支持具备云-端融合特性网构软件的开发.

如图3所示, 该构件模型的实现按照MVC风格分为三部分: 模型(model)定义了构件所需的数据对象类及从本地或远程数据源获取数据对象并实例化的方法, 以实现云-端数据源动态绑定、云-端协同数据缓存策略等; 视图(view)定义用户界面中包含的元素及其展示和布局信息, 以实现对设备显示资源的适配, 并定义了如何将元素上的用户操作转换为控制器中的功能; 控制器(controller)定义了实现构件的主要业务逻辑的函数, 开发人员在配置文件中声明在本地和远程上执行的程序逻辑及相应的触发条件, 并实现一致性和正确性保障机制, 如上下文环境同步与校验、函数重构、动态分载(offloading)等. 构件接口依然包含业务接口和反射接口两部分, 在实现上, 每个业务接口对应一个控制器函数, 每个反射接口则通过事件驱动机制来捕捉构件状态的变化, 以便其他构件获知该构件状态的改变并执行必要的函数. SMVC构件中模型、视图和控制器部分均可独立开发、演化和适配, 从而提高了软件适应快速变化的能力; 多个SMVC构件之间可以通过接口相互调用以组装形成更为复杂的应用逻辑与更为丰富和个性化的用户体验[32].

本质上, 上述开发方法依然是全生命周期软件体系结构驱动的网构软件开发方法的一种延伸和扩展, 其核心是通过SMVC这一符合云-端融合特性的软件体系结构风格, 来描述和控制移动互联网环境下软件自适应的复杂性, 如在需求阶段基于应用更新日志、设备特征和用户行为的功能分析[33,34], 在设计和开发阶段面向SMVC模型的领域特定编程语言[35]、面向移动应用内深度链接的接口自动化生成技术[36], 以及基于移动应用商店开发生态模式分析的持续反馈机制等[37].



图 3 面向服务的模型-视图-控制器构件模型
Figure 3 Service oriented model-view-controller component model

2.2 面向云-端融合的网构软件运行平台

云-端融合场景下, 信息资源覆盖数据中心(云)、通信网络(网)和智能终端(端). 相应地, 软件运行平台需要管理海量异构异质的资源(计算、存储、网络、电能、数据等), 支持更加多样化的应用, 支持新型的应用模式和计算模式, 支持用户以多种不同终端设备访问所需的服务. 虽然运行时软件体系结构能为网构软件运行平台提供高层的状态描述和行为规约, 但在操作层面还面临着资源手工管理、静态配置、难以变化等问题, 难以满足网构软件应用对资源进行个性定制、灵活调度、场景适应、开放共享的需求.

针对云-端融合场景下各类资源原生管理能力异构多变、关系交织难控、高级管理任务开放多样的问题, 基于软件定义和运行时模型相结合的网构软件运行支撑方法, 扩展了MOF/QVT国际建模标准语言机制[38,39], 提出了一种运行时软件体系结构模型驱动的云管理语言, 将控制台、管理脚本、配置文件等常见形式的云管理功能自动化地封装和组装为管理程序[38], 自动生成给定计算任务的资源管理功能代码, 手工编程量大幅度减少. 还提出了一组跨域跨层的资源协同管理技术, 如运行时代码动态分载、分布式内存共享与缓存协作等, 优化同层资源、多层资源、跨域资源、业务系统与管理系统之间的资源消费与供给, 有效提升各类资源利用率和业务性能[40,41].

2.3 面向大数据互操作的软件平台

形态上，网构软件是由分布于网络上各个节点的软件实体，通过跨网络的功能和数据互操作而形成的软件系统. 在互联网计算时代, 功能的互联互通互操作一直是网构软件关注的重点. 在移动互联网计算时代, 大数据开启信息化发展的新阶段，数据互联互通互操作的重要性愈发凸显, 特别是, 需要高效地开放已有的大量“信息孤岛”(包括PC应用、Web应用和App等)中蕴藏的“存量”数据资源，是网构软件所面临的一项重大需求. 而传统的“白盒式”的技术手段，如数据库表拷贝、二次开发等，均建立在对原系统内部细节有充分了解的基础上，效率低、难度大、成本高且风险难控. 特别是，由于很多遗产系统的源代码或开发文档等信息不完整甚至不可用、原开发团队缺位等原因，“白盒式”途径往往难以适用.

在前期ABC方法为遗产系统逆向恢复软件体系结构的基础上，形成了客户端驱动的“黑盒式”互操作方法，如图4所示：以上述网构软件SMVC构件模型为切入点，将客户端视为信息孤岛的外部感受器与效应器, 基于信息系统内部基于云-端融合特性的计算反射机理，通过客户端驱动的软件体系结构资源反射方法与机制，逆向理解和重构其运行时软件体系结构并实时监测与控制客户端运行状态，基于细粒度远程调用在线重构、应用构件透明加载、内存状态等价指令序列执行、运行状态按需动态持久化、状态图路径无损复原等技术，保证了客户端运行状态调整能够被实时无误地执行. 在此基础上, 该方法提出人机协同的互操作接口自动化生成技术，将用户行为视为信息孤岛互操作的情境实例，构造语义模型并自动生成互操作接口及其实现体的代码. 通过反馈制导的内核编译优化、计算分载、数据预取、自适应缓存、配置动态适配等全栈式互操作接口运行优化技术，保证了互操作接口正确高效的运行. 该方法消除了对系统源码、数据库表、后台权限、原开发团队等的“白盒式”依赖，将数据开放效率平均提升了两个数量级，为数据互操作提供了高效的网构软件技术手段.

3 展望

互联网延展和覆盖的深度与广度不断加大，正在推动人类社会、信息空间、物理世界的交叉融合，形成“高速泛在、天地一体、云网融合、智能敏捷”的人机物融合泛在计算环境. 作为互联网的延伸, 泛在计算环境标志着我们从终端互联、用户互联、应用互联开始走向万物互联. 作为信息技术“灵魂”的软件将“无处不在”, 并正处于“基础设施化”. 一方面, 软件自身是信息技术应用基础设施的重要构成成分，以平台方式为各类信息技术应用和服务提供基础性能力与运行支撑. 另一方面，软件正在融入整个人类经济社会运行的基础设施中，正在对传统物理世界基础设施和社会经济基础设施进行重塑与重构，通过软件定义的方式赋予其新的能力和灵活性，成为促进生产方式升级、生产关系变革、产业升级、新兴产业和价值链的诞生与发展的重要引擎. “软件定义一切、万物均需互联、一切皆可编程”的人机物融合泛在计算环境为网构软件研

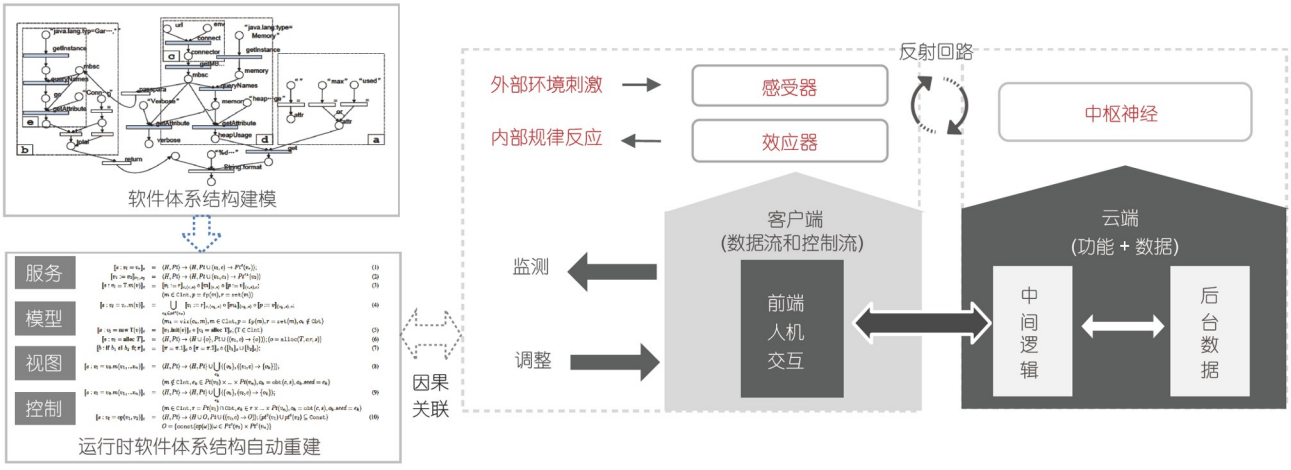


图 4 基于计算反射的云-端融合系统体系结构恢复
Figure 4 Reflection-based software architecture recovery for cloud-client-convergence systems

究开辟了更为丰富的应用场景和更为广阔的发展空间.

网构软件模型方面, 人机物融合泛在计算环境蕴含着海量异构资源, 软件的编程抽象需要泛化到包括计算、存储、网络、传感、电能、数据、服务等在内的各类信息资源, 各种数字化机电设备和可传感物体对象在内的各类物理资源, 乃至可通过激励机制调配的人力资源, 这就对网构软件模型的抽象层次和适应能力提出了更高的要求[42]. 例如, 大量新型专用计算资源(如GPU、TPU、DPU等)的广泛应用, 要求现有面向通用目的开发的软件模型以场景为中心按需分布到异构计算设备或平台上, 从而适应应用按需融合与自适应、自演化的要求. 因此, 软件模型就需要建立一种解构再重构的新机制, 使得各类系统可以通过软件定义的方式对原有的软硬件和服务资源进行解构, 然后基于软件平台进行重构. 相应地, 程序设计语言的研究也应充分考虑泛在资源的各种编程抽象, 探索语言演化和生长机制, 特别是构建领域特定的程序设计语言[43], 为软件范型发展提供编程模型和语言基础.

网构软件开发方法方面, 人机物融合泛在计算环境下, 软件的辖域范围和性质发生了根本性变化, 软件系统将呈现出复杂系统的特征, 其功能数量和规模呈指数级提升, 复杂场景分析与建模将成为网构软件构造的核心难点, 需充分考虑软件与硬件、软件与人交互的紧密性和持续性, 特别是要考虑如何将人作为一种核心要素纳入, 建立“人在回路”的资源调度机制. 软件开发活动中, 开源软件、应用商店、众包、全球化软件开发等带来了开发的群体化[44]. 相应地, 群体协同将成为新的软件技术体系发展驱动力, 应深入研究个体学习、群体协作和组织演化的机理, 特别是理解和利用大规模开源软件项目的供应链行为, 并探索软件生态的形成和可持续机制. 软件开发环境需要以支持高质量、高效率和低成本的软件开发甚至软件自动化为目标, 应支持低代码/无代码开发、人机协作编程、开发运维的智能一体化等, 特别应重视人类智能和机器智能协同融合(如实现部分的机器自动编程), 实现软件系统的快速持续开发和交付.

网构软件的运行平台方面, 人机物融合泛在计算环境中的海量、异质、异构、自主资源导致资源管理的复杂度呈指数级增加, 应用需求的领域性、专用性和个性化更加突出, 万物互联导致安全问题更为突出, 多域融合导致多方多元互信关系更加错综复杂. 相应地, 在资源虚拟化和功能可编程的软件定义方法指导下[45], 网构软件运行平台应在资源栈的解耦合方法、资源的有效虚拟化方法、场景驱动的资源管理方法、系统持续适应演化的支撑机制等关键问题上有所突破, 满足软件作为信息和社会基础设施在适应、演化、体验、效能等方面的诸多严格要求, 形成具有泛在感知、智能互联、反馈控制、自然交互、随需应变等新能力的新形态操作系统, 即“泛在操作系统”(ubiquitous operating system, UOS)[46,47].

特别地, 数据是人机物融合的主要载体, 智慧城市、智能制造、智慧医疗等人机物融合应用不断涌现, 正在推动以机器为中心的互联网衍生出以数据为中心的数联网(internet of data)的重大技术变革. 从软件定义的视角看, 数联网是互联网上数据传、存、算任务调度的控制面[48], 为网构软件向数据为中心的计算模式探索提供了新思路. 在已有的大数据互操作平台基础上, 网构软件运行平台需基于数字对象架构(digital object architecture, DOA), 突破多协议自适应的数据路由、高效数据发现和高性能数据存证等数联网基础软件关键技术[49,50].

4 结语

本文以软件范型作为切入点, 以互联网及其延伸所形成的复杂计算环境作为线索, 介绍了网构软件研究进展, 并展望了未来的研究工作. 需要指出的是, 限于篇幅, 本文着墨于网构软件的开发和运行, 未能对网构软件质量评估与保障相关的研究作充分的介绍. 相对于传统软件, 网构软件的质量主要体现在特定应用环境下软件系统的系统质量对用户期望质量的满足程度[51], 更关注“指标比较综合的以服务质量和可信度为主的使用质量”. 服务质量大多是开发者和用户约定, 由开发者提供保障, 具有相对客观性. 而可信度更多关注用户个性化的体验, 由用户评价, 具有相对主观性. 对服务质量和可信度相关属性需求的满足及对变化的适应, 仍然需要通过软件开发过程和运行支撑机制来实现: 开发过程保障围绕开发制品(如源代码、测试用例、文档、配置文件等[22~57])和开发活动(如开源软件开发个体参与和群体协同[58~61])开展研究, 提出了一系列相关的方法和技术; 运行支撑保障则主要是从软件自适应角度来开展研究, 如基于RSA的可靠性保障、反模式检测与消除、容错体系结构风格选择、冗余数据传输消减等[62~69]. 特别地, 人机物融合泛在计算环境推动软件基础设施化, 这就使得软件质量问题会直接

或间接地对物理世界中的社会经济活动产生重大安全威胁, 需深入理解和度量人机物融合导致的软件系统行为复杂性和不确定性对服务质量与可信度带来的新问题, 并研究相应的保障方法与机制. 希望本文有助于对网构软件感兴趣的同行了解相关研究现状和趋势, 为其从事软件技术相关研究和实践提供有价值的参考.

参考文献

1 Yang F Q, Mei H, Lü J, et al. Some discussion on the development of software technology (in Chinese). Acta Electron Sin, 2002, 26: 1104–1115 [杨芙清, 梅宏, 吕建, 等. 浅论软件技术发展. 电子学报, 2002, 26: 1104–1115]

2 Mei H, Huang G, Xie T. Internetware: A software paradigm for internet computing. Computer, 2012, 45: 26–31

3 Mei H, Liu X Z. Software techniques for internet computing: Current situation and future trend (in Chinese). Chin Sci Bull, 2010, 55: 1214–1220 [梅宏, 刘譞哲. 互联网时代的软件技术: 现状与趋势. 科学通报, 2010, 55: 1214–1220]

4 Lü J, Ma X X, Tao X P, et al. Internetware: Research and progress (in Chinese). Sci China Ser E Inf Sci, 2006, 36: 1037–1080 [吕建, 马晓星, 陶先平, 等. 网构软件的研究与进展. 中国科学E辑: 信息科学, 2006, 36: 1037–1080]

5 Yang F Q, Lü J, Mei H. Technical framework for internetware: An architecture centric approach. Sci China Ser F Inf Sci, 2008, 51: 610–622 [杨芙清, 吕建, 梅宏. 网构软件技术体系: 一种以体系结构为中心的途径. 中国科学E辑: 信息科学, 2008, 38: 818–828]

6 Lü J, Tao X P, Ma X X, et al. Research on agent-based Internetware model (in Chinese). Sci China Ser E Inf Sci, 2005, 35: 1233–1253 [吕建, 陶先平, 马晓星, 等. 基于Agent的网构软件模型研究. 中国科学E辑: 信息科学, 2005, 35: 1233–1253]

7 Wang P W, Jin Z, Liu H Y. Description and its discovery for internetware entity functionality (in Chinese). Sci China Ser F Inf Sci, 2009, 39: 1271–1287 [王璞巍, 金芝, 刘红岩. 网构软件实体的功能描述及其发现. 中国科学F辑: 信息科学, 2009, 39: 1271–1287]

8 Lü J, Ma X X, Tao X P, et al. Context-driven model and supporting technologies for Internetware (in Chinese). Sci China Ser E Inf Sci, 2008, 38: 864–900 [吕建, 马晓星, 陶先平, 等. 面向网构软件的环境驱动模型与支撑技术研究. 中国科学E辑: 信息科学, 2008, 38: 864–900]

9 Chang Z M, Mao X J, Qi Z C. Component model and its implementation of internetware based on Agent (in Chinese). J Softw, 2008, 19: 1113–1124 [常志明, 毛新军, 齐治昌. 基于Agent的网构软件构件模型及其实现. 软件学报, 2008, 19: 1113–1124]

10 Mei H, Huang G, Zhao H Y, et al. An architecture-centric software development approach for internetware (in Chinese). Sci China Ser E Inf Sci, 2006, 36: 1100–1126 [梅宏, 黄罡, 赵海燕, 等. 一种以软件体系结构为中心的网构软件开发方法. 中国科学E辑: 信息科学, 2006, 36: 1100–1126]

11 Mei H, Chang J, Yang F. Software component composition based on ADL and middleware. Sci China Ser F Inf Sci, 2001, 44: 136–151

12 Zhang W, Mei H. A feature-oriented domain model and its modeling process (in Chinese). J Softw, 2003, 14: 1345–1356 [张伟, 梅宏. 一种面向特征的领域模型及其建模过程. 软件学报, 2003, 14: 1345–1356]

13 Jin Z. Automatically multi-paradigm requirements modeling and analyzing: An ontology-based approach. Sci China Ser F Inf Sci, 2003, 46: 279–297

14 Pan Y, Wang L, Zhang L, et al. Relevancy based semantic interoperation of reuse repositories. In: Proceedings of the 12th ACM SIGSOFT Symposium on Foundations of Software Engineering. New York: Association for Computing Machinery, 2004. 211–220

15 Mei H, Zhang W, Zhao H. A metamodel for modeling system features and their refinement, constraint and interaction relationships. Softw Syst Model, 2006, 5: 172–186

16 Hou L, Jin Z, Wu B. Modeling and verifying web services driven by requirements: An ontology-based approach. Sci China Ser F Inf Sci, 2006, 49: 792–820

17 Zhao W, Zhang L, Liu Y, et al. SNIAFL. ACM Trans Softw Eng Methodol, 2006, 15: 195–226

18 Mei H, Huang G, Zhao H, et al. A software architecture centric engineering approach for internetware. Sci China Ser F Inf Sci, 2006, 49: 702–730

19 Mei H, Huang G, Lan L, et al. Architecture-based adaptation approach for internetware (in Chinese). Sci China Ser E Inf Sci, 2008, 38: 901–920 [梅宏, 黄罡, 兰灵, 等. 基于体系结构的网构软件自适应方法. 中国科学E辑: 信息科学, 2008, 38: 901–920]

20 Huang G, Wang Q X, Cao D G, et al. A domain-oriented component runtime support platform (in Chinese). Acta Electron Sin, 2002, 30: 39–43 [黄罡, 王千祥, 曹东刚, 等. 一种面向领域的构件运行支撑平台. 电子学报, 2002, 30: 39–43]

21 Huang G. Runtime software architecture based on reflective middleware. Sci China Ser F Inf Sci, 2004, 47: 555–576

22 Zhong H, Zhang L, Xie T, et al. Inferring resource specifications from natural language API documentation. In: Proceedings of 24th IEEE/ACM International Conference on Automated Software Engineering. Los Alamitos: IEEE Computer Society Press, 2009. 307–318

23 Huang T, Chen N J, Wei J, et al. OnceAS/Q: A QoS-oriented web application server (in Chinese). J Softw, 2004, 15: 1787–1799 [黄涛, 陈宁江, 魏峻, 等. OnceAS/Q: 一个面向QoS的Web应用服务器. 软件学报, 2004, 15: 1787–1799]

24 Huang G, Liu T C, Mei H, et al. Towards autonomic computing middleware via reflection. In: Proceedings of the 28th Annual International Computer Software and Applications Conference. Los Alamitos: IEEE Computer Society Press, 2004. 122–127

25 Shen J R, Sun X, Huang G, et al. Towards a unified formal model for supporting mechanisms of dynamic component update. In: Proceedings of the 5th Joint Meeting of the European Software Engineering Conference and ACM SIGSOFT Symposium on the Foundations of Software Engineering. New York: Association for Computing Machinery, 2005. 80–89

26 Mei H, Cao D G. ABC-S2C: A cross-cutting oriented technique for separation-of-concern of component-based software (in Chinese). J Comput, 2005, 28: 2036–2044 [梅宏, 曹东刚. ABC-S2C: 一种面向贯穿特性的构件化软件关注点分离技术. 计算机学报, 2005, 28: 2036–2044]

27 Wang Q, Shen J, Wang X, et al. A component-based approach to online software evolution. J Softw Maint Evol-Res Pract, 2006, 18: 181–205

28 Huang T, Ding X, Wei J. An application-semantics-based relaxed transaction model for internetware. Sci China Ser F Inf Sci, 2006, 49: 774–791

29 Huang G, Liu X Z, Mei H. Online approach to feature interaction problems in middleware based system. Sci China Ser F Inf Sci, 2008, 51: 225–239

30 Huang G, Liu X Z, Zhang Y. A mobile web application platform with synergy of cloud and client (in Chinese). Sci Sin Inform, 2013, 43: 24–44 [黄罡, 刘譞哲, 张颖. 面向云-端融合的移动互联网应用运行平台. 中国科学: 信息科学, 2013, 43: 24–44]

31 Huang G, Liu X, Ma Y, et al. Programming situational mobile web applications with cloud-mobile convergence: An internetware-oriented approach. IEEE Trans Serv Comput, 2019, 12: 6–19

32 Zhao X, Liu X Z, Wang X D, et al. A mechanism for dynamically constructing homogeneous Mashup components pool (in Chinese). J Front Comput Sci Technol, 2013, 7: 14–24 [赵星, 刘譞哲, 王旭东, 等. 同质Mashup构件池动态构造机制. 计算机科学与探索, 2013, 7: 14–24]

33 Lu X, Chen Z P, Liu X Z, et al. PRADO: Predicting app adoption by learning the correlation between developer-controllable properties and user behaviors. Proc ACM Interact Mob Wearable Ubiquitous Technol, 2017, 1: 79

34 Lu X, Liu X Z, Li H R, et al. PRADA: Prioritizing Android devices for apps by mining large-scale usage data. In: Proceedings of the 38th International Conference on Software Engineering. New York: Association for Computing Machinery, 2016. 3–13

35 Liu X, Xu M, Teng T, et al. MUIT: A domain-specific language and its middleware for adaptive mobile web-based user interfaces in WS-BPEL. IEEE Trans Serv Comput, 2019, 12: 955–969

36 Ma Y, Hu Z N, Liu Y X, et al. Aladdin: Automating release of deep-Link APIs on Android. In: Proceedings of the 2018 World Wide Web Conference on World Wide Web. New York: Association for Computing Machinery, 2018. 1469–1478

37 Liu X, Li H, Lu X, et al. Understanding diverse usage patterns from large-scale appstore-service profiles. IEEE Trans Softw Eng, 2018, 44: 384–411

38 Ferry N, Chauvel F, Song H, et al. CloudMF: Model-driven management of multi-cloud applications. ACM Trans Internet Technol, 2018, 18: 16

39 Chen X P, Huang G, Chauvel F, et al. A framework for the integration of MOF-compliant analysis methods. In: Proceedings of the Second Asia-Pacific Symposium on Internetware. New York: Association for Computing Machinery, 2010. 1

40 Song H, Raj A, Hajebi S, et al. Model-based cross-layer monitoring and adaptation of multilayer systems. Sci China Inf Sci, 2013, 56: 082101

41 Huang G, Song H, Mei H. SM@RT: Towards architecture-based runtime management of internetware systems. In: Proceedings of the First Asia-Pacific Symposium on Internetware. New York: Association for Computing Machinery, 2009. 9

42 Xu C, Qin Y, Yu P, et al. Theories and techniques for growing software: Paradigm and beyond (in Chinese). Sci Sin Inform, 2020, 50: 1595–1611 [许畅, 秦逸, 余萍, 等. 可成长软件理论方法和实现技术: 从范型到跨越. 中国科学: 信息科学, 2020, 50: 1595–1611]

43 Ji R Y, Liang J J, Xiong Y F, et al. Question selection for interactive program synthesis. In: Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation. New York: Association for Computing Machinery, 2020. 1143–1158

44 Zhang W, Mei H. Software development based on collective intelligence on the Internet: Feasibility, state-of-the-practice, and challenges (in Chinese). Sci Sin Inform, 2017, 47: 1601–1622 [张伟, 梅宏. 基于互联网群体智能的软件开发: 可行性、现状与挑战. 中国科学: 信息科学, 2017, 47: 1601–1622]

45 Mei H. Understanding “software-defined” from an OS perspective: Technical challenges and research issues. Sci China Inf Sci, 2017, 60: 126101

46 Mei H, Guo Y. Toward ubiquitous operating systems: A software-defined perspective. Computer, 2018, 51: 50–56

47 Mei H, Cao D G, Xie T. Ubiquitous operating system: Toward the blue ocean of human-cyber-physical ternary ubiquitous computing (in Chinese). Bull Chin Acad Sci, 2022, 37: 30–37 [梅宏, 曹东刚, 谢涛. 泛在操作系统: 面向人机物融合泛在计算的新蓝海. 中国科学院院刊, 2022, 37: 30–37]

48 Huang G. Internet of data: Digital space infrastructure (in Chinese). Commun CCF, 2021, 17: 58–60 [黄罡. 数联网: 数字空间基础设施. 中国计算机学会通讯, 2021, 17: 58–60]

49 Huang G, Luo C R, Wu K D, et al. Software-defined infrastructure for decentralized data lifecycle governance: Principled design and open challenges. In: Proceedings of 39th IEEE International Conference on Distributed Computing Systems. Los Alamitos: IEEE Computer Society Press, 2019. 1674–1683

50 Liu X Z, Sun S X, Huang G. Decentralized services computing paradigm for blockchain-based data governance: Programmability, interoperability, and intelligence. IEEE Trans Serv Comput, 2020, 13: 343–355

51 Wang Y, Lü J, Xu F, et al. A trust measurement and evolution model for internetware (in Chinese). J Softw, 2006, 17: 682–690 [王远, 吕建, 徐锋,

等. 一个适用于网构软件的信任度量及演化模型. 软件学报, 2006, 17: 682–690]
52 Mei H, Hao D, Zhang L, et al. A static approach to prioritizing JUnit test cases. IEEE Trans Softw Eng, 2012, 38: 1258–1275
53 Yang Z Q, Jin Z. Requirements modeling and system reconfiguration for self-adaptation of internetware. In: Proceedings of the 4th Asia-Pacific Symposium on Internetware. New York: Association for Computing Machinery, 2012. 11
54 Lou Y L, Chen J J, Zhang L M, et al. History-driven build failure fixing: How far are we? In: Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis. New York: Association for Computing Machinery, 2019. 43–54
55 Shen B, Zhang W, Zhao H Y, et al. IntelliMerge: A refactoring-aware software merging technique. Proc ACM Program Lang, 2019, 3: 170
56 Liu M W, Peng X, Jiang Q T, et al. Searching Stack Overflow questions with multi-faceted categorization. In: Proceedings of the 10th Asia-Pacific Symposium on Internetware. New York: Association for Computing Machinery, 2018. 10
57 Jiang H, Nie L, Sun Z, et al. ROSF: Leveraging information retrieval and supervised learning for recommending code snippets. IEEE Trans Serv Comput, 2019, 12: 34–46
58 Zhou M, Mockus A. Who will stay in the FLOSS community? Modeling participant's initial behavior. IEEE Trans Softw Eng, 2015, 41: 82–99
59 Wu Y W, Zhang Y, Wang T, et al. Exploring the relationship between developer activities and profile images on GitHub. In: Proceedings of the 11th Asia-Pacific Symposium on Internetware. New York: Association for Computing Machinery, 2019. 7
60 Jiang Y, Zhang W, Wang P, et al. A knowledge graph construction method based on internet-based collective intelligence (in Chinese). J Softw, 2022, 33: 2646–2666 [蒋逸, 张伟, 王佩, 等. 基于互联网群体智能的知识图谱构造方法. 软件学报, 2022, 33: 2646–2666]
61 Zhang W, Mei H. A constructive model for collective intelligence. Natl Sci Rev, 2020, 7: 1273–1277
62 Liu X Z, Yu M H, Ma Y, et al. I-Jacob: An internetware-oriented approach to optimizing computation-intensive mobile web browsing. ACM Trans Internet Technol, 2018, 18: 14
63 Zhu J, Rong G P, Huang G C, et al. JLLAR: A logging recommendation plug-in tool for Java. In: Proceedings of the 11th Asia-Pacific Symposium on Internetware. New York: Association for Computing Machinery, 2019. 16
64 Wang Q X, Li X. Bug localization via searching crowd-contributed code. In: Proceedings of the 6th Asia-Pacific Symposium on Internetware. New York: Association for Computing Machinery, 2014. 1–10
65 Zhang Y Y, Zhang Y, Wu Y W, et al. Exploring the dependency network of docker containers: Structure, diversity, and relationship. In: Proceedings of the 13th Asia-Pacific Symposium on Internetware. New York: Association for Computing Machinery, 2020. 199–208
66 Ouyang X, Garraghan P, Primas B, et al. Adaptive speculation for efficient internetware application execution in clouds. ACM Trans Internet Technol, 2018, 18: 15
67 Wang X, Zhang L, Xie T, et al. Locating need-to-externalize constant strings for software internationalization with generalized string-taint analysis. IEEE Trans Softw Eng, 2013, 39: 516–536
68 Li J G, Chen X P, Huang G, et al. Selecting fault tolerant styles for third-party components with model checking support. In: Proceedings of the 12th International Symposium on Componet-Based Software Engineering. Singapore: Springer, 2009. 69–86
69 Wang W, Liu Z H, Jiang Y, et al. EasyCache: A transparent in-memory data caching approach for internetware. In: Proceedings of the 6th Asia-Pacific Symposium on Internetware. New York: Association for Computing Machinery, 2014. 35–44

Summary for “网构软件研究: 回顾与展望”

Research on internetware: Review and prospect

Hong Mei[1,2*], Gang Huang[1,2], Xuanzhe Liu[1,2] & Ying Zhang[2]

[1] *School of Computer Science, Peking University, Beijing 100871, China;*
[2] *Key Lab of High-Confidence Software Technologies (Peking University), Ministry of Education, Beijing 100871, China*
* Corresponding author, E-mail: meih@pku.edu.cn

Software is the core component of IT industry and an important “infrastructure” that supports social operations in the digital economy era. Since the 21st century, the internet has evolved into a global ubiquitous computing environment and open platform. Its open, dynamic, and uncontrollable nature requires the corresponding changes in the basic form and characteristics of software, the conceptual framework, and the logical connotation, and thus leads to substantial challenges to software theory, methods, and technologies. In 2000, researchers from China proposed the term “internetware”, which indicates a new software paradigm for the internet computing. After more than 20 years of efforts, a series of important research achievements have been made in aspects of the basic model, development methodology, runtime support, quality assessment and assurance of internetware, and resulted in systematic innovation results and produced a wide range of academic and industrial impacts. In internetware, the software model consists of a set of autonomous software entities distributed and/or decentralized over the internet and other extensions like internet of things and 4G/5G, together with a set of connectors for enabling collaborations among these entities in various ways. Internetware software entities are able to sense dynamic changes of the underlying environments, and continuously adapt to these changes by means of structural and behavioral maintenance and evolution. From the micro perspective, internetware software entities collaborate with each other on demand and on the fly. From the macro perspective, the entities can self-organize to form an application or community of interest and even decentralized autonomous organizations. As a result, the development and evolution of a software application with internetware can be viewed as continuous and iterative composition of various “disordered” resources into “ordered” software applications. Thus, software development with internetware is a process being bottom-up, inside-out, spiral, and to some extent, similar with the complex adaptive systems. For example, the internetware paradigm proposed the theory of software architecture modeling (called ABC methodology) covering the whole life-cycle, where the core artifacts and activities of every single stage are unified into the software architecture model and its iterative refinement and transformation. In this way, the internetware paradigm greatly improves the efficiency and quality of software development and evolution. In addition, the internetware paradigm expands the software architecture from the development phase to the runtime, by proposing the concept of runtime software architecture (RSA). The RSA has been widely applied for a large number of information “silo” systems to enable the functionality and data interoperability. This solution is a significantly disruptive technical invention, or namely the “black box” mode, based on the client-driven resource reflection mechanism to achieve automatic recovery of the system runtime architecture and automatic generation of data access interfaces. Compared to the traditional “white box” interoperability solution, the internetware paradigm eliminates the need of accessing source code, documentation, and original development team, and improves the interoperability efficiency between information silos, with more than 100X acceleration. This article reviews the research and practice of internetware from the perspective of software paradigm, following the internet computing environment and its extensions as a clue. It also discusses the future research outlook of internetware, especially for the ubiquitous computing environments and data centric technologies.

internet, software paradigm, internetware, software architecture

doi: 10.1360/TB-2022-0585