第33卷 第15期 计 算 机 工 程 2007年8月
Vol.33 No.15 Computer Engineering August 2007
·软件技术与数据库· 文章编号：1000—3428(2007)15—0083—03 文献标识码：A 中图分类号：TP311

管理领域面向方面软件体系结构及软件过程

王 斌，盛津芳，桂卫华
(中南大学信息科学与工程学院软件系，长沙 410083)

摘 要：软件体系结构设计是构建大型管理领域系统的关键步骤，同时面向方面的软件开发已成为解决软件复杂性的有效方法。该文基于面向方面软件开发方法提出了管理领域系统建设的“4+1”关注点视图，针对“4+1”关注点视图提出了面向管理领域、基于服务实现的面向方面软件体系结构。基于该软件体系结构开发的管理业务支持平台及其支持下的软件开发过程可以缩短软件开发周期，改善软件一致性和可维护性，使软件具有更好的演化能力。

关键词：关注点视图；面向方面；领域软件体系结构；管理业务支持平台；软件演化

Aspect-oriented Software Architecture and Software Process for Management Domain

WANG Bin, SHENG Jin-fang, GUI Wei-hua
(Dept. of Software, College of Information Science and Engineering, Central South University, Changsha 410083)

【Abstract】 Software architecture design becomes a key step of building large, complex management domain specific system, such as human resource management system(HR), office automation system(OA) and customer relationship management system (CRM), while aspect-oriented software development(AOSD) is emerged as a promising approach to solve software complexity. This paper proposes a “4+1” concern view of management domain specific system based on AOSD method. According to the “4+1” concern view, it provides an aspect-oriented software architecture, which is to be implemented by services. The management business support platform constructed based on the proposed architecture and the corresponding development process can be applied to effectively shorten software development cycle, improve software consistency and maintainability and provide better software evolution capability.

【Key words】 concern view; aspect-oriented; domain-specific software architecture; management business development support platform(MBDSP); software evolution

1 概述

分离关注点(separation of concerns)是软件工程降低复杂性、改善软件可重用性(reusability)以及增强演化能力(ability of evolution)的重要方法[1,2]。软件的开发过程在本质上是寻找需求空间到实现方法空间的映射，而传统的软件开发强制将所有的需求依附于主流的、占统治地位的需求——业务功能需求，如图1所示的软件体系结构描述[3]。

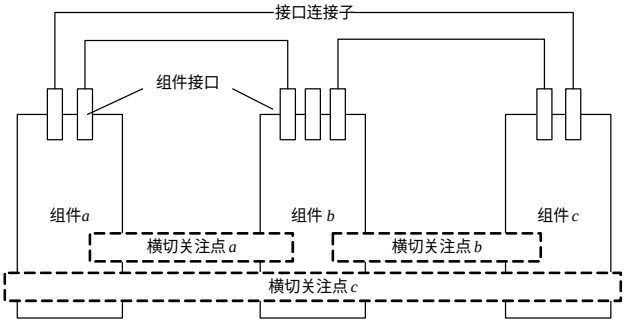

图1 关注点横切的软件体系结构

图1中组件*a*、*b*、*c*分别代表了3个不同功能的业务组件，而横切关注点*a*、*b*和*c*代表了非功能需求的关注点，如：日志，认证授权，事务等操作[4~6]。由于系统是按照业务功能需求去开发实现，因此只要是在组件*a*、*b*、*c*中需要调用横切关注点(非功能组件)的时候，就会出现业务功能组件被非功能组件横切的现象。在关注点横切的软件体系结构中业务功能组件中会出现大量对非功能的调用，导致代码的分散和纠缠，为系统扩展和系统维护带来了困难。

面向方面软件开发方法可以有效地克服关注点横切带给软件系统的上述缺陷[7~9]。目前面向方面的软件开发在语言级上得到较强的支持，如：AspectJ, Hyper/J和Aspect C++[10~12]，但是由于缺少面向方面的集成开发平台，面向方面语言很难支持大型软件的开发过程。面向方面支持关注点的分离，而关注点是系统中对需求的描述。由于在具体的应用领域中用户的非功能需求关注点相对固定，因此为具体的应用领域建立面向方面的业务支持平台是可行的。

2 “4+1”关注点视图

目前国内的大型企业的运营规模越来越大，能够提供的业务也由简单到复杂。这些企业在业务运营上已经逐渐建立了自己的IT架构，但是需要进一步提升管理的信息化，从而为业务支持系统提供一个统一的接入和展示平台。这些企业的管理组织和管理支持平台具有如下特点：

基金项目：中南大学博士后科学基金资助项目；湖南省自然科学基金资助项目(05JJ40312)
作者简介：王 斌(1973－)，男，博士，主研方向：构件组装；盛津芳，讲师、博士研究生；桂卫华，教授、博士生导师
收稿日期：2006-08-27 **E-mail：**wb_csut@mail.csu.edu.cn

(1)拥有大量的分公司甚至是海外公司，组织架构呈现分层、多级的结构。

(2)企业中存在大量已经建设好的各种管理系统，如 HR、CRM 等，使系统的最终用户发生重叠。

(3)管理领域系统能够支持水平管理和垂直管理。水平管理使企业中同级部门相互协同，垂直管理使企业中具有同等性质、不同级别的部门相互协同。

(4)企业中的管理流程能够根据业务发展的需要灵活定制，这样就需要企业管理信息系统具备良好的可扩展性和演化能力。

从不同角度去分解管理领域系统这些非功能需求，可以发现导致系统演化的关注点。根据“开-闭原则”将管理领域系统中能够发生演化的关注点进行封装，得到图 2 所示的管理领域系统“4+1”关注点视图。

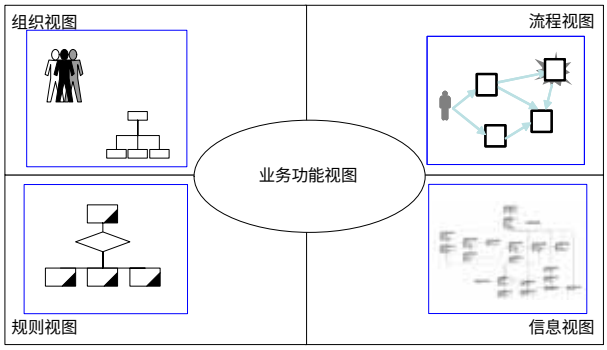

图 2　管理领域系统“4+1”关注点视图

在管理领域系统“4+1”视图中，业务功能视图处于关注点视图的核心，该视图封装了管理领域系统中的业务功能，其他视图中提供的功能作为对业务功能的支持。

组织视图封装了企业的组织结构和人员信息。由于任何一个管理领域系统都需要操作权限模块，因此组织视图是管理领域系统中必须考虑的基础设施之一。组织视图独立封装了管理系统用户的变化情况，可以有效地解决目前大型企业中存在的不同管理信息系统操作人员重叠的问题，实现单点登录。

流程视图支持业务流程定制、流程统计和流程管理。流程视图封装了企业管理需求的变化。用户可以根据管理需求的变动调整或者新建业务流程，适应管理需求的变化。

规则视图封装了业务功能的相关约束、系统访问控制和授权、事务等可配置需求的支持。规则视图使这些横切在管理系统业务功能模块中的关注点独立出来，避免了这些非功能需求在实现过程中将代码分散在业务功能组件中，使业务功能组件具有更好的模块性和可维护性。

信息视图是企业管理流程中不同环节需要传递和展示的数据封装。信息视图不是支持管理系统的底层数据库，而是依赖于数据库表的数据集。业务流程的各个环节从各自的数据集提取需要的数据。

3　管理领域面向方面的软件体系结构

根据管理领域系统“4+1”关注点视图中关注点分离的原则，下面给出管理领域面向方面软件体系结构的基于服务的描述，具体见图 3。

基于服务的管理领域面向方面软件体系结构描述如下：

(1)业务功能组件(business function components)和接口连接子(interface connectors)。业务功能组件只完成管理系统需要的具体业务功能。不同业务功能组件间的功能调用通过“接口连接子”完成，对接口连接子的描述可以遵循标准的接口定义语言来定义。

(2)切点(point cut)。业务功能组件嵌入非功能执行代码的入口。对于一些需要横切业务功能组件的关注点用服务来描述，例如：业务功能组件 *a* 需要使用组织结构管理服务的一个具体功能，那么，利用面向方面的技术，可以将实现该功能的代码(源代码或执行代码)编织(weave)入业务功能组件 *a* 的“切点”中。

(3)方面(aspect)。业务功能组件中的“切点”和服务代码的执行点二者的连接线称为方面。方面中需要定义如下内容：

1)向目标模块切入代码的条件；

2)引入的服务模块或库；

3)切入代码的执行逻辑(在方面内部定义或在引入服务模块内定义)。

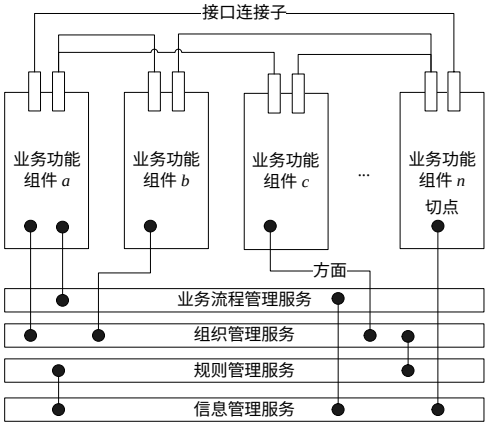

图 3　基于服务的管理领域面向方面软件体系结构

4　管理业务支持平台及软件过程

根据图 3 所示的基于服务的管理领域面向方面的软件体系结构，可为管理领域系统提供通用的支持平台——管理业务开发支持平台(management business development support platform, MBDSP)。MBDSP 是将管理领域中非功能需求设计为支持业务功能需求实现的框架。图 4 所示为管理业务支持平台框架。

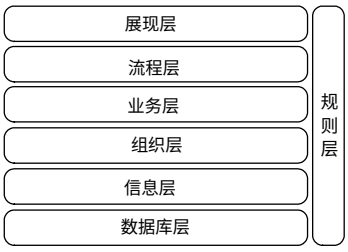

图 4　管理业务开发支持平台的框架

MBDSP 中的信息层(layer of information)、组织层(layer of organization)、流程层(layer of process)和规则层(layer of rules)，分别对应“4+1”关注点视图中的信息视图、组织视图、流程视图和规则视图。在 MBDSP 中信息层、组织层、流程层和规则层作为提供给业务层的服务被集成到框架中的。在 MBDSP 中只有业务层(layer of business)中的组件需要独立开发，而非功能需求支持层中的功能是配置上去的，所以利用 MBDSP 开发管理领域系统的开发方式和过程与传统的开发方式及过程不同。下面定义针对该平台的软件开发过

程——管理业务模型驱动的软件过程(MBMdSP)。

MBMdSP 在获取用户需求的基础上，初始化管理业务支持平台 MBDSP 的开发环境。针对具体企业进行企业组织建模、企业信息建模、企业规则建模、企业业务建模和企业流程建模，并利用 MBDSP 将上述模型生成程序代码、数据库表和数据集，然后进行编译和测试。对不满足用户需求的部分进行迭代求精。图 5 为管理业务模型驱动的软件过程。

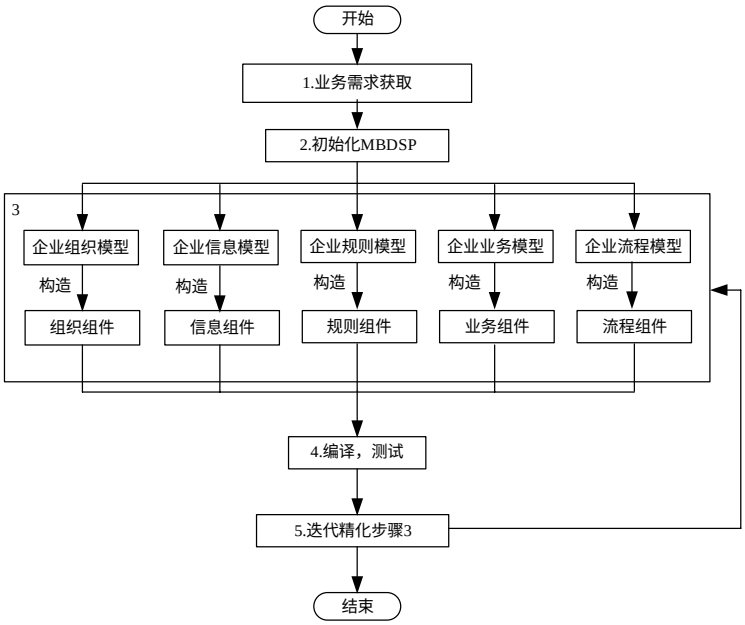

图 5　管理业务模型驱动的软件过程

管理业务模型驱动的软件过程具有如下优点：

(1)缩短开发周期。通过管理业务支持平台生成代码，而不是由开发人员手写代码，这样开发人员不需要消耗过多的时间去编写模板代码，从而缩短软件开发周期。

(2)发挥体系结构的优势。通过管理业务支持平台开发系统，开发人员首先要考虑的是管理领域系统中的高层实体，这样就会迫使开发人员去考虑系统的体系结构及对象模型，而不会直接去开发代码。开发人员通过使用管理业务支持平台发挥了基于软件体系结构开发的优势。

(3)改进代码的一致性和可维护性。开发人员在系统的开发过程中保持应用体系结构以及软件代码的一致性一直是软件工程领域中的难题。管理业务支持平台的底层实现机制——面向方面的软件开发方法保证生成的目标代码具有一致的算法，而且代码是由标准的、具有一致设计模式的平台生成的，为系统的维护带来极大的便利。

(4)提升软件的演化能力。管理业务模型驱动的软件过程中企业组织模型、企业信息模型、企业规则模型和企业流程模型封装了管理领域系统绝大部分非功能需求来支持业务模型的变化。企业组织模型、企业信息模型、企业规则模型和企业流程模型可以根据需求的变化调整和配置，快速生成代码以适应新的业务功能需求。

5　结论

面向方面的软件开发为软件工程中分离关注点、更好地模块化提供了技术基础。利用面向方面的思想以及成熟的、基于服务的实现机制，可为领域相关的软件系统提供高效的开发支持平台。本文分析了传统的关注点横切软件体系结构存在的缺点，说明了体系结构支持关注点分离的重要性。同时分析了企业管理领域系统的一些关键的非功能需求，提出了面向管理领域系统的“4+1”关注点视图。根据“4+1”关注点视图，给出了针对管理领域、基于服务实现的面向方面软件体系结构。基于该软件体系结构开发的管理业务支持平台及软件开发过程缩短了软件开发周期，改善了软件代码一致性和可维护性，使软件具有更好的演化能力。

参考文献

1 Tarr P, Ossher H, Harrison W, et al. N Degrees of Separation: Multi-dimensional Separation of Concerns[C]//Proc. of International Conference on Software Engineering, Las Angeles. ACM Press, 1999-05: 107-119.

2 Miller S K. Aspect-oriented Programming Takes Aim at Software Complexity[J]. Computer, 2001, 34(4): 18-21.

3 Navasa A, Pérez M A, Murillo J M, et al. Aspect Oriented Software Architecture: a Structural Perspective[C]//Proc. of International Conference on Aspect-oriented Software Development. 2002.

4 Filman R E, Friedman D P. Aspect-oriented Progr-amming is Quantification and Obliviousness[C]// Proc. of Workshop on Advanced Separation of Concerns. 2000.

5 Da L, Cooper K. Modeling and Analysis of Non-functional Requirements as Aspects in a UML Based Architecture Design[C]//Proc. of the 6th Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing. 2005: 178-183.

6 Solar G V, Bañuelos L G, Martini J L Z. Toward Aspect Oriented Services Coordination for Building Modern Information Systems[C]// Proceedings of the 5th Mexican International Conference on Computer Science. 2004: 353-360.

7 Noro M, Kumazaki A. On Aspect-oriented Software Architecture: It Implies a Process as Well as a Product[C]//Proc. of the 9th Software Engineering Conference. 2002: 276-285.

8 Elrad T, Filman R E, Bader A. Aspect-oriented Programming: Introduction[J]. Communications of the ACM, 2001, 44(10): 29-33.

9 Czarnecki K, Eisenecker U W. Generative Programming: Methods, Tools, and Applications[M]. Boston: Addison Wesley, 2000.

10 Chavez C, Garcia A F, Lucena C J P. Some Insights on the Use of AspectJ and Hyper/J[C]//Proc. of Workshop on Aspect Oriented Programming and Separation of Concerns, Lancaster, UK. 2001-08: 23-24.

11 Hanenberg S, Unland R. Using and Reusing Aspects in AspectJ[C]// Proc. of Workshop on Advanced Separation of Concerns in Object-oriented Systems. 2001.

12 Spinczyk O, Gal A, Preikschat W S. AspectC++: An Aspect-oriented Extension to the C++ Programming Language[C]//Proceedings of the 40th International Confernece on Technology of Object-oriented Languages and System, Sydney, Australia. 2002: 53-60.