适应性软件体系结构研究

Research on Adaptive Software Architecture

李 刚 金茂忠

(北京航空航天大学计算机系 北京100083)

Abstract In this paper, the necessity of researching on adaptive software architecture is expounded from the angle of impacts of Internet on software; the status quo of software architecture and software adaptability is summarized; the open problems and our viewpoints are presented. To contribute to improving software architecture adaptability, much attention should be paid to the following: (1) the theory and methodology of software surviving environment system and system analyzing, (2) the model of adaptive software architecture and the formal theory, (3) the methodology of adaptive software architecture-based software development and architecture requirement analysis.

Keywords Adaptive software architecture, Architecture-based development

随着计算机技术的发展,信息系统渐渐深入到社会的各个领域,Internet 正逐渐发展成第四媒体,成为社会生活中不可分割的一部分。电子商务的兴起使互联网应用从最初单纯的科研领域扩展到社会经济领域,也促进了 Internet 乃至整个 IT 产业更加迅猛的发展。Web 的商业价值越来越重要,到2003年全世界电子商务的总支出估计将达1.3亿美元[2]。根据CNNIC2001年1月的中国互联网络发展状况统计报告[1],目前我国已有上网计算机892万台,WWW 站点265405个,其中商业站点221988个,占83.64%。Web 的发展使 Internet 环境下软件的需求越来越大。

Internet 环境下的软件分布、异构,多用多种语言开发,复杂性大,需求也更具易变性。现有开发方法和软件结构已暴露出一些不足,如:不适合大型复杂系统开发,不能从整体上对软件进行规划和设计,软件灵活性、重用性差等。同时,Internet 缩短了人们的时空距离,加剧了软件市场竞争。加剧的行业竞争要求方法不仅要能降低开发费用,而且能迅速适应市场变化,开发出来的产品能在一定时期、范围内经受住市场、用户等的变化,结构化方法和面向对象方法在这方面有明显欠缺,在现有构件模型基础上开发的构件也只能在功能、结构完全或近似匹配的环境下才能较好地使用,当出现体系结构不匹配时很难对其做出必要修改。适应性的不足限制了构件重用,Internet 环境下这种矛盾更加突出,要真正实现基于 COTS 构件的组装式开发还有许多问题需要解决。

在此背景下,适应性软件体系结构的研究显得尤为重要[2]。具有良好适应性的构件、软件体系结构及基于该软件体系结构的开发方法能够从软件的局部和整体以及开发过程上为问题的解决提供新思路。

本文首先从 Internet 环境下软件的特点入手论述了研究适应性软件体系结构的必要性,然后从软件体系结构和软件适应性两方面简要介绍了与适应性软件体系结构相关的工作,并在此基础上提出了适应性软件体系结构研究的三个方向。

1 Internet 对软件的影响

在开放的 Internet 环境下,软件要对各种不可预知的事件做出响应,应能随环境要素的变化及时演化,面向对象语言、框架、设计模式尚不能满足软件在动态性、适应性等方面的设计需要[10]。技术、用户、市场等环境要素更具易变性,要求软件过程/体系结构应具有一定的适应性。由于 Internet 的迅猛发展和应用形式的不断更新,层次化体系结构暴露出一些不足,如在系统性能、负载平衡、动态界面等方面还不尽人意[11]。尽管目前的 Web 应用已能提供个性化服务,但在对用户环境、需求、商务逻辑等变化的适应上仍存在不足。

与其它环境下的软件相比,Internet 使软件呈现以下特点:

(1)处理的信息量大,形式多样,不确定性强。Internet 环

平台总体的所有优点之外,还可以实现可分离,从整个平台中分离出来,用于指导分配需求的管理。

(3)需求管理子平台只是在总体框架上对需求管理进行了研究,并没有拘泥于某具体管理方式和技术,这有利于组织根据自己的需要进行定制,只要是按这个框架进行,就能最大限度地保证需求管理的效率和质量。

(4)软件本身在体系结构上具有可扩展性,组织可以根据自己的需求管理的力度来定制需求管理活动的最小粒度,进一步细化上述的过程流程。

参 考 文 献

1 [美]威伊格斯(KarlE. Wiegers)著,陆丽娜等译. 软件需求. 北京:机械工业出版社, 2000.7

2 Paulk M C. et al. Key Practices of the Capability Maturity Model SM , Version 1.1. Mary Beth Chrissis , Marilyn Bush: [CMU/SEI-93-TR-025]

3 Thyer R H, Dorfman M, eds . Software Requirements Engineering, 2d ed. Los Alamitos, CA : IEEE Computer Society Press, 1997

4 I Sommerville, Sawyer P. Requirements Engineering: A Good Practice Guide. Chichester, England: John Wiley&Sons. 1997

5 McConnell S. Rapid Development: Taming Wild Software Schedules . Redmond. WA: Microsoft Press. 1998. Software Project Survival Guild. Redmond WA: Microsoft Press, 1996

6 李怀璋,李明树. 基于 ISO9000和 CMM 的软件质量保证平台的研究:[第一届中国博士后大会学术会议论文]. 2000.10

7 ISO9000-3质量管理和质量保证标准选用或实施指南 ISO9001在软件中的应用, 1994

境下的软件涉及大量数据，其中既有结构化的也有非结构化的，数据的组织、存储方法及交互格式存在很大差别，分布式特点增大了信息更新、处理、发布的不确定性。

(2)运行平台分布、异构。C/S 模式是 Internet 环境下的基本计算模式，PC 机上集中、同构的计算结构在 Internet 环境下已经很少见，应用不仅在逻辑上而且在物理上分成多个层次，开发、调试、运行更加复杂。软件系统边界发生了很大变化，甚至包括了与之或将要与之发生交互的系统，实现方式、运行平台往往异构、不确定。即使是同一商务逻辑，在很多情况下也需要运行在不同的平台上[11]。

(3)用多种语言开发，复杂性大。Internet 环境下的软件采用程序＝脚本描述＋构件的设计范型，已很少用一种语言开发。涉及多种语言的构件组装、与遗留系统的集成、商务逻辑的易变性、平台的异构与分布、数据的多样性等诸因素都使系统结构和行为呈现出复杂、动态的特点。真正分布式的 Web 程序并没有在扩展性、易维护性等方面取得新进展，甚至反而使问题变得更加复杂。

(4)环境开放、易变。与传统 C/S 的持续性连接不同，Internet 下的 C/S 是一种间断性连接，Web 页面间信息的传递也更加复杂、不确定。在 B2B 交易中企业存在更多的潜在客户群和合作伙伴，客户关系管理更加复杂，物流和供应链也经常发生变化，供应链甚至延伸到一些不确定或不可知的生产商和供货商。各企业生产模式、管理方式、交易模型的变化往往互相影响，引起软件系统的变化。Internet 不仅要求服务个性化而且软件也更加个性化，电子商务领域尤其如此。此外，资源组合不确定、新技术不断出现等使软件对各种变化的适应性显得更加重要。

(5)安全性要求高。Internet 环境下软件对安全性的要求已是不争的事实。

上述特点要求 Internet 环境下的计算不仅要稳定、可靠，而且还要有相当的适应能力。一方面在网络基础设施、平台发生变化或故障，影响到网络及应用的安全、效率和功能时软件能及时做出响应并平衡负载，另一方面用户的功能性/非功能性需求或其他环境要素发生变化时软件能进行自动、半自动或手工的重组、配置、扩展或更新，较好地适应变化。Internet 环境的特点要求软件能适应多种环境要素变化，同时也使软件重用更加困难。

从体系结构入手进行软件开发及维护不仅可以控制问题复杂性，提高开发效率和系统质量，而且还可以通过结构改善使目标系统的适应性、演化性得到保证。为此，需要对适应性软件体系结构进行研究，下面将首先对软件体系结构的有关情况做简要介绍。

2 软件体系结构

60年代，软件体系结构的思想萌芽最早在 Dijsktra 和 Spooner 的论述中出现[3]，80年代末90年代初 Shaw，Perry 和 Wolf 等人将其发展成为一个独立的研究领域。目前软件体系结构还没有标准、统一的定义。在收集了众多观点、研究[12,13]之后，我们对其中的60个进行了分析和比较。对软件体系结构的研究主要有以下几个方面：软件体系结构表达方式、软件体系结构形式化、软件体系结构分析、基于体系结构的软件开发与演化、软件体系结构恢复和重用、软件体系结构的设计和选择等。文[4]对相关现状及存在问题有详细介绍，在此仅对其欠缺部分加以补充。

软件体系结构从高层、宏观上讲是整个系统的简略、抽象的框架结构，如模块结构等，从低层、微观上讲是高层框架结构的实现，是具体的构件及其相互作用关系，包括数据流、控制流、对象间的消息传递关系等，体现为类图、合作图、活动图、配置图等多种形式。软件体系结构的重用、设计与具体应用环境有关。

对软件体系结构(风格)本质特性和理论基础的研究是抽象的，其成果有较为广泛的指导意义；特定领域的软件体系结构是具体的，如：安全结构、医院管理信息系统体系结构、操作系统体系结构等，对特定领域软件的大规模重用及软件质量的提高有重要意义。

不同视角看到的体系结构是不同的。从用户角度看，它应描述软件的功能分布、使用方法；从开发人员角度看，它是一种高层次设计蓝图；从开发过程角度看，它对应一组与结构相关的活动，这组活动能将体系结构转换为软件实现；从管理过程角度看，它体现为开发组织的结构划分，并与一组管理活动相关；从工程原则角度看，它体现为一组指导工程实践的原则或约束；从生命周期角度看，它体现为需求层、设计层、代码层上的体系结构描述；从静态/动态特性角度看，可分为动态、静态结构；从结构构件抽象层次角度看，它体现为概念结构、模块互连结构、代码结构；从与硬件环境的关系及软件分布的角度看，它体现为软件系统物理配置和分布，与系统物理模型相对应。

不同观察角度会导致不同的描述模型，即使是在同一种模型中不同的侧重点也会导致不同的结构视图。不同体系结构模型的用途也有所不同，如：概念模型对理解问题空间意义较大，模块互连结构对资源分配、项目组织和规划、系统结构重组有很大帮助，代码结构描述了代码层上的构件及其相互关系和如何将源代码、二进制代码、程序库在开发环境中组织起来的方法，与程序最终实现关系密切。

随着 Internet 环境下软件复杂性的增加及用户需求的飞速增长，构件、软件体系结构在适应性上的需求也越来越迫切。正如 Gérard Roucairol 在 WCC2000上所说：软件系统范型正在从软件＝实现＋功能＋预先确定的环境(资源、工具等)转换到软件＝实现＋功能＋(动态)变化的环境。适应的起因往往是外部的，即由环境要素变化造成的，与软件体系结构无关，而变化对软件的影响及软件对变化的适应能力是与结构有关的。

迄今对软件体系结构的研究多集中在共性上，忽视了适应性的研究，虽已开始对动态特性的研究，但多局限于结构本身，很少将环境因素考虑在内。在软件体系结构适应性研究上的欠缺已给重用和基于体系结构开发的大规模使用造成了很大障碍[8]。

3 软件的适应性

另一个与适应性软件体系结构相关的研究领域是软件适应性，在此将对有关状况做详细介绍。

适应性与软件的扩展性、修改性、动态特性有密切关系。为了提高软件的修改性、扩展性及动态特性人们做了很多努力，问题虽然得到了一定程度改善，但远没有得到较好解决。例如：虽然面向对象的信息隐藏和继承机制为软件中的变化提供了一定支持，但为了适应变化软件构造者必须熟练、清楚地掌握类继承层次，同时还要限制适配器的使用范围，以保持结构的完整性。此外，由于缺乏语义信息软件构造者无法从接

口上获得适应性修改所需的必要信息，信息隐藏的优点在环境变化时恰恰阻碍了对软件的适应性修改。面向对象还不能对软件适应性提供直接有效支持。

ISO/IEC 9126中将软件适应性定义为移植性的一个子特征，与软件不采取任何步骤或方法就能适应不同环境的机会有关。文[3,4]都对适应性进行了有益探讨(表2给出了软件适应性研究中较有代表性的8种观点)。由于软件生存环境及其变化多种多样，适应性的类型和模式也纷繁复杂。对适应性的分类有多种。按开放性可分为：

开放自适应：适应性受环境中其他软件或硬件实体的影响；

封闭自适应：适应性是孤立的不受其他实体的影响。

按对不可预见变化的考虑深度可分为：

主动适应：对不可预见变化的适应能力较强，无需对软件实体进行改动就可以适应环境变化，是一种完全自适应。

被动适应：只能适应可预见的变化，必要时要对软件实体进行修改，又称为可适应。

按实现方式可分为：

条件表达式：这是一种比较简单的适应性行为实现方式，它根据对条件表达式的评估结果执行相应程序。

在线算法：适应性建立在未来事件(输入)不确定假设基础上，利用问题域或输入域知识实现自适应，效率较高。

类属算法：通过类型实例化或外部输入达到行为参数化的目的，从而实现对变化的适应。

算法选择：用该方法实现的软件能根据运行环境特点及约束在一组算法中选择最有效的算法执行。

利用遗传算法及演化计算实现的适应：能根据运行环境特点和适应环境的历史记录，生成新算法，其中要用到机器学习、演化计算、遗传算法等人工智能知识。

表1

	观点与定义	提出者	时间
著作与论文观点	适应性软件是一种可对体系结构实例化的类属软件，它由三部分组成：小子图规约、初始行为规约和行为增强性描述。	Kar Licberherr	1995
	适应性程序应带有一个与之一致的功能规约，而一个自适应规约与它的一个部分实现相关联，通过逻辑结构和操作行为的交互式演化可以实现软件的适应性。	Dusko Pavlovic	1998
	适应性软件是能随其操作环境的变化改变行为的软件。	Peyman Oreizy	1999
	适应性系统应由主构件和适应器两部分组成，主构件实现系统的预期功能，适应器完成系统的适应性任务。	Akos Ledeczi	2000
Web 论点	适应性软件是能随环境变化而改变行为的软件。不同含义的环境会导致不同的适应性。		
	适应性软件是一种类属软件，它建立在抽象数据模型基础上，包含了一组程序，其体系结构有很大可变性。		
	适应性软件由数据和功能两部分组成，这两部分通过约束松散的耦合在一起。		
	就多媒体软件而言，适应性可以看作是一种问题的转换：用户说明一个应用于同步多媒体描述形式的断言，然后根据该断言生成客户化的描述形式。		

目前，软件领域较有代表性的软件适应性研究工作有：

(1)美国东北大学的 Karl Lieberherr 等人提出了一种适应性软件设计方法——Demer 方法[7]。Demer 方法建立在面向对象思想基础上，认为类结构也是一种可变的环境因素，在类结构变化时应用程序也应发生相应变化，其中用于实现软件适应性的核心机制是可增殖模式中的可遍历小规约。Demer 虽然偏重于结构风格程序设计，但它是一种比较完整、系统的适应性软件开发方法。

(2)Peyman 等人提出了一种基于软件体系结构的适应性软件开发方法[5]，该方法分为演化管理和适应管理两部分，前者对适应性软件的生命周期进行了粗略描述，后者简要给出了软件适应性修改的方法。该方法的创新之处在于将软件体系结构、适应性等思想集成到了一个综合性方法中，但该方法只是框架性的，对动态软件体系结构、基于事件的监测与评价及软件定制的支持还很欠缺。

(3)麻省理工学院人工智能研究所对软件适应性及适应性软件基础做了一些研究[6]。他们在动态领域体系结构基础上构造适应性软件，动态领域体系结构将应用领域划分为多个通用服务层，每个服务都有数个可用于不同环境的实现。此外还在软件体系结构层上提供了一种目标连接机制，用于描述如何将多个服务构件组织在一起以实现系统整体目标。环境变化自动诊断功能的设计与实现是该项研究的最成功之处，但在应用框架和体系结构的研究上存在不足，对构件组装方法的研究也很欠缺，缺少用于适应性软件开发的通用方法。

(4)伍斯特工学院计算机系的 George T. Heinman 等人对适应性构件进行了研究，他们认为灵活的规约是灵活构件重用的前提，并在 JavaBean 基础上引进了主动接口、端口和角色机制，对复杂构件的接口进行详细说明，以方便构件的适应性修改。目前，他们还没有建立起基于规约修改的构件自适应方法和可增加代码的构件扩展机制[9]。

综合其他研究，在软件适应性这一领域还有以下问题需要进一步探讨：

(1)由适应性修改引起的与软件安全性、可靠性、正确性等相关的问题。运行时动态改变体系结构或构件及对软件的适应性修改往往会对软件的正确性、一致性和安全性造成损害。这些问题目前还没有得到很好解决。

(2)对环境变化的分析。适应性强的软件应能适应多种环境和复杂变化，而变化的不可预见性增大了适应性软件设计的困难，如何对软件生存环境进行定义，分析其中的可变因素，确定适应类型，制定适应策略，目前尚未进行相应研究。缺少描述适应性软件应用环境的方法，这种方法一方面要能描述适应类型、应用条件，另一方面要易于理解，易于使用，同时能充分表达适应性语义。

(3)适应性与程序复杂性、演化性等的矛盾还没有一个很好的解决方法。将软件体系结构模型与程序分离，增大了系统灵活性，但也增大了系统开销。参数化代码也增大了代码的复

杂性。实践证明适应性软件明显比普通软件庞大、复杂。如何解决这方面的矛盾还需进一步研究。

(4)对适应性的定性或定量度量还没有很好的方法,定量度量尤其困难。

(5)对适应性构件、软件体系结构的研究还很欠缺。在实现上还需对动态体系结构进一步研究,以便适应用户需求变动和软件体系结构修改的要求。

结构是最终解决适应性问题的基础,无论是适应性修改,还是自适应算法都需要通过一定的数据结构、计算结构实现。具有良好适应性的软件体系结构不仅有助于控制问题复杂性,解决适应性与其他质量属性的矛盾,而且有助于软件重用。在复杂、易变的 Internet 环境下研究适应性软件体系结构及基于该结构的软件开发方法显得更加现实和有意义。但目前,对适应性软件体系结构的研究还很欠缺。

4 研究方向

在对现有工作进行分析的基础上,我们将对适应性软件体系结构做进一步研究。鉴于软件体系结构概念尚不统一,在工作中将采用如下定义:软件体系结构是对软件系统静态和动态拓扑结构的刻画,可以通过结构构件、构件连接和结构约束来描述。

结构构件(以下简称构件)是体系结构的重要组成部分,构件连接可以是构件(可以通过过程调用、消息机制、共享变量实现,也可以通过连接构件实现),体现了构件间静态或动态的连接关系。按照构件在体系结构中的地位和作用,可将其分为计算构件、数据构件和连接构件。结构约束从质量属性、结构、应用领域、功能需求、非功能需求、风格,甚至外观和内在美感等方面对体系结构进行了限定。软件体系结构实现形式多种多样(如 Len Bass 曾在其论著中给出了软件体系结构的10种不同视图[12]),研究中抽象原则的应用角度也不尽相同。对软件体系结构的描述应从静态和动态两方面进行。

对于适应性将采用如下定义:适应性是指当软件生存环境发生变化时软件实体的结构或行为可以随之改变的特性。适应可分为两个层次,一类实体本身能够自动适应环境变化,称为完全自适应;另一类需要对其修改、配置(人工或自动)后才能适应环境变化称为可适应。具有上述特性的软件、软件体系结构及构件分别称为适应性软件、适应性软件体系结构和适应性构件。

在此软件实体是指软件、软件体系结构(框架)或构件,软件生存环境是指影响实体的一系列有利或制约条件。

针对当前研究现状我们将从软件生存环境系统理论、适应性软件体系结构模型、基于适应性软件体系结构的开发方法三方面进行深入探讨。

(1)提出软件生存环境系统理论,研究软件生存环境要素分析的理论和方法,分析其中的可变因素和不变因素,以指导面向适应或面向重用的体系结构、构件或程序设计。生存环境系统理论将系统地对影响软件的诸要素及其与各质量/价值属性的关系进行研究,以提高软件实体的适应性,延长其使用周期。

(2)研究适应性软件体系结构模型,从构件、构件连接和结构框架三个层次上研究体系结构适应性。通过结构接口实现构件结构修改,通过构件运算与变换提高复合构件的适应性。定义基于角色构件的连接模型,包括基于角色构件连接的结构、角色特征分类及其描述方法、构件交互协议设计、变化管理。将计算结构与控制结构分离,在框架层上提高结构的适应性。

(3)研究基于适应性软件体系结构的软件开发方法,引入体系结构需求分析的概念和方法,并探讨开发过程的适应性。环境变化要求开发过程也应具有适应性;另外我们将从环境要素分析入手获取、分析体系结构需求,通过体系结构需求分析可以综合各项软件的质量要求,减少它们之间的冲突。

文中所述是我们阶段性研究成果的一部分,目前已经提出了软件生存环境系统的理论框架及灰盒构件模型,并在理论框架指导下结合工程实践总结了适应性构件、体系结构设计的一般性原则。有关内容将另文介绍。

结束语 本文从 Internet 对软件的影响入手论述了适应性软件体系结构研究的必要性,对相关领域现状进行了综述、分析,指出了存在的问题。并在此基础上提出了在适应性软件体系结构及软件设计方法研究中应解决的主要问题。

参 考 文 献

1 Garlan D, et al. Architectural mismatch. In: Proc. Intl. Conf. on Software Engineering, 1995

2 Hilliard R. Aspects, concerns, subjects, views. the OOPSLA'99 Workshop

3 Baragy J, Reed K. Why is it so hard to define software architecture?. 5th Asia-Pacific Software Engineering Conf., 1998

4 Perry, Wolfe. Foundations for the study of software architecture. ACM SigSoft, 1992, 7(4)

5 Self-adaptive software for automated design of complex. Engineering Systems. http://www.cs.rutgers.edu/hpcd/sas/proposal/

6 Oreizy P, et al. An architecture-based approach to self-adaptive software. IEEE intelligent systems, 1999(May/June): 54~62

7 Licberherr K, et al. Workshop on adaptable and adaptive software. http://www.ccs.neu.edu/research/demeter/adaptable-systems/

8 Heineman G T. Adaptation and software architecture. http://www.cs.wpi.edu/~heineman/workshop/position.html

9 Ohlenbush H, et al. Complex ports and roles within software architecture: [Technical Report, WPI-CS-TR-98-12, WPI]. 1998

10 Kitayama F, et al. Design of a framework for dynamic content adaptation to Web-enabled terminals and enterprise application. In: Proc. of the Asia Pacific Software Engineering Conf., 1998

11 Cho E S, et al. Object-oriented Web application architectures and development strategies. APSEC '97 / ICSC '97, 1997

12 Bass L, Kazman R. Architecture-based development: [Technical Report, CMU/SEI-99-TR-007]. CMU, 1999

13 Shaw, Garlan. Software architecture: perspectives on an emerging discipline. Prentice-Hall, 1996