软件体系结构论文

动态软件体系结构演化分析

姓　　名　李琦

学　　号　18721802

论文评分

2019 年 3 月 15 日

目　录

[摘要]本文总结了现有的动态体系结构模型的演化及其优缺点，首先介绍了动态体系结构的基本概念，特点以及研究中所设计的基本问题，指出动态体系结构研究中存在的难点。本文着重于基于构建的动态模型介绍，该模型分为三层，每层各司其职，各自执行相应的更新请求，并在本末与本人硕士期间的开源项目 Vaa3d 相结合，进行了实例演示验证，结果表明，该模型能够很好地支持系统的动态更新。

[关键字]动态软件体系结构，构件，模型，Vaa3d

Abstract: This paper summarizes the evolution of the existing dynamic architecture model and its advantages and disadvantages. Firstly, it introduces the basic concepts, characteristics and basic problems of the dynamic architecture, and points out the difficulties in the research of dynamic architecture. This paper focuses on the introduction of the dynamic model based on the construction. The model is divided into three layers. Each layer performs its own functions and each performs the corresponding update request. In the end, it combines with the open source project Vaa3d during the master's degree. The results show that the model can well support the dynamic update of the system.

Key words: Dynamic software architecture, components, models, Vaa3d

一 绪论

1.1 软件体系结构模型的起源

自 NATO(北大西洋公约组织)于 1968 年提出软件工程概念以来,软件工程界已经提出了一系列的理论、方法、语言和工具,解决了软件开发过程中的若干问题。但是,软件固有的复杂性、易变性和不可见性,使得软件开发周期长、代价高和质量低的问题依然存在。[2]大量实践统计表明:大系统软件开发中 70%的错误是由需求和软件设计阶段引入的;而且错误在系统中存在的时间愈长则愈难发现,解决这些错误的代价也愈高。

为了提高软件需求和软件设计的质量,软件工程界提出了需求分析工程技术和各种软件建模技术。但是在需求与设计之间仍存在一条很难逾越的鸿沟,即缺乏能够反映做决策的中间过程,从而很难有效地将需求转换为相应的设计。为此,软件体系结构概念应运而生,并试图在软件需求与软件设计之间架起一座桥梁,着重解决软件系统的结构和需求向实现平坦地过渡的问题。[1]

1.2 动态软件体系结构模型概述

历经传统的结构化开发方法和面向对象开发方法,基于软件体系结构、构件的开发方法已逐渐成为当前软件开发的主流,软件开发的基本单元已从传统的代码行、对象类转变为各种粒度的构件,构件之间的拓扑结构形成了软件体系结构。这种转变给软件开发带来更多的灵活性,可以通过构件重用和替换来实现,即实现构件的“即插即用”。而灵活性的一方面是动态性。在体系结构层实现动态性会给大型软件系统的开发提供可扩展性,用户自定义和可演化性。而且,软件体系结构的动态改变和演化对于需要长期运行或具有特殊任务的系统是尤其重要的。

由于系统需求、技术、环境、分布等因素的变化而最终导致软件体系结构的变动,称之为软件体系结构演化。软件系统在运行时刻的体系结构变化称为体系结构的动态性,而将体系结构的静态修改称为体系结构扩展。体系结构扩展与体系结构动态性都是体系结构适应性和演化性的研究范畴。[3]

体系结构的动态性主要分为三类:(1)交互式动态性,例如允许在复合构件的固定连接中改变数据;(2)结构化动态性,例如允许对系统添加或删除构件或连接件;(3)体系结构化动态性,例如允许构件的整个配置的改变。

1.3 动态软件体系结构模型发展现状

尽管自 1994 年召开了首届软件体系结构国际研讨会以来,软件体系结构研究领域取得了很多成果,但在应用方面,软件体系结构仍然很不成熟,动态体系结构由于其自身的复杂性更是如此。目前,动态体系结构的研究工作主要分为两类:模拟和描述体系结构动态更新的语言和支持体系结构动态更新的执行工具。

ADL 提供了一种形式化机制来描述软件体系结构,这种形式化机织主要通过提供语法和语义描述来模拟构件、连接件和配置。但是大多数 ADLS 只描述系统的静态结构,不支持对体系结构动态性的描述。[5]

近年来,对这一方面的研究主要集中在对现有的一些 ADLS 扩展以支持体系结构的动态性,现已研究出一些支持动态体系结构的 ADLS。在动态体系结构建模和描述方面,C2 支持结构动态性,它定义了专门支持体系结构修改的描述语言 AML。Darwin 对体系结构的修改采用相应的脚本语言[3],它具有惰性和动态实例化两种

动态机制,但它只涉及计算结构单元,对于事务逻辑和体系结构配置的互动没有考虑。

对于动态体系结构应用方面的研究,还很不成熟。目前,支持动态体系结构的机制主要有 ArchStudio 工具集和软件体系结构助理。

1.4 动态体系结构研究的必要性

当前,软件体系结构研究主要集中在静态体系结构上,这种体系结构在运行时不能发生改变。但是,对于一些需要长期运行并具有特殊使命的系统(例如金融系统、航空航天系统、交通系统、通信系统等),如果系统需求或环境了发生变化需要更新,此时停止运行进行更新或维护,将会引起高额的费用和巨大的风险,对系统的安全性也会产生很大的影响。静态体系结构缺乏表示动态更新的机制,很难用它来分析描述这样的系统。因此,动态体系结构的研究应运而生,动态体系结构主要研究软件系统由于特殊需要必须在连续运营情况下的体系结构变化与支撑平台。

研究动态软件体系结构模型的主要作用是:

(1)减少系统开发的费用和风险,由于采用动态体系结构,一些具有特殊使命的系统能够在系统运行时根据需求对系统进行更新,并降低更新的费用和风险;

(2)增强用户自定义性和可扩展性,动态体系结构能够为用户提供更新系统属性的服务。

二 动态体系结构的基本问题

当前主流的体系结构模型 CORBA、COM/DCOM、EJB 等,都不能支持体系结构的动态更新。同时,动态体系结构由于其自身的复杂性,比静态体系结构需要更多的形式化描述机制和分析工具,形式化描述机制用来描述运行时的更新。分析工具用来帮助验证这些更新的属性。由于缺乏通用的结构模型、有效的形式化描述机制和分析工具,使得目前学术界对于动态体系结构的研究还不成熟,处于摸索阶段。[4]

2.1 动态更新时期

一般来说,体系结构的动态更新主要发生在以下四个时期:设计时期、预运行时期、限制性运行时期和运行时期。设计时期是当前体系结构研究的重点;运行时期,代表了灵活的体系结构运行更新;另外两个中间时期与纯动态实例相比提供的灵活性较少,但是它们易于分析和执行。

(l)设计时期:更新发生在体系结构模型和与之相关的代码编译之前。由于仅仅是系统的抽象模型发生更新,这样的更新相对来说比较容易理解和实现。

(2)预运行时期:更新发生在编译之后,但在运行之前。由于系统并未运行,在更新时无需考虑系统状态。这些更新要求系统模型包括添加、删除构件等机制。

(3)限制性运行时期:当满足一定的预先描述的限制时,更新才能执行。当应用系统处于一个与更新相关的“安全”状态时,允许更新基于程序状态的限制条件。

(4)运行时期:更新发生在系统运行时期。这四个时期的动态更新以及应对策略都是不同的,应该对每个时期的情况进行研究分析,从而找出相对应的策略。本文主要研究运行时期发生的更新(动态体系结构的更新)及其形式化描述和分析方法。

2.2 运行系统

运行系统是体系结构动态更新的引擎和核心部分,运行系统的职责包括:

(l)维护系统的安全性和稳定性。由于体系结构需要在系统运行时被更新系统必然会受到影响,运行系统必须确保系统的安全性和稳定性;

(2)确保所请求的体系结构更新在所允许的更新限制的范围内;

(3)利用环境所提供的机制来执行更新,例如,构件的动态装载和链接,将构件映射到处理元素的处理迁移机制和构件间通信的进程交互机制等等。如果缺乏这些机制,将会限制动态更新的范围。运行系统有多种形式,它可能在系统编译时被编译为一个库函数,也可能是更新操作的一部分,或者是一个独立的部分。它的形式依赖于很多因素,包括更新操作的类型、更新限制的类型和执行环境的属性等等。

2.3 动态更新操作

体系结构的动态性允许系统在运行时进行演化。描述这种系统的体系结构必须能够被允许这些改变的 ADL 所表示。现有的大多数 ADLS 仅支持静态体系结构描述,而不提供支持体系结构动态性的机制。通常,动态体系结构的更新操作包括:

(1)添加新的构件:在系统运行时添加新的构件,不可假定系统正处于此构件的初始状态,构件必须检测系统的状态,并执行必要的操作来同步系统的内部状态;

(2)升级或替换已存在的构件:系统的某一构件能够被另外的具有更好性能的构件替换,需升级或替换的构件状态要转移到新的构件上,在变化过程中,两个构件不能同时被激活,这个替换需要通过相同的接口;

(3)删除不必要的构件:系统不再使用某一个构件时,这个构件将被删除,这种删除包括临时删除和永久删除。每个具体的应用中有合适的条件管理构件的移除。例如,当一个构件中的任何一个功能仍在系统执行进程的堆栈,系统的运行环境就会禁止构件移除;

(1)应用体系结构的重新配置:通过添加或删除构件间的连接改变系统的拓扑结构,例如构件和连接间的重新连接;

(5)系统体系结构的重新配置:将某一构件从一台主机上移到另一台主机上。这样,体系结构必须支持构件到处理器之间映射的修改;

(6)查询系统元素的属性,例如获取当前版本的信息;

(7)查询当前系统的拓扑结构。

通常,系统的一个更新会引起与之相关的一些操作。例如,在更新某一已存在的构件之前,需要做一些验证工作:(1)存在的构件的版本比更新的构件版本旧;(2)依赖于处理器的构件不能受到影响;(3)存在新的处理器构件依赖的构件。如果验证工作(2)或(3)失败了,将更新已存在的构件。

2.4 动态更新方法

对于运行系统的动态更新问题,一般来说,有以下几种更新方法:

(1)动态链接:动态链接允许动态更替对象,但不要求从对象中获取任何功能。

这种方法主要适用于不具有持久状态的对象。

(2)静态状态转换:适用于具有持久状态的对象。这种状态转换有一些步骤:首先一个程序状态被保存,此程序停止运行,然后用新的版本取代旧版本的对象,旧版本先前的状态转移到新版本上。如果该程序不能被停止,例如它是系统的核心部分,那就要用到下面的方法。

(3)动态状态转换:利用状态转换函数,可以把一个运行系统的状态转换到新的版本,而且这个函数能够被自动生成。

2.5 更新限制和更新优化

体系结构更新的最大风险就是破坏系统的整体性,因此,在体系结构发生更新时,需要一些机制来维护系统的完整性。

更新限制主要是限制更新操作、特殊构件、更新时期或者是系统内部的组装等等。在一个更新执行之前,它们要求一些功能属性被验证。如果所有限制不能够同时作用,与系统行为属性相关的更新允许被替换。举个例子来说,对于一个整体性依赖于实时限制的系统来说如果没有足够的工具来验证运行系统的一些属性,那么体系结构更新就被局限于某个子系统中,而不能作用于实时系统中。

动态更新排除了静态体系结构优化使用的一些类。例如,在静态体系结构模型中,一个常用的优化方式是用直接过程调用去执行体系结构连接件。由于在运行时新的构件可能会添加到连接件上,因此不能将这种方式应用到动态体系结构上。但如果对体系结构加以限制,例如“新的构件不能被添加到连接件 X 或从连接件 X 上删除”等等,这样虽然优化了性能,但牺牲了灵活性。因此,动态体系结构的设计者在设计时必须平衡灵活性和性能之间的关系。

三 CBDA 动态体系结构模型

3.1 基本概念

在不同的环境下,构件、连接件有不同的理解。在本文提出的模型中,它们的定义如下:定义 1 构件是一些单个且独立的实体,它可以是一个单独的程序也可以是整个的系统应用。构件包含执行和接口,一个构件可能包含多个对象。接口定

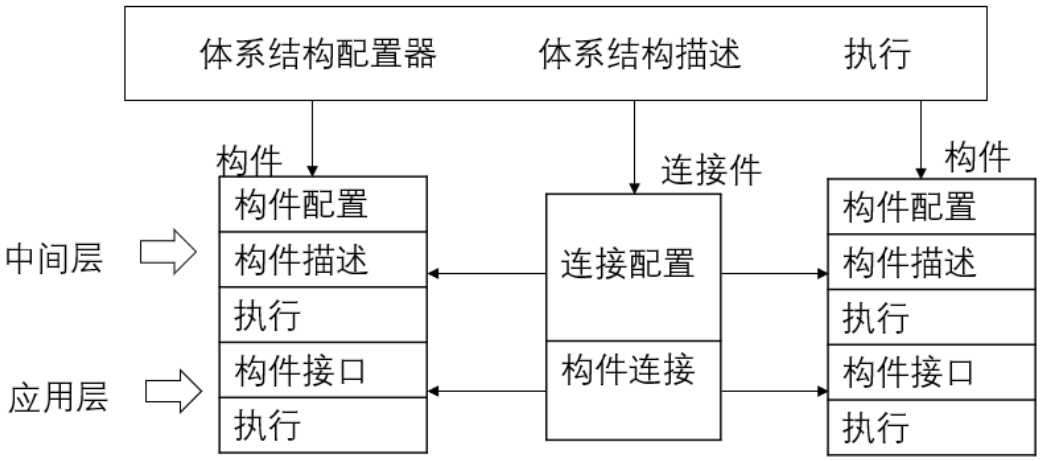
义了处理构件间通信的对象。所有与构件配置有关的信息被存储在构件描述中。

定义 2 连接件是封装了构件间通信的实体。构件间不能直接通信,它们通过连接件进行通信。连接件存储了更新过程中通信通道之间的状态。由于连接件和构件的定义类似,连接件可以被看作构件。与构件不同的是,连接件不对应于执行系统中的单元。

3.2 CBDA 模型介绍

本节介绍了一种基于构建的动态体系结构模型，如下图所示：CBDA 模型支持运行系统的动态更新,它分为三层:应用层、中间层和体系结构层。

应用层处于最底层,包括构件连接、构件接口和执行。构件连接定义了连接件如何与构件的相连接;构件接口说明了构件提供的服务,例如消息、操作和变量等等。在这一层,可以添加新的构件、删除或更新已存在的构件。[3]



中间层包括连接件配置、构件配置、构件描述及执行。连接件配置主要是管理连接件及接口的通信配置;构件配置管理构件的所有行为;构件描述对构件的内部结构、行为、功能和版本等信息加以描述。在这一层,可以添加版本控制机制和不同的构件装载方法。

体系结构层位于最上层,控制和管理整个体系结构,包括体系结构配置器、体系结构描述和执行。其中,体系结构描述主要是描述构件以及它们相联系的连接件的数据;体系结构配置控制整个分布式系统的执行,并且管理配置层;体系结构描述主要是对体系结构层的行为进行描述。在这一层,可以更改和扩展更新限制,

更改系统的拓扑结构,更改构件到处理元素之间的映射。[6]在每一层都有一个执行部分,主要是对相应层的操作进行执行。

在更新时,必要情况下将会临时孤立所涉及的构件。在更新执行之前,要确保:(1)所涉及的构件停止发送新的请求;(2)在更新开始之前,连接件的请求队列中的请求全部已被执行。而且,模型封装了连接件的所有通信,这样可以很好的解决动态更新时产生的不一致性问题

3.3 更新请求描述

更新可以由用户提出,也可以由系统自身发出请求。

一般来说,一个更新描述包括以下几个部分:

(1)更新类型(updateyteP):更新类型包括添加、删除和更新一个新的构件;

(2)更新对象列表(listofupdated。bjeCtS):需要更新的对象类的 ID 号;

(3)对象的新版本说明(enwversionSoftheobjects):对象的新版本执行情况;

(4)对象更新方法(updatemethod):更替、动态及静态;

(5)更新函数(uPdatefunction):用来更新一个执行对象进程的状态转换函数;

(6)更新限制(updateconstraints):描述更新(包括子更新)和它们之间的关系的序列,例如只有对象 A 的版本)2.0 时,对象 A 才能被更新。

3.4 更新执行步骤

按照 CBDA 模型的结构,对系统进行更新,一般来说,有以下几个步骤:

(l)检测更新的范围。在更新执行之前,首先要判断是局部更新还是全局更新,局部更新作用于需更新构件的内部而不影响系统的其他部分,全局更新影响系统的其他部分,全局更新需要发送请求到更高的抽象层。

(2)更新准备工作。如果更新发生在应用层,构件配置器等待参与的进程(或线程)发出信号,以表明它们已处于可安全执行更新的状态;[6]如果更新发生在配置层,就需要等待连接件中断通信和其他构件配置器已完成它们的更新;如果更新发生在体系结构层,就直接执行。

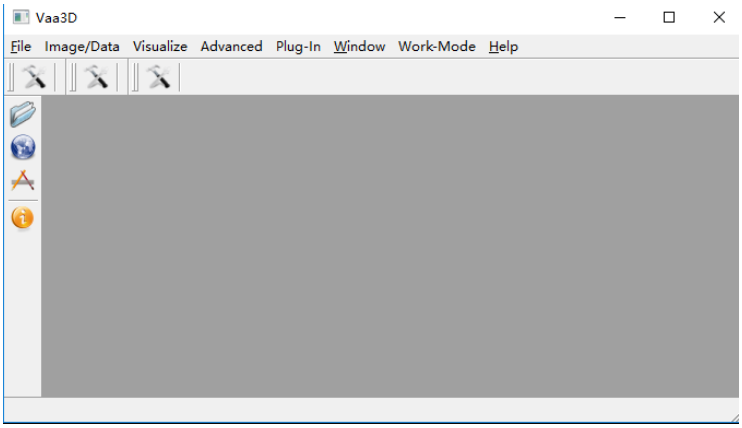
(3)执行更新。执行更新,并告知更新发起者更新的结果。

(4)存储更新。将构件或体系结构所作的更新存储到构件或体系结构描述中。

四 Vaa3d 实例分析

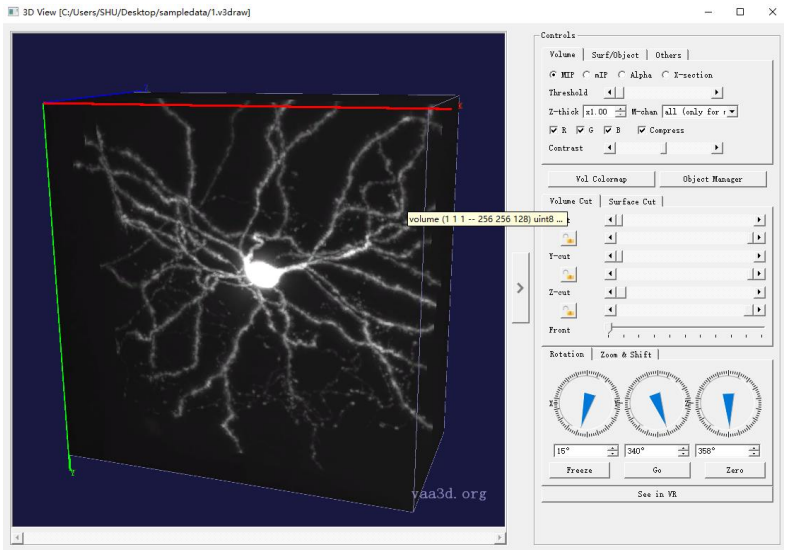
本章采用硕士期间的开源项目 Vaa3d 实例，并用两种更新（局部更新和全局更新）来分析 CBDA 模型如何支持体系结构动态更新。

4.1Vaa3d 布局简介



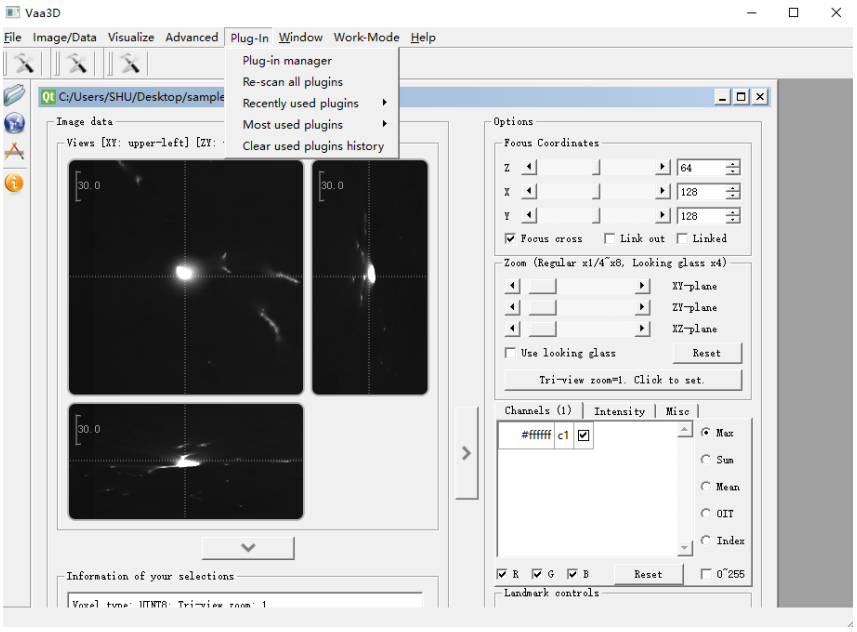
Vaa3d 主要作用于神经元图像分析和标注工作。其软件价值主要在于可以标注全脑级别的神经元，并依靠软件中提供的工具以及插件功能，帮助标注人员对神经元更快的重建，来获得大量的数据。

如下图所示，软件可以对神经元进行三维方向上的观察，统计以及标注。从而把图像数据转化为计算机可以分析的文本数据。



4.2 局部更新

当我们对软件功能有了一定更新之后，比如添加了新的插件，来帮助用户处理图像，Vaa3d 提供了新的插件接口，使得插件（作为一个构件）和程序主体不需要同时进行第二次编译。这里，我们把新编译的插件生成为在 windows 操作系统下的 DLL 动态链接库，只需要放入 Vaa3d 对应的文件夹下面，Vaa3d 在不需要重启的情况下，可以通过插件菜单直接观察到新生成的插件，如下图所示。从而直接完成了构件 A 到连接件之间的准备更新以及更新之后重新连接的过程。（主要通过 C++动态链接库来实现）



整个过程可以描述如下：

1.更新发起者发出一个更新请求,这个请求被送到构件 A 的配置器中,构件配置器将分析更新的类型,从而判断它是接口对象的局部更新;

2 由于更新为局部更新,构件 A 的配置器发出一个信号给连接件以隔离构件 A 的通信,准备执行更新;

3.构件 A 的配置器开始执行更新。

4.更新执行完毕后,构件 A 的构件描述被更新,并且构件 A 发送一个消息给连接件 B,两者间的连接被重新存储起来;

5.将更新结果返回给更新发起者。

4.3 全局更新

当我们不仅仅对插件功能进行更新，而是对软件的整体功能进行修改，比如三维图像的显示时。由于我们的整体项目部署在 Github 上，除了开发人员，大部分用户只需要拉取可执行版本即可，而不需要像开发人员一样对整个项目进行重新编译再进行使用。所以当我们对整体进行更新的时候，开发人员再本地进行更新，提交更新请求，把代码传入 Github，运营人员下载最新代码，进行更新检查，然后通知用户，当准备就绪时，运营人员直接将最新版本提供给标注人员，通知更新结果并返回相应信息。

在这个全局更新的过程中，github 仓库充当了连接件的角色，运营人员充当了体系结构的配置器。开发者作为更新发起者，通知各部件进行更新，在没有影响原先程序运行的情况下，维护了系统的一致性。

参考文献

[1] 李琼,姜瑛.动态软件体系结构研究综述[J].计算机应用研究,2009,26(06):2352-2355.

[2] 李萧玮.计算机体系结构软件模拟技术[J].电子技术与软件工程,2018(21):38.

[3] 于振华,蔡远利,徐海平.动态软件体系结构建模方法研究[J].西安交通大学学报,2007(02):167-171.

[4] 陈向东.动态自适应软件体系结构重配置研究[J].计算机科学,2015,42(06):185-188

[5] 吕平,刘勤让,邬江兴,陈鸿昶,沈剑良.新一代软件定义体系结构[J].中国科学:信息科学,2018,48(03):315-328

[6] 晏郑勇. 基于动态软件体系结构的软件自适应性研究[D].华中师范大学,2014.