软件动态演化技术

李玉龙，李长云
（湖南工业大学 计算机与通信学院，湖南 株洲 412008）

摘 要：为适应开放环境和用户需求的变化，软件应该具备在运行时刻自演化的能力，从而满足各种需求变化，因此深入研究动态演化技术显得尤为重要。介绍了软件动态演化的开放性、层次性和整个演化活动的具体过程，讨论了动态演化的语言、模型和平台三者之间的关系以及当前的研究成果，并对已有成果中的实现方法、实用性和特色进行了比较，指出其不足之处。最后提出动态演化技术应重点研究的关键问题，并对其发展前景进行展望。

关键词：软件体系结构；演化；D－ADL
中图分类号：TP311 **文献标识码**：A **文章编号**：1673－629X(2008)09－0083－04

Dynamic Evolution Technology of Software

LI Yu-long, LI Chang-yun
(College of Computer and Communication, Hunan University of Technology, Zhuzhou 412008, China)

Abstract: In order to adapt to the open Internet environment and the variable user's requirements, software systems should be able to evolve dynamically, therefore, it is especially important to research the evolution in software. After introducing the usual features and process of software dynamic evolution, the relation among the language, model and platform of dynamic evolution is discussed; then, the achivement on hand is analized, including their implementing method and application. Finally, the dynamic evolution technology is forecasted.

Key words: software architecture; evolution; D－ADL

0 引 言

软件是对现实世界中问题空间与解空间的具体描述，是客观事物的一种反映。现实世界是不断演化的，软件系统面对的问题域也是不停地发展变化的，这使得用户期望软件系统能够很好地适应外界要求的变化。因此，演化性[1]是软件的基本属性。特别是在Internet成为主流软件运行环境之后，网络的开放性和动态性使得客户需求与硬件资源更加频繁地变化，导致软件的演化性和复杂性进一步增强。在这样的背景下，软件的动态演化方法引起了人们的注视，对软件系统演化的理论依据和执行手段进行了重点研究。目前已有的研究成果从函数、类、构件、体系结构等动态演化的不同粒度和层次出发，提出了各自相应的动态演化方法。

收稿日期：2007－12－05
基金项目：国家自然科学基金资助项目(60773110)；湖南省教育厅优秀青年项目(06B023)；湖南省学位与研究生研究课题(06B28)
作者简介：李玉龙(1976－)，男，安徽淮南人，硕士研究生，研究方向为软件体系结构；李长云，博士，教授，硕士生导师，研究方向为软件体系结构、软件自动化。

1 动态演化技术

软件演化可基本上分为两种：静态演化(Static Evolution)和动态演化(Dynamic Evolution)。静态演化是指软件在停机状态下的演化。其优点是不用考虑运行状态的迁移，同时也没有活动的进程需要处理。然而停止一个应用程序就意味着中断它提供的服务，造成软件暂时失效。动态演化是指软件在执行期间的软件演化。优点是软件不会存在暂时的失效，具有不间断服务的明显优点，但由于涉及到状态迁移等问题，比静态演化从技术上更为错综复杂，包括动态更新、增加和删除构件，动态配置[2]系统结构等问题，它已经成为软件演化研究领域倍受关注的一个热点问题。

1.1 动态演化技术是开放软件的关键特性

在开放的Internet环境下，动态性和多变性的需求越来越明显，而目前的主流软件技术还属于比较封闭和静态的软件体系框架，无法构造一个可充分利用丰富资源的计算平台。许多依托于互联网的软件技术，如网构软件、自治计算、网格计算等，都需要动态演化技术的支持。

网构软件是在Internet开放、动态和多变环境下

软件系统新的存在形态,其结构可根据应用需求和网络环境变化而发生动态演化,主要表现在其实体元素数目的可变性、结构关系的可调节性和结构形态的动态可配置性。演化性能是网构软件系统适应 Internet 开放、动态和多变环境的保障。

在很多关键的领域中,都需要软件能够感知环境的变化,并根据环境的变化改变自身行为,采取适应性动作以适应资源的可变性、用户需求的变化以及系统错误等,为了实现这种自管理的计算模式,人们提出了自治计算概念,它是指一个自动调节以满足正在其中运行的应用需要的基础结构,从自治计算系统的自调整性来看,它的最终目标都是要求能实现系统的动态演化。因此,追求动态演化是自治计算最本质的目的,同时也是最基本的出发点。

网格计算是伴随着互联网技术而迅速发展起来的,专门针对复杂科学计算的新型计算模式,它利用互联网把分散在不同地理位置的节点组织成一个"虚拟的超级计算机",实现各种资源的全面共享。网格的引入增强了计算能力,然而其要解决的关键问题就是远程调用和控制某台计算机的计算资源与存储资源,以及对网络流量的调节,这些都需要一种动态演化技术为其提供支持。因此,动态演化技术作为网格计算的基础提出,也有着其特殊的作用和意义。

1.2 动态演化过程

一次完整的动态演化过程分为三个阶段:动态演化触发、动态演化策略生成、动态演化执行。图 1 为动态演化的过程示意图。

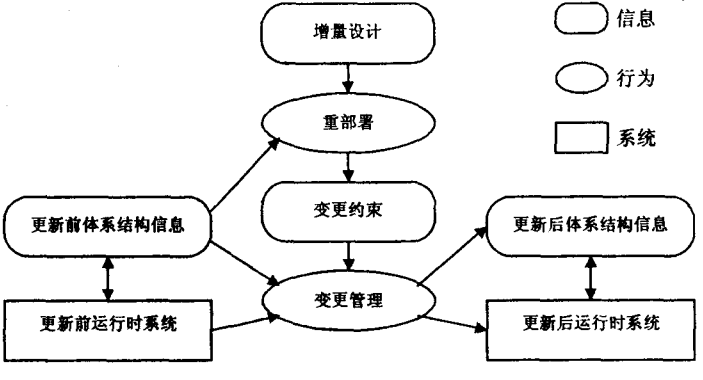

图 1 动态演化过程

为实现系统功能更新或 QoS 管理,动态演化被触发。触发动态演化的主体可能是系统设计师,在某些自适应系统中,也可能是系统本身。在实际应用中,更普遍的情况是人与系统一起决定是否触发以及如何触发动态演化。无论以何种模式触发动态演化,都将生成对动态演化具体内容的描述。基本动态演化内容包括:构件的删除、添加、替换、迁移,连接件的建立、删除、重定向,以及构件属性设置等。

动态演化平台根据动态演化内容和系统正确性约束生成动态演化策略。动态演化策略描述了实现动态演化的具体步骤。在动态演化本身合法的前提下,动态演化策略必须保证系统在动态演化期间和动态演化后处于正常运行状态,这就是在动态演化期间的系统一致性。动态演化必须保证系统一致性。

动态演化平台依据动态演化策略,通过监控构件的行为和状态,实现动态演化。由于在动态演化的实施过程中,系统仍然处于运行状态,因此必须尽可能提高动态演化的性能,减少对系统的影响。

1.3 动态演化具有层次性

在动态演化技术中,演化是分层次的,在不同的演化层次上,都以不同的粒度为演化形式。低层次的演化技术是高层次演化的基础,由低向高逐层构造了一个层次分明且相互关联的动态演化技术体系。演化的层次基本有函数层次、类层次、构件层次以及体系结构层次。

1.3.1 函数层次

一般来说,早期的动态链接库 DLL 的动态加载就是以 DLL 为粒度的函数层次的演化。DLL 的调用方式分为加载时的隐含调用和运行期的显式调用。加载时的隐含调用由编译系统完成对 DLL 的加载和应用程序结束时 DLL 卸载,属于静态调用方式;运行期的显式调用由编程者用 API 函数加载和卸载 DLL 来达到调用 DLL 的目的,使用上较复杂,是编制大型应用程序时经常采用的较为灵活的一种重要手段。DLL 的设计方法为应用程序提供一定的可扩展性和动态特性,但未成为一个规范。

1.3.2 类层次

类层次的动态演化方法最常见的就是代理机制下的类的动态替换。在软件运行期间,为某个对象提供一个代理对象,任何一个访问该对象的操作都必须通过代理对象来进行,这样就可以在调用实际对象前或调用后利用消息传递做一些调用预处理和收尾处理等工作。调用预处理工作用于完成类的版本判别、对象的替换、执行对象调用等操作。当一个对象请求调用另一个对象时,代理对象首先取得调用请求的信息,然后识别被调用对象对应类的版本是否更新,若已更新则重新装载该类并替换被调用对象;对象替换时,代理对象采用反射技术[3]获取旧版本对象的状态并传递给新版本对象以保障对象替换的状态一致性;最后完成新版本对象的行为调用,保证对象替换的引用一致性。类层次的动态演化是更侧重于代码的一种演化方

法,它作为技术手段为构件层次的演化提供了技术支持。

1.3.3 构件层次

在传统的软件系统中,构件之间是直接的交互引用的,而在动态体系结构中,构件之间的交互通过连接件来实现,很大程度上降低了构件之间的耦合度,增加软件系统动态演化的可能性。在构件层次的演化过程中,构件之间的调用请求以及构件状态可以被演化平台获取,演化平台在演化时刻将构件的调用请求截获,同时检测构件状态,当接收到演化命令时,将该调用请求阻塞,根据体系结构配置信息将构件进行运行时的重新组装和部署,最后完成连接件的重定向操作并释放构件的调用请求。构件层次[3]的演化与类层次的演化有一定的联系和区别,构件运行时的实例作为更为复杂的对象,在替换时仍然需要借助类层次的演化方法,但构件层次的演化需遵循体系结构的约束,保持演化前后的结构一致性。

1.3.4 体系结构层次

除了函数层次、类层次以及构件层次的动态演化之外,还有体系结构层次[3]的动态演化。体系结构层次的演化可以保证软件演化的一致性、正确性以及其它一些所期望的特性。体系结构将开发人员的关注点从代码转移到了粗粒度的构件和构件之间互连的结构上。这样使得设计者从难以理解的细粒度的软件编程细节中脱离出来从更高的高度来关注体系结构视图:系统的结构、构件之间的交互,构件的调度等。软件体系结构的一个显著特征就是显性地构造连接件,连接在构件间起着桥梁的作用,它同时还管理构件之间的交互,从而将构件的计算功能从构件通信中分离出来,最大程度上地减小了构件之间的相互依赖关系,使得构件的计算逻辑与适配逻辑相分离,方便对系统的理解、分析和演化。

2 语言、模型和平台

2.1 语 言

软件体系结构研究的主要成果表现为体系结构描述语言(Architecture Description Language,简称 ADL)。目前已出现了许多 ADL,如 Darwin,Rapide 语言和 C2 等,为软件体系结构动态机制的研究奠定了理论基础。

为准确、充分地表达运行时软件体系结构,提出基于高阶多型 π 演算的动态体系结构描述语言 D-ADL[4]。这种 D-ADL 以高阶多类型的 π 演算作为语义基础,使得其可以被机器自动化处理,便于模型检查、执行、求精和演化;提供体系结构的运行时支持,包括结构、行为的运行时演化;可被运行环境解释执行,成为整个系统调度的依据,进行可运行二进制物理构件的发现、执行和通信,从而驱动用户应用的运行;几乎所有的语法都有着抽象定义,包括行为抽象和数据抽象。D-ADL 为真正的运行时的演化提供了理论支持。

2.2 模 型

目前已有的研究成果中,提出的软件动态演化模型[2]各有特色与不同,但代表性的模型基本都分为元结构、元数据[5]、元协议三个部分。

元结构包括基层、元层(动态演化或重配置平台)、元控层。基层是被动态演化的目标应用系统,元层是实施动态演化或重配置的运行平台,元控层是动态演化或重配置的驱动者,负责监控基层,制定动态演化目标。

元数据包括结构元数据和行为元数据。结构元数据描述着应用软件体系结构的信息,包括组成应用系统的构件、构件之间的连接关系以及构件的物理位置等。行为元数据是指应用系统中构件的状态信息与构件的行为信息,如描述构件是否完成正在响应的请求的信息就是构件状态信息,构件是否可以启动事务或恢复发送请求就是构件行为信息。

在动态演化过程中,动态演化平台利用结构元数据和行为元数据,在驱动和判定系统处于动态演化的安全状态后,实施对系统的动态演化。这一过程中需要元协议来指导整个演化过程,使得动态演化过程在满足系统一致性约束的前提下进行。

2.3 代表性的动态演化平台

目前在各个不同的演化层次上,都有极具代表性的动态演化平台,如类层次的演化平台有 mChaRM,构件层次的演化平台有基于 CORBA 的 StarDRP[2]、基于 J2EE 的 PKUAS[6],而且这些演化平台相当一部分都具备体系结构层次上的演化特性。

2.3.1 mChaRM

意大利学者 Cazzola 等人设计了 mChaRM 系统,该系统解决了传统分布式面向对象系统中对象之间交互时计算逻辑与适配逻辑交错在一起的问题以及中间件平台对实体之间通信监控不足的问题,他们将实体之间的调用、通信具体化成一个多信道元层,以便检测到系统中实体之间的通信,在该多信道元模型中,系统实体之间一次方法调用以消息的形式在逻辑多信道中出现,而这个逻辑多信道则建立在请求服务的实体和提供服务的实体之间。

2.3.2 StarDRP

StarDRP[2]是由国防科技大学研制的基于 CORBA 的动态配置模型,该模型由系统信息库(System Infor-

mation Library)、动态配置算法库(Reconfiguration Algorithm Library)、动态配置算法生成器(Reconfiguration Algorithm Generator)、动态配置管理器(Dynamic Reconfiguration Manager,)、容器和部署基础设施六部分组成。

系统信息库中存放了系统结构和语义信息,这些都是通过解析部署描述文件,收集到应用的组成构件、构件的物理位置、构件间的连接关系等系统静态体系结构信息。

动态配置算法生成器则根据配置者通过图形用户界面 GUI 提交的动态配置意图,从动态配置算法库中取出对应的动态配置算法模板,生成针对此次动态配置意图的动态配置算法,提交给动态配置管理器作为中心控制器,负责动态配置算法的解析和执行。

根据动态配置管理器的指令,利用基于 CORBA 平台的基础设施提供的请求截获和重定向等功能,记录构件响应请求的状态信息,检测构件状态,并实施对请求的阻塞、释放和重定向等控制。

动态配置管理器根据整个动态配置算法的执行进度,在适当时机调用由构件实现的方法,完成相应的控制功能。

2.3.3 PKUAS

PKUAS 是基于 J2EE 的构件演化平台,该平台为北京大学自主研发,支持 ABC[3] 方法,是面向电力领域的一个成功案例。

借鉴操作系统微内核思想,PKUAS 通过抽取一组最基本功能形成一个内核,将平台内部的其他功能封装在各个相对独立的模块内。

PKUAS 将平台自身的计算实体划分为四种系统构件:

(1)容器系统:容器是构件运行时所处的空间,负责构件的生命周期管理以及构件运行需要的上下文管理。

(2)服务:这些服务可通过微内核动态增加、更换、删除,并结合元编程机制可进行服务功能的动态调用机制。元编程机制[3]作为一种有效的解决方案被引入中间件,该机制能够松散系统行为与资源、公用特性的紧耦合关系,提高系统的适应能力。

(3)工具:辅助用户使用和管理 PKUAS 的工具集,主要包括部署工具、配置工具与实时监控工具,允许用户实时观察系统的运行状态并作出相应调整。

(4)微内核:PKUAS 继承 JMX 并抽象出一个平台内核,其主要功能是有效地管理容器、服务、管理工具等系统构件。

3 演化技术的不足与展望

引起软件演化的原因是多方面的,如环境的改变、功能的增加、更优算法的发现等等,所以,对软件演化进行理解和控制显得比较复杂而又困难,还存在许多问题有待解决。在软件动态演化的研究过程中,须重点关注以下问题:

(1) 如何预先推导变更的结果及其影响范围。在发生软件变更之前,对变更之后的软件是否满足了需求?变更后的软件是否符合系统的约束?这些都应该有合适的机制进行自动的预测和评估,并决定是否能够进行软件的动态演化。

(2) 对于软件计算成员的替换,能够保证替换前后成员的外部行为的一致性。因为构成软件的元素相互协作、相互通信,软件的一个组成成员的功能执行可能需要另外的组成成员的配合来完成,每一个成员都对和它进行协作的成员有一个期望的交互方式和行为约束。这就意味着在替换软件计算元素时,不仅要使得它们的接口保持兼容,而且它们的可观察的外部行为也要保持一致性。

(3) 动态演化过程中事务机制的研究与实现;具备控制变更过程的手段,以保持应用的完整性。例如在变更过程中发生了错误,可以采取措施进行回滚,撤消最近的变更。

(4) 实时、较准确地对变更前后状态进行切换的机制,以维持演化期间的上下文一致性。当软件执行到某一断点时,从系统配置上撤换下来,它保持一定的状态信息。当该构件重新连接进入系统时需要从撤换时的断点继续开始执行。

参考文献:

[1] 李长云.基于体系结构的软件动态演化研究[D].杭州:浙江大学,2005.

[2] 窦 蕾.面向构件的复杂软件系统中动态配置技术的研究[D].长沙:国防科技大学,2005.

[3] 梅 宏,陈 锋,冯耀东,等. ABC:基于体系结构、面向构件的软件开发方法[J].软件学报,2003,14(4):721-732.

[4] 李长云,李赣生,何频捷.一种形式化的动态体系结构描述语言[J].软件学报,2006,17(6):1349-1359.

[5] Cazzola W, Chiba S, Saake G. Software evolution: A trip through reflective, aspect, meta-data oriented techniques [C]// In: Malenfant J, Østvold B M. Proc. of the ECOOP 2004 Workshop Reader. Oslo: Springer-Verlag, 2005:118-132.

[6] 黄 罡,王千祥,梅 宏,等.基于软件体系结构的反射式中间件研究[J].软件学报,2003,14(11):1819-1826.