软件服务的在线演化

王怀民 史佩昌 丁 博 尹 刚 史殿习
(国防科学技术大学计算机学院 长沙 410073)

摘 要 软件服务的在线演化技术是当前可信软件研究的一个重要方向,对于实现快速、低成本的成长式可信演化具有重要意义.与离线的演化技术相比,在线演化强调软件系统在结构修改和功能调整期间仍能够持续提供服务.文中在给出软件服务在线演化基本定义、归纳其结构模型和一般性过程模型基础上,提出涵盖演化范畴、演化类型和演化方式等方面的分类模型,并以此分类模型为比较框架,对目前几种具有代表性的演化使能平台和可信演化系统做了综述和比较.最后对值得进一步研究的问题进行了分析和展望.

关键词 软件服务;在线演化;软件演化;可信软件
中图法分类号 TP311 **DOI 号**: 10.3724/SP.J.1016.2011.00318

Online Evolution of Software Services

WANG Huai-Min SHI Pei-Chang DING Bo YIN Gang SHI Dian-Xi
(*School of Computer Science, National University of Defense Technology, Changsha* 410073)

Abstract The technology to enable online evolution of software services is an important direction in the current research of trustworthy software, which is important for realizing quick, low cost and incremental trustworthiness-oriented evolution. In contrast with the offline evolution technology, it doesn't break down the services provided by the software while its structure and functionalities are modified. In this paper, on the basis of introducing the concept of online evolution of software services, inducing its basic structural model as well as its general process model, the authors present a taxonomy model for online evolution which covers the evolution scope, type and fashion. And the authors use this taxonomy model as a framework to summarize and compare a set of typical evolution-enabling platforms and trustworthiness-oriented evolution systems. Finally, the analysis and prospect of future research challenges are given.

Keywords software service; online evolution; software evolution; trustworthy software

1 引 言

当今社会越来越依赖于持续提供服务的大规模分布式软件系统[1-2].这类软件系统部署在开放网络环境中,各类要素交互协同,呈现出规模可持续成长、多目标相互制约、动态自适应等复杂系统新形态.与传统软件系统相比,其安全性、可用性与可靠性等可信问题变得越来越难以保证,特别是提供7×24持续服务的特性,使其越来越强调在运行中动态调整功能和结构,以确保在应对各种改变[3-7]的同时仍能持续提供服务.在线演化相关技术[8-14]是

收稿日期:2010-06-30;最终修改稿收到日期:2010-09-11.本课题得到国家自然科学基金(90818028)、国家"九七三"重点基础研究发展规划项目基金(2011CB302600)资助.**王怀民**,男,1962 年生,博士,教授,主要研究领域为分布式计算、信息安全和计算机软件. E-mail: whm_w@163.com. **史佩昌**,男,1981 年生,博士研究生,主要研究方向为分布式计算. **丁 博**,男,1978 年生,博士,主要研究方向为分布式计算、适应性软件. **尹 刚**,男,1975 年生,博士,讲师,主要研究方向为分布式计算、信息安全. **史殿习**,男,1966 年生,博士,副教授,主要研究方向为分布式计算、普适计算.

这类软件系统实现可信演化[1-2]的重要途径，甚至是必须的技术途径之一.软件服务的在线演化是在软件维护、演化、自适应等相关概念基础上衍生的一种粒度更细、关注点更集中的概念，其目的是在不中断软件系统所提供服务的前提下，提升软件系统的可信性、提高软件系统适应需求的能力.其相关技术主要解决两类核心问题：(1)如何使提供服务的软件系统具有运行时可被改变的能力；(2)如何利用这种能力来实现软件服务的在线演化，从而提升其可信性和适应需求的能力.

纵观软件演化技术的发展过程，从20世纪60年代提出软件演化概念至今，可以认为其经历了3个阶段.第1阶段是软件演化概念形成阶段(60年代～80年代中期)：该阶段Halpern[15]和Couch[16]首先提出了演化术语；Lehman和Belady比较了演化与维护在系统设计、修改方案、层次性质和操作主体等方面的区别与联系[17]，初步给出了软件演化的若干定律[18]，进一步明确了软件演化的必要性和内涵.第2阶段是软件演化研究的发育阶段(80年代中期～90年代后期)：软件演化的概念开始得到较为广泛的认可[19-20].研究者对为什么演化(Why)及如何演化(How)等问题展开初步探索，提出了Bohem螺旋模型[21]、Bennet分段模型[22]等演化过程模型，并且在程序语言[23-25]、体系结构[26-30]等层面开始考虑对在线演化的支持.这一阶段工作规模较小，演化动作缺乏组织性、目标单一.第3阶段是软件演化研究的跃升阶段(90年代后期至今)：主要研究探索大规模软件的演化，特别是部署在开放网络环境中、提供持续服务的大规模分布式软件系统的演化.其中具有明显里程碑意义的工作是Oreizy等所提出的基于体系结构的在线演化方法[31-32]，在其基础上衍生出了ArchStudio[33-34]、PKUAS[35-37]、Artemis-*[38-39]、OSGi[40]、Fractal[41]和Archware[42]等演化使能平台以及CASA[43]、MADAM[44]、K-Component[45-48]、Rainbow[3-4]和MDB[49]等利用软件系统运行时可改变能力来实现在线演化的可信演化系统.软件演化是一个年轻的领域，它的关注点甚至它的基础概念都在不断变化[6]，其实现技术、尤其是软件服务在线演化的实现技术远未成熟，仍需要深入研究.

软件服务在线演化的相关研究主要围绕软件维护[50-51]、演化[6]、适应[5]和反射[52]等相关问题展开，这些概念自提出以来就一直互有重叠.本文第2节首先给出软件服务在线演化的定义，提出软件服务在线演化的理想结构模型和一般性过程模型；第3节给出涵盖了演化范畴、演化类型和演化方式等方面的软件服务在线演化分类模型；第4节以在线演化分类模型为比较框架，系统梳理和比较若干具有代表性的研究项目；第5节对该领域值得进一步研究的问题进行分析和展望；最后一节总结全文.

2　基本概念和模型

2.1　基本概念

软件的变化可能发生在编译时、装载时或运行时等软件生命周期各个阶段[53].在线演化主要关注软件运行阶段的变化，目前尚无统一定义.近年来，软件自适应[5,54]、动态适应[12]和自组织[55]等方向的研究为在线演化赋予了更深刻的内涵.自适应[54]是指软件自我评价自身行为，当评价结果表明软件不能达成既定目标或需增加功能、优化性能时，改变自身行为.动态适应[12]是软件在运行时处理如下事件的一种适应能力：用户需求的改变，系统入侵或失效，运行环境和资源的变化等.自组织[55]是系统的一个动态适应过程，该过程中系统在不受外界控制的情况下通过自身内部元素的交互维持其结构.上述概念从演化能力、演化形态等方面丰富了在线演化的内涵.

文献[56]指出在线演化是这样一种软件演化：在不中断正在运行程序前提下升级该程序.程序持续运行是在线演化的根本标志；不中断则强调在用户请求不被取消或拒绝的前提下修改软件.综合已有工作和实际应用需求，本节以部署在开放网络环境中、提供持续服务的大规模分布式软件系统为背景，从技术机理层面给出软件服务在线演化的定义.

定义1(软件服务的在线演化).　软件服务的在线演化是指为了提高软件系统适应需求的能力，在不中断其所提供服务的前提下，自动或借助外部动态指导发生的软件改变活动.

这里，不中断是指对使用者而言，软件系统所提供的服务持续在线；借助外部动态指导则强调了整个在线演化过程中人作为不可或缺的因素参与其中；改变活动主要包括软件功能和结构等方面的改变，例如对软件体系结构拓扑及其元素(构件、连接子等[57-58])的修改活动.定义中没有显式定义非功能属性[59]演化，因为其一般会触发功能、结构等的演化.

上述定义所给的在线演化概念是一种运行时演

化,但其面向的是网络环境中的大规模分布式系统,更强调整个系统层面上“服务持续在线”这一特征,而不强求系统所有成分都必须持续运行.此外,与软件维护通常由专门维护人员来实施不同,在线演化动作既可以由第三方驱动,也可以由软件系统主动实施,如依据某些预定义的规则.

2.2 结构模型

结构模型给出了参与在线演化活动的实体及其交互关系,所要回答的核心问题是:如何使软件系统具备运行时可改变的能力(即演化使能)以及由谁、以何种方式来驱动演化过程(即演化决策).本节对现有研究和工程实践进行总结提炼,给出特定、基于平台和基于引擎三种在线演化结构模型,在此基础上提出软件服务在线演化的理想结构模型.这些模型均可被分为演化使能和演化决策两部分,在图 1 中使用深灰和浅灰区分.此外,图 1 中实线交互关系是不可缺少的,虚线交互关系是可选的.

(a)特定演化模型. 早期的演化技术直接将与演化相关的监测模块和执行模块硬编码到应用系统中,监测模块可以输出应用系统的内部状态信息,执行模块则可以通过参数重配置等形式对应用系统实施在线调整.演化决策通常由管理维护人员完成.此类模型如图 1(a)所示,其代表性工作主要集中于小规模、单目标和无系统层支持的软件演化研究[23-30].

(b)基于平台的演化模型. 随着中间件等基础软件的广泛应用,软件开发与运行平台对软件演化的支持有了显著提高.这使得演化使能机制从应用程序中分离出来,成为相对独立的公共支撑设施.这种分离提高了软件复用程度,增强了软件系统的在线演化能力.此类模型如图 1(b)所示,其代表性工作主要集中于在线演化研究中的演化使能平台[33-42].

(c)基于引擎的演化模型. 相对于基于平台的模型,这类模型的主要特点是增加了演化决策模块,试图实现一种通用的软件演化引擎,代替人来实时、自动化地做出演化决策.这将大幅提升软件演化的自主性和时效性,但在具体实践中,由于对演化引擎能力的期望过于理想化,反而限制了其应用范畴.此类模型如图 1(c)所示,其代表性工作主要集中于在线演化研究中的可信演化系统[4,43-49].

(d)理想结构模型. 软件服务在线演化的理想结构模型是(b)类模型和(c)类模型的有机组合,即演化决策可以由软件系统依据事先制定的规则等做出,也可以在必要时由人来驱动演化决策,如图 1(d)所示.该模型综合了(b)类和(c)类模型的优点,回答了软件系统本身应该为在线演化承担何种责任和提供何种程度支持的问题.鉴于现实中大量应用系统部署在不支持软件演化的平台上,因此如何利用这种理想模型解决遗留系统的演化也是目前过渡时期面临的一个重大挑战.

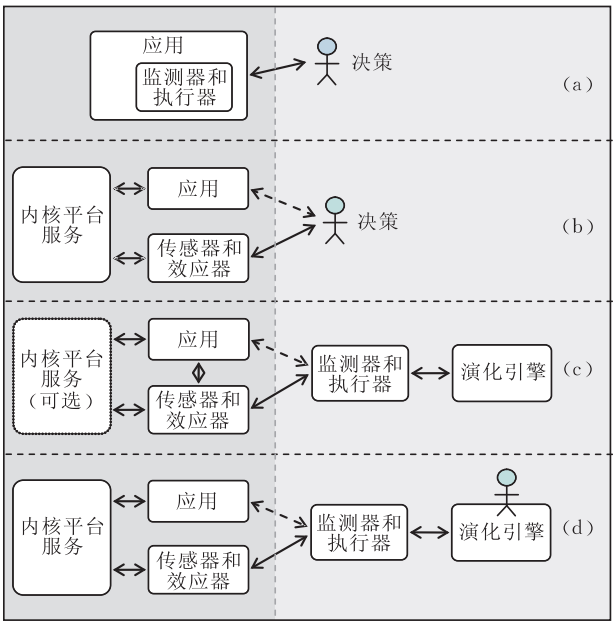
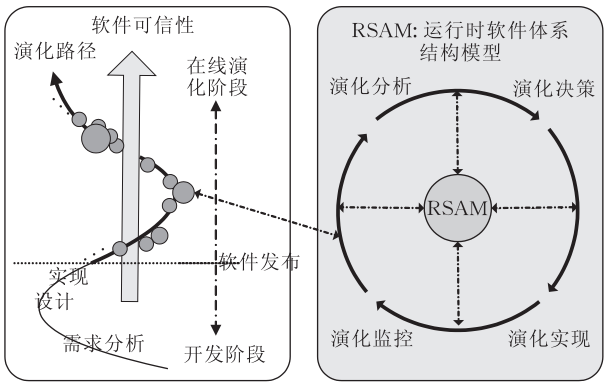


图 1 在线演化结构模型

2.3 过程模型

文献[1]认为形式化方法不能在软件系统发布前完成其可信性证明,测试方法也因找不到完备的测试集而无法全面验证软件的可信性.历时 22 年总结出的 Lehman 定律[18,60-63]和自组织临界性理论[64]从不同角度证明了软件系统必定是不断演化的.“演化论”比“构造论”更能表达软件系统该方面的特性.Bennett[22]将“演化论”下软件系统的生命周期划分为初始开发阶段、演化阶段、维持阶段和淡出阶段.梅宏等[65]在传统软件系统生命周期的基础上引入了“后开发阶段”,主要包含维护、演化、复用等方面.从“演化论”下软件系统生命周期的相关研究可以看出,演化已经成为软件生命周期中的一个重要阶段.

如何认识“演化论”下在线演化的基本过程是一个很重要的方面.在 Kephart 模型[66]和 Dobson 自适应六过程闭循环[67]、Oreizy 适应管理周期[32]等研究工作基础上,本文提出了处在运行阶段的在线演化过程模型.图 2 结合软件服务在线演化的螺旋上升曲线,给出的这一模型.



(a) 在线演化螺旋上升曲线　　(b) 在线演化一般性过程模型

图 2　在线演化过程模型

图 2(a)为软件服务在线演化的螺旋上升曲线，运行中软件系统可信性的提升符合文献[68-69]中产品质量螺旋上升的客观规律. 图中软件发布前为软件开发阶段，包括需求分析、设计和实现等；软件发布之后的在线演化阶段主要由一系列不断提升可信性的演化活动组成. 图 2(a)中每个大小不等的圆映射到图 2(b)中均为一个在线演化基本过程，即由监控、分析、决策和实施所构成的循环. 演化基本过程中的每一步骤都与运行时软件体系结构模型 RSAM(Runtime Software Architecture Model)密切相关，体现了体系结构模型[51,70]对在线演化过程的重要指导作用. 图 2(b)中在线演化基本过程每循环一次，软件可信性就能得到维持或提升，能够更加适应用户需求. 文献[71]指出，在实施演化前，一般需要对所制定演化方案的可行性进行分析和评估；演化实施后还需要对演化效果及开销进行评估. 加上上述两类评估，在线演化基本过程模型覆盖了软件服务在线演化所要解决的核心问题，对在线演化研究工作具有一定的宏观指导意义.

3　在线演化分类模型

软件演化最初的分类是从软件维护相关分类借鉴而来[15,17-18]，典型的软件演化相关分类模型有 Swanson[50]、IEEE[72]、Chapin 等人[73]、Wang 等人[56]、Mens 等人[53,76]和 Salehie 等人[5]的. 上述分类分别从主观目的、客观凭证和演化原因等角度对软件演化、改变、自适应和维护等进行了分类. 虽然分类方法是软件演化领域的一个研究热点，但鉴于目前软件演化尚无统一定义，演化分类方法也难有一致标准. 综合已有研究和本文所给软件服务在线演化的定义，本节提出一种从演化范畴、演化类型和演化方式等 3 个方面对在线演化活动进行分类的分类模型.

3.1　演化范畴

演化范畴从演化粒度、功能相关性和层次 3 个维度刻画在线演化活动的目标对象.

从软件体系结构的角度，演化粒度可分为构件、连接子和配置演化. 构件演化是指业务模块的参数重配置、添加、删除、替换等操作；连接子演化是指对业务模块间交互关系的调整；配置演化则涉及到多个构件和连接子的演化，可能还伴有体系结构约束[70]等其它元素的演化. 分布式系统中具有本地内存、相互之间通过消息进行通信的实体被称为结点，相应地，其演化层次可以区分为单结点和跨结点演化：单结点是指演化活动仅在单个结点内部进行；跨结点演化则涉及到多个结点. 例如 OSGi 主要关注单个结点内部的演化活动，Rainbow 则可以通过建立严格的分层控制方式来实现跨结点演化活动. 功能主要用来区分演化是否会改变软件系统对外所提供的服务：如果仅对代码的格式、文档等进行修改，或者代码修改仅仅是为了重构等目的，则称之为功能无关的演化，反之则称为功能相关的演化.

3.2　演化类型

演化类型从主动性、可预期性和自动化程度 3 个维度刻画在线演化活动的外部特征.

演化主动性可以分为反应型和前摄型两种. 反应型演化是指在外部压力刺激下做出的演化活动，例如用户需求变化后对软件实施的必要修改；前摄型演化则是指在对软件系统可能面临风险进行预测的基础上，所实施的有前瞻性的演化活动. 演化预期性可以分为两种：预期型是指在软件开发阶段即可预期的改变活动，其实现相对简单，例如可以通过在应用系统中预留扩展点等形式来支持；非预期型则是开发阶段无法预期的在线改变活动，一般而言需要在开发工具、平台、语言、构件模型等层面上给予支持. 自动化程度方面，自动化的演化过程是一个不需人参与的闭循环，由软件依据开发阶段所预设的策略做出演化决策；人参与交互式演化则是一个需要人参与的开循环，由人来全部或者部分地做出决策.

3.3　演化方式

演化方式从一般性、体系结构和演化策略 3 个维度刻画在线演化活动的内部实现机制.

依据是否与特定体系结构风格紧密耦合，演化实现机制可以分为两类：具有通用性的一般(general)方法；仅能用于特定系统、解决特定问题的特定方

法.例如 Rainbow 中的演化机制可适用于不同的体系结构风格,具备一般性;ArchStudio 主要针对 C2、REST 等软件体系结构风格,则可以划分为特定方法.在体系结构方面,基于体系结构的方法是指在软件运行过程中维护显式的体系结构模型,在该模型指导下实施演化活动;非基于体系结构的方法则不强调显式的体系结构模型维护,例如没有模型指导的动态 AOP(Aspect-Oriented Programming)、Java Hotswap 机制[25]等.演化策略主要针对自身具有决策能力的可信演化系统,关注自动化演化过程中如何做出决策,可以分为基于规则和基于学习两类.前者一般在开发阶段预测系统可能碰到的各种问题,并设置规则(动作策略、效用函数等);后者则可以通过机器学习等方法在运行时扩展其决策能力.

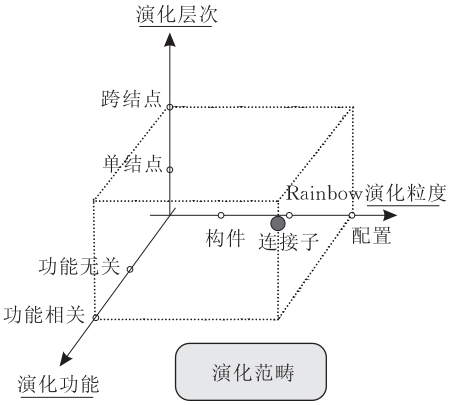


图 3 在线演化分类模型之演化范畴

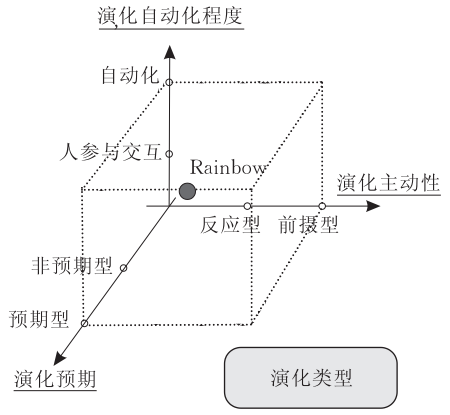


图 4 在线演化分类模型之演化类型

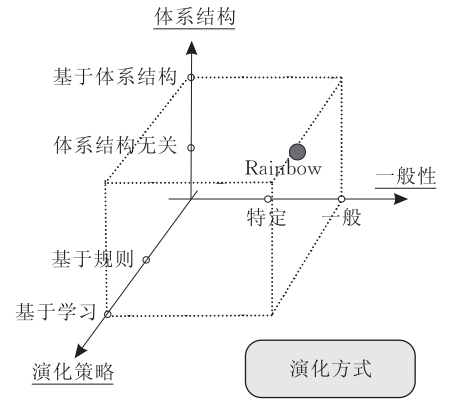


图 5 在线演化分类模型之演化方式

4 典型项目概述与比较

本节以在线演化分类模型为基本框架,对目前有代表性的典型项目进行梳理分析,重点比较它们在演化范畴、演化类型和演化方式等方面的异同.通过对这些系统的分析比较,总结该领域的研究现状.其中演化使能平台对应于图 1(b)基于平台的演化模型,可信演化系统对应于图 1(c)基于引擎的演化模型.

4.1 演化使能平台

4.1.1 OSGi 框架[40]

OSGi(Open Services Gateway Initiative)是由 OSGi 联盟制定的服务平台规范,其核心组件是 OSGi 框架.OSGi 框架实现了面向服务的构件模型——应用由高度模块化、可动态管理的服务构件(被称为 Bundle)组织而成,框架则为 Bundle 运行提供一个通用基础设施.OSGi 框架对在线演化的支持突出表现在 Bundle 的动态管理机制上:在 OSGi 框架中,Bundle 的安装、加载、启动、停止、卸载都可以在运行时动态实施,服务由哪个 Bundle 提供也是由框架中的动态发现机制确定的,因此可以在运行时对部分 Bundle 进行修改更新,而无需重启整个系统.

4.1.2 Fractal 构件模型及其平台[41]

Fractal 是一种支持反射的构件模型.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,实现构件的控制接口.控制接口的存在使得构件的行为和结构可以在运行时内省或调整,并且这种反射能力并未固化于构件模型中,应用开发者可根据其需求来进行扩展.通过反射,Fractal 可以支持软件系统的在线演化,例如当用户需求变化时对现有系统进行调整,或者将现有系统动态集成到不同的环境中等.文献[41]同时给出了一个支持 Fractal 构件模型、基于 Java 技术的运行平台 JULIA.

4.1.3 Artemis-* 平台

Artemis-* 平台包括 Cogent[39]、M3C/MAC[74]和 ARC[59]等.Artemis-M3C 基于截获器(Interceptor)机制,构建了 PCM(Programmable Coordina-

tion Media)元层来描述基层服务软件实体的协同行为，并通过 PCM 的编程配置，提高服务软件实体对开放网络环境的动态适应能力. 自演化软件使能支撑平台 Artemis-MAC 采用了 RDF/OWL 技术对情境进行建模，为感知决策提供了一个统一的框架，并将体系结构层面推理的动作序列直接作用于体系结构对象. Artemis-ARC 是面向体系结构的软件服务平台，由体系结构元模型提供的动态体系结构视图指导应用的运行时变化，能够为服务的集成和演化提供有效支撑.

4.1.4　PKUAS 平台[35-37]

PKUAS 是基于体系结构的反射式构件中间件平台，能够支持中间件及其上应用系统状态与行为的在线观测调整. PKUAS 引入软件体系结构以实现反射体系对系统整体的表示和控制，主要解决的问题有两类：现有中间件平台过于注重系统局部或单个实体的反射而缺乏全局视图；过于注重中间件平台内部功能的反射而对上层应用反射不够. PKUAS 系统中的实体可分为系统构件(如容器系统、公共服务、工具)和微内核两类，其中微内核负责对系统构件的加载、配置、卸载以及启动、停止、挂起等状态管理.

4.1.5　ArchStudio 平台[33-34]

ArchStudio 平台是基于软件体系结构来实现软件系统开发和演化的环境，整合了体系结构描述、体系结构变化的表示、演化结果呈现、演化规划执行等工具. 早期 ArchStudio 的工作基于 C2 体系结构风格，在其后续版本中，陆续增加了对 xADL 等体系结构描述语言的支持以及基于知识的自适应管理等. 由于 C2 本身是一种较灵活的动态体系结构风格，所以 ArchStudio 可以较好地支持体系结构演化. ArchStudio 在实施演化计划之前，一般会评估演化方案，包括演化效果的预测分析和演化的影响评估等.

4.2　可信演化系统

4.2.1　K-Component[45-48]

K-Component 的目标通过自底向上方式创建具有适应性的分布式系统，所谓“自底向上”是指各个结点具备自管理和自演化能力，结点间则通过协同来实现系统层面上的自管理. K-Component 的工作包括：(1)支持反射的构件模型；(2)体系结构反射协议及具体化该协议的容器，容器在运行时维护显式的体系结构模型，依据适应契约(adaptation contract)驱动演化动作；(3)协同强化学习(collaborative reinforcement learning)机制，实现从个体自演化到群体自演化的过渡. 在 K-Component 中，由于每个构件只拥有局部视图，因此单个构件演化能力是受限的[74]，而系统层面上的协同强化学习方法也只能适用于某些特定场景[47].

4.2.2　RainBow[3-4]

Rainbow 的目标是构建一个以体系结构为中心、可重用的自适应软件基础设施，使得遗留系统或新开发系统可以以较低的代价获得可信演化能力. Rainbow 在运行时维护一个显式的体系结构模型，基于该模型所收集的信息和预定义的策略触发自适应动作. Rainbow 的总体架构由系统层、翻译设施和体系结构层组成，每一层又包括若干部件及对这些部件进行定制的实体(如策略、映射规则、操作子等). 这一架构对于软件重用的支持主要体现在：(1)体系结构层设施是与具体体系结构风格无关的，它通过翻译设施与应用系统交互；(2)可以通过 Acme 体系结构描述语言和 Stitch 策略语言来指定策略、操作子等，从而使基础设施为不同应用服务；(3)Rainbow 也支持系统层设施、翻译设施及系统自适应知识的重用[3].

4.2.3　MBD 系统[49]

MBD 是一种基于模型诊断实施在线演化的框架，利用对上下文的诊断降低演化复杂性，提高软件鲁棒性. MBD 模型诊断的主要作用包括：诊断异常行为的源头并纠正偏离行为；诊断程序失效和性能缺失；为重新制定目标和上下文重配置提供依据；为新策略的选择提供依据；使程序具有理解、监控和修改自身行为的能力等. MBD 与传统的离线、需要人参与的演化不同，主要表现在：在运行时检测演化需求，并实施所需演化；采用自建模方式，其自建模程序包含在运行程序中，这体现了 MBD 内在驱动演化和自动化操作等特性；评估、修订和重配置等都由软件操作模型驱动；每个构件都有描述信息，以便系统的构件可在运行时选择和调度；每种描述都以模型形式显现，即描述必须包含一个重要的功能抽象，以便支持描述所涉及的操作.

4.2.4　MADAM[44]

MADAM(Mobility- and Adaptation-Enabling Middleware)是面向移动环境下自适应软件构造和运行的中间件. 在 MADAM 的设计中，同一构件可以有适合不同场景的多个实现体，在线演化主要体现为依据上下文和预定义效用函数切换实现体的过程，例如当应用迁移到不同设备上时加载不同人机

交互界面. MADAM 为此维护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,加载了哪些具体的实现体. 在运行时,MADAM 依据当前环境状态和效用函数进行规划,检查理想体系结构配置与当前实例体系结构模型的差异,据此进行实现体切换等演化操作.

4.2.5 UbiStar[75]

UbiStar 是支持自适应软件开发、运行和管理的平台. 在 UbiStar 平台上,应用由感知构件、行为构件和包括策略连接子等在内的各类连接子组装而成. 其中,感知构件负责感知环境变化,行为构件负责实现业务逻辑,策略连接子则指定了在环境变化时执行何种适应行为(如参数调整或构件增删替换). 在运行时刻,平台中的 Auxo 软件框架维护显式的环境模型和体系结构模型,并基于这些模型和策略连接子的内容来驱动软件的在线变化. 同时,UbiStar 也支持第三方在必要时对正在运行软件的体系结构进行在线调整,部分地体现了本文所给软件演化理想结构模型的思想.

4.3 比 较

前述典型项目所支持的软件演化活动及其实现机制如表 1 所示.

表 1 在线演化相关部分典型项目的比较

项目	演化范畴			演化类型			演化方式		
	粒度	层次	功能	主动性	预期	自动化程度	一般性	体系结构	策略
OSGI	构件	单结点	相关	第三方决定	非预期	人参与交互	特定	无关	—
Fractal	构件	单结点	相关	第三方决定	非预期	人参与交互	特定	无关	—
Artemis-ARC	配置	跨结点	相关	第三方决定	非预期	人参与交互	一般	有关	—
PKUAS	配置	跨结点	相关	第三方决定	非预期	人参与交互	一般	有关	—
ArchStudio	配置	单结点	相关	第三方决定	非预期	人参与交互	特定	有关	—
K-Component	配置	跨结点	相关	反应型	预期+非预期	自动化	特定	有关	规则+学习
Rainbow	配置	跨结点	相关	反应型	预期	自动化	一般	有关	基于规则
MBD	配置	单结点	相关	前摄型	预期	自动化	—	有关	基于规则
MADAM	构件	单结点	相关	反应型	预期	自动化	特定	有关	基于规则
UbiStar	配置	跨结点	相关	反应型	预期+非预期	自动化+人参与交互	特定	有关	基于规则

注:以上各栏均基于各项目引用率较高的经典参考文献,某些项目后期所做的扩展工作未计入.

在演化范畴方面,现有项目可以从构件、配置等不同粒度上为在线演化活动提供支持;部分项目在设计时考虑了对跨结点演化活动的支持;所有项目均能够支持软件系统功能的演化,而大多功能相关的演化都会涉及功能无关的演化,例如对源代码的可读性做不改变语义的调整、对软件文档的修改补充等.

在演化类型方面,可以从演化使能平台和可信演化系统两个角度进行分析. 对于演化使能平台,由于实施何种演化活动是由作为第三方的管理维护人员决定的(参见图 1(b)),因此演化活动的性质也取决于第三方,且这些演化活动往往是开发阶段无法完全预期的. 可信演化系统可以自动化地实施某些演化活动,但目前的工程实现仍以相对较为简单的、基于规则的反应型为主,其能力受制于开发阶段所给定的演化规则. 这导致软件服务很难应对开放环境的挑战,因为开放环境意味着很难在开发阶段预期到所有可能的环境变化[76]. 结合演化方式一栏中相关赋值可以看出,解决这一问题目前有两种基本途径:采用机器学习等人工智能方法(如 K-Component),但此类方法通常只适用于某些特定的应用场景[47];在平台/系统层面允许第三方对演化策略等进行动态部署和修改(如 UbiStar),从而在线扩展软件的可信演化能力.

在演化方式方面,近期在线演化研究工作大多是基于体系结构的,这是因为一方面体系结构信息对软件演化有指导作用,另一方面运行时体系结构模型与实际系统因果关联,可以作为体系结构修改的入口. 此外,具有通用性的一般性演化方法在理论方面研究较多,而绑定到体系结构风格的特定方法在实现层面的研究较多.

5 未来研究趋势

综合目前在线演化相关工作的研究现状以及大规模分布式系统本身的规模持续成长、多目标互相制约、动态自适应等特点,本文认为软件服务在线演化的未来研究趋势主要包括:

(1) 大规模系统在线演化的改变管理技术

改变管理(change management)涉及到演化改变范畴的确定、改变前后的一致性保证、改变动作的开销等一系列问题[31,77]. 在软件系统持续改变和复

杂性渐增[6,18]等特点越来越明显的情况下，改变管理对于大规模分布式软件系统在线演化十分重要.一方面，在演化改变边界确定的情况下，大型复杂演化问题可以分解为局部的演化问题，有助于在线演化的实现.另一方面，部署在开放网络环境中的大规模分布式系统具有与传统软件系统所不同的特点，例如失效常态性、潜在的目标冲突、可在线成长等[78]，这些特点使传统改变管理技术中可建立全局视图、需维护严格一致性等假设有可能不再成立，为改变管理带来挑战的同时也带来了新的机遇.

(2)复杂分布环境下的在线演化决策技术

演化决策是指在给定演化需求、约束和策略等之后，基于当前系统所处状态，给出一个有关如何实施演化的操作序列的过程[5,32,79].演化决策技术是实现软件可信演化最有挑战性的技术之一，这不仅仅是因为目前还不能期望软件完全“自主”地做出各种决策，同时也是由复杂分布式环境的特点所决定的，例如：在复杂分布式环境下可能无法获得全局视图，诸如非线性规划等定量决策技术可能无法在系统层面上直接应用；演化决策需要聚合来自不同结点的体系结构和状态信息，而在结点间共享这些信息可能会限制系统可伸缩性[80]；在结点属于不同管理域的情况下，各个结点的决策结果需要兼顾全局效果和个体偏好，因而可能需要引入准则制约的交互(law-governed interaction)[81]、涌现行为控制[82]等机制，等等.

(3)在线演化的预评估和效果评估技术

要求软件系统能够持续提供符合用户预期的服务，对演化动作的效果进行评估是必须的.这种评估可以发生在演化动作实施之前，以防患于未然，例如为了保证演化过程中事务一致性而预先进行的必要检查[76]；也可以发生在演化动作实施之后，以确定演化目标是否已达到，并为后续演化动作提供依据.文献[83]已经证明完全自动化的通用演化评估方法是计算不可判定的.如何针对具体类型的软件演化动作，设计合理的评估机制是未来研究重点.

(4)软件服务在线演化基础设施

部署在开放网络环境中的大规模软件系统本身极为复杂.要将在线演化理想结构模型推向工程实践，合理的软件基础设施是实现“化繁为简”的关键.基础设施能够以重用部件形式沉淀改变管理、演化决策、演化评估等技术，有效降低软件服务在线演化的开发和运行成本.尽管本文第4节给出了若干此类基础设施，然而它们距离实际需求尚有差距.例如，基于体系结构技术的基础设施是当前工作主流，但现有项目大多需要维护全局层面上的体系结构模型[4,44]，这在跨多个管理域的分布式系统中并不一定成立；基础设施如何应对环境开放性带来的挑战等问题也需要进一步的探索[75-76]，等等.

6　结束语

本文以部署在开放网络环境中、提供持续服务的大规模分布式系统为背景，给出了软件服务在线演化的定义，指出了其与一般软件运行时演化概念在侧重点上的不同之处.在此基础上，对在线演化的现有结构模型进行了抽象提炼，提出了软件服务在线演化的理想结构模型；对在线演化的螺旋上升规律进行了分析，提出了软件服务在线演化的一般性过程模型.本文进而给出了涵盖演化范畴、演化类型和演化方式等方面的软件服务在线演化分类模型，并以其为框架，系统梳理和比较了目前几种具有代表性的演化使能平台和可信演化系统.最后结合目前软件服务在线演化的实际应用需求和研究现状，从改变管理、演化决策、演化评估、基础设施等方面分析了软件服务在线演化技术的未来研究趋势.

参　考　文　献

[1] Wang Huai-Min, Yin Gang. The trustworthiness-oriented software evolution in the network era. Communications of the CCF, 2010, 6(2): 28-34(in Chinese)
(王怀民，尹刚. 网络时代的软件可信演化. 中国计算机学会通讯，2010，6(2)：28-34)

[2] Liu Ke, Shan Zhiguang, Wang Ji et al. Overview of major research plan of trustworthy software. Science Foundation in China—Discipline Progress and Prospect, 2008, 22(3): 145-151(in Chinese)
(刘克，单志广，王戟等. “可信软件基础研究”重大研究计划综述. 中国科学基金，2008，22(3)：145-151)

[3] Cheng S W, Huang A C, Garlan D, Schmerl B R, Steenkiste P. Rainbow: Architecture-based self-adaptation with reusable infrastructure. IEEE Computer, 2004, 37(10): 46-54

[4] Cheng S W. Rainbow: Cost-effective, architecture-based self-adaptation [Ph. D. dissertation]. Carnegie Mellon University, Pittsburgh, 2008

[5] Salehie M, Tahvildari L. Self-adaptive software: Landscape and research challenges. ACM Transactions on Autonomous and Adaptive Systems, 2009, 4(2): 1-42

[6] Godfrey M W, German D M. The past, present, and future of software evolution//Proceedings of the International Conference on Software Maintenance. Beijing, 2008: 129-138

[7] Georgas J C, Van der Hoek A, Taylor R N. Using architectural models to manage and visualize runtime adaptation. IEEE Computer, 2009, 42(10): 52-60

[8] Taylor R N, Medvidovic N, Oreizy P. Architectural styles for runtime software adaptation//Proceedings of the Joint Working IEEE/IFIP Conferenceon Software Architecture & EuropeanConferenceon Software Architecture. Cambridge, 2009: 171-180

[9] Oreizy P, Medvidovic N, Taylor R N. Runtime software adaptation framework, approaches, and styles//Proceedings of the International Conference on Software Engineering. Leipzig, 2008: 899-910

[10] Salehie M, Li S, Tahvildari L. Employing aspect composition in adaptive software systems: A case study//Proceedings of the Workshop on Linking Aspect Technology and Evolution. Charlottesville, 2009: 17-21

[11] Dusparic I, Cahill V. Using distributed W-learning for multi-policy optimization in decentralized autonomic systems//Proceedings of the International Conference on Autonomic Computing. Barcelona, 2009: 63-64

[12] Fox J, Clarke S. Exploring approaches to dynamic adaptation//Proceedings of the International DiscCoTec Workshop on Middleware-Application Interaction. Lisbon, 2009: 19-24

[13] Saudrais S, Staikopoulos A, Clarke S. Using specification models for runtime adaptations//Proceedings of the International Workshop on Models@run. time. Denver, 2009: 1-15

[14] Fritsch S, Senart A, Schmidt D C, Clarke S. Scheduling time-bounded dynamic software adaptation//Proceedings of the International Workshop on Software Engineering for Adaptive and Self-Managing Systems. Leipzig, 2008: 89-96

[15] Halpern M. The evolution of the programming system. IEEE Datamation, 1964, 10(7): 51-53

[16] Couch R F. Evolution of a toll MIS at bell Canada//Proceedings of MIS. Copenhagen, 1971: 163-188

[17] Lehman M M, Belady L A. Program Evolution: The Process of Software Change. New York: Academic Press, 1985

[18] Lehman M M. On understanding laws, evolution and conservation in the large program life cycle. Systems and Software, 1980, 1(3): 213-221

[19] Arthur L J. Software Evolution: The Software Maintenance Challenge. Hoboken: John Wiley & Sons, 1988

[20] Oman P W, Lewis T G. Milestones in Software Evolution. New York: IEEE Computer Society Press, 1990

[21] Boehm B W. A spiral model of software development and enhancement. IEEE Computer, 1988, 21(5): 61-72

[22] Bennett K H, Rajlich V T. Software maintenance and evolution: A roadmap//Proceedings of the International Conference on Software Engineering. Limerick, 2000: 73-87

[23] Gupta D, Jalote P, Barua G. A formal framework for on-line software version change. IEEE Transactions on Software Engineering, 1996, 22(2): 120-131

[24] Hicks M. Dynamic software updating [Ph. D. dissertation]. University of Pennsylvania, Pennsylvania, USA, 2001

[25] Dmitriev M. Safe class and data evolution in large and long-lived java applications [Ph. D. dissertation]. University of Glasgow, Glasgow, UK, 2001

[26] Inverardi P, Wolf A L. Formal specification and analysis of software architectures using the chemical abstract machine model. IEEE Transactions on Software Engineering, 1995, 21(4): 373-386

[27] Metayer D L. Describing software architecture styles using graph grammars. IEEE Transactions on Software Engineering, 1998, 24(7): 521-553

[28] Luckham D C, Vera J. An event-based architecture definition language. IEEE Transactions on Software Engineering, 1995, 21(9): 717-734

[29] Magee J, Kramer J. Dynamic structure in software architectures//Proceedings of the 4th ACM SIGSOFT symposium on Foundations of Software Engineering. New York, USA, 1996: 3-14

[30] Allen R J, Douence R, Garlan D. Specifying and analyzing dynamic software architectures//Proceedings of the Conference on Fundamental Approaches to Software Engineering. Lisbon, 1998

[31] Oreizy P, Medvidovic N, Taylor R. Architecture-based runtime software evolution//Proceedings of the International Conference on Software Engineering. Kyoto, 1998: 177-186

[32] Oreizy P, Gorlick M, Taylor R et al. An architecture-based approach to self-adaptive software. IEEE Intelligent Systems, 1999, 14(3): 54-62

[33] Dashofy E M, Hoek A, Taylor R. Towards architecture-based self-healing systems//Proceedings of the Workshop on Self-Healing. Charleston, 2002: 21-26

[34] Georgas J C, Taylor R. Towards a knowledge-based approach to architectural adaptation management//Proceedings of the ACM SIGSOFT Workshop on Self-Managed Systems. New York, USA, 2004: 59-63

[35] Mei H, Huang G. PKUAS: An architecture-based reflective component operating platform//Proceedings of the International Workshop on Future Trends of Distributed Computing Systems. Suzhou, 2004: 26-28

[36] Huang G. Research on theory and technology of reflective middleware [Ph. D. dissertation]. Peking University, Beijing, 2003(in Chinese)
(黄罡. 反射式软件中间件原理与技术研究[博士学位论文]. 北京大学, 北京, 2003)

[37] Huang Gan, Wang Qian-Xiang, Mei Hong, Yang Fu-Qing. Research on architecture-based reflective middleware. Journal of Software, 2003, 14(11): 1819-1826(in Chinese)
(黄罡, 王千祥, 梅宏, 杨芙清. 基于软件体系结构的反射式中间件研究. 软件学报, 2003, 14(11): 1819-1826)

[38] Lu Jian, Zhang Ming, Liao Yu, Tao Xian-Ping. Research on component software framework based on mobile Agent technology. Journal of Software, 2000, 11(8): 1018-1023(in Chinese)

(吕建，张鸣，廖宇，陶先平. 基于移动 Agent 技术的构件软件框架研究. 软件学报，2000，11(8)：1018-1023)

[39] Hu Hai-Yang, Yang Mei, Tao Xian-Ping, Lu Jian. Research and implementation of late assembly technology in Cogent. Chinese Journal of Electronics, 2002, 30(12): 1823-1827(in Chinese)
(胡海洋，杨玫，陶先平，吕建. Cogent 后组装技术研究与实现. 电子学报，2002，30(12)：1823-1827)

[40] OSGi Service Platform Specification Release 4.2. OSGi Alliance, 2009

[41] Bruneton E, Coupaye T, Leclercq M et al. An open component model and its support in java//Proceedings of the International Symposium on Component-Based Software Engineering. Ddinburgh, 2004: 7-22

[42] Oquendo F, Warboys B, Morrison R et al. Archware: Architecting evolvable software//Proceedings of the European Workshop on Software Architecture. Andrews, 2004: 257-271

[43] Mukhija A, Glinz M. Runtime adaptation of applications through dynamic recomposition of components//Proceedings of the International Conference on Architecture of Computing Systems. Innsbruck, 2005: 124-138

[44] Floch J, Hallsteinsen S, Stav E et al. Using architecture models for runtime adaptability. IEEE Software, 2006, 23(2): 62-70

[45] Dowling J, Cahill V. Dynamic software evolution and the K-Component model//Proceedings of the Workshop on Software Evolution. Vienna, 2001: 1-6

[46] Dowling J, Cahill V. The K-Component architecture meta-model for self-adaptive software//Proceedings of the International Conference on Metalevel Architectures and Separation of Crosscutting Concerns. Kyoto, 2001: 81-88

[47] Dowling J. The decentralized coordination of self-adaptive components for autonomic distributed systems [Ph. D. dissertation]. Dublin: Trinity College, 2004

[48] Dowling J, Cahill V. Self-Managed decentralized systems using k-components and collaborative reinforcement learning//Proceedings of the ACM SIGSOFT Workshop on Self-managed Systems. Newport Beach, 2004: 39-43

[49] Robertson P, Laddaga R. Model based diagnosis and contexts in self-adaptive software//Proceedings of the Conference on Self-* Properties in Complex Information Systems. Bertinoro, 2005: 112-127

[50] Swanson E B. The dimensions of maintenance//Proceedings of the International Conference on Software Engineering. San Francisco, 1976: 492-497

[51] IEEE standard glossary of software engineering terminology. IEEE Standard 610.12-1990. IEEE, 1990

[52] Smith B C. Reflections and semantics in a procedural language [Ph. D. dissertation]. Cambridge, MA: Massachusetts Institute of Technology, 1982

[53] Mens T, Buckley J, Rashid A et al. Towards a taxonomy of software evolution//Proceedings of the Workshop on Unanticipated Software Evolution. Warsaw, 2003: 50-59

[54] Laddaga R. Self-adaptive software. New York: The Defense Advanced Research Projects Agency, TR: 98-12, 1998

[55] Wolf T D, Holvoet T. Emergence and self-organization: A statement of similarities and differences//Proceedings of the International Workshop on Engineering Self-Organizing Applications. New York, 2004: 96-110

[56] Wang Q X, Shen J R, Wang X P et al. A component-based approach to online software evolution. Journal of Software Maintenance and Evolution: Research and Practice, 2006, 18(3): 181-205

[57] Bass L, Clements P, Kazman R. Software Architecture in Practice. 2nd Edition. Boston: Addison Wesley Professional, 2003

[58] IEEE recommended practice for architectural description of software-intensive systems. IEEE Standard 1471-2000. IEEE, 2000

[59] Yu Ping, Ma Xiao-Xing, Lu Jian, Tao Xian-Ping. A dynamic software architecture oriented approach to online evolution. Journal of Software, 2006, 17(6): 1360-1371(in Chinese)
(余萍，马晓星，吕建，陶先平. 一种面向动态软件体系结构的在线演化方法. 软件学报，2006，17(6)：1360-1371)

[60] Lehman M M. Programs, cities, students, limits to growth? London: Imperial College of Science, Technology and Medicine, 1974

[61] Lehman M M. Laws of program evolution-rules and tools for programming management//Proceedings of the Infotech State of the Art Conference. Pergamon, 1978: 15-44

[62] Lehman M M. Software engineering, the software process and their support. IEE Software Engineering Journal, 1991, 6(5): 243-258

[63] Lehman M M. Laws of software evolution revisited//Proceedings of the European Workshop on Software Process Technology. Nancy, 1996: 108-124

[64] Wu J W. Open source software evolution and its dynamics. University of Waterloo, Waterloo, 2006

[65] Mei Hong, Shen Jun-Rong. Progress of research on software architecture. Journal of Software, 2006, 17(6): 1257-1275 (in Chinese)
(梅宏，申峻嵘. 软件体系结构研究进展. 软件学报，2006，17(6)：1257-1275)

[66] Kephart J O, Chess D M. The vision of autonomic computing. IEEE Computer, 2003, 36(1): 41-50

[67] Dobson S, Denazis S, Fernández A et al. A survey of autonomic communications. ACM Transactions Autonomous Adaptation System, 2006, 1(2): 223-259

[68] Juran J M, Godfrey A B. Juran's Quality Handbook. 5th Edition. New York: McGraw-Hill, 1998

[69] Juran J M, Gryna F M. Quality Planning and Analysis. 3rd Edition. New York: McGraw Hill, 1993

[70] Taylor R N, Medvidovic N, Dashofy E. Software Architecture: Foundations, Theory, and Practice. New York: John Wiley & Sons, 2009

[71] Mei H, Huang G, Lan L et al. A software architecture-centric self-adaptation approach for Internetware. Science in China Series F: Information Sciences, 2008, 51(6): 722-742

[72] IEEE Standard for Software Maintenance. IEEE Standard 1219-1998. IEEE, 1998

[73] Chapin N, Hale J, Khan K, Ramil J, Tan W. Types of software evolution and software maintenance. Journal of Software Maintenance and Evolution: Research and Practice, 2001, 13(1): 3-30

[74] Lu Jian, Ma Xoap-Xing, Tao Xian-Ping et al. Research and progress on internetware. Science in China (Series E), 2006, 36(10): 1037-1080(in Chinese)
(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, 2006, 36(10): 1037-1080)

[75] Ding Bo, Wang Huai-Min, Shi Dian-Xi, Cao Jian-Nong. Taming software adaptability with architecture-centric framework//Proceedings of the International Conference of Pervasive Computing and Communication. Mannheim, 2010: 145-151

[76] Baresi L. Toward Open-World Software: Issue and Challenges. IEEE Computer, 2006, 39(10): 36-43

[77] Kramer J, Magee J. The evolving philosophers problem: Dynamic change management. IEEE Transactions on Software Engineering. 1990, 16(11): 1293-1306

[78] Northrop L, Feiler P, Gabriel R P et al. Ultra-large-scale systems: The software challenge of the future. Pittsburgh, USA: Carnegie Mellon University, TR: 15213-3890, 2006

[79] Salehie M, Tahvildari L. A quality-driven approach to enable decision-making in self-adaptive software//Proceedings of the International Conference on Software Engineering. Leipzig, 2007: 103-104

[80] Georgiadis I, Magee J, Kramer J. Self-organizing software architectures for distributed systems//Proceedings of the Workshop on Self-Healing Systems. Charleston, 2002: 33-38

[81] Minsky N H. On conditions for self-healing in distributed software systems//Proceedings of the International Workshop on Active Middleware Services. Seattle, 2003: 86-92

[82] Parunak H, Brueckner S A. Engineering swarming systems. Methodologies and Software Engineering for Agent Systems. New York: Springer, 2004: 341-376

[83] Gupta D. On-line software version change[Ph. D. dissertation]. Indian Institute of Technology, Kanpur, 1994

WANG Huai-Min, born in 1962, Ph. D., professor, Ph. D. supervisor. His research interests include distributed computing, information security and computer software.

SHI Pei-Chang, born in 1981, Ph. D. candidate. His current research interests focus on distributed computing.

DING Bo, born in 1978, Ph. D.. His research interests include distributed computing and adaptive software.

YIN Gang, born in 1975, Ph. D., lecturer. His research interests include distributed computing and information security.

SHI Dian-Xi, born in 1966, Ph. D., associate professor. His research interests include distributed computing and pervasive computing.

Background

Today more and more large-scale distributed software systems are deployed. The challenge of ensuring the trustworthiness of those systems is arousing great attention in both academic and engineering circles. Since such a system usually runs in an open network environment and is expected to provide continuous services without interruption, an important way to achieve this goal is the online evolution, i. e., modifying the functionality and structure of the software system without the breakdown of its services.

In this paper, we mainly reviewed the existing work related to this topic. This paper also contains some original insights, such as the definition of online evolution of software services, several models related to this topic and a set of challenges we believe to be the most important ones in the near future. We hope that they can help and inspire the researchers and practitioners who concern this issue. This work is supported by the National Natural Science Foundation of China under Grant No. 90818028: "Behavior Monitoring and Trustworthiness-Oriented Evolution of Large-Scale Distributed Software Systems". This project aims at the methodology and a set of necessary techniques for online evolution of software services. This work is also supported by National Grand Fundamental Research 973 Program of China under grant No. 2011CB302600: "Basic Research on Effective and Trustworthy Internet-Based Virtual Computing Environment (iVCE)", whose purpose is to design basic models and mechanisms for effective and trustworthy virtual computing environment.