科学通报 2010 年 第 55 卷 第 13 期：1214 ~ 1220
评 述 www.scichina.com csb.scichina.com
《中国科学》杂志社
SCIENCE CHINA PRESS

互联网时代的软件技术：现状与趋势

梅宏[①②]，刘譞哲[①②]

① 高可信软件技术教育部重点实验室(北京大学)，北京 100871；
② 北京大学信息科学技术学院软件研究所，北京 100871
E-mail: meih@pku.edu.cn

2009-08-17 收稿，2009-11-11 接受
国家重点基础研究发展计划(编号：2009CB320700)和国家自然科学基金(批准号：60821003)资助项目

摘要 互联网(Internet)的快速发展为信息技术与应用扩展了更为广阔的空间，同时也为传统的软件开发理论、方法和技术带来了一系列的挑战．回顾了现有基于网络的新型计算范型和应用模式．探讨了软件模型、软件运行平台和支撑机制、软件开发方法和软件质量评价与保障体系的现状和发展趋势．讨论并展望了进一步的研究工作．

关键词
互联网
软件技术
软件方法学
网构软件

计算机软件是计算机执行某项任务所需的程序、过程及文档的集合．从功能上看，计算机软件可以分为应用软件、系统软件和支撑软件，系统软件和支撑软件也称为基础软件．计算机软件的最终目的，是为人类提供更好的计算服务．因此，满足各种应用需求，就成为推动软件技术发展的直接驱动力．

在各种应用需求的驱动下，软件技术的研究基本上沿着两条并行的脉络展开．一条是软件系统技术的研究，核心内容是软件的本质、特征和模型，即软件的基本元素、体系结构、交互协议、行为模式、效率机制和质量保障机制，基本途径是逐层的虚拟化技术和系统优化技术．例如，操作系统可以视为构架在硬件资源上的软件虚拟机，数据库、中间件和应用软件可视为架构在下层资源上的一层虚拟机，以提供更高效的资源管理和更自然的人机交互界面．另一条是对软件开发技术的研究，主要内容包括软件开发方法论及相应的工程原则、支撑工具和环境等，即软件工程学科的研究范畴．

本质上，软件是对客观事物的虚拟反映，是知识的固化、凝炼和体现，这就驱动了软件的基本模型必然不断地追求更具表达能力、更符合人类思维模式、更具备可构造性和演化性的结构特征．例如，结构化软件试图适应人分析问题和解决问题的思维模式，面向对象软件试图适应客观世界的问题结构．另一方面，软件的主要作用是指挥计算机系统进行问题求解并实施相应的管理，这就驱动着软件不断地追求更高效合理地发挥硬件资源所提供的计算能力．例如，在操作系统发展过程中，随着计算机硬件的发展，单机操作系统从最早的引导程序到管理程序，再发展到发挥 CPU 能力及外设功用的多道程序支撑，进而发展为追求计算机软硬件资源高效利用的资源共享与管理系统．因此，随着应用领域和信息技术的持续发展变化，软件技术及系统规模也不断地升级、演化，呈现出不同的形态[1]．

互联网(Internet)无疑是 20 世纪最为伟大的技术创造之一．进入 21 世纪以来，互联网的快速发展极大地促进了全球化的广度和深度，也为信息技术及其应用开辟了更为广阔的发展空间．人们对互联网的认识，已经从“以网络连接的多个计算机”逐步发展到以不同角度抽象的“统一的计算机”．如何充分利用这样一台全球泛在的“统一的计算机”，成为计算机科学与技术的重大挑战，由此产生了多个基于网络的新型应用模式或计算范型．这些应用模式或计算范型，以及互联网平台自身开放、动态、难控的

英文版见: Mei H, Liu X Z. Software techniques evolved by the Internet: Current situation and future trend. Chinese Sci Bull, 2010, 55, doi: 10.1007/s11434-010-0163-x

特性, 导致软件形态呈现新的变化, 由此对传统的软件理论、方法和技术提出了新的挑战[2].

本文将探讨互联网环境下软件技术的现状和进展. 首先, 回顾现有基于网络的应用模式或计算范型. 其次, 结合这些研究实践, 围绕软件方法和技术体系中的四大要素——软件形态与模型、软件运行平台、软件开发方法和软件质量, 分析相关理论、方法和技术的现状以及发展趋势. 最后, 讨论并展望软件技术的进一步发展方向.

1 基于互联网的计算范型和应用模式

互联网产生至今, 已经出现了多种基于网络的新型应用模式或计算范型. 下面简要介绍目前一些主要的概念及其相关技术.

网格计算(Grid Computing)的概念最初于 1990 年代中期提出, 其目标是有效整合分布于互联网上的计算资源、存储资源、通信资源、软件资源和知识资源等, 为用户提供一体化透明的服务. 近年来出现的云计算(Cloud Computing)可以看作是这种理念的一种延伸. 相对于网格计算中资源地域分布的特点, 云计算更强调将互联网上丰富的虚拟化资源(软件、数据甚至计算平台)都集中在数据中心(Data Center)之上, 并以服务的形式供用户使用, 来满足不同计算任务的需要. 网格计算和云计算得到了学术界和产业界的广泛关注, 并取得了一些进展, 但在资源的按需分配与聚合、资源的可信协同与管理等问题上仍面临诸多挑战. 值得一提的是, 我国学者在国家重点基础研究发展计划项目(973)的支持下, 开展了"虚拟计算环境"的研究[3], 为解决开放环境下的按需聚合、分布自治资源的自主协同以及聚合与协同的计算性质等问题作出了有益的尝试.

普适计算(Pervasive Computing), 类似的概念还有泛在计算(Ubiquitous Computing)等在 1990 年代初提出, 其目标是无所不在的、随时随地可以进行计算的一种方式, 试图将信息空间和人们生活的物理空间有机集成为一个整体. 以各种网络环境(如互联网、无线网、传感网)为应用环境, 以各种嵌入式计算设备(如手机、便携式电脑、PDA 等)为载体, 普适计算已经取得了一些重要进展. 特别地, 近年来, 随着信息技术向网络化、泛在化和服务化飞速发展, 离散的计算过程和连续的物理世界的结合更加紧密, 并出现了一种融合计算、通信和控制的新型系统 CPS(Cyber-Physical System)[1]. 美国国家自然科学基金委员会(NSF) 认为, CPS 可广泛应用于非传统计算设备, 如汽车电子、智能医疗等, 将极大程度地改进人类和现实世界的互动方式, 并已将 CPS 列为最为重要的研究课题之一. 充分考虑计算设备、网络环境和物理世界之间的互动关系和对系统行为产生的影响, 实现有效的情境感知机制、实时控制机制和可信性保障机制, 为人们提供一致的交互方式和更好的用户体验, 将是未来普适计算需要重点解决的问题.

服务计算(Services Computing)的概念在 20 世纪 90 年代末提出, 其目标是以服务作为应用开发的基本单元, 能够以服务组装的方式快速、便捷和灵活地生成增值服务或应用系统, 并有效地解决在分布、异构的环境中数据、应用和系统集成问题. 软件服务最本质同时也是最核心的理念, 就是人们不再需要拥有软件产品本身, 而是直接使用软件所提供的功能. 目前, 网络上已经存在了相当规模的软件服务资源. 围绕软件服务的主要支撑技术——Web Services, 学术界和工业界均展开了大量研究与实践. 但是, 在开放网络环境下, 仍有一些问题亟待解决, 包括快速准确的服务发现、明晰一致的服务语义、按需的服务协同、灵活的服务组装、可信的服务质量、跨域服务的安全保障等.

语义网(Semantic Web)的概念在 21 世纪初提出, 其最初目标是克服目前基于 Web 信息共享方式在资源定位查找上的不足, 希望通过为网页增加计算机可处理的语义信息, 建立一种使网络资源的语义能够自表示、便于机器理解、基于互联网的信息共享平台, 以此来支持 Web 信息的有效共享和利用. 进一步地, 语义网希望对网络上的各种资源都赋予明晰无歧义的语义信息, 从而使计算机可以分辨和识别, 并对其进行自动化地解释、交换和处理. 语义网试图使整个互联网成为一个统一、通用的信息交换平台, 但是, 实现这一目标仍然面临诸多挑战, 包括为网络资源定义无歧义的形式化语义、构造良构、标准的元数据词汇表(即本体)、保障资源描述的可信性等. 特别地, 如何实现对网络上已有的海量资源进行自动化/半自动化的语义化处理, 是语义网所面临的主要困难.

万维学(Web Science)的概念由万维网的创始人

1) Lee E A. Cyber-Physical Systems-are computing foundations adequate? Position paper for NSF Workshop on Cyber-physical systems: Research motivation, techniques and roadmap. October 16–17, 2006, Austin, TX

Tim Berners-Lee 于 2006 年提出[4], 其目标是掌握 WWW 的发展, 并产生更为有效、有益的网上行为和模式. 万维学是以 WWW 这一目前最为成功的网络应用模式作为研究对象来理解和分析万维网的演化和发展, 并将与万维网有关的基础理论和技术作为一门独立的学科来研究, 同时强调其与社会学、心理学、管理学等相关学科的融合. 近年来兴起的 Web 2.0 和社交网络(Social Networking)均可被纳入万维学的范畴. 目前, 万维学的研究包括新型网络应用模式、基于网络的虚拟社会形成及关系管理、用户行为分析、网络资源的发现和使用机制等.

上述产生于互联网环境下的新型计算范型和应用模式均是从某种视角或层次对“统一计算机”的应用提出新的理念和技术体系. 例如, 网格计算和云计算可以看成是从资源共享与管理的角度探讨未来网络系统的应用与构造模式; 普适计算可以看成是从人机交互的角度来探讨未来网络系统的应用模式; 语义网则是从语言学和自然语言处理角度探讨智能化地使能或辅助网络资源共享. 本质上, 这些计算模式或范型均离不开软件技术作为其基础支撑, 这就需要软件的基本形态和特征、概念框架、逻辑内涵都发生相应的变化, 同时也对软件方法与技术提出了全方位的挑战.

我国学者从软件技术角度出发, 于 2002 年正式提出了网构软件(Internetware)[5]的概念. 网构软件是互联网环境下的一种新型软件形态. 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, 为了适应开放、动态、多变的互联网环境及应用领域全球化、个性化、持续成长等特点, 网构软件也有其独有的基本特征, 如自主性、演化性、协同性和反应性等. 在国家重点基础研究发展计划项目“Internet 环境下基于 Agent 的软件中间件理论和方法研究”的支持下, 网构软件的研究与实践已取得了一些进展, 形成了一套以软件体系结构为中心的技术体系, 主要包括一种基本实体主体化、构件化、服务化以及实体间开放按需协同的网构软件模型、一个支持网构软件基本实体运行和自治管理的网构软件中间件平台、一套以全生命周期软件体系结构为核心、模型驱动的网构软件开发方法等.

2 软件模型的演变

计算机软件技术的发展历史表明, 软件方法学是软件技术的基础, 同软件方法学发展最为密切的 3 个要素, 便是计算机平台、人的思维模式和问题域的特性. 而软件模型则是软件方法学的核心, 集中体现了一种软件方法及其技术支撑体系以何种基本视角来看待软件系统之成分构成和运作机理[1]. 考察软件模型的发展脉络, 可以发现, 软件模型始终在追求更具表达能力、更符合人类思维模式、易构造、易演化的目标. 随着计算平台从封闭、静态走向开放、动态, 所面临的问题特性也从单一固定变为多重多变, 软件模型也必将随之发生改变.

软件模型描述软件实体和实体之间的交互连接关系. 软件实体是构成软件系统的基本元素. 在计算机应用与支撑平台快速发展的驱动之下, 软件实体的发展经历了语句、函数/过程、模块、抽象数据类型/对象、构件、服务等粒度和层次. 在这个过程中, 软件实体的内容自包含性、功能独立性和实体适应性在不断增强, 以满足开发者和使用者对于系统复杂性控制的要求. 例如, 结构化软件试图适应人分析问题和解决问题的思维模式, 而面向对象进一步地试图适应客观世界的问题结构. 作为目前主流的软件模型, 对象封装了一类实体的属性和操作, 并通过继承、多态和动态绑定在一定程度上解决了控制复杂性和应变性的问题, 使得软件能够对问题的开放性(即功能的易变性)具备一定的应变能力. 以统一建模语言(Unified Modeling Language)、模型驱动体系(Model Driven Architecture)为代表的对象技术取得了较大成功. 出于软件复用的考虑, 人们希望寻求比对象粒度更大、更易于复用的基本单元, 因此对软件实体的自包含性和功能独立性提出了更高的要求, 软件构件技术得以迅速发展起来. 软件构件包括构件实现体和构件接口两个部分, 前者实现构件的具体功能, 后者则对构件的功能进行规约, 为第三方进行构件组装提供基础支撑. 产业界已经有了具有代表性的软件构件技术, 如 OMG 组织的 CORBA/CCM, Microsoft 公司的 COM/COM++和 SUN 公司的 J2EE/EJB 等. 为了应对开放网络环境中各类资源的共享和集成对计算机软件技术的挑战, 软件服务在近年来得到了广泛关注, 被认为是软件发展的重要方向[6]. 软件服务使得其使用者能够在不拥有软件产品的前提下, 直接使用软件的功能, 从而实现“即拿即用”的效果. 这种“只求使用, 不求拥有”的特性, 也只有通过互联网和软件的组合才可能实现. 软件服务的使用者可

以更加灵活地进行服务组装以生成增值服务，并能不断地选择最优服务来满足适应性的需求. 以 Web Services 为支撑技术，软件服务广泛应用于业务流程管理、企业应用集成等领域. 近年来随着 Web 2.0 技术的兴起，软件服务的支撑技术和应用模式呈现出多样化、个性化的特点，如 RESTful Web 服务、RSS/Atom, Mashup 等，带来了互联网上软件服务的进一步繁荣.

软件实体之间需要通过建立通信连接和交互约束，相互协作达成既定应用目标，这个过程就是软件协同. 例如，语句之间的 3 种基本控制结构(顺序、循环和选择)、函数之间的子程序调用、对象之间的消息传递，都是不同软件模型的协同机制. 在传统的结构化和面向对象技术中，与软件实体相比，协同处于从属地位. 软件协同逻辑往往被固化在过程或方法调用之中，和计算逻辑紧密地耦合在一起. 在开放的互联网环境下，软件实体往往是高度构件化或服务化的，协同必须以尊重这种构件化或服务化为前提以达成应用目标，而且协同方式必须灵活多样且能动态调整，以适应用户需求和环境的变化. 因此，实体之间的协同机制就需要被从软件实体的计算逻辑中剥离出来，作为相对独立的部分加以研究和实现，并通过对其灵活的配置、多样的集成和分层次的演化来适应用户需求和环境的变化. 例如，服务计算中以服务流程语言(如 WS-BPEL 和 WS-CDL)作为服务之间的协同机制，网构软件提出以软件体系结构中可定制的自定义连接子(Connector)来刻画自主构件之间的协同机制[7]等.

传统的软件系统运行于静态、封闭、固定的环境中，其运行效果可以预计，因此传统的软件模型较少考虑环境因素，如环境状态信息的获取、软件实体与环境中其他未知实体的同步、协作关系等. 而在动态、开放的互联网环境下，软件实体可能随时加入或退出，软件实体之间的交互更加灵活多变，用户和软件系统也处于持续互动过程之中，这些都会导致系统状态和行为不断发生变化，从而表现出一种“涌现”性质. 因此，外部环境因素成为构造软件系统必须考虑的基本要素之一，其基本特征、构成成分和变化规律对于软件基本模型的影响也日益重要. 新的软件模型必须充分考虑其所处环境的信息，感知其变化并作出相应的适应性调整. 近年来，随着软件技术的发展和变革、软件应用的不断深化和领域延伸，关于环境特征、环境建模以及环境信息处理等方面的工作得到了重视并取得了初步进展. 例如，环境感知(Context-Aware)机制是普适计算最为重要的支撑技术[8]；网构软件提出环境驱动模型显式化地刻画环境信息和互动方式，从而支持软件系统自主感知并适应外部环境变化[9]；基于 Web 的情境应用(Situational Applications)[10]的开发也需要建立在充分感知并理解外部环境的变化的基础上.

3 软件运行平台的演变

软件运行平台是一组软件基础设施，属于系统软件的范畴，为应用软件的运行提供各种必要的支撑. 从系统角度看，软件运行平台负责管理和协调应用软件对各种底层软硬件资源的使用，充分发挥底层软硬件资源所提供的计算能力，有效地桥接各种异构资源，增加互操作性. 从使用角度看，软件运行平台需要对底层资源细节进行抽象，为使用者提供更方便易用的人机界面，提供其上应用软件的编程模型. 从非功能角度看，软件运行平台需要为应用程序的运行提供性能和易用性等非功能保证，为平台自身运行提供性能优化能力.

计算机操作系统是典型的单机软件运行平台. 最早的计算机软件都是以机器语言或者汇编语言书写、运行在硬件裸机上的，资源管理完全由应用程序自身负责，软件开发效率和资源使用效率都很差，此时其实没有软件运行平台的概念. 单机操作系统可以看成是最早的软件运行平台. 单机操作系统的发展，是从最早的引导程序到管理程序，再发展到发挥 CPU 能力及外设功用的多道程序支撑，进而发展为追求计算机软硬件资源高效利用的资源共享与管理系统. 网络操作系统的出现主要是支持多个计算机之间的网络通信和资源共享，分布式操作系统主要是负责管理分布式系统资源和控制分布式程序运行，并行操作系统是为了管理大规模的并行处理任务，嵌入式操作系统为嵌入式电子设备实现各种灵活功能提供信息处理系统平台. 从操作系统的发展过程看，软件运行平台始终在追求更高效合理地发挥软硬件资源所提供的计算能力.

当软件的运行环境从单机环境迈入网络环境后，软件运行平台需要连接并管理网络上数量众多的异构、自治的硬件资源和软件资源，包括主机、操作系统、数据库和应用等. 在这种需求的驱动之下，软件

中间件便成为网络环境下典型的软件运行平台. 软件中间件是在网络环境下处于操作系统等系统软件和应用软件之间的一种起连接作用的分布式软件, 抽象了底层分布环境(网络、主机、操作系统、编程语言)的复杂性和异构性, 使得网络化应用程序只需关注应用逻辑自身. 早期的中间件只关注异构网络环境下分布式应用软件的互连与互操作问题, 以提高应用系统的易移植性为主要目标. 随着网络环境的快速发展, 中间件的定义也走出了狭义的空间, 逐步形成更为广义的内涵, 呈现出新的发展趋势[7]. 首先, 原有各种不同功用的中间件, 如数据访问中间件、消息中间件、分布计算中间件等, 正在不断融合, 呈现出一体化的趋势. 其次, 中间件正在从传统的企业运算领域延伸到新的计算范型, 例如服务计算中的 Web Services, Web 2. 0 中的 REST、支持嵌入式和移动计算设备的轻量级中间件等等, 使得中间件广泛地存在于各种计算模型和系统[13]. 其三, 中间件开始具备自适应和自治管理的能力, 尽可能多地自动执行监测、分析、规划、实施等管理任务并尽可能自动化地衔接各个任务, 以更好地适应动态、多变的环境[14]. 例如, IBM 提出的自治计算, 以及我国学者提出的体系结构驱动的网构软件自适应技术[11], 均是针对自适应能力的工作.

尽管操作系统(面向单机环境)和中间件(面向网络环境)最初的出发角度不尽相同, 但是二者在功能上正呈现出互相渗透的趋势. 一方面, 中间件开始提供一些原操作系统的功能, 另一方面, 操作系统也开始向中间件能力发展[13]. 然而, 操作系统和中间件均产生于相对静态、封闭、固定的环境, 无法适应互联网环境更加复杂、动态和多变的特性, 这就对软件运行平台提出了更高的要求. 互联网环境下的软件运行平台必须支持构件化、服务化的软件模型, 提供更好的动态适应能力、开放演化能力和可信保障能力. 通过对互联网上的资源进行虚拟化, 软件运行平台需要从支持本地化进程管理转向支持网络化的软件协同, 从支持本地化的文件管理转向支持网络化的数据管理, 从支持本地化的设备管理转向支持网络化的资源聚合. 目前, Google 和 Microsoft 各自推出的云计算平台, 以及我国学者提出的“虚拟计算环境”和“网构软件中间件”, 都在这方面作出了有益的尝试.

4 软件开发方法的演变

软件开发方法决定了软件开发过程所应遵循的原则和方向, 其目的是尽可能地提高软件开发的效率和质量. 软件开发方法随着软件模型和软件运行平台的变化而变化. 例如, 结构化方法通过划分模块(子系统)并构造它们之间的控制关系来构造系统, 面向对象方法以对象和对象间关系构造系统, 基于构件的软件开发(component based software development, CBSD)通过组装可复用软件构件构造系统, 等等. 总体上看, 在封闭、静态的环境下, 软件开发过程基本采用自顶向下、逐步求精的途径, 确定系统的范围(即 scoping)总是需求分析的第一步, 然后通过分解而实施分而治之的策略, 整个开发过程处于有序控制之下.

在互联网环境下, 软件系统往往由分布在网络各个节点上的、具有高度构件化或服务化的软件实体构成, 以各种协同方式进行跨网络的互连、互通和协作, 并且需要适应网络环境的变化和满足用户的个性化需求. 相应地, 软件开发活动呈现为通过原本“无序”的基础软件资源组合为“有序”的基本系统, 随着时间推移, 这些系统和资源在功能、质量、数量上的变化导致它们再次呈现出“无序”的状态, 需要进行新一轮的有序化. 这种由“无序”到“有序”的过程往复循环, 基本上是一种自底向上、在线组装、渐趋稳态的途径. 互联网环境下的软件系统在规模、组成、时间、空间和使用方式上呈现的新型特性, 使得软件开发方法产生了显著的转变. 随着互联网上可复用的软件资源愈加丰富多样, 以复用为基础的软件构件/服务技术、模型驱动的开发方法成为软件技术的重要方向之一[12]. 例如, 服务计算中, 通过标准化的接口描述将软件发布成服务(如 Web Services), 然后按照一定的业务逻辑将其组装成为更大粒度的复合服务(如 WS-BPEL); 针对网构软件提出的开发方法 ABC(architecture-based component composition)[7]以构件组装为基本手段, 使用软件体系结构的理论与概念来指导软件开发的全生命周期, 以提高系统开发的效率和质量, 包括对互联网上“无序”软件资源的有效建模、组织和管理, 网构软件自适应建模技术, 以及支持网构软件特征的支持工具和运行平台. 出于积累可复用软件资源、提高软件开发效率的目的, 对可复用软件资源进行挖掘和组织日益得到学术界和工业界的重视. 目前, 以基于互联网的大型软件构件库为代表的软件基础资源库的构建, 正在成为软件开发组织甚至整个软件行业基础设施建设的重要

构成成分. 例如, 开放源代码软件社区如 Apache[2)]、Google 的开放应用程序接口库[3)]、北京大学的软件构件库[4)]等. 此外, 针对互联网环境下中小型软件项目或个人应用的需求, 新型的轻量级软件开发方法开始得到广泛关注. 例如, 支持敏捷开发过程的极限编程(Extreme Programming)、最终用户编程(End User Programming)[15]等.

5 软件质量目标和保障技术的演变

软件系统的最终目的, 是为人类提供更好的计算服务, 而衡量软件系统能力的指标就是软件质量. 按照 ISO-9126 标准的定义, 软件质量是软件产品满足规定和隐含需求的能力有关的特征和特性的总和. 好的软件质量始终是软件研发追求的目标, 相关研究主要围绕其质量目标和保障手段展开. 软件质量目标主要涵盖功能性(functionality)、可靠性(reliability)、效用性(efficiency)、可用性(usability)、可维护性(maintainability)、可移植性(portability)等具体质量属性和指标. 软件质量保障主要采用软件测试验证、开发保障和运行保障 3 种技术途径[16].

对于运行于静态、封闭、可控环境下的传统软件, 人们往往只关注软件质量属性的局部, 如正确性、安全性等, 即侧重指标相对单一的系统质量. 相应地, 软件质量保障技术体系就以系统质量为核心, 即, 软件测试与验证针对程序的正确性, 软件开发方法主要关注功能需求和功能正确性, 软件运行支撑则凝练共性应用功能以保障系统的正确运行.

在开放、动态、多变的互联网环境下, 软件系统往往由分散在网络各个节点并持续自主演化的软件实体构成, 实体间的交互方式更为复杂多样, 系统整体行为也更加难以预测. 同时, 以软件服务方式提供计算能力正在成为趋势. 这些新型的软件特性, 以及用户使用方式的变化和个性化需求, 使得人们对软件质量的要求发生了变化. 软件质量目标的重心开始从系统质量转移到使用质量, 软件可信性收到了高度关注. 近年来, 可信性研究十分活跃, 出现了如可信计算基(Trusted Computing Base)、可信计算(Trusted Computing)、开放式可信计算(Open Trusted Computing)、高可信软件(High Confidence Software)等多个代表性研究工作. 这些工作均从不同的使用层面和不同的综合化角度对“可信”进行了阐释. 我国学者在国家重点基础研究发展计划项目的支持下, 以网构软件为载体, 开展了网络环境下软件质量和保障技术的研究, 将软件的使用质量划分为服务质量和可信度. 其中, 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. 服务质量和可信度关注的是软件系统在使用过程中表现出来的具体属性, 相关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, 这就对软件质量保障技术提出了新的要求. 软件质量保障技术体系需要以系统质量为基础, 将使用质量置于核心地位, 综合考虑软件形态、环境特点、用户体验和满意度等多重因素. 软件开发需要在满足功能需求的同时保障可信度和服务质量, 从而需要建立新的软件开发方法及其质量保障体系; 运行支撑需要凝练共性管理功能以保证软件可信、高服务质量运行, 从而需要研究新型的软件运行平台及其质量保障机制.

6 讨论和展望

互联网的发展为信息技术的应用开启了新的篇章, 由此产生了从不同层次或角度研究互联网技术与应用的多种新型应用模式或计算范型. 而软件则为这些应用模式或计算范型提供了基础支撑. 本文从软件模型、软件运行支撑平台、软件开发方法、软件质量目标和保障技术 4 个角度分析当前开放网络环境下软件研究与实践的现状, 并总结相关发展趋势.

随着下一代网络(Next Generation Network)的基础框架日趋完善, 多网融合的大趋势使得软件系统将运行在一个包括互联网、无线网、电信网等多种异构网络的复杂网络环境之中, 也为软件技术提出了新的挑战: 从资源共享的角度, 需要解决跨网络的异构资源的协同共享问题, 实现统一、自治、可控的资源管理;从人机交互的角度看, 需要提供一致、和谐的交互方式, 实现无处不在的无缝计算; 从应用模式的角度看, 需要支持基于服务的架构, 实现主动可信的

2) Apache. http://www. apache. org
3) Google Code. http://code. google. com
4) 公共软件构件库. http://libs. gdsp. net:8005/notice. jsp

服务计算. 为了达到上述目标, 就需要建立一套适用于基于网络的复杂软件的新型软件理论、方法和技术体系, 为实现开放、动态、多变的网络环境中的可信软件系统提供支撑.

参考文献

1 Yang F Q, Lu J, Mei H. Technical framework for Internetware: An architecture centric approach. Sci China Ser F-Info Sci, 2008, 51: 610—622

2 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

3 Lu X C, Wang H M. Internet-based virtual computing environment (iVCE): Concepts and architecture. Sci China Ser F-Info Sci, 2006, 49: 681—701

4 Hendler J, Shadbolt N, Hall W, et al. Web science: An interdisciplinary approach to understanding the web. Commun ACM, 2008, 51: 60—69

5 杨芙清, 梅宏, 吕建, 等. 浅论软件技术发展, 电子学报, 2002, 30: 1901—1906

6 江泽民. 新时期我国信息技术产业的发展. 上海交通大学学报, 2008, 42: 1589—1607

7 Mei H, Huang G, Zhao H Y, et al. An architecture centric engineering approach to Internetware. Sci China Ser F-Info Sci, 2006, 49: 702—730

8 Satyanarayanan M. Pervasive computing: Vision and challenges. IEEE Pers Commun, 2001, 8: 10—17

9 Lu J, Ma X X, Tao X P, et al. On environment-driven software model for Internetware. Sci China Ser F-Info Sci, 2008, 51: 683—721

10 Balasubramaniam S, Lewis G, Simanta S, et al. Situated software: Concepts, motivation, technology, and the future. IEEE Softw, 2008, 25: 50—56

11 Mei H, Huang G, Lan L, et al. A software architecture centric self-adaptation approach for Internetware. Sci China Ser F-Info Sci, 2008, 51: 722—742

12 France R B, Rumpe B. Model-driven development of complex soft-ware: A research roadmap. In: Brand L C, Wolf A L, eds. Proceedings of Future of Software Engineering, 2007 May 23—25, Minneapolis MN. Los Alamitos, CA: IEEE Computer Society Press, 2007. 37—54

13 Issarny V, Caporuscio M, Georgantas N. A perspective on the future of middleware-based software engineering. In: Brand L C, Wolf A L, eds. Proceedings of Future of Software Engineering, 2007 May 23—25, Minneapolis MN. Los Alamitos, CA: IEEE Computer Society Press, 2007. 244—258

14 Krammer J, Magee J. Self-managed systems: An architectural challenge. In: Brand L C, Wolf A L, eds. Proceedings o f Future of Software Engineering, 2007 May 23—25, Minneapolis MN. Los Alamitos, CA: IEEE Computer Society Press, 2007. 259—268

15 Jazayeri M. Some trends in web application development. In: Brand L C, Wolf A L, eds. Proceedings of Future of Software Engineering, 2007 May 23—25, Minneapolis MN. Los Alamitos, CA: IEEE Computer Society Press, 2007. 199—213

16 Wang H M, Tang Y B, Yin G, et al. Trustworthiness of Internet-based software. Sci China Ser F-Info Sci, 2006, 49: 759—773