一种适应 WEB 环境的混合体系结构风格①

汪保杰 王如龙 (湖南大学 软件学院 湖南 长沙 410082)

摘 要： 软件体系结构风格是软件体系结构领域的一个重要研究方向，它约束了系统生命周期中的各项活动，提高了系统的复用级别。在现有风格的基础上提出了更加适应 WEB 环境的混合体系结构风格—RROS(Rich Representational Orthogonal Service)，此风格可作为 WEB 应用系统开发的参考风格，用来指导 WEB 应用系统的设计、定义和部署。通过在 CKM(客户知识管理)系统中的应用，证明了其实际应用价值。

关键词： 软件体系结构；软件体系结构风格；混合风格；客户知识管理

A Mixed Architecture Style Suited to Web Environment

WANG Bao-Jie, WANG Ru-Long

(Software School, Hunan University, Changsha 410082, China)

Abstract: Software architecture style is important in the software architecture area. It restrains the system's activities in life cycle and improves the system's reuse level. This paper proposes a mixed architecture style－RROS (Rich Representational Orthogonal Service) which is more suitable for WEB environment based on the existing style. This style of WEB application system can be used as reference for the development and to guide the design, definition and deployment of WEB application system. The paper demonstrates the value of its practical application to the CKM (Customer Knowledge Management) system.

Keywords: software architecture; software architecture style; mixed style; customer knowledge management

1 引言

随着信息系统的规模和复杂度不断扩大，单种软件体系结构风格已经不能满足系统开发约束性的要求，必须引入其他风格形成一种混合风格以便指导系统的开发。经过长期的大型软件设计与开发，人们总结了一系列风格，为系统级别的复用提供了可能，并带来了有效的框架复用和编码复用。

风格的研究与应用是一个重要的领域，M. Shaw 等人指出模式和风格是一个工程领域成熟的标志[1]。目前，在体系结构风格的总结以及形式化描述等方面有些研究成果，但在风格组合以及组合的指导方法上研究很少。在现有风格基础上，依据 WEB 应用系统设计规格约束提出了一种混合风格 RROS(Rich Representational Orthogonal Service)，并在 CKM (Customer Knowledge Management)系统中进行了初步应用。

2 RROS 体系结构风格定义

2.1 软件体系结构风格定义

软件体系结构风格是对一组相似系统在模式、语义特性和约束上的抽象，是描述某一特定应用领域中系统组织方式的惯用模式[2]。一种体系结构风格包括一个术语表和一组约束集合，术语表中包含一些构件和连接件类型，而约束集合指出系统如何将这些构件和连接件组合在一起。事实上，一种体系结构风格决定了一种体系结构框架。文献[3]给出如下定义：软件体系结构风格可由三元组 SAS={Components, Connectors, Design Criteria} 来表示，其中：

① 基金项目:国家科技支撑计划项目(项目编号 2006BAF01A13)
收稿时间:2009-07-12;收到修改稿时间:2010-03-05

(1) Components 是构件的集合，他提供了系统的基本功能和操作，构成了一个系统的基本框架。

(2) Connectors 是连接件的集合，表示构件之间的交互方式，定义构件之间的交互规则，如交互特性和交互的数据类型。

(3) Design Criteria 包括了构件和连接件的使用、选择以及约束限制等。

文献[4]对风格进行了分类总结。文献[5]对几种风格进行了提取研究，为风格的发现提供了指导方法。体系结构风格是基于不同的视角和层次抽象出来的，不同风格之间往往有交叉现象，比如：层次风格是一切复杂系统的基本构架方法。基于上述分析，本文对软件体系结构风格的定义如下：

定义. 软件体系结构风格是一组术语表和约束的集合，是在某一视图下的软件体系结构。SA 表示软件体系结构，AS 表示 SA 的风格。

AS= {SA| abstract views, terms, constraints}.

且 SA=instance(AS)，其中 abstract views 表示抽象视图，terms 表示术语表，constraints 表示约束集。

2.2 软件体系结构风格比较表

软件体系结构风格的形成是多年研究和工程实践的经验积累。体系结构风格决定着体系结构模型，决定着构件的映射和构件间的连接[6]。风格只描述系统的整体结构框架，不传达系统的细节，带有整体性、普遍性、一般性、抽象性的特点。

风格比较主要从风格所提供的约束对系统性能指标的支持粒度进行比较。有的研究者从比较矩阵的角度来支持对风格的选择[7]，并进行了适当的量化，但是比较矩阵还是依赖于经验值。本文没有进行量化，而是通过对风格的约束所导致的一些属性的比较分析来指导风格的选择，依赖于经验值。比较表主要从性能、可伸缩性、简单性、可进化性、可扩展性等几个指标来评价每种风格的导致属性。系统属性是相对的，引入一种风格、添加一种约束可能会增强也可能会减弱另一个系统属性[6]。风格在应用于不同领域的系统时，会有不同的效果，本文只讨论 WEB 模式下企业应用系统开发领域的体系结构设计。

表 1 展示了我们总结的风格比较表，其中减号表示消极影响，加号表示积极影响，加减号表示依赖于问题领域或问题场景。该表格并不是一个所有风格以及所有属性的清单，而是列出了对 WEB 构建有指导意义的属性和风格。

表 1　软件体系统结构风格比较表

系统属性 / 风格名称	性能	可见性	简单性	可靠性	可伸缩性	可进化性	可扩展性	可定制性	可配置性	可重用性
管道 / 过滤器风格		+	+		+	+	+		+	+
基于事件的隐式调用风格		−			+	+	+	+	+	+
缓存风格	+									
C/S 风格	+									
B/S 风格			+							
按需代码风格	+	−			+			+	+	+
C2风格		+	+		+	+	±		+	+
插件风格		+	+	+	+	++	++	+	+	+
REST风格	+	+		+	+	+	+		+	+
RIA 风格	+									
正交风格		+	+	+	+	++	++		+	+
SOA风格	−	_	++	−	++	++	++	++	+	++

2.3 WEB 应用系统的期望约束集

越来越大、越来越复杂的应用系统的创建，尤其是将几个属于不同体系结构风格的子系统集成为一个完整的复杂大系统，仅靠单独一种体系结构风格的指导是很难满足需求的。解决这个问题的一种趋势是通过将若干种风格集成，再用来指导系统的构建。

表 2　WEB 应用系统的期望约束集

设计约束	候选风格
统一接口	REST风格　RIA风格　SOA风格　…
方便集成	REST风格　SOA风格　正交风格　…
方便扩展	插件风格　正交风格　SOA风格　…
分层部署	REST风格　RIA风格　SOA风格　…
高重用性	REST风格　RIA风格　SOA风格　…
可靠/容错能力	REST风格　正交风格　RIA风格　…
按需代码	REST风格　RIA风格　SOA风格　…
缓存数据	缓存风格　REST风格　…
关注点分离	C/S 风格　SOA风格　REST风格　…
高用户体验	RIA风格　…
简单性与通用性	REST风格　C2风格　…
与业务对齐	SOA风格　…
低成本开发	REST风格　RIA风格　SOA风格　…
子系统独立开发	正交风格　…
减少网络延时	RIA风格　…
增强的安全性	REST风格　插件风格　…
…	…

基于 WEB 模式的企业业务系统更多的关注开发成本、业务敏捷性、接口的统一、关注点的分离以及方便集成扩展和更好的用户体验。表 2 识别出了 WEB 系统所期望得到的属性，并结合第 2.2 节介绍的风格比较表，我们选择了会导致这些属性的体系结构风格，将他们与早期的 WEB 风格结合形成一种新的、混合的现代 WEB 风格，这种风格能够更好的反应现代 WEB 系统所期望的属性。比如通过简化和标准化接口来满足“统一接口”和“方便集成”的约束，这方面 REST 风格和 SOA 风格做得最好，我们选择了 SOA 风格；WEB 系统往往通过缓存来提高服务性能，这方面缓存风格和 REST 风格可

以满足，而 REST 风格本身就关注缓存和中间件技术，所以这里引入了 REST 风格；再如通过引入富客户端技术来满足“减少网络延时”的约束，这里引入了 RIA 风格。经过这样的比较分析我们确立了 RROS 混合风格。

本表格并不是穷尽 WEB 应用系统的所有约束，对于约束，在软件开发中任何需求、管理、过程等的变动都会引入约束，本表只关心 WEB 系统中最核心的开发实现约束集。

2.4 RROS 混合体系结构风格特点

RROS 风格叠加了 RIA[8]、REST [9,10]、正交[11,12]、SOA[13]风格的约束，增强了 WEB 开发的特性。RROS 风格是对 WEB 应用系统的一种抽象，忽略的构件实现和协议语义的细节，以便聚焦于构件之间交互的约束上，它代表了 WEB 应用行为的本质。

按照第 2.1 节体系结构风格定义，对 RROS 风格的抽象视图不做限制，可依据具体需求提供适当的视图，比如系统的上下文图、应用体系结构图、接口视图、业务视图等等；RROS 的术语表包括资源、服务、线索、层次、缓存、中间件、用户体验、业务对齐等；对于约束集如表 2 所示。

RROS 风格是一组系统构架约束，当作为一个整体应用时，它强调构件之间交互的可伸缩性、接口的通用性、构件的独立部署、减少交互延迟和增强用户体验、最大限度的利用现有系统及对资源的服务封装。

RROS 风格体现的六大核心特点：

(1) 支持标准化、开放性的理念和技术

(2) 统一资源描述、简化操作接口和领域概念

(3) 结构上，基于线索的层次划分

(4) 关注缓存和中间件技术

(5) 关注网络交互效率和用户体验

(6) 面向服务的集成和业务对齐理念

3 RROS在CKM系统中的应用

3.1 CKM 系统简介

面对日益加剧的竞争环境，企业必须比以往任何时候更积极与客户打交道，使他们参与企业的产品研发、市场营销、测试和销售。也就是说企业必须实施客户知识管理(CKM，Customer Knowledge Management)[14]，以提升核心竞争力。

CKM 系统是构建客户统一视图、进行客户研究、管理并传播客户知识的开放平台，为运营商实施以客户为中心的信息运营提供支撑。CKM 可以说是 CRM 与 KM 的结合，使得 CKM 既具有知识管理的特性，又有客户管理的理念。

3.2 CKM 上下文体系结构与业务体系结构

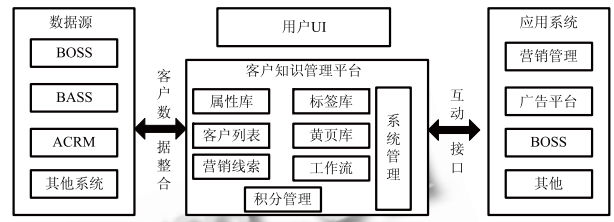

图 1 客户知识管理平台系统上下文

图 1 是 CKM 系统上下文体系结构，客户知识管理平台主要业务功能是客户的提取、客户行为分析以及客户知识积累等。数据源来自 BOSS，CRM 等业务运营系统，在本平台中形成客户知识，并把知识传播到营销管理平台和广告平台等运营系统中，实现了客户提取的智能化。

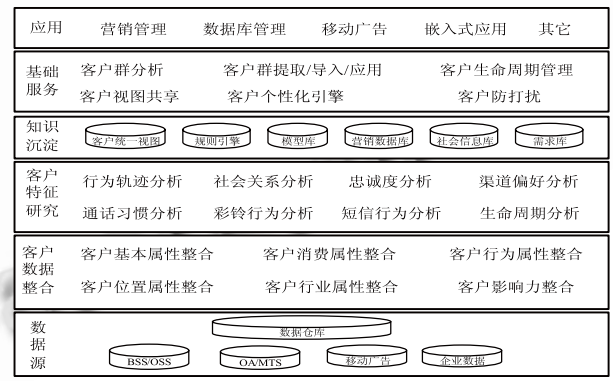

图 2 客户知识管理平台业务体系结构

图 2 是 CKM 系统的业务体系结构。从业务角度、客户知识平台可以划分为 5 个层次：(1)客户数据整合层：平台提供广泛的数据访问接口，从各个数据源获得客户数据，并根据业务规则对数据进行整合，形成客户统一视图。(2)基础功能组件：提供平台所需的基本报表分析，规则管理的功能。(3)业务组件层：用户可以借助客户统一视图，利用第三方工具对客户做专题分析，如通话习惯分析、渠道偏好分析、行为轨迹分析、分析所得到的规则、模型等研究成果可以通过平台统一管理以丰富客户知识。(4)对外服务层：客户

知识管理系统通过标准的接口为各运营系统提供客户知识服务，如为营销平台和广告平台提供目标客户提取服务，为其他行业应用提供统一客户视图。(5)公共组件层：提供系统管理的功能。

CKM 需要和几个业务运营系统进行数据交互，数据量比较大，因此接口设计上应尽量简单和标准化；在客户知识概念的描述上也应该简化；方便其他异构系统提取某个客户的基本知识和扩展知识；CKM 系统要求有更好的用户体验和网络交互性能 通常 CKM 以轻量级、嵌入式的形式集成到其他系统，因此对可集成性要求较高。从上面的总结可以发现 CKM 所期望的约束集是 RROS 风格所提供的约束集或者特性的一个子集，下面将介绍为支持 CKM 开发所做的技术和框架的选型。

3.3 满足 RROS 风格约束条件的技术和框架

依据 CKM 的上下文体系结构和业务体系结构需求可以发现 RROS 风格完全满足了其构架需求。为了满足 RROS 风格的设计约束，CKM 系统在体系结构设计时选择了一系列开发框架如图 3。

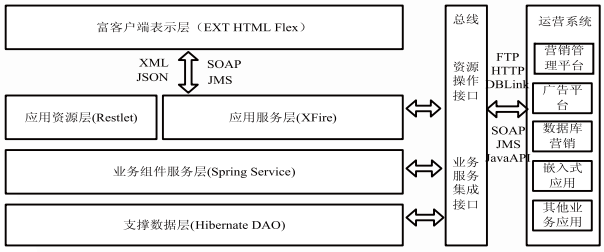

图 3　客户知识管理技术支持框架

为了满足系统的开放性和标准化，我们选择了 J2EE 平台，并把开源的 Liferay 系统作为开发基础，CKM 在此基础上演化，从而最大化地复用已有的构件和技术；为了满足用户体验约束，在界面表示层我们选择了 Ext 做客户端实现技术；为了简化客户知识概念，在系统应用层对原有的客户属性、客户标签、客户列表等概念进行了资源封装，这里我们选择了 Restlet 资源引擎，统一了资源访问接口；在可集成性上，我们选择了 XFire 技术作为服务封装的实现框架；系统功能结构上采用水平层次和垂直线索式的划分，在构件依赖配置和事务管理方面选择了 Spring 框架，在数据持久封装化方面采用了 Hibernate 框架；在大数据量集成上，采用 ETL 工具和 FTP 协议以及标准化的数据文件和校验文件，简化了数据集成接口；体系结构设计工具我们采用了 Structure101，方便了设计师设计体系结构以及更好的约束开发人员的开发活动，这几大框架和技术的联合满足了 RROS 风格在 CKM 系统上的主要约束，也体现了 RROS 风格所定义的六大核心特性。

3.4 RROS 应用效果分析表

表 3　RROS 应用效果

关注点	应用效果
简单性	对数据和服务的资源封装，简化和统一了访问接口
网络性能	关注缓存和RIA技术，简化网络交互，提高用户体验
集成性	引入SOA理念，方便与周边系统集成和知识共享
平台无关性	本身基于J2EE平台，并引入服务框架，使系统跨平台、跨语言。
开发与管理	水平层次和垂直线索划分，使开发和项目管理更简便

RROS 风格强调对数据和服务的资源封装，简化了客户端操作接口；关注缓存和中间件以及 RIA 技术最大化地减少网络交互，并提高了用户的网络操作体验；基于 J2EE 平台和服务理念使系统具备跨平台、跨语言特性；首次引入了库集的概念，包括了风格库、设计模式库、构件库、服务库；系统开发分工可依据层次或者单个线索，线索之间是相互独立的，可并行开发，对于系统变动，可以将变动限制在一个线索或者一个线索集中，同时也方便管理人员对进度和开发情况的追踪。总之 RROS 风格最大化地简化了基于 WEB 的应用系统开发活动。

4　结语

RROS 风格在系统的不同层次不同抽象级别上应用了不同的风格，增强了系统的互操作性、重用性、简单性、可伸缩性等，方便分工和管理，极大地缩短了开发时间。不同项目的系统设计要求选择不同的风格，每一种风格都有其优缺点，切不可生搬硬套，选择最合适的风格才是成功的关键。下一步将对 RROS 风格进一步细化和量化、研究如何用一种形式化的语言来描述 RROS 风格，并研究风格的统一描述方法，使各种风格易于比较、对照、选择和集成。

参考文献

1 Shaw M, Garlan D. Software architecture: perspectives on an emerging discipline.北京:清华大学出版社.

1998.19－32.
2 孙昌爱,金茂忠,刘超.软件体系结构研究综述.软件学报, 2002,13(7):1228－1237.
3 张广泉,朱雪阳,郑建丹.基干时态逻辑语言 XYZ/E 的软件体系结构(II)-常见体系结构风格的描述.重庆师范学院学报(自然科学版), 2002,19(1):1－3.
4 Shaw GM. An introduction to software architecture. CMU-CS-94-166, 1994.
5 叶俊民,赵恒,曹瀚,王鸿丰,王振宇.软件体系结构风格的实例研究.小型微型计算机系统, 2002,10(23):1158－1160.
6 袁兆山,李莹莹,李宏芒等.软件体系结构评价指标体系与评价方法的探讨.全国软件与应用学术会议(NASAC2004)论文集, 2004,136－140.
7 李莹莹,孙全玲,袁兆山.基于评价矩阵的软件体系结构风格选取.淮北煤炭师范学院学报. 2005,26(4):66－67.
8 周林,谢峰.基于 RIA 架构的应用开发改进方案.微计算机信息, 2007(2):221－223.
9 Fielding R, Architectural Styles and the Design of Network-Based Software Architectures, PhD thesis, Univ. of California, Irvine, June 2000.
10 许卓明,栗明,董逸生.基于RPC和基于REST的Web服务交互模型比较分析.计算机工程与应用, 2003, 29(20):6－8.
11 李海洋,李柏林,郭荣佐.正交软件体系结构设计和演化方法.计算机应用研究, 2007,24(1):78－79.
12 张友生,陈松乔.正交软件休系结构的设计与进化.小型微型计算机系统, 2004,25(2):295－299.
13 徐黎明,姚耀文. SOA 开发框架的研究和实现.计算机应用, 2008,28(B06):307－309.
14 王战平,柯青.客户知识管理概念研究.情报科学, 2004,(1):19－21.