ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.17, No.6, June 2006, pp.1338−1348
DOI: 10.1360/jos171338
© 2006 by *Journal of Software*. All rights reserved.

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

一类通用的适应性软件体系结构风格研究*

黄双喜,+ 范玉顺, 赵 彧

(清华大学 自动化系,北京 100084)

Research on Generic Adaptive Software Architecture Style

HUANG Shuang-Xi+, FAN Yu-Shun, ZHAO Yu

(Department of Automation, Tsinghua University, Beijing 100084, China)

+ Corresponding author: Phn: +86-10-62899634, Fax: +86-10-62770351, E-mail: huangsx@tsinghua.edu.cn, http://www.tsinghua.edu.cn

Huang SX, Fan YS, Zhao Y. Research on generic adaptive software architecture style. *Journal of Software*, 2006,17(6):1338−1348. http://www.jos.org.cn/1000-9825/17/1338.htm

Abstract: The generic adaptive software architecture style is studied. Through the analysis of the basic characteristics of adaptive software architecture, the generic framework for adaptive software architecture is extracted. The integrated reference model of adaptive software architecture (ASA-IRM) is proposed based on multiview modelling theory. The rules of model mapping and evolution in ASA-IRM are discussed. The algorithm of model mapping consistency is proposed based on metamodelling and graph transformation theories. Finally, an integrated support environment for ASA-IRM is developed. The adaptive software architecture can be used in the development of complex software system, especially in open, networked environment.
Key words: adaptive software architecture; multiview modelling; model mapping

摘 要: 研究并提出一类通用的适应性软件体系结构风格.通过对适应性软件体系结构的基本特性分析,抽取适应性软件体系结构的通用框架,建立基于多视图建模理论的集成化适应性软件体系结构参考模型,给出多视图模型的演化与映射规律.基于元建模和图转换理论,提出模型映射一致性算法.最后,建立了适应性软件体系结构支撑环境.适应性软件体系结构适用于复杂软件系统,特别是网络环境下大型、开放式软件系统的开发和实施.
关键词: 适应性软件体系结构;多视图建模;模型映射
中图法分类号: TP311 文献标识码: A

由于软件规模的不断增大和软件复杂性的不断提高,软件系统的体系结构设计显得越来越重要.在软件体系结构研究领域中,软件体系结构风格的研究与应用是一个重要的领域.1992 年,Perry 和 Wolf 提出了软件体系结构风格的概念[1],将那些公认的、被多次成功使用的软件系统结构称为软件体系结构风格.

Metayer 给出了一个较为直观的定义[2]:软件体系结构风格或者说是软件体系结构惯用模式(idiomatic paradigm),是一类具有许多共同特征的软件体系结构的共同描述.软件体系结构可以看作是某种软件体系结构风格的具体实例.

* Supported by the National Natural Science Foundation of China under Grant No.60504030 (国家自然科学基金)
Received 2006-01-09; Accepted 2006-03-13

© 1994-2008 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net
© 中国科学院软件研究所 http://www.jos.org.cn

体系结构风格的形成是多年研究和工程实践的经验积累结果.一个设计良好的、通用的风格往往是这个工程技术领域成熟的标志.经过多年的发展,已经产生许多种软件体系结构风格.

Mary 和 David 曾对已有的体系结构风格进行了分类[3]:数据流系统,包括批处理风格、管道-过滤器风格;调用/返回系统,包括主程序/子程序风格、面向对象风格、层次结构风格;独立构件,包括进程通信风格、基于事件的隐式调用风格;虚拟机,包括解释器风格、基于规则的风格;以数据为中心的系统,包括数据库、超文本系统、黑板风格.除了这些分类外,另外还有:特定领域体系结构风格,如过程控制、模拟器;特定结构体系结构风格,如分布式处理、状态转移系统;不同体系结构集成的异构体系结构风格等.

近年来,随着网络和许多新兴软件技术,如 AGENT、P2P、网格计算、普适计算、移动计算、网构软件等的发展,对软件系统体系结构提出了许多新的要求,如体系结构的适应性、可靠性、安全性、扩展性、复用性、可用性等[4].Perry 认为,目前软件体系结构领域最为重要的 3 个研究方向是:体系结构风格、体系结构连接件和动态体系结构[5],明确指出了体系结构风格以及动态体系结构研究的重要性.这里,Perry 所说的动态性与本文中所说的适应性具有相同的内涵.即指当软件生存环境发生变化时,软件实体结构与行为可以随之改变并满足新环境需要的特性.文献[6]对软件体系结构的动态性和适应性的概念进行了说明,认为软件体系结构的动态性是实现手段,而软件体系结构的适应性是一种实现目标.在本文中,统一使用适应性来表示软件系统随环境变化的能力.具有适应性的软件体系结构就是适应性软件体系结构.

目前,具有代表性的适应性软件体系结构风格包括基于多代理的软件体系结构[7,8]、开放网格服务体系结构[9,10]、网构软件体系结构[11,12]、面向服务的体系结构[13,14].以上体系结构都是在特定领域中产生的特定风格的体系结构.这些体系结构大多依赖于特定计算环境和实现技术.对于通用的适应性软件体系结构风格的研究,则需要从已有风格中抽取和扩展.从适应性软件的本质出发,探讨适应性的产生机理,研究适应性软件体系结构的共性特征和通用理论与方法.

本文通过对适应性软件体系结构的基本特性分析,提出适应性软件体系结构的通用框架,进一步抽取出适应性软件体系结构参考模型(integrated reference model for adaptive software architecture,简称 ASA-IRM).该参考模型独立于具体的计算环境,具有普遍适用性.文章对该参考模型的组成以及软件生命周期阶段中模型间的映射关系和模型演化过程进行了深入的研究,并建立了相应的集成化开发环境,在工具上为基于适应性体系结构的软件开发、部署与运行维护提供支持.文章最后以一类复杂企业应用系统——企业资源管理系统的设计、开发与实施运行为例,说明了适应性软件体系结构风格的应用方法与过程.

1 适应性软件体系结构通用框架

1.1 适应性软件体系结构的本质特征

软件体系结构是软件系统中各组成元素之间相互协调与处理的形式说明.软件体系结构是一个复杂软件系统的高层结构,它高度抽象,超越了算法和数据结构,基本着眼点是系统结构和需求与实现之间的交互,是一个用于理解系统级目标的框架结构.软件体系结构可以用 3 个基本要素来描述:构件、连接件和配置[3].无论是哪一种软件体系结构,都可以将它拆分为这 3 方面来进行研究.决定软件体系结构风格的正是所拥有的组成要素的特性.适应性软件体系结构同样可以用这 3 个基本要素来描述,只不过适应性软件体系结构定义了一组特定的、具有适应性的构件、连接件以及它们之间应该如何连接的配置约束.其适应性分别来自于构件适应性、连接件适应性和配置适应性[15].

1) 构件适应性:主要针对构件组装过程中对环境变化的适应性.可分为两种情况:(l) 构件不具备满足新环境需求的结构和功能;(2) 构件不能与环境很好地集成.前者可以从构件本身入手解决,需要从功能/行为及结构两方面提供构件的适应性;后者还需要通过构件连接解决;

2) 连接件适应性:软件系统构件和结构的变化都会引起构件连接的演化.体系结构应在连接层具备一定适应性.连接件适应性体现为对连接件接口、语义、约束等的修改能力.连接件适应性实现了构件间交互的动态性;

© 1994-2008 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

© 中国科学院软件研究所 http://www.jos.org.cn

3) 配置适应性:是指软件系统的拓扑结构可以根据环境的变化而改变.分为静态适应性和动态适应性两种.静态适应性发生在软件系统设计阶段,在设计时根据需求的变化修改和调整系统的结构;动态适应性发生在系统运行阶段,根据实际的运行环境动态地建立构件间的连接关系,从而实现系统的动态拓扑结构.

一般来讲,软件系统的体系结构在设计期间并不能保证准确地反映和完成系统的功能、质量和性能要求.这就要求系统的体系结构可以根据需求、技术、环境、分布等因素的变化,在开发和运行期间进行修正和完善,即体系结构必须具有适应性.Oriezy 等人定义了软件体系结构适应性发生的 4 个阶段[16]:

1) 设计时:修改变化是在体系结构模型和相关代码被编译前做出的.由于只是一个抽象的体系结构模型被替代或修改,这时的适应性相对容易实现;

2) 预执行时:修改发生在执行之前,编译之后.由于这时应用程序并未执行,做出修改时可以不考虑应用程序的状态,但需要考虑系统的体系结构,并且需要系统具有添加和删除构件的机制;

3) 受约束的运行时:修改只发生在某些特定约束满足时,基于程序状态的约束允许.当应用系统发生改变时处于“安全”的状态;

4) 运行时:系统的体系结构在运行时不能满足要求时发生的改变,包括添加构件、删除构件、升级替换构件、改变体系结构的拓扑结构等.

适应性软件体系结构的研究目的就是实现和提高软件系统体系结构在这 4 个阶段的适应性.不同的适应性软件体系结构可以采用不同的方法和技术来实现不同层次和类型的适应性.如 CORBA 采用中间件和标准的接口定义语言实现构件的完整封装,并可以根据需要进行构件的删减;MAS 系统通过 AGENT 本身的适应性和动态性实现系统结构的动态性;开放式网格服务体系结构基于标准的接口和协议,通过对网格服务的动态管理来实现适应性;P2P 通过消除客户端和服务器的界限,增强节点的自主性和协作性来提高系统的适应性.

虽然软件体系结构的适应性可以通过不同的方法和技术来实现,但这些不同的适应性体系结构具有一些相同的特性.正是由于这些相同特性的存在,才使我们从中抽取出一个通用的适应性软件体系结构风格成为了可能.这些特性包括:

1) 松散耦合:将构件接口作为与构件实现分离的实体而存在.能够在不影响使用者的情况下进行构件修改;

2) 可重配:能够支持系统体系结构的构件和连接件在运行时的动态变化.动态变化包括构件的增加、删除以及检查已添加的构件与连接件间的一致性关系;

3) 层次化:通过系统结构的分层和功能的正交分解,使系统的变动可以分解到不同的层次和构件.同时,也使局部的变动不会影响到系统的整体运行;

4) 可管理性:体系结构的变动必须满足一定的约束,需要一种根据体系结构修改保持系统整体性的机制,包括对变动的方式和范围的控制.

1.2 适应性软件体系结构通用框架

Chen 和 David 等人曾提出了一种 3 层的适应性软件体系结构框架[10]:顶层是任务层,负责系统任务和需求管理,决定系统执行的操作,并设置系统性能目标和资源约束;任务层的信息将传达给中间层,即模型层,用以指导系统达到更好的适应性;模型层包含一个由系统中交互的构件和连接件构成的全局体系结构模型及一个体系结构管理器,是系统适应性产生的核心;最下层是运行层,包括系统的具体实现代码以及它所处的运行环境.该结构虽然给出了适应性体系结构的层次,但并未显式地定义出适应性体系结构的基本组成要素及要素间的关系.基于前面对适应性体系结构特性的分析,我们对该层次模型进行了扩展,提出了一个通用的适应性软件体系结构框架,可以满足体系结构在可重配、松耦合、层次化及可管理性方面的需求.

该体系结构由用户层、控制层、服务层和基础层 4 个层次构成(如图 1 所示).

(1) 用户层:用来描述系统需求及任务,可以采用 UML、工作流等多种方法来描述系统需求.用户层可以请求系统提供服务,通过配置管理器,一个需求可以对应一个或多个服务.这种映射可以显著提高系统的适应性.当用户需求发生变化时,只要简单地添加或修改需求和服务之间的映射关系,就可以动态地实现适应性.而这种

© 1994-2008 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

© 中国科学院软件研究所 http://www.jos.org.cn

适应性的实现则需要其他几个层次的支持.

(2) 控制层:由配置管理器和服务管理器组成.配置管理器中维护了系统内部和外部环境的信息,用于支持系统体系结构层次上的配置.配置管理器由感知器和控制器组成.感知器接收环境的变化,进行变化辨识,并将变化信息传递给控制器;控制器根据变化作出相应的调整决策,并向服务管理器发出控制消息.服务管理器维护了一个服务目录,服务目录中注册了所有的服务,管理器提供了服务的动态注册、加载、卸载、服务的导出、服务的预定以及服务的安全管理等功能.服务管理器实际上实现系统计算结构的控制.服务管理器根据控制器的指令动态调整计算结构.

(3) 服务层:服务层由许多构件构成,提供各种功能服务.在这里,构件均被封装为标准的服务形式.服务请求者不必关心构件的具体细节,从而透明地获得所需服务.服务可以动态加载、卸载和导出,可以相互通信.

(4) 基础层:负责构件间的通信和协作.包括消息管理、交互管理和通信语言.消息管理提供构件间消息传输的机制,主要提供消息的接收、发送和过滤等;交互管理提供构件间的基本交互协议;通信语言为构件之间的消息传递提供规范的语义定义和格式,包括一整套消息类型,每种消息类型都有基于形式化的语义描述,表示某种特定的意思.

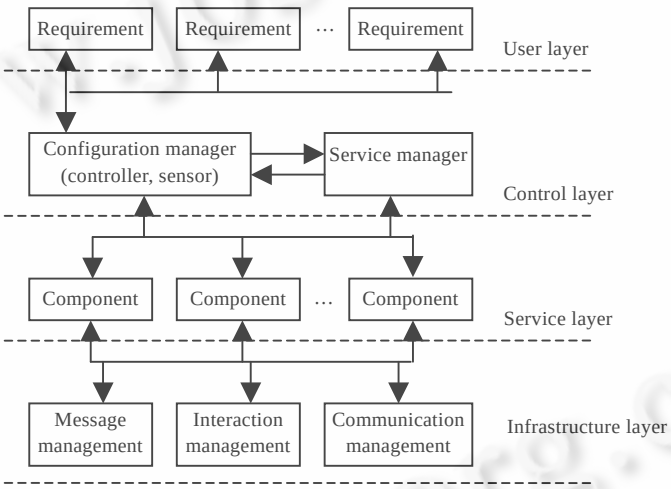

Fig.1 Generic framework for adaptive software architecture

图 1 适应性软件体系结构通用框架

从概念上讲,适应性体系结构有 3 个主要的抽象级别:需求、服务和操作.适应性软件体系结构的目的是通过这种高层次的抽象,将环境的变化和动态性映射到软件的具体实现.具体来说,在用户层,用户可以通过工作流等工具对需求进行动态建模,并通过配置管理器实现需求到服务之间的动态映射,这种映射关系可以在系统运行前或运行过程中进行动态修改.当一个服务失效时,可以指定另外一个可用服务替代.与 CORBA,AGENT,SOA 等目前具有适应性的体系结构相比,本文将体系结构的计算部分和控制部分分离,并显式地进行控制部分的实现,即控制部分作为一个软件实体存在于系统中.CORBA 等体系结构只在构件和构件连接层次上实现一定程度的适应性,即只能实现局部的、计算结构的适应性,缺乏在整体上、更高层次上的适应能力.控制结构与计算结构的分离使得系统可以在更高层次上实现适应性,可以根据环境变化动态调整系统的计算结构,为体系结构整体上的适应性调整提供支持.

2 适应性软件体系结构多视图模型

研究软件体系结构的首要问题是如何表示软件体系结构,即如何对软件体系结构建模.比较著名的软件体系结构模型有 Kruchten 提出的“4+1”视图模型、ISO 的开放分布式处理参考模型——RM-ODP(reference model

© 1994-2008 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net
© 中国科学院软件研究所 http://www.jos.org.cn

of opening distributed processing)和 Zachman 框架等.

Kruchten 的“4+1”视图模型从 5 个不同的视图——逻辑视图、过程视图、物理视图、模块视图和场景视图来描述软件体系结构[17].逻辑视图(也称概念视图)主要支持对系统功能方面需求的抽象描述,即系统最终将提供给用户什么服务;开发视图(也称模块视图)主要侧重于系统之间的组织;过程视图主要侧重于描述系统的动态行为,即系统运行时的特性.它着重解决系统的并发性、分布性、容错性等;物理视图主要描述如何把系统软件映射到硬件上,通常要考虑系统的性能、规模、容错等;场景视图(部署视图)与上述 4 个视图相重叠,希望综合它们的主要内容,作为开发人员辨别要素和验证设计方案的辅助工具.

RM-ODP 是一个标准的分布式处理系统开发框架[18].RM-ODP 提供了一个拥有 5 类视图的框架来描述系统:(1) 企业视图:描述信息系统需要做什么,这个视图中的模型用来捕获业务和管理需求,以及描述用来判断和指导信息系统设计的策略;(2) 信息视图:描述信息系统的信息结构、信息流和信息操作的约束;(3) 计算视图:描述信息系统的操作,即改变信息的过程的计算特性(算法);(4) 工程视图:描述信息系统需要的支持分布式处理的工程资源;(5) 技术视图:描述实现信息系统需要的实际元素.通过这 5 类视图,RM-ODP 完成了业务需求、体系结构和信息技术之间的解耦.

Zachman 框架是 1987 年 IBM 公司的 Zachman 提出来的面向信息系统设计和开发的体系结构[19].Zachman 框架经过了数次修正,现在对外宣称为 Zachman 企业体系结构[20],其出发点是一种基于体系结构的软件系统实施方法.最新的 Zachman 框架用一张二维表格来描述,在这个二维结构中,纵向定义了面向 6 类用户的 6 个视图用于描述和分析系统的不同抽象层次,对应“问题定义→软件需求→软件体系结构→软件设计→软件实现”这样的基于体系结构的信息系统实施过程.每一个抽象层次又都从 6 个侧面(横向)进行建模,包括数据、功能/过程、网络/通信、组织、时间和动机.体系结构还定义了每个单元格中模型描述的内容和基本建模元素,每个单元格允许采用不同的描述方式.Zachman 框架非常全面地定义了从业务需求开始的软件系统实施全过程,并且强调了模型的支持,对建模框架的设计有很强的指导意义.

但是,以上这些体系结构模型大多定位于对静态体系结构的描述,对于软件全生命周期过程中的体系结构演进,即体系结构的动态性和适应性描述不够,并且缺少对视图之间的映射机制和一致性维护机制的说明.而视图间的映射和一致性维护是适应性体系结构关注的重点,也是保证体系结构不同抽象层次(即需求-实现-运行维护)之间一致性映射的基础[21].

近年来,动态体系结构的研究力求解决软件需求模型到实现及运行模型的动态映射问题.在基于模型的软件适应性研究中,BANN 公司的基于动态建模的 DEM 方法初步实现了业务需求与软件系统的一致和同步.BANN/DEM 方法是:首先抽取出应用系统基本功能,并封装成标准对象放置在分布式对象运行环境中;然后在参考模型的基础上,建立业务工作流程,通过配置工具实现从系统设计模型到系统实施模型的映射,利用工作流管理工具灵活地组合成适应业务模型的软件系统,从而实现业务与信息系统之间的一致和同步.

基于以上软件体系结构模型的研究,本文提出了一个集成化的适应性软件体系结构参考模型(ASA-IRM).它综合了多视图体系结构建模、动态体系结构建模、面向领域体系结构建模的思想和方法,提供了一个覆盖软件开发全生命周期、具有多视图模型定义及一致性维护、面向通用和专用领域的集成化体系结构参考模型.

ASA-IRM 是由视图维、生命周期维和通用性层次维组成的三维结构.ASA-IRM 强调模型驱动的适应性,主要表现在对于不同生命周期阶段,ASA-IRM 根据建模目的和抽象层次的不同,组成的视图模型是不同的.也就是说,生命周期维和视图维是紧密关联的.同时,在软件开发的生命周期的 4 个阶段中(需求分析、体系结构规划、系统实施和运行维护),从前到后各个阶段的模型又可以相互转换或者映射.ASA-IRM 的通用性层次维分成通用层、部分通用层和领域专用层 3 个层次,使建模过程能够模块化或者模板化,体现重用的概念和方法,缩短建模周期,提高建模质量.通用层由一些建模构件、术语、规则和协议等组成,是组成模型的基本单元;部分通用层是一组适合于某一类行业/领域的部分通用模型(模板),即参考模型,拥有该类行业/领域中大部分实例的共有的典型结构;专用层则是指利用建模构件和参考模型建立的适用于一个特定领域/系统的模型.

根据不同的软件开发阶段和层次,在 ASA-IRM 中共抽取出 4 类视图和 12 种视图模型(如图 2 所示):

© 1994-2008 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

© 中国科学院软件研究所 http://www.jos.org.cn

1) 业务视图:软件设计首先需要确定未来要实现的业务,需要完成对当前业务的描述、分析、诊断和优化,确定核心业务过程.因此在这个阶段,需要完成的业务视图模型包括业务过程模型、业务功能模型和业务组织模型.这 3 个视图模型的核心是业务过程模型,在过程模型中通过业务活动的属性定义实现过程与组织、资源的集成.

2) 体系结构视图:体系结构视图是实现业务需求到具体软件实现的桥梁.按照体系结构的基本组成.体系结构视图模型包括构件模型、接口(连接件)模型和结构配制模型.由于构件模型和接口模型均是通过业务需求中过程模型映射而来,因此可以保证系统体系结构与业务需求的一致性.

3) 系统设计视图:系统设计视图模型包括表示系统静态结构信息的功能模型和信息模型,同时也有描述系统动态行为特征的交互模型.同时,通过部署模型来描述系统的具体实现.在这里,系统设计模型的生成采用的是基于体系结构的方法,通过体系结构的求精,实现软件体系结构的细化和实现.

4) 系统运行视图:系统运行维护阶段需要完成的工作是监控、管理系统的运行,并进行维护工作保持系统始终与变化中的企业战略、业务需求同步一致,是体系结构动态适应性产生的根源.ASA-IRM 在这个阶段定义了工作流模型和业务-组件匹配模型.工作流模型实际上是前面需求阶段过程模型的实例化,在系统部署模型定义的物理环境中执行.业务-组件匹配模型则是建立系统运行阶段时业务与组件的实际匹配关系,是一种动态模型,可以描述业务和组件的动态匹配关系.当业务和组件任何一方根据环境发生了变化,在维护阶段都可以通过对映射约束的检查来同步调整业务-组件的匹配关联,保证业务与系统的同步一致.

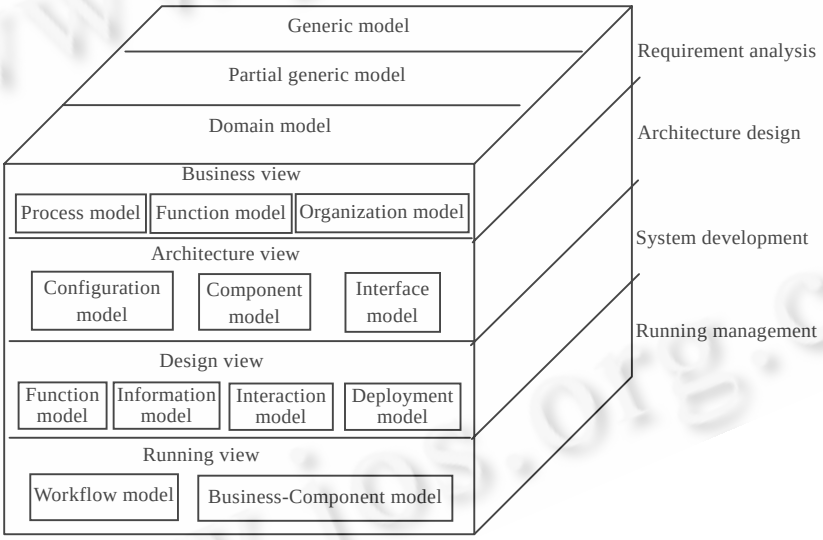

Fig.2 Integrated reference model for adaptive software architecture

图 2 集成化适应性软件体系结构参考模型

ASA-IRM 中各生命周期阶段中模型间的映射关系和模型演化过程如图 3 所示.

3 适应性软件体系结构中的模型映射一致性

在基于体系结构的软件开发中,需求与实现之间的一致性是通过体系结构中不同阶段视图模型间的转换和映射来实现的.模型转换和映射是基于体系结构软件开发中的核心问题.在体系结构的模型转换和映射方面,目前研究主要集中在模型的静态转换上.对于软件运行过程中模型的动态映射问题研究相对较少.映射可以看作是一类特殊的转换操作.模型静态转换完成后,在转换前后件模型之间就建立起了一定的映射关系.转换后件模型是前件模型基于此映射关系的一个映像.当作为映射原像的前件发生变更时,原像和映像之间的映射关系约束可能会被破坏,这就产生了映射一致性问题.保证需求模型和实现模型之间的映射一致性是适应性软件体

© 1994-2008 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net
© 中国科学院软件研究所 http://www.jos.org.cn

系结构中的关键问题.

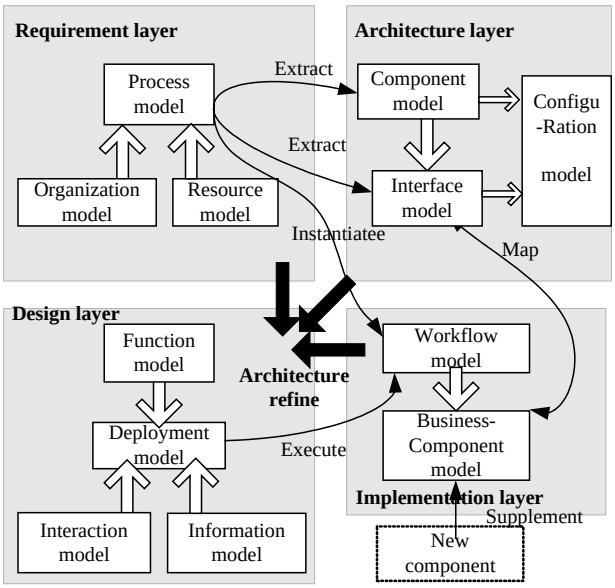

Fig.3 Model mapping & revolution in ASA-IRM

图 3 ASA-IRM 中模型映射与演化过程

这里,我们首先给出映射一致性的形式化定义,然后基于元建模和图转换理论,利用一致性恢复问题中的变换谓词逻辑方法,给出映射一致性处理算法.由于篇幅限制,这里只对该过程的主要步骤和所采用的理论方法进行简单说明.详细的理论介绍和推导请参见文献[21].

定义(映射一致性). 给定两种模型语言 A 和 B,映射关系 $R:A\rightarrow B$.若模型 $M_A\vdash A,M_B\vdash B,(M_A,M_B)\vdash R$,且有 M_A 上的一个建模操作 $\delta:M_A\rightarrow M_A^l$ 满足 A 上的建模一致,则当且仅当 $(M_A^l,M_B)\vdash R$,称为 δ 在 R 上满足映射一致.

模型映射一致性算法.

1. 若模型 $A_i,i=1,...,n$ 参与映射,首先构建一个 $\cup_i A_i$ 上的视图 V,根据需求在 V 中基于模型之间的关联关系定义映射约束;
2. 将 V 中的映射约束转换成为一阶谓词逻辑公式 P,并推演出用变化谓词表示的一致性恢复计算公式 $V^{+-}(V=\neg P)$;
3. 利用图转换规则描述已发生的模型变更操作,并根据图转换规则的语义定义转换成为对应的逻辑式 O;
4. 将 O 代入映射约束的一致性恢复计算公式 V^{+-},推演计算约束一致性恢复操作,将计算结果规范为前束析取范式 $P_1\vee P_2\vee...\vee P_n\equiv 1$;
5. 将 $P_i\equiv 1$ 转换成为图转换规则 p_i;
6. 在 p_i 中选取一个合适的恢复操作作用在模型中,完成模型的互动.

4 适应性软件体系结构支撑环境

一种风格的实现与推广应用离不开相应的工具和环境的支持.适应性软件体系结构支撑环境应具有支持体系结构的静态和动态调整的能力,并对软件开发提供全生命周期支持.比起通用的软件工程环境,适应性软件体系结构支撑环境更强调对需求与软件实现之间一致性的实现和维护.基于 ASA-IRM,本文建立了一个适应性软件体系结构支撑环境,提供了一套基于模型的适应性软件开发工具集(如图 4 所示),将软件需求到实现的一致性问题转化为体系结构中不同模型之间的映射一致性问题.通过对模型的一致性维护,实现不同阶段的软件适应性.在系统分析阶段,基于元模型技术,通过对业务需求模型和体系结构模型的一致性建模,实现需求和体系

© 1994-2008 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net
© 中国科学院软件研究所 http://www.jos.org.cn

结构的一致性映射;在系统设计阶段,通过对体系结构模型的动态求精,实现软件体系结构模型到系统功能模型的映射,保证当体系结构变化时系统设计模型的一致性变化;在系统运行时,通过工作流管理工具,实现业务模型和软件功能模型之间的动态映射,保证软件系统能够在运行时根据环境的变化动态调整体系结构和实现模型.整个环境可以划分为工具层、平台层和数据层 3 个层次.

(1) 工具层:为基于体系结构/模型的适应性软件系统开发的生命周期提供工具支持.其中,规划工具搜集和评价来自业务的需求,得到软件系统规划方案;建模与诊断工具用于支持ASA-IRM框架中定义的模型的建立与操作,并基于模型进行诊断分析.由于模型是适应性体系结构的核心,这个工具也是工具集中的核心工具;CASE工具负责支持信息系统的设计、开发直至代码生成工作;工作流管理工具负责软件系统的执行与维护,执行软件系统中各应用组件的导航和执行控制;运行监控工具实现软件运行状态的监控、分析,检测环境和软件本身的变化,并将变化反馈到工作流管理系统中.

(2) 平台层:由一些平台组件构成,这些组件为应用和辅助应用工具提供基础服务.模型访问组件完成模型数据的访问和操作,提供应用工具到数据存储的接口;模型操作引擎提供通用模型操作支持,完成模型操作步骤(尤其是模型转换和映射)的定义和自动执行,并维护模型操作的一致性;工作流引擎实现活动的流转、资源的调度和数据的传递,为工作流管理系统提供与模型实例和运行日志数据之间的接口;可视化引擎定义数据的可视化规则,实现可视化效果.

(3) 数据层:由若干存储工具集应用数据的库组成,包括用于业务及信息系统描述的模型库和相关文档库、用于软件系统配置的组件库、用于系统运行的业务实例库和日志数据库等.

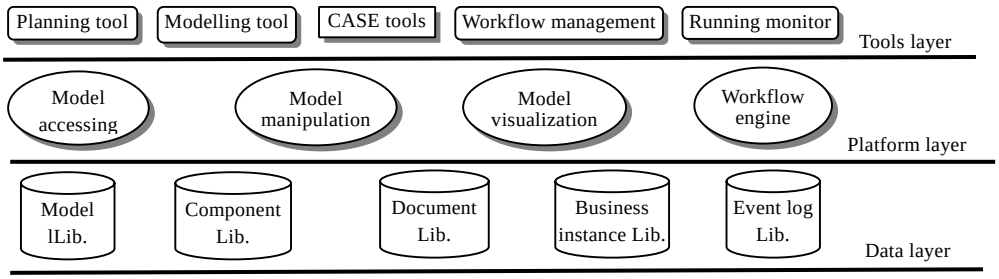

Fig.4 Support tool set for adaptive software architecture

图 4 适应性软件体系结构支撑工具集

5 应用实例

本文提出的适应性软件体系结构风格已经在复杂企业应用系统——企业资源管理系统(ERP)的设计开发中得到应用.针对现有 ERP 系统的开发模式在开发、实施、维护和使用方面的一些问题,如系统开发缺乏灵活性、系统实施与维护难度大、系统易用性较差以及软件系统不能适应迅速变化的企业业务流程等,采用本文所提出的适应性软件体系结构及支撑工具集,支持新一代适应性 ERP 系统的设计和开发[22].

基于适应性软件体系结构的 ERP 系统开发与实施方法同传统 ERP 系统开发方法相比,其本质差别是:它把体系结构从具体的软件实现中分离出来,并以体系结构/模型为核心来设计和构建 ERP 系统,并控制其运行.该方法通过对ERP业务流程的分析,抽取出原子级的企业业务活动,并基于这些活动进行ERP构件的设计与开发,实现构件与这些原子级企业活动的映射.在运行阶段,通过工作流执行服务按照所建立的业务模型进行业务的执行,并在业务-组件匹配模型的控制下实现企业业务和软件实现的全面集成.

利用适应性软件体系结构支撑工具集,可以设计和建立一个适应性 ERP 系统开发支撑环境,用来支持 ERP 系统的业务流程分析、业务构件抽取、业务系统构建和业务系统执行(如图 5 所示).整个环境可以为 ERP 系统的开发与实施在方法和工具上提供支持:在业务分析阶段,可以利用工作流建模与分析工具详细分析 ERP 的业务流程和相关的组织/人员情况,生成完整的 ERP 业务过程模型,明确 ERP 软件的实际需求,即谁在什么时间做什么事情、需要什么信息、生成什么信息;而在 ERP 组件设计中,可以依据业务分析阶段的业务过程模型,整理

© 1994-2008 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net
© 中国科学院软件研究所 http://www.jos.org.cn

出 ERP 业务的原子活动.针对这些原子活动,分析其需要由什么样的组件来实现,并确定每个组件的输入/输出接口,从而抽取出 ERP 系统的组件规划方案.至此,可以得出构成 ERP 系统的基本组成元素——原子活动和组件;在第 3 步中,就可以根据企业业务需求组合这些原子活动和组件,以建立一个可实际运行的 ERP 系统.组合过程可以使用工作流建模工具来完成.在这一步骤中,可以动态关联原子活动和组件,并定义各活动和组件间输入/输出关系;最后,所得到的 ERP 系统可以通过工作流引擎进行执行.引擎根据定义好的业务逻辑推动 ERP 系统的运转,负责流程的启动、导航、结束等操作,并实现对组件的调用、活动和组件间数据的交互以及用户与系统的交互.

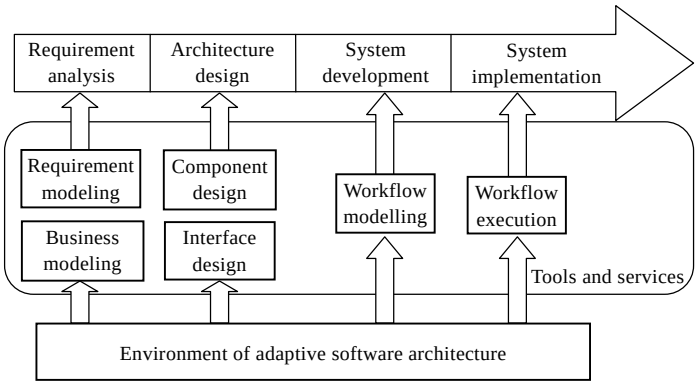

Fig.5 Adaptive software architecture based development and implementation of ERP

图 5 基于适应性体系结构的 ERP 系统开发与实施

与传统的基于功能的 ERP 系统开发与实施方式相比(见表 1),基于适应性软件体系结构的方法可以大幅度提高 ERP 系统的开发与实施效率.首先,基于适应性软件体系结构的方法,采用 Bottom-up 方式进行系统开发,与传统的基于功能分解的 Top-Down 开发方式相比,采用 Bottom-up 方式通过组件动态聚合形成系统,效率高;传统的 ERP 开发方式只能实现组件级别的重用,难以实现体系结构层次的重用,因此许多成功的设计经验难以积累.基于体系结构的开发可以实现系统层次的重用,快速将行业经验转化为解决方案,提高设计效率;传统 ERP 系统由于体系结构上不具适应性,当环境变化产生新的需求时,只能通过二次开发来满足需求,工作量大.而基于适应性软件体系结构的方法可以在不影响系统运行的情况下,通过动态改变体系结构来满足新的系统需求.

Table 1 Comparison between traditional ERP development and adaptive architecture based development

表 1 传统 ERP 开发方式与基于适应性体系结构开发方式的比较

Traditional ERP development	Adaptive architecture based ERP development
Using top-down development method	Using bottom-up development method
Reuse in component level	Reuse in architecture level
Updated by reprogramming	Updated by reconfiguration of architecture

6 结 论

软件系统必须适应千变万化的需求和环境的持续变化,并对这些变化作出快速和高效的反应.采用基于体系结构的软件开发方法可以提高软件系统的开发效率和质量,这是复杂软件系统必须采用的方法.实现软件系统的适应性,最好的方法就是从软件体系结构入手,从高层次,即系统体系结构层次上使系统具有适应性.适应性软件体系结构的研究是近几年来软件体系结构的一个研究热点.目前,对适应性软件体系结构的风格、描述语言、支撑工具等已经进行了一些研究,取得了一些成果,但距离形成成熟的、通用性的适应性软件体系结构风格还有很长一段路要走.本文在通用的适应性软件体系结构风格方面进行了一些研究工作,力求抽取出适应性体系结构的公有特性和通用参考模型.有了这些研究基础,下面的研究重点将是适应性体系结构的形式化分析、视图一致性维护及动态有效性判定,以及基于适应性体系结构的自动求精等方面的研究.

© 1994-2008 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

© 中国科学院软件研究所 http://www.jos.org.cn

致谢 在此,我们向对本文的工作给予支持和建议的同行,尤其是清华大学自动化系网络化制造实验室的老师和同学表示感谢.

References:

[1] Perry DE, Wolf AL. Foundations for the study of software architecture. ACM SIGSOFT Software Engineering Notes, 1992,17(4):44−52.

[2] Metayer DL. Describing software architecture styles using graph grammars. IEEE Trans. on Software Engineering, 1998,24(7): 521−533.

[3] Mary S, David G. Software Architecture: Perspectives on An Emerging Discipline. 2nd ed., Beijing: Science Press, 2004. 19−32.

[4] Zhang GY. Research on the attributes of Adaptive software architecture [MS. Thesis]. Taiyuan: Taiyuan University of Technology., 2003 (in Chinese with English abstract).

[5] Perry DE. An overview of the state of the art in software architecture. In: Proc. of the 19th Int'l Conf. on Software Engineering. New York: ACM Press, 1997. 590−591.

[6] Yu XL. Analysis of Software Architecture and Its Examples. Beijing: Science Press, 2004. 78−108 (in Chinese).

[7] Sun ZY. Research on architecture & modeling of multi-agent system [Ph.D. Thesis]. Hefei: Hefei University of Technology, 2004 (in Chinese with English abstract).

[8] The foundation for intelligent physical agents. FIPA Abstract Architecture Specification, 2002. http://www.fipa.org/specs/fipa00001/SC00001L.pdf

[9] Foster I, Kesselman C. The Grid: Blueprint for a Future Computing Infrastructure. 2nd ed., Beijing: China Machine Press, 2005. 215−259.

[10] Cheng SW, David G, Bradley S. Software architecture based adaptive for pervasive systems. 2005. http://www-2.cs.cmu.edu/afs/cs/project/able/ftp/arch-arcs02/arcs-final.pdf

[11] Yang FQ, Mei H, Lu J, Jin Z. Some discussion on the development of software technology. Acta Electronica Sinica, 2002,30(12A): 1901−1906 (in Chinese with English abstract).

[12] Li CX. Research of integrated transportation platform based on Internet ware theory [MS. Thesis]. Dalian: Dalian University of Technology, 2005 (in Chinese with English abstract).

[13] Hu CM, Huai JP, Sun HL. Web service-based grid architecture and its supporting environment. Journal of Software, 2004, 15(7):1064−1073 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/15/1064.htm

[14] Lu ZJ. The research of design method of the enterprise's SOA system. Industrial Engineering Journal, 2004,7(6):14−19 (in Chinese with English abstract).

[15] Li G, Jin MZ. Research on adaptive software architecture [Ph.D. Thesis]. Beijing: Beijing University of Aeronautics and Astronautics, 2002 (in Chinese with English abstract).

[16] Oreizy P, Gorlick MM, Taylor RN, Heimhigner D, Johnson G, Medvidovic N, Quilici A, Rosenblum DS, Wolf AL. An architecture-based approach to self-adaptive software. IEEE Intelligent Systems, 1999,14(3):54−62.

[17] Kruchten P. The 4+1 view model of architecture. IEEE Software, 1995,12(6):42−50.

[18] Steen MWA, Derrick J. ODP enterprise viewpoint specification. Computer Standards & Interfaces, 2000,22(3):165−189.

[19] Zachman JA. A framework for information systems architecture. IBM Systems Journal, 1987,26(3):276−292.

[20] The Zachman Institute for Framework Architecture. The zachman framework for enterprise architecture. 2004. http://www.zifa.com/framework.pdf

[21] Zhao Y. Research on the model-driven approach and consistency problem for enterprise informatization engineering [Ph.D. Thesis]. Beijing: Tsinghua University, 2005 (in Chinese with English abstract).

[22] Huang SX, Fan YS. Workflow based development and implementation of ERP. Computer Integrated Manufacturing System, 2004, 10(2):139−143 (in Chinese with English abstract).

附中文参考文献:

[4] 张国有.适应性软件体系结构的属性研究[硕士学位论文].太原:太原理工大学,2003.

© 1994-2008 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

© 中国科学院软件研究所 http://www.jos.org.cn

[6] 余雪丽.软件体系结构及实例分析.北京:科学出版社,2004.78−108.
[7] 孙志勇.多 Agent 系统体系结构及建模方法研究[博士学位论文].合肥:合肥工业大学,2004.
[11] 杨芙清,梅宏,吕建,金芝.浅论软件技术发展.电子学报,2002,30(12A):1901−1906.
[12] 李程旭.基于网构软件理论的交通综合平台研究[硕士学位论文].大连:大连理工大学,2005.
[13] 胡春明,怀进鹏,孙海龙.基于 Web 服务的网格体系结构及其支撑环境研究.软件学报,2004,5(7):1064−1073. http://www.jos.org.cn/1000-9825/15/1064.htm
[14] 卢致杰.SOA 体系设计方法研究.工业工程,2004,7(6):14−19.
[15] 李刚,金茂忠.适应性软件体系结构研究[博士学位论文].北京:北京航空航天大学,2002.
[21] 赵彧.企业信息化工程中的模型驱动方法及一致性问题研究[博士学位论文].北京:清华大学,2005.
[22] 黄双喜,范玉顺.基于工作流的 ERP 系统开发与实施.计算机集成制造系统,2004,10(2):139−143.

黄双喜(1972－),男,四川射洪人,博士,助理研究员,主要研究领域为软件体系结构,企业建模,工作流.

赵彧(1981－),男,博士,主要研究领域为企业建模,企业信息化工程.

范玉顺(1962－),男,教授,博士生导师,主要研究领域为企业建模,工作流.

© 1994-2008 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net
© 中国科学院软件研究所 http://www.jos.org.cn