第 36 卷 第 1 期 计 算 机 工 程 2010 年 1 月
Vol.36 No.1 Computer Engineering January 2010
·软件技术与数据库· 文章编号：1000—3428(2010)01—0055—03 文献标识码：A 中图分类号：TP311

基于构件的网构软件系统动态演化

朱 庆，王小平，薛小平，司文婷
(同济大学电子与信息工程学院，上海 200092)

摘 要：为适应网构软件系统在线演化的需求，提出一种基于构件的网构软件系统动态演化模型。该模型以构件为基本单位，基于软件体系结构部署和实施演化，给出构件添加、删除和替换需求的演化算法，通过引入一致性检查机制保证演化的安全和可靠。该模型实施简单、具有普遍适用性，软件系统无须进行大量的改动即可适应该演化模型。

关键词：动态演化；网构软件；构件；软件体系结构；一致性

Dynamic Evolution of Internetware System Based on Component

ZHU Qing, WANG Xiao-ping, XUE Xiao-ping, SI Wen-ting
(School of Electronics and Information Engineering, Tongji University, Shanghai 200092)

【Abstract】 A dynamic evolution model of the internetware system based on component is proposed in order to meet the need that internetware system is able to evolve during the runtime. The model takes component as elementary unit and bases on software architecture to deploy and implement the evolution. It takes evolutionary algorithms of component adding, deleting and replacing into account, and introduces consistency checking mechanism to guarantee safety and reliability of the evolution. The model is easy to implement and has universal applicability, so that software systems is able to adapt to the evolution model without a great deal of modification.

【Key words】 dynamic evolution; internetware; component; software architecture; consistence

1 概述

随着 Internet 的发展，软件系统的运行环境逐渐由封闭、静态走向开放、动态及多变。以构件等技术支持的软件实体将以开放、自主的方式存在于 Internet 的各个节点之上，从而形成一种与当前 WWW 类似的 Software Web。Web 不再仅仅是信息的提供者，而是各种服务的提供者，这种新的软件形态被称为网构软件[1]。

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。相对于传统的软件系统，它具有如下的基本特征[1]：(1)自主性，(2)协同性，(3)反应性，(4)演化性，(5)多态性。

网构软件的演化性是指网构软件系统可根据网络运行环境的变化以及服务需求的变化进行自我调整的能力。软件的演化可分为静态演化和动态演化，这 2 种演化方式的不同点在于演化时机不同，静态演化通常是在设计时对软件系统进行更新，而动态演化则是在软件系统运行的过程中进行，且演化过程不中断软件的服务。动态演化技术在某些要求必须连续运行的系统中尤为重要，如网络服务系统、空中交通管制系统、全球性金融交易系统、工业控制系统，若在演化过程中停止系统的运行将导致不可接受的延迟、代价和危险。

本文针对网构软件的动态演化，提出支撑动态演化的具体实施方案。

2 支持动态演化的系统模型

模型由 3 个部分构成：运行中心，监控中心和演化中心，见图 1。其中，运行中心由构件和连接器[2]构成，连接器在构件间起桥梁作用，有效分离构件的计算功能和通信功能，最大程度减小了构件间的耦合关系，方便对系统的理解、分析和演化；监控中心监督和控制系统的运行，并辅助演化中心完成演 130

化任务，包括数据库和控制部件，其中，数据库存储软件系统中构件和连接器的相关信息，控制部件提供一系列构件服务功能；演化中心负责实现系统的演化任务，模型中引入安全性检查部件以保证演化动作的安全性，防止危险的演化行为导致系统崩溃。

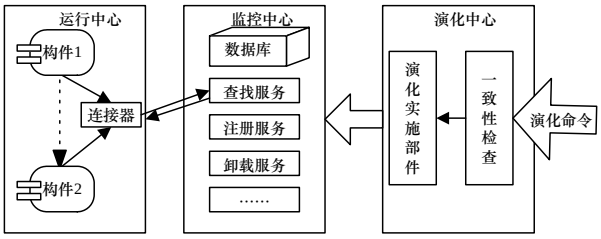


图 1 支持动态演化的系统模型

2.1 运行中心

传统的构件系统中，构件间通过直接调用的方式实现交互，如图 2 所示。但这种方法不能很好地支持系统的演化，演化通常会带来较大的改动成本。

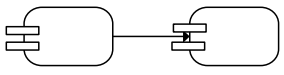


图 2 传统构件间交互方式

图 2 中若构件 B 修改了其服务接口，所有调用构件 B 的构件(如构件 A)为了能够得到相应的服务必须重新编写构件

作者简介：朱 庆(1986－)，女，硕士研究生，主研方向：软件工程，软件构件，软件体系结构；王小平，教授、博士；薛小平，副教授、博士；司文婷，硕士

收稿日期：2009-06-08 **E-mail：**zqing1986@gmail.com

代码、修改构件接口，这无疑会大大增加演化的成本。为了解决传统方式的局限性，本文采用构件和连接器的方式实现构件间的交互，如图 3 所示。

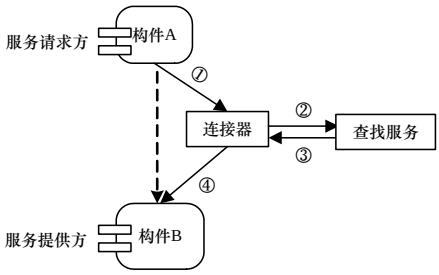


图 3　构件和连接器交互方式

构件 A 不是通过直接调用构件 B 获取服务的，其获取服务过程如下：

(1)A 向连接器发送请求，并传递请求内容及接口信息。

(2)连接器通过查找服务获取能够为 A 提供服务的构件信息。

(3)查找服务返回所需构件信息，如构件名称、ID、位置。

(4)连接器连接到构件 B，传递 A 的请求，实现 A, B 间的通信。

在这种交互方式下构件无须关心其他构件的存在，而是通过查找服务获取所需服务。若构件 B 更新其接口，A 无须重新编写以适应 B 的变化，此时只需修改查找服务中的映射函数即可实现 A 对 B 的请求服务，这种修改相对于传统方法是轻量级的，且无须中断系统的运行。

2.2　监控中心

监控中心是模型的核心部分，运行中心和演化中心的正常运行都需要从监控中心获取相应的服务。它主要实现以下功能：

(1)构件注册服务。当系统中增加构件时，监控中心必须完成对构件的登记工作，包括记录构件的类型描述、实现其功能映射函数、为其分配 ID，并将这些信息存入数据库中。

(2)构件卸载服务。系统中删除构件时，监控中心从数据库中删除该构件的相关信息，包括 ID、类型描述、功能映射等，以防止其他构件的非法调用。

(3)构件查找服务。查询数据库，根据功能映射函数选取满足需求的构件，并返回该构件 ID。为了提高查询效率，可以为映射函数设置访问频度，构件注册时生成的映射函数频度设置为 0，每调用 1 次频度加 1，查询则按照频度由高到低的顺序进行。

(4)监控构件实时运行状态。由于系统中构件间的交互通过监控中心实现，因此构件的状态可以从监控中心获取，当构件请求服务时，通过连接器发送消息给监控中心进行查找服务，此时监控中心便可获知该构件处于运行状态。

(5)存储系统的体系结构，并以 GUI 的形式显示给用户[3]。当系统因演化导致体系结构发生改变时，监控中心应相应地修改存储的体系结构，并更新 GUI 视图。

2.3　演化中心

演化中心管理系统的演化，由一致性检查部件和演化实施部件构成。当演化中心收到演化命令时，首先通过一致性检查部件检查演化命令是否符合安全性要求(详见第 4 节)，若符合，则由演化实施部件执行演化命令。

为保证演化的正确性，演化中心在同一时间只能执行一个演化命令，因此，必须设置缓冲队列保存请求的演化命令，当演化实施部件收到演化命令并准备执行时，应对自己上锁，在执行结束并正确返回时解锁，继续执行缓冲队列中的演化命令，见图 4。

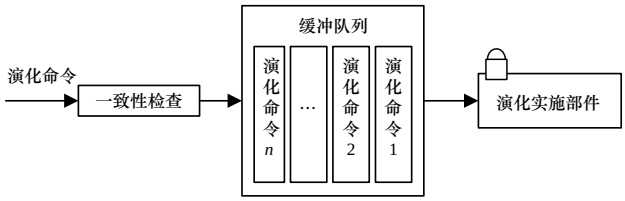


图 4　演化实施流程

演化分为简单演化和复杂演化[4]，简单演化是指构件添加、删除和替换，复杂演化是由简单演化组合构成的。

3　动态演化算法

基于构件的动态演化是以构件为粒度对软件系统进行的演化，包括构件添加、删除和替换，下面分别给出这 3 种演化操作对应的算法。

3.1　构件添加算法

由于构件的添加不影响当前系统的运行，因此可以直接进行添加，算法如下：

(1)上传新构件及其描述性文件。

(2)通过监控中心的注册服务完成构件的注册，为构件分配 ID，描述其请求和服务接口，并根据接口设置映射函数。

(3)更新监控中心数据库中的体系结构。

3.2　构件删除算法

构件的删除要判断构件当前的状态，根据不同状态采取不同的策略，具体算法如下：

(1)通过一致性检查部件检查卸载命令是否合法，若不合法则通知演化失败。

(2)查询构件的当前状态，若处于执行状态，则阻塞构件的请求队列，等待构件的缓冲队列中所有请求执行结束。

(3)从数据库中删除构件的注册信息和对应的映射函数。

(4)卸载构件。

(5)更新数据库中的体系结构。

3.3　构件替换算法

文献[5-6]总结了动态更新构件需要解决的 4 个问题：

(1)构件引用透明问题。如图 5 所示，构件 A 调用构件 B 的服务，这样构件 A 中就有了 B 的引用，如果将构件 B 替换为 B’，那么所有拥有 B 句柄的构件都需要将引用改成 B’。当多个构件都有 B 的句柄，甚至这些构件间也有相互依赖关系时，在运行时要保证构件替换造成的引用改变使用户的透明性较难实现。

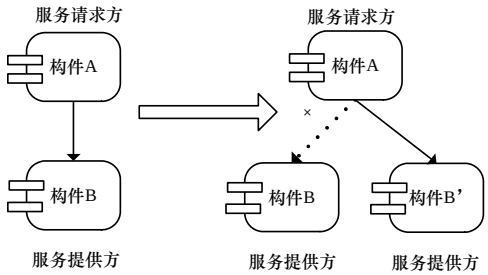


图 5　构件替换

(2)构件状态迁移问题。当被替换的构件有实例存在时，需要将这些实例也替换成新构件的实例，并将属性方法执行状态复制到新的实例中，同时保持系统的持续运行。

(3)相互引用问题。当多个构件相互依赖，在替换时需要将这多个构件在同一事务操作中替换，且替换顺序要合理，否则可能导致演化失败。

(4)构件有可能处于一个长事务中，如果替换则会造成构件执行语义不一致，所以，只有等事务完成或者回滚后才能替换构件。

为了解决上述问题，在演化模型的基础上引入构件状态转移自动机，保证构件状态迁移的正确性，如图 6 所示。系统通过演化命令添加构件，此时构件完成注册进入初始化状态，当初始化结束后构件进入空闲状态等待接收服务请求，运行状态表示构件正在执行请求服务。构件在初始化、空闲、运行状态时都可以通过演化命令进行演化。

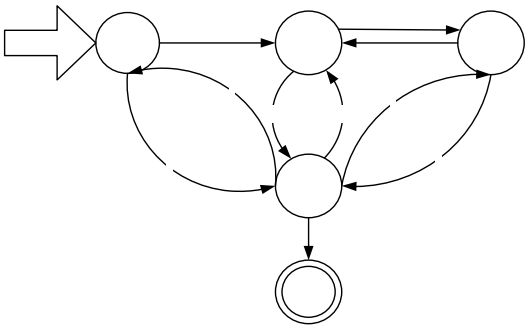


图 6　构件状态转移自动机

下面给出不同状态转换情况下构件替换的算法:

(1)初始化->演化。由于此时并没有创建构件实例，因此可以直接进行替换，更新数据库中构件的注册信息，若新构件修改了构件接口，需相应地修改映射函数，同时更新体系结构视图，保证体系结构的同步更新。演化完成后进入初始化状态。

(2)空闲->演化。此时构件接收了服务请求，创建了构件实例，因此，演化时必须同时更新构件实例，具体算法如下:

1)阻塞构件消息队列，禁止其他构件对其的服务请求。

2)更新缓冲队列中的构件实例: 通过反射机制获取构件实例的属性、方法和构造函数，利用反射机制中的一组 API 对反射出的属性、方法和构造函数进行操作，从而满足新构件的需求。

3)替换构件，修改数据库中的相关内容。

4)更新体系结构视图。

5)进入空闲状态，释放被阻塞的队列。

(3)运行->演化。此时构件处于运行状态，如果立即更新构件，会导致系统进入不安全状态，合理的做法是等待执行结束进入空闲态，再依据空闲->演化的演化算法进行演化，完整的状态转移依次为运行->演化->运行->空闲->演化->空闲。

4　演化一致性检查

本文的一致性检查指演化前对演化命令进行的检查。通过对演化命令的检查保证系统集成行为不因错误添加构件系统而产生错误。

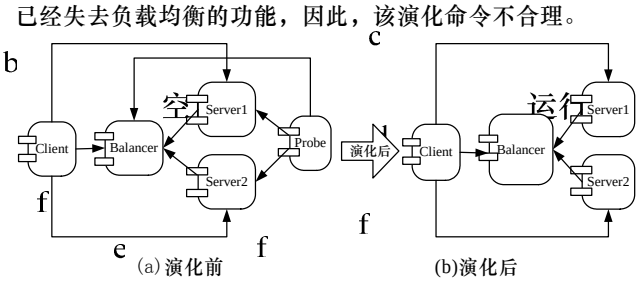
为了便于检查，演化命令以 XML 的形式描述，以构件添加、删除和替换为例，其表示如下:

```
<Evolve >
<Add_component>
<name>构件名称</name>
<description>描述性文件</ description >
<loc>构件位置</loc>
</Add_component>
<Del_component>
<name>构件名称</name>
</Del_component>
<Rep_component>
<name>需替换的构件名称</name>
<description>新构件描述性文件
</ description >
<loc>新构件位置</loc>
</Rep_component>
</Evolve >
```

检查的内容包括演化意图的正确性、演化命令能否保证构件间相互关系的一致性。

检查采用预演的方法，首先从数据库中提取当前系统的体系结构信息，在该体系结构上执行演化命令(指通过演化命令修改体系结构视图)，演化预演结束后由系统管理员观察演化后的体系结构视图是否符合演化意图，若符合，则表明演化命令满足一致性检查，否则禁止该命令的执行。

以负载均衡系统[4]为例，其体系结构(省略连接器)如图 7(a)所示。在该系统中，Balancer 记录了系统中所有 Server 的引用和负载情况，Client 为了获得 Server 提供的服务，首先通过 Balancer 查询负载记录并根据一定策略将一个 Server 的引用返回给 Client。Probe 负责收集部署在 Server 上的负载情况，报告给 Balancer。给定演化命令为删除构件 Probe，通过预演，体系结构如图 7(b)所示，从图中可以看出，该系统已经失去负载均衡的功能，因此，该演化命令不合理。



(a) 演化前　　(b)演化后

图 7　负载均衡系统体系结构

5　结束语

本文介绍了基于构件的网构软件系统动态演化模型，该模型分为 3 个部分，每个部分实现的功能相对独立，其中，运行中心采用构件和连接器模型，使动态演化成为可能；监控中心集合了系统中的所有信息和服务，辅助和监控系统的运行和演化；演化中心负责系统的动态演化，通过引入一致性检查验证演化的安全性。在演化模型的基础上，给出了构件的添加、删除和替换 3 个原子操作的演化算法。

进一步的工作将从以下 3 个方面展开:

(1)丰富和完善演化一致性检查的内容。

(2)引入回滚机制，当演化过程中发生错误时，提供相应的回滚操作恢复到演化前的状态。

(3)实现支撑该模型的软件平台，使该模型能够应用于通用软件。

(下转第 60 页)