一种灵活的软件在线演化机制

陈洪龙 李仁发
(湖南大学计算机与通信学院 长沙 410082)

摘 要 随着因特网成为主流软件的运行环境,计算模式逐步进入开放、动态以及用户需求频繁变化的环境,导致软件系统需动态地调整软件的组织结构或者功能行为来满足外界变化的需求。基于设计模式中的命令模式,利用方法调用与方法执行分离的原则和采用集中调度控制所有方法执行的方式,设计出一种灵活的软件在线演化机制。通过实例验证,本机制能在运行时刻灵活方便地变更软件的功能行为和流程逻辑。

关键词 自适应软件,设计模式,命令模式,集中调度
中图法分类号 TP40 **文献标识码** A

Flexible Online Evolution Mechanism for System Software

CHENG Hong-long LI Ren-fa
(College of Computer and Communication, Hunan University, Changsha 410082, China)

Abstract With the internet being the main runtime environment, the computer model is increasingly under the more open, dynamic and variable environment. So the software must dynastically adjust the organization or behavior to satisfy the changing requirement. Based on the command pattern, utilizing the isolation of method invocation and method execution and using centralized schedule, we designed a flexible online evolution mechanism. Finally the mechanism was testified to be convenient for changing system function and logic flow.

Keywords Self-adaptive software, Design pattern, Command pattern, Centralized schedule

随着因特网的迅速发展,网络已基本上深入到生活的每个角落,为我们提供了一个全球范围的计算平台,计算已从封闭的局域网扩展到以 Internet 为主干、广域网和局域网为分支的广域开放式分布网络环境中。而由于大量体积微型、带有网络的嵌入式设备的普及,软件也从单纯的桌面系统扩展到移动计算、普适式计算等新型应用领域。因此,在面临开放、用户需求易变并且不能提前预测的复杂网络环境的情况下,软件系统不能像在传统的集中封闭的环境中那样,从一开始就能充分预见和考虑到各种各样的业务流程,而需要在设计和开发时提供灵活机制,使软件能够在运行的时刻根据环境的变化进行修改或调整,达到预期设定的目标,延长软件的生命周期。

自适应演化软件被认为是解决上述问题的方案之一。但当前对于自适应演化软件的研究,主要集中在体系结构、自适应特性的形式化描述与适应前后的完整性验证的理论研究等方面,对于系统的业务逻辑以及系统行为的可变的实现研究较少。本文基于设计模式,通过方法的调用与方法的直接执行分离,设计出一种灵活的支持在线进化的机制(FOEM),它使软件可以根据环境的变化对系统运行过程中的逻辑以及行为进行在线的演化。

1 相关研究

国内的北京大学和南京大学的研究人员针对软件环境的变更进行研究,提出网构软件[1]的概念,指出网构软件是在 Internet 开放、动态和多变环境下软件系统基本形态的一种抽象,具有区别传统软件形态的基本特征,其中包括自主性和演化性,即软件可根据应用需求和网络环境变化而发生动态演化。文献[2]中指出,网络环境的开放、动态和多变性,以及用户使用方式的个性化要求决定软件不再像经典软件那样一蹴而就,它应能感知外部环境的动态变化,并随着这种变化按照功能指标、性能指标或可靠性指标等进行静态(离线)的调整和动态(在线)演化。国内外的研究人员从各方面对软件的自主性与在线演化进行了研究,IBM 于 2001 年提出自治计算的概念[4];文献[3,5-7]提出基于体系结构的自适应与在线演化方法;文献[8]中提出面向构件化的在线演化方法,但主要是注重演化前后软件的一致性、完整性与追溯性以及相应的理论证明,对在线演化软件的具体实现机制研究得不多。并且在演化过程中,代码总是需预先集成在软件系统中,等相应的自适应规则触发后,才执行相应流程。或者如 Jboss 中面向方面的方案一样,对已编译完成的中间代码上进行目标代码的变更(指令的变更),按配置在相应位置注入相关指令。ACE(Adaptive Communication Environment)提供动态服务配置框架,用于运行时动态加载服务,但主要局限于整体服务的替换,并不能重用旧服务中的部分功能,即服务中涉及的类及对象需全部替换,因此是粗粒度的,对于资源有限的嵌入式

到稿日期:2009-06-02 返修日期:2009-08-16 本文受国家自然科学基金(60673061,60873074),湖南省重点科技计划项目(S2007G21211841)资助。

陈洪龙(1974-),博士生,主要研究方向为软件中间件、软件自适应;李仁发(1957-),博士生导师,主要研究方向为嵌入式计算、无线网络,E-mail:cchenhonglong@sina.com。

系统来说是不合适的[9]。

设计模式[10]是指在实践中被证明良好的解决方案，为提高软件的复用性，而重复使用的那些解决方案，最初是指城市和建筑的模式。Christopher Alexander 说过："每一个模式描述了在我们周围不断重复发生的问题以及该问题解决方案的核心。这样，你就能一次又一次地使用该方案而不必做重复劳动。"设计模式不仅能提高软件的复用能力，而且通过面向对象的继承与多态性，可以获取运行时刻的动态性，即通过在运行时依据具体的子对象类型而决定系统的行为。但该动态性基本上是在编译阶段完成的，只在运行时刻才表现出来。

其中命令模式是设计模式中的对象行为型模式之一，其意图是将请求封装为对象，从而使你可用不同的请求对客户进行参数化，以统一的方法发送请求并执行不同的行为。本文基于设计模式的命令模式，通过分离方法的调用与方法的执行，设计一种灵活支持在线进化机制(FOEM)。它通过运行时刻的动态配置，能在系统的运行过程中灵活地添加、删除以及替换系统中已有的相关行为方法，使系统能够在线修改其服务处理逻辑，达到适应环境变更的目标。

2 POEM 的结构

本机制的设计原理是把系统中所有的方法调用、封装成命令请求，加入主动队列，通过操作系统的同步触发机制，统一集中调度。在具体调度时，检查内部配置文件，根据配置的具体对象类型、方法名称以及相关动作，动态增加或者替换该方法调用，从而达到在线变更软件系统行为的目标，其结构如图 1 所示。

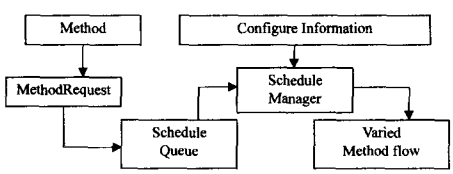

图 1 FOEM 软件结构

从以上软件结构可知，主要的机制是把方法的调用与方法的具体执行分离开来，即类中的方法调用不是直接执行，而是通过类封装为方法请求类，并且把方法的执行延迟到调度管理器中。通过这样简单的分离，系统能够在运行时依据配置从外界注入代码，替换或者变更已有的方法的执行顺序，使软件系统能够达到在线适应环境或需求变化的目标，从而达到延长软件生命周期的目标。

3 初步实现及示例

在具体的实行过程中，为把方法的调用与执行分开，设计了 FOEM. Method 和 FOEM. MethodRequest 基类。其中 FOEM. Method 基类具有 Execute 的虚拟方法，类中所有的方法均基于该基类，并且按照自身的功能，实现具体的 Execute 方法；而 FOEM. MethodRequest 与 FOEM. Method 具有组合关联，一个 MethodRequest 实例有且仅有一个 FOEM. Method 的实例；MethodRequest 保存方法调用的实例以及方法调用中的所有参数信息。

为使方法可以集中执行，设计了 FOEM. ActiveQueue，FOEM. ScheduleManager。其中 FOEM. ActiveQueue 是相当于 WINDOWS 的消息队列类，负责存储与访问所有的方法调用请求。为方便所有类的存储，其实例为全局唯一。FOEM. ScheduleManager 负责方法的具体执行，其中拥有线程执行的方法 Exec()。该线程阻塞等待方法调用请求，当方法调用请求存入队列时，通过事件通知该线程，从队列中取出一个方法调用请求，然后根据配置执行。

为使系统能按照配置执行，先设置配置格式如下：类名称(全名)、动作、方法名称。其中动作包括 PreMethod，PostMethod，ReplaceMethod。PreMethod 表示在方法执行前，先执行配置中加入的方法；PostMethod 表示在方法执行后，再执行配置中加入的方法，而 ReplaceMethod 为用配置中的方法替换原始的方法。并且，为避免由于配置导致性能的损失，系统通过 HashTable 保存，建立映射关联，以后该方法的调用直接从 HashTable 中获取实例执行，不用每次都重新加载。具体的类图如图 2 所示。

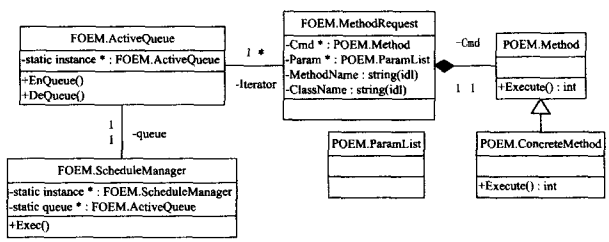

图 2 FOEM 主要类图结构

其中 POEM. ActiveQueue 及 POEM. Schedule-Manager 为 Singleton 模式，在整个系统中只存在唯一的实例。以下用伪代码示例系统的动态性，如图 3 所示。系统的加方法，在应用的过程被动态在线地替换为乘方法：

```
Class Operation {
1. int Add(int xInput,int yInput) {
2. m= New FOEM.MethodRequest(New CAdd( new POEM.ParamList(xInput,yInput))
3. FOEM.ActiveQueue::Instance->EnQueue(m);
4. SetEvent(&e) } }
```

以上代码用于客户端调用时，把方法的调用转化为方法的执行请求，并发送至集中调度队列，由队列根据反射机制，利用相关信息，直接交由真正的类执行具体的业务逻辑。

```
Class CAdd: public FOEM.Method {
Int Execute(FOEM.ParamList Param) {
        Int a = Param[0];   Int b = Param[1];
        Return a +b; }}

    Class FOEM.SchedualManager {
        Void Execute() {
1.  While(True){
2.      WaifForSingleObject(e,Null,0,0);
3.      M=FOEM.ActiveQueue::Instance->DeQueue();
4       g_M=GetClassHashMap(M.cmd->className);if(g_M==NULL){
5.      IF (GetConfigEvent(M.cmd->ClassName)){
          Value=Config.Find(M.cmd->ClassName);
6.        IF (value.Action == "ReplaceMethod")  {g_M = Value.LoadNewMethod();
7.      g_M.cmd->Execute(param); SetHashMap(M.cmd->ClassName,g_M);} }}

Class CMultiply: public FOEM.Method {
    int Execute(POEM.ParamList param) {
        return param[0]*param[1];  } }
```

图 3 FOEM 机制具体示例

4 相关比较及分析

在解决类似问题的思路上，亚特兰大大学的 OpenORB[11](针对其 1.0 的 C++版本)也采取方法替换、在原始方法调用前增加新的调用以及在原始方法调用之后增加新的调用来获得软件的适应性，因此具有一定的可比性。在 OpenORB 中，主要是通过使用委托代理器，保存其代理的实际接口的虚拟函数表的地址入口，并且把实际接口的虚拟函数表的地址指向代理本身提供的函数表的地址。这样，通过

替换代理提供的函数地址表中相应位置的函数，或者设置多至32个预先处理函数以及32个后续处理函数后，当实际接口调用对应函数时，其实是调用替换的函数，这样来获取软件系统的适应性。本文提出的机制显然更加灵活。

另外，SOA(面向服务体系结构)也能用于在线动态更换服务，但主要是粗粒度的，需重写整个服务。FOEM能针对类中的方法进行替换，从而变更整个类，主要是细粒度的，但利于系统变更时尽可能复用不变的代码，提高软件的复用性，这对于资源依赖强烈的嵌入式系统有一定的优势[9]。

从FOEM的设计可知，集中调度是整个系统的瓶颈，但我们可以相关调度策略来进行优化，这方面的研究很多，也不是本文研究的重点，因此没有详细展开。另外，由于通过方法的调用请求和方法的具体执行分开机制，在获得演化的灵活性的同时也带来一定的性能影响。但FOEM只对存在需求可变化的类才采用该机制，并且FOEM的主要开销是类的加载。从演化的技术来看，运行时单次加载所带来的性能开销是可接受的。表1是FOEM与OpenORB和ACE的具体相关比较(其中Load表示装载演化，Runtime表示运行时演化，AF表示抽象工厂模式，FM表示工厂方法模式)。

表1 相关比较

	动态演化粒度	运行时演化	动态配置	设计模式	演化类别	反射
FOEM	函数/类/对象	支持	支持	Command	Load/Runtime	不支持
Open-ORB	函数	不支持	不支持	Proxy	Load	支持
ACE	组件	支持	支持	AF/Proxy/FM	Load/runtime	不支持

结束语 随着计算环境的变化，给软件设计开发提出了新的挑战，尤其是如何快速地适应环境和需求的变化。而自适应演化软件正是解决该类问题的方案之一。当前对于自适应演化软件的研究从各方面展开，但主要是在理论研究方面，比如服务组合以及组合后的验证、动态体系结构语言的描述、动态体系结构的建模等。而关于在具体应用中如何实现自适应，特别是在线的软件自适应演化方面的实现研究却不多。本文基于设计模式中的命令模式，通过隔离方法的调用请求和方法的具体执行，提出一种灵活的在线演化机制，并通过初步的实现验证本机制的有效性。下一步对其性能进行改进，使之适合对系统性能要求高但资源严格受限的实时嵌入式环境。

参考文献

[1] 杨芙清，梅宏，吕建，等. 浅论软件技术发展[J]. 电子学报，2002，30(12A)：1901-1906

[2] 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)：1037-1080

[3] 余萍，马晓星，吕建，等. 一种面向动态软件体系结构的在线演化方法[J]. 软件学报，2006，17(6)：1360-1371

[4] Kephart J，Chess D. The Vision of Autonomic Computing[J]. IEEE Computer，January 2003：41-51

[5] 陈磊，李三立. 网格数据复本管理的动态自适应软件体系结构[J]. 软件学报，2006，17(6)：1436-1447

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

[7] Garlan D，Schmerl B. Using architectural models at runtime：research challenges[C]// Oquendo F，Warboys B，Morrison R，eds. Proc. of the 1st European Workshop on Software Architectures. LNCS 3047，St. Andrews：Springer-Verlag，2004：200-205

[8] 王晓鹏，王千祥，梅宏. 一种面向构件化软件的在线演化方法[J]. 计算机学报，2005，28(11)：1890-1897

[9] Vandewoude Y，Berbers Y. Run-time Evolution for Embedded Component-oriented Systems[C]// Proceedings of the International Conference on Software Maintenance (ICSM'02)

[10] Gamma E，Helm R，Johnson R，et al. 设计模式-可复用面向对象软件的基础[M]. 李英军，等译. 机械工业出版社，2007

[11] Blair G S，Coulson G，Andersen A，et al. The design and implementation of open ORB 2[EB/OL]. IEEE Distributed Systems Online，2001，2(6). http://dsonline. computer. ogr/0106/features/bla0106-print. html.

(上接第90页)

[3] Joux A. A One Round Protocol for Tripartite Diffie-Hellman[C]// ANTS 4，LNCS 1838. Springer-Verlag，2000：385-394

[4] Al-Riyami S S，Paterson K G. Tripartite Authenticated Key Agreement Protocols form Pairings[C]// IMA Conference of Cryptography and Coding 2003，LNCS 2898. 2003：332-359

[5] Liu S，Zhang F，Chen K. ID-Based Tripartite key agreement protocol with pairing[C]//2003 IEEE International Symposium on Information Thory. 2003：136-143

[6] Shim K，Woo S. Weakness in ID-based One Round Authenticated Tripartite Multiple-Key Agreement Protocol with Pairings[J]. Applied Mathematics and Computation，2005，166：523-530

[7] Bao F，Deng R，Zhu R. Variations of Diffie-Hellman problem[M]. Springer-Verlag，2003：301-312

[8] Shim K. Efficient ID-Based Authenticated Key Agreement Protocol Based on the Weil Pairing[J]. Eectronic Letters，2003，39(8)：653-654

[9] Juang W-S，Wei-Ken Nien W-K. Efficient Password Authenticated Key Agreement Using Bilinear Pairings，Mathematical and Computer Modelling[M]. New York：Pergamon- Press，2008

[10] Chien H Y，Lin R Y. Identity-based key agreement protocol for mobile ad-hoc networks using bilinear pairing[C]//IEEE International Conference of Sensor Networks，Ubiquitous，and Trustworthy Computing. vol. 1，June 2006：520-529

[11] Chung H R，Ku W C. Impersonation attacks on a simple three-party key exchange protocol[C]// 17th Information Security Conference. June 2007

[12] Lee N Y，Wu C N，Wang P C. Authenticated multiple key exchange protocols based on elliptic curves and bilinear pairings[J]. Computers and Electrical Engineering，2008，34(1)：12-20

[13] Chien H Y. Highly Efficient ID-based Ring Signature from Pairings[C]// 2008 IEEE Asia-Pacific Services Computing Conference，IEEE APSCC 2008. Yilan，December 2008

[14] Guo Y J，Yu Z Q. Trust architecture in dynamic systems[C]// International Symposium on Advances in Computer and Sensor Networks and Systems. 2008：69-74