摘要

随着移动增值业务的迅速发展和普及，人们通过短信、彩信、WAP等增值业务方式参与媒体的活动也越来越频繁，由于目前大部分媒体增值业务管理系统功能单一，没有集成短信、彩信等多种增值业务类型，所使用的技术过于陈旧等缺陷，应用范围也存在一定的局限性，因此研究新的通用的媒体增值业务管理系统将具有非常大的意义，而本论文就是要研究基于J2EE技术构建一个功能齐全、集成大部分移动增值业务的通用媒体增值业务管理系统。

商用的媒体增值业务管理系统主要包括直播、统计、帐单、交友等模块，采用最新的J2EE技术，包括hibernate关系数据库影射技术、spring轻量级动态注入技术以及webwork表示层技术等。而在研究过程中，将取得以下成果：

(1) 实现跨平台的互通和集成，实现和第三方系统包括asp.net ror、asp、php等的互相通信和集成，并能实现跨平台的部署和集群。

(2) 解决多种增值业务集成问题，通过一个通用的管理系统集成多种增值业务。

(3) 提供二次开发功能，方便新业务的开展，适应新需求的变化。

(4) 技术上采用最新的web2.0技术，给予用户全新的体验。

(5) 提供帐单管理功能，方便SP公司和合作方的费用结算。

(6) 采用hibernate ORM技术，增加短信的吞吐能力。

在取得以上研究成果基础上，采用软件工程的方法，把研究成果运用到媒体增值业务管理系统当中，并通过实际运营逐步完善和修改该增值业务管理系统，使之更适合媒体行业增值业务方面的应用和管理。

关键词：hibernate，J2EE，web2.0，ORM,增值业务，SP,webwork,spring

ABSTRACT

With the development of mobile value-added service, more and more people participate in media activity by SMS, MMS, WAP, and so on. Currently most mobile value-added management systems have too simple functions, for example, there is no integration of SMS, MMS and others, the technology is out of date, and the applied range has limitation. So there will be huge meaning for researching common media value-added management system. This paper research how to building a common media value-added management system with complete functions based on J2EE.

Commercial media value-added management system includes broadcast, statistics, billing and friends modules, use the newest J2EE technology, including hibernate ORM technology, spring IOC technology and webwork and so on. In process of research, we will get the following gains:

(1) It realizes the cross platform intercommunication and integration, also realizes the third-party system intercommunication and integration, such as asp.net, ROR, ASP, PHP and so on. What's more, it realizes the cross platform deployment and cluster.

(2) Solves many kinds of value-added services integration problems, through a common management system integrate many kinds of valued-added services.

(3) Provides the re-development functions, facilitates the new service development, and adapts the new requirements.

(4) In the technology uses the newest WEB2.0 technology, gives users a brand-new experience.

(5)Provides the billing management function, facilitates business settlement between SP companies and the cooperation companies.

(6)Use Hibernate technology to increase short message handling capacity.

Based on above researches, use the method of software engineering, apply research result to media value-added management system, and improve and modify this value-added manage system by exercise it in fact. Make it more suitable for

media industry value-added application and management.

Key Words: hibernate,J2EE, web2.0, ORM, value-added service, SP,webwork,spring

学位论文版权使用授权书

本人完全了解同济大学关于收集、保存、使用学位论文的规定，同意如下各项内容：按照学校要求提交学位论文的印刷本和电子版本；学校有权保存学位论文的印刷本和电子版，并采用影印、缩印、扫描、数字化或其它手段保存论文；学校有权提供目录检索以及提供本学位论文全文或者部分的阅览服务；学校有权按有关规定向国家有关部门或者机构送交论文的复印件和电子版；在不以赢利为目的的前提下，学校可以适当复制论文的部分或全部内容用于学术活动。

学位论文作者签名：吴锋彬

2017 年 01 月 15 日

经指导教师同意，本学位论文属于保密，在　　年解密后适用本授权书。

指导教师签名：　　学位论文作者签名：吴锋彬

年　月　日　　2017 年 01 月 15 日

同济大学学位论文原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，进行研究工作所取得的成果。除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人创作的、已公开发表或者没有公开发表的作品的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。本学位论文原创性声明的法律责任由本人承担。

签名：吴镜彬

2019年 7月 15 日

第1章 引言

1.1 课题背景及意义

短信、彩信等增值业务经历过一段洗礼之后，很多不规范的SP公司(service provider,移动增值业务领域的服务提供商）逐渐被淘汰，移动增值业务市场由此慢慢开始步入一个规范、成熟的发展时期。而短信、彩信等增值业务在电视台、电台等媒体行业的应用也开始慢慢成为媒体行业一个重要的宣传和沟通的工具。

目前很多的运营商和SP公司开始借机进入媒体行业，开始大力推广手机增值业务在媒体行业的具体应用。移动公司也开始大力推广手机增值业务，制订了一系列的业务规范和措施，逐渐规范和完善增值业务市场。因此研究和开发一套通用的适合媒体行业应用的移动增值业务管理系统将具有非常大的意义。

移动增值业务在媒体行业当中的运用，将增加电台电视台节目的互动性，提高用户的参与积极性，同时对于媒体行业本身以及运营商来说也是一种全新的宣传方式，实现双赢的局面。

移动增值业务目前已经被用户所广泛接受，增值业务也开始进入媒体行业，电台、电视台、报纸等机构利用移动增值业务，和用户形成良好的互动。而且目前大部分媒体机构都开通了短信等移动增值业务，但在技术上目前支持不够，只能开展一些简单的投票类的活动，总得来讲，SP公司提供的增值业务平台主要还存在以下问题：

(1)功能上比较单一，目前还没有公司能提供彩信等其他增值业务功能，因此存在一定的应用局限性。

(2)功能不够齐全，不能方便二次开发，增加新应用困难，与其他第三方系统互通困难。

(3)使用的技术太过陈旧，安全不够，不能满足大负荷应用，大多数SP公司提供的增值业务管理系统采用的是ASP、PHP较老的技术，技术上没有推陈出新。

(4)没有解决结算帐单分级管理，目前几乎没有这样的管理系统能提供详细

的帐单管理功能。

本课题研究的主要内容是在技术上有所创新，采用 J2EE 技术，部分采用 WEB2.0 规范设计，集成短信、彩信等多种移动增值业务，提供一个功能齐全，全新用户体验的适合媒体机构的增值业务管理系统。首先要解决多种增值业务集成问题，实现多种移动增值业务集成在一个通用的管理系统之内。其实是提供二次开发功能，方便新业务扩展，适应新需求的变化；同时，技术上推陈出新，部分采用 WEB2.0 规范，给用户全新的体验；最后，由于目前增值业务管理系统几乎没有帐单管理功能，而该功能是 SP 公司和媒体机构结算的依据，因此本课题要解决该问题。

通过该增值业务管理系统，媒体机构可以通过统一的一个管理系统，实现短信、彩信等移动增值业务的管理，和用户形成良好的互动，彻底改变目前系统支持增值业务形式单一的缺点（只支持短信，并不支持彩信等其他增值业务）。

通过该通用的增值业务管理系统，媒体机构或 SP 公司可以方便的进行二次开发，提供第三方接口，开展一些特殊类的活动，比如当前 CBA 比赛的短信字幕程序，都是另外单独开发服务器程序，或者直接由字幕员手工编辑的，运用该系统提供的跨平台跨应用的通用接口，可以方便进行二次开发，集成短信字幕应用，且不影响解说员阅读短信、彩信。

目前的增值业务平台基本上都是短信平台没有开通彩信，而且采用的技术也比较陈旧，采用目前最新的技术开发该通用的媒体增值业务管理系统，将给用户全新的体验，可以解决留白等问题（网络速度慢的时候，每次刷新会出现一段时间白屏的情况），在技术上可以说做了比较大的创新。

由于目前很多公司提供的增值业务管理系统还没有详细的结算功能，因此，本系统将解决结算问题， 包括按条以及包月的帐单，做到隔天出帐。

1.2 研究现状及发展动态

本课题是手机增值业务在媒体行业的应用项目，短信、彩信等增值业务在媒体行业的应用具有方便性，安全性，快捷性等优势，在目前短信、彩信应用步入成熟期的背景下，移动增值业务在媒体行业的应用将越来越具有广阔的前景。

目前国外短信等增值业务在媒体等行业的应用已经比较普及，技术上也已

经比较成熟，由于风俗习惯上的原因，短信在亚洲的受欢迎程序远远高于其他欧美国家，而目前短信等增值业务在国内的媒体行业应用也才刚刚起步。一个是短信经历了一段不规范运作之后，目前的市场刚刚进入规范期，另一个是在媒体行业的应用也刚刚被老百姓接受。

在手机短信与传统媒体结合模式中，广播获益颇多，电视媒体凭借其广大的影响面和火爆的节目收视率，在利用手机短信投票和竞猜中也获利颇丰。虽然媒体中的短信收费和短信投票引发了不少争议，但是受众还是有和媒体互动、参与节目、表达意见的需求。

虽然目前移动增值业务（包括短信、彩信、WAP 等）在媒体行业应用已经取得了一些成就，但大多数公司提供的移动增值业务管理系统存在以下问题：

(1)功能单一，没有彩信等其他增值业务功能，目前几乎没有一家公司提供支持彩信功能的媒体增值业务管理系统，这样媒体机构就无法开展彩信相关的一些业务，比如电台交友、彩信参与互动节目等等。

(2)功能不够齐全，不能方便二次开发，现在的软件系统在一定程度上需要方便跟其他第三方应用的集成与结合，目前的媒体增值业务管理系统一般是简单的一个投票系统，或者是一个简单的短信直播平台，功能比较单一，系统比较分散，使用极不方便；另外各个系统之间交互集成困难，比如电视台上需要字幕播出用户发上来的短信，由于系统没有提供开放的接口，不得不重写服务器端程序，因此，虽然媒体行业的增值业务应用比较广泛，但在技术上支持不够，各个应用太分散，无法集中管理、方便搭建新的应用。

(3)使用的技术太过陈旧，安全不够，不能满足大负荷应用，目前的增值业务在媒体行业的应用还局限在短信上，所使用的技术一般也是 ASP 等比较老的技术，在安全上存在一定缺陷，容易被攻击，且在大负荷运营当中存在一定问题，短信的吞吐速度不够；此外，由于这些系统相开发较早，没有运用一些比较热门的技术（比如 web2.0 ajax 等），在用户体验上有一定差距，比如很多电台等媒体机构使用的网通的宽带接入，而有些公司提供的增值业务管理系统是电信的宽带接入，因此访问速度非常慢，一旦刷新页面就会出现白屏的情况，严重影响节目的播出！

(4)没有解决结算和帐单分级管理,目前的媒体增值业务管理系统，由于系统分散，没有集中管理，导致结算也分散，而且提供的结算方式也过于简单（只显示一个短信的收发数量），没有处理包月、定制等业务形式；因此本课题将解决

结算问题，集中结算集中管理，做到隔天结算！

综上所述，研究和开发一个通用的，囊括各种移动增值业务的，集中管理的，方便第三方系统交互集成的，方便扩展新应用的适合媒体机构使用的增值业务管理系统，将非常具有意义。

1.3 主要研究内容和关键技术

首先，具有商业使用价值的通用媒体增值业务管理系统，主要包括直播、统计、抽奖、帐单、交友等模块，整个系统需要实现跨平台的互通和集成，目前的媒体增值业务管理系统由于采用的技术上过于陈旧，开发和提供通用的接口变得相对比较困难，因此为了能和 dotnet、asp 以及 php 等其他第三方应用互通和集成，系统必须提供通用的接口。比如电视台常见的直播字幕程序，需要读取增值业务管理系统的直播数据，按照目前的系统，需要额外开发服务器端应用，以返回给字幕程序相应的数据。本课题就是要解决目前媒体增值业务管理系统中存在的这个问题，提供一个通用的开放的接口，方便集成其他应用。此外，本系统采用 J2EE 技术，也实现了在应用上的跨平台部署和集群。

其次，为确保数据安全，媒体行业增值业务管理系统因为面向的用户是电视台电台等媒体行业用户，因此可以通过 CA 认证登录该系统，实现数据交换的安全性，服务器端可以确保连接的客户端是合法的，同理，客户端也可以验证服务器不是伪造的，而且，客户端与服务器之间通讯的数据都是经过加密的，保障的用户的短信等隐私不被窃取。

最后，用 UML 对基于 J2EE 的媒体增值业务管理系统进行建模，本课题意在通过 UML 建模，将整个系统划分成直播、抽奖、答题、统计等模块，并对模块进行单独建模分析，各个模块之间通过单一接口互相通讯。而本课题的目的是通过基于 J2EE 的架构，开发一个能通用的媒体行业增值业务管理系统，在这个系统上，方便媒体用户操作，方便搭建新的应用，方便 SP 公司和各个媒体机构结算等等，这些需求设计都通过 UML 来建模。

1.4 论文主要成果及创新

在使用J2EE技术重新构建媒体增值业务管理系统，结合实际的工作经验，取得了一定的创新和工作成果：

(1)集成了多种移动增值业务，使用一个通用的管理系统就可以管理各种增值业务，进行统一管理，方便的媒体行业使用。

(2)数据层进行了缓冲，极大的提高了系统的压力承受能力。

(3)方便快捷的二次开发，系统提供通用的二次开发接口，使系统应用直接的互相集成变得非常方便。

(4)增加完善的帐单管理功能，做到隔天出帐单。

1.5 论文组织结构

论文共 6 章，围绕着增值业务管理系统的研究与实现，论文的各章节内容组织如下：

第 1 章：绪论。介绍课题的研究背景、意义及现状，阐述了课题的主要研究内容、关键技术、主要成果及创新点。

第2 章 ：简单介绍了通用媒体增值业务管理系统的相关技术。

第3 章 ：简单介绍了通用媒体增值业务管理系统的需求。

第4 章 ：介绍了通用媒体增值业务管理系统的各个模块设计和实现。

第5 章 ：典型业务的系统实现。

第6 章 ：结论与展望，对论文作了小结，同时指出了进一步工作的方向。

第 2 章 相关技术

2.1 基于工作流的系统实现

2.2.1 工作流基本概念

工作流的概念起源于生产组织和办公自动化领域。它是针对日常工作中具有固定程序的活动提出的一个概念。通过将工作活动分解成定义良好的任务、角色，规则和过程来进行执行和监控，达到提高企业经营管理水平和工作效率的目标。工作流通常用于过程的自动化，通过将文档、信息或任务按照预先定义好的规则和流程在参与者之间传递，从而帮助用户完成整个经营目标。

一个工作流有三个基本元素：活动、活动之间的连接关系和参与活动的角色及组织单元。活动对应于经营过程中的任务，主要是反映经营过程中的执行动作或操作。活动之间的连接关系代表了经营过程的规则和业务流程。一个工作流就是一个用一组连接关系组合起来的一组活动组成的一个反映企业某个业务过程的模型。在工作流模型中定义的参与活动的角色及组织单元则描述了企业的经营过程是由谁来完成的。

由于工作流提供了经营过程逻辑与信息支撑系统相分离的环境，这种应用逻辑和过程逻辑分离的方式可以大大提高工作流系统的重用率，它可以在不修改具体功能模块实现方式（硬件环境、操作系统、数据库系统、编程语言、应用开发工具、用户界面）的情况下，通过重新定义过程模型来改进系统性能，实现对生产经营过程部分或全部地集成管理，发挥系统最大效能。

关于工作流的定义不同的研究者和工作流产品供应商从不同的角度给出了许多定义，到目前为止还没有一个完全统一的定义。在这里我们给出工作流管理联盟对工作流所下的定义：工作流是一类能够完全或者部分自动执行的经营过程，它根据一系列过程规则，使文档、信息或任务能够在不同的执行者之间传递、执行。

2.2.2 工作流技术的起源和发展

在 PC 机出现前，纸张是各行各业中进行日常业务活动所不可替代的信息载

体，如表单、文件、信函、技术资料等在内的各类文档都是以纸张的形式来传递的。这种古老的载体在信息的处理、组织、存储以及查询检索方面都是很低效的。到八十年代初期，随着PC机的出现，越来越多的信息处理可以通过PC机来完成，因此，人们希望以方便有效的电子方式来替代纸张作为信息处理的载体。于是一些企业便建立了一种无纸化的、计算机智能的工作环境，用于实现日常表单处理的电子化与自动化。这种系统便是现在工作流管理系统的原始雏型。

到八十年代中期，FileNet、ViewStar等公司率先开拓了工作流产品市场，他们把图像扫描、复合文档、结构化路由(structured routing)、实例跟踪、关键字索引以及光盘存储等功能结合在一起，形成了一种全过程支持某些业务流程的集成化的软件(包)，推出了一些商用工作流管理系统。比较典型的有FileNet于1984年推出的WorkFlow Business System， ViewStar于1988年推出的 ViewStar。很显然，这种增值性质的集成化软件系统为企业简化与重组自己的关键业务流程提供了一种非常合适的方案。工作流从最初的诞生之日起便是作为一种面向过程的系统集成技术而出现的，但限于当时的计算机发展水平，它所集成的功能较为简单。

进入九十年代，随着计算机的普及、网络的延伸，现代企业的信息资源越来越表现出一种异构、分布、松散耦合的特点，企业的分散性、决策制定的分散性、对日常业务活动详尽信息的需求以及 Client / Server 体系结构、分布式处理技术(CORBA、 WWW、 OLE、JAVA)的日益成熟，都说明了这样一个事实——集中式信息处理的时代已经过去，实现大规模的异构分布式执行环境，使得相互关联的任务能够高效运转、并接受密切监控已成为一种趋势。在这种不可抗拒的技术背景下，工作流管理系统也由最初的创建无纸办公环境，转而成为同化企业复杂信息环境、实现业务流程自动执行的必要工具。因而人们开始从更深的层次、更广的领域上对工作流技术展开研究、开发。

为了实现不同工作流产品之间的兼容性，于1993年成立了工作流技术的标准化组织工作流管理联盟(Workflow Management Coalition，WFMC)。WFMC在工作流管理系统的相关术语、体系结构和编程接口等方面制定了一系列标准。它的成立标志着工作流技术在计算机应用研究领域之中被明确地划分出了自己的一席之地，相应的概念与术语也得到了人们的承认。WFMC的成立进一步推动了工作流技术的研究以及相关产品开发的发展，更多更新的技术被集成进来，文

件管理系统、数据库、电子邮件、移动式计算、Internet 服务等都已被容纳到工作流管理系统之中。

纵观工作流软件产品由八十年代的萌芽到九十年代后的繁荣，大约经历了三个阶段：第一阶段，主要为应用于某些特定领域的、相对独立的应用系统，比如图像、文档管理系统；第二阶段，主要表现为具有底层的通讯基础结构、能够实现任务协作的应用系统，比如具有消息传递功能的工作流系统；第三阶段，具有图形用户界面的过程定义工具、用户定义与任务执行完全分离的工作流系统，其体系结构基本上符合工作流管理联盟所提出的标准结构。经历了这三个阶段的发展，工作流产品基本上确定了它在计算机应用软件市场上的独立位置。

2.2.3 工作流管理系统

关于工作流管理系统，WFMC 给出了如下的定义：工作流管理系统是一个软件系统，它完成工作流的定义和管理，并按照在计算机中预先定义好的工作流逻辑推进工作流实例的执行。

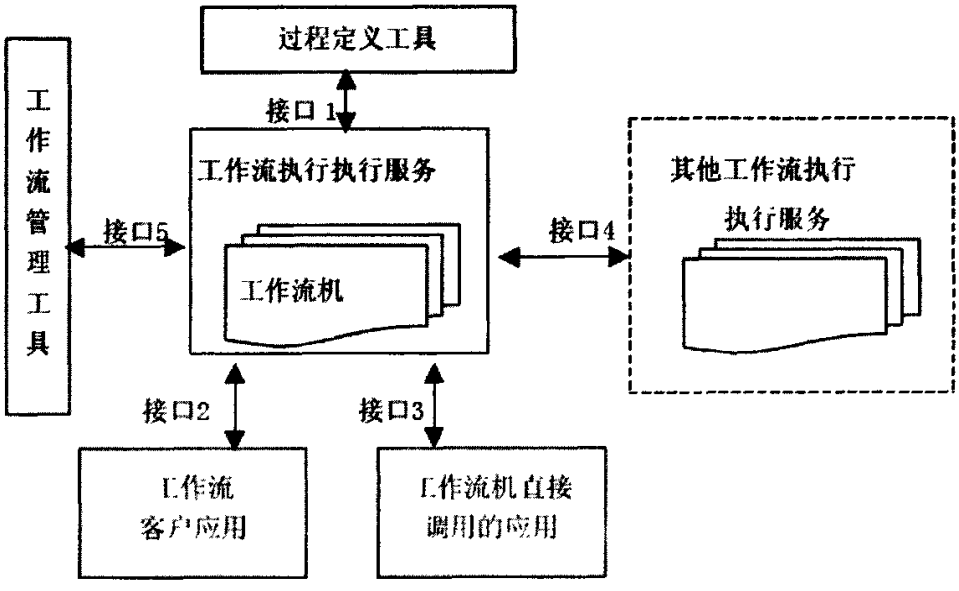
根据工作流的基本概念，我们可以这样认为，工作流管理系统是指运行在一个或多个称为工作流机的软件上的用于定义、实现和管理工作流运行的一套软件系统，它和工作流执行者(人、应用)交互，推进工作流实例的执行，并监控工作流的运行状态。工作流管理系统类似于在单个计算机上的操作系统，它为企业的业务系统运行提供一个软件支撑环境，只不过工作流管理系统支撑的范围更大、环境更复杂而已，所以也有人称工作流管理系统是业务操作系统(BOS —— Business Operating System)。

虽然不同的工作流管理系统具有不同的应用范围和不同的实施方式，但它们还是具有许多共同的特性。从比较高的层次上来抽象地看工作流管理系统，可以发现所有的工作流管理系统都提供了三种功能：一是建立阶段的定义功能：主要考虑工作流过程和相关活动的定义和建模功能；二是运行阶段的控制功能：在一定的运行环境下，执行工作流过程，并完成每个过程中活动的排序和调度功能；三是运行阶段的人机交互功能：实现各种活动执行过程中用户与 IT 应用工具之间的交互。

为了实现工作流技术的标准化和开放性，WFMC 提出了一个工作流体系结构参考模型，约定了工作流管理系统的体系结构，应用接口及特性。图 2-1 为 WFMC 提出的工作流参考模型，并给出了五类接口：

• 接口 1：工作流服务和工作流建模工具间接口，包括工作流模型的解释和读写访问；

• 接口 2：工作流服务和客户应用间接口，这是最主要的接口规范，它约定了所有客户应用和工作流服务之间的功能访问方式：



图：2-1 工作流管理系统参考模型

• 接口 3：工作流机和工作流应用间的直接接口；

• 接口 4：工作流管理系统之间的互操作接口；

• 接口 5：工作流服务和工作流管理工具之间的接口；

系统各部分功能如下：

1）工作流执行服务：由一个或多个工作流机组成（在分布环境下，由多个工作流机组成），提供了过程实例和执行的运行环境，具体完成以下功能：

• 解释过程的定义，生成过程实例，并管理其实施过程，包括开始、结束、挂起、恢复等。

• 依据工作流相关数据为过程的活动导航，包括顺序或并行操作、期限安排等。

• 维护工作流控制数据并向用户传递必要的相关数据。

• 与外部资源交互完成各项活动。

工作流执行服务通过下面两种途径使用外部资源：

• 客户应用接口：工作流机通过任务项列表管理器来管理资源，任务项列表管理器负责从任务项列表中选择并监督工作项的完成。

• 直接调用应用接口：工作流机直接调用相应的应用来完成一项任务。这主要是针对基于服务器的无需用户参与的应用，那些需要用户操作的活动则通过任务项列表管理器来调用。

在分布式的工作流执行服务中，多个工作流机系统协同工作，推进工作流实例的执行，每一个工作流机控制过程一部分，并使用相关的资源和应用工具，这种执行服务需要共同的命名和管理范围，便于过程定义和用户/应用名称的一致，分布式的工作流系统采用特定的协议来同步各工作项的工作流执行服务。由于各个厂家的协议不尽相同，因此当选用不同的工作流系统产品时，各工作流机之间需要一个标准来进行控制过程实例的生成，使之能够在异构的工作流机间传递过程、子过程及活动、支持共同的管理职能。

2）工作流机：也称工作流引擎，是一个为工作流实例提供运行执行环境的软件服务。它主要提供以下功能：

• 对过程定义进行解释；

• 控制过程实例的生成、激活、挂起、终止等；

• 控制过程活动间的转换，包括串行或并行的操作、工作流相关数据的解释

等；

• 支持用户操作的界面；

•维护工作流控制数据和工作流相关数据，在应用或用户间传递工作流相关数据；

• 提供用于激活外部应用以及提供工作流相关数据的界面；

• 提供控制、管理和监督的功能。

3）工作流过程定义工具：主要用于分析、建模、描述并记录经营过程。它输出一个能被工作流机动态解释的过程定义。不同的工作流产品其过程定义工具输出和格式是不同的，接口 1 不仅使工作流的定义阶段和运行阶段分离，使用户可以分别选择建模工具和执行产品，还可以使不同的工作流产品合作提供一个过程定义的运行服务环境。

4）工作流管理工具：主要负责对工作流实例的运行进行监控。管理员可以通过工作流管理工具获得目前各个活动的运行情况报告，并干预实例的推进。

工作流管理系统的实施：

如图 2-2，工作流管理系统的实施应用一般分为三个阶段，即过程建模阶段、过程实例化阶段和过程运行阶段。过程建模阶段通过利用工作流建模工具完成企业经营过程模型的建立，将企业的实际经营过程转化为计算机可处理的工作流模型。在过程实例化阶段为每个过程设定运行所需的参数，并分配每个活动执行所需要的资源（包括资源、人员、应用）。过程运行阶段完成经营过程的执行，在这个过程中重要的任务是完成人机交互和应用的执行，并对过程与活动的执行情况进行监控与跟踪。

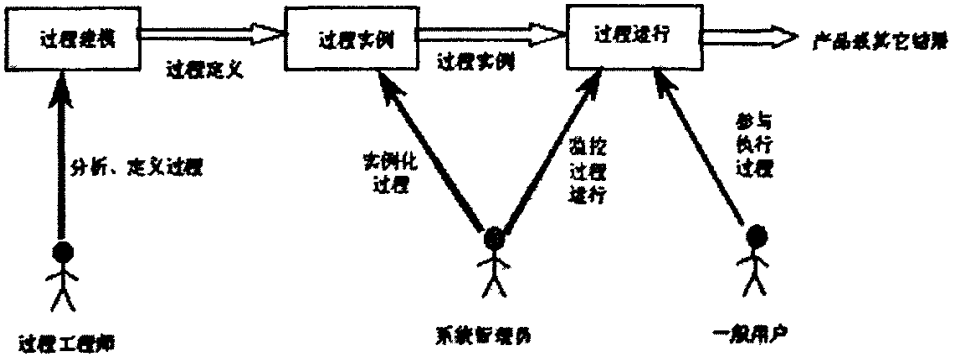


图 2-2 工作流管理系统实施三个阶段

一、过程建模阶段

工作流管理系统过程建模阶段的功能主要是完成经营过程的计算机化的定义。在这个阶段，利用一个或多个建模技术与工具，完成实际的经营过程到计算机可处理的形式化定义的转化。所得到的定义通常可称为过程模型、过程模板、过程元数据或过程定义。因此，在工作流建立阶段主要完成过程建模工作。在 WFMC 定义的工作流管理系统中，将过程建模得到的结果统称为过程定义。

过程建模主要解决如何根据过程目标和系统约束条件，将系统内的活动组织为适当的经营过程的问题。过程建模的作用体现为：

1)用于准确描述企业的经营过程，供流程分析和优化(如经营过程重组)使用。

2)用于在不同的组织和信息系统间共享经营过程知识。

3)根据设计的企业过程模型进行相应的功能构件配置，使得所建立的系统能够按过程实现横向集成，而不是按传统的部门划分结构实现纵向集成，从而满足企业核心价值流的要求。按过程模型进行系统构件配置还能够实现柔性更好的过程集成。

有很多方法可以用来进行工作流(过程)模型的定义与描述。使用者可以通过一套完整有效的描述经营过程的建模语言对流程的的逻辑顺序结构，如顺序、分支、汇合、条件、循环、并行进行描述。目前较为广泛接受的建模语言有 CIMOSA 的经营过程描述语言、工作流管理联盟 WFMC 定义的工作流描述语言、Keller

等人提出的EPCM模型等，这些工作流描述语言的描述形式与程序设计语言中语义结构的定义方式类似。其它一些方法是采用传统项目管理中使用的概念和模型来表述经营过程，例如PERT图或其它各种形式的网络图等。

二、过程实例化阶段

在完成了过程模型的定义后，所生成的工作流模型将由工作流执行服务软件进行实例创建并控制其执行过程。一般把工作流管理软件称为工作流机。由工作流机对使用工作流模型描述的过程进行初始化、调度和监控过程中每个活动的执行，在需要人工介入的场合完成计算机应用软件与操作人员的交互，从而实现在模型中定义的经营过程与现实世界中实际过程之间的连接。

工作流机除了完成过程的创建、删除、活动的执行与控制外，它的另外一个重要的功能是完成与应用软件及操作人员的交互。

企业经营过程的执行通常需要若干个应用软件和若干人员的参与才能够完成，随着计算机与网络技术的迅速发展，特别是Internet应用日益普及的情况下，企业信息系统往往运行在不同地点的不同计算机系统上，计算机硬件环境、操作系统、数据库管理系统平台也不尽相同。为了能够支持这样一种分布性和异构性的计算机网络环境，作为工作流管理控制软件的工作流机同样需要能够在分布异构的环境中运行。

按照工作流机管理系统设计开发的难易程度，可以采用不同的设计方法来满足对工作流机的分布性要求。工作流管理系统的分布性可以分为分布式的工作流用户与应用接口、分布式工作流机和分布式工作流模型三种主要的分布方式。分布式的工作流用户与应用接口通常是工作流管理系统必须提供的分布处理功能，因为企业的应用软件和用户本身是分布在不同的计算机环境和不同的工作地点。

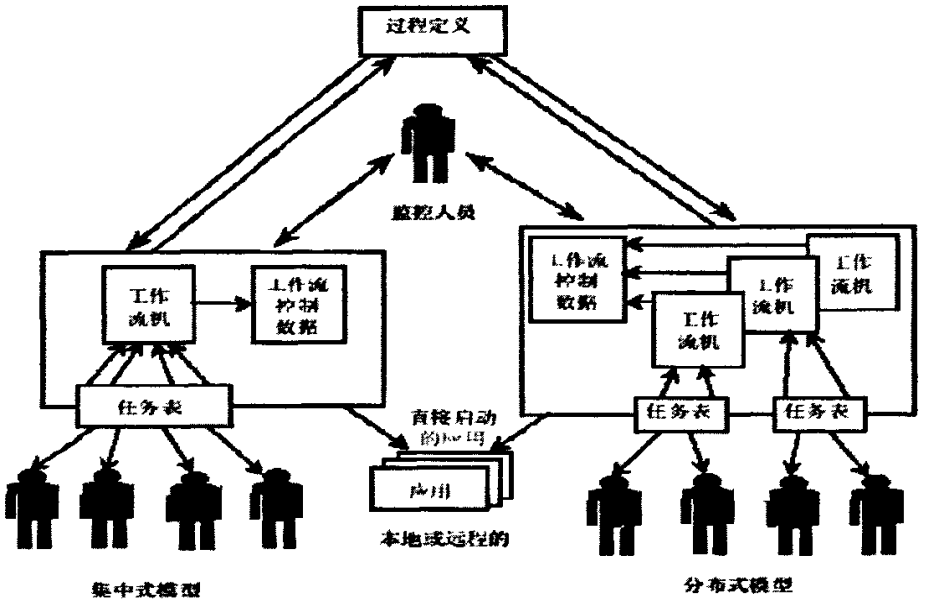


图 2-3 分布工作流机与应用结构

图2-3给出了一种分布式的工作流执行服务情况。其中左面表示的是集中式的工作流机模型，右面是分布式工作流机模型，整个系统是一个由异构分布工作流机构成的工作流执行服务环境。对于工作流模型和工作流机集中，而工作流接口分布的工作流管理系统的结构，所有计算机上的活动执行由一个工作流机来控制。而对于由多个工作流机协作执行一个过程实例这种情况，被控制的过程实例的控制数据必须是这些不同的工作流机都可以访问的。控制数据可以集中存放在一个主机上作为一个共享资源使用，也可以将它分布到不同的工作流机环境中。在将控制数据分布到不同的环境中时，必须定义一套机制来保证这些控制数据之间的一致性[16] 。

三、过程运行阶段

在工作流管理系统的运作过程中，人和应用是完成整个业务过程的主体。工作流定义工具、工作流执行服务和任务表管理器都是为完成业务过程和支持人员工作提供的运行环境和工具。在过程运行阶段主要进行人机交互：按照工作流任务管理器提供的任务项，完成具体的业务处理工作(如填写表格、启动一个应用来计算生产计划、查询库存情况等)，同时监控系统运行状态：检查、监视系统的执行情况，对于系统中出现的意外情况进行紧急处理，如终止、恢复

某个过程实例的执行，改变某个活动的状态以便整个系统能够继续执行等。

2.2.4 工作流模型定义

根据WFMC提出的工作流体系结构，一个完整的工作流管理系统是由建模工具（流程定义工具）、工作流机、工作流管理工具以及用户界面等模块组成的。从这一体系结构来看，工作流模型是整个系统的基础，它的确定性保证了系统内各部分之间交互的一致性。因而要建立一个工作流管理系统，关键是要设计一个正确、交互性强的工作流模型。

我们知道企业中业务流程是由一系列最基本的生产活动按照一定的逻辑顺序规则组成的，这些生产活动与它们之间的逻辑关系可以很直观地映射成为一个由节点和连接弧组成的有向图。有向图中的节点即表示一个可执行的活动单元，连接于两个节点之间的有向弧即表示活动间的先后顺序关系。为了便于描述，我们给出以下定义：

(1)有向图G={N, L}为一个二元组，N={n1，n2,…，ns}为节点的集合，L={l1，l2,…，lr}为连接弧的集合，其中li=（nj，nk）为nj到nk的连接弧，nj，nk∈N。

(2)对于任意ni，nj∈N，若有l=(ni，nj)∈L，则ni为nj的一个前趋节点，nj为ni的一个后继节点，l称为nj的一条输入连接弧，或者ni的一条输出连接弧。

(3)若Ñ包含于N，且Ñ={n|(n′ ，n)∈L}，则Ñ为节点n的前趋节点集，记为Pre(n)； 若Ñ包含于N，且Ñ={n|(n ，n′)∈L}，则Ñ为节点n的后继节点集，记为 Post(n)。

(4)节点状态：对于任意n∈N，有状态函数State(n)={0，1}，当节点n处于非执行状态时，State(n)=0；当节点n处于执行状态时，State(n)=1。初始时刻，任意n∈N，State(n)=0。

(5)转移函数：对于任意l∈L，有转移函数Trans(l)={0，1}，如果Trans(l)=1，则连接弧l允许发生转移(是否发生转移，则要取决于后面的演进规则)；如果Trans(l)=0，则连接弧l不能转移。

(6)演进规则：有向图的演进是由节点状态的改变与连接弧发生转移这两个动态因素相互作用而完成的，因此，规则包含如下两个方面：

a、对于任意n∈N， 当State(n)=0时，若存在l=(Ñ，n)发生转移，则State(n)=1； 当State(n)=1时，若节点 n执行完毕，则 State(n)=0

b、对于任意l=(n ，n′)∈L，当State(n)从1变为0时，若Trans(l)=1，则

连接弧l发生转移；若 Trans(l)=0，则连接弧l不发生转移，直至下一次 State(n)从l变为0的时候再使用此规则。

1. 节点的类别

有向图中的节点代表了具有如下特征的多种实体：

(1)与企业中实际存在的事件或活动有着直接的对应关系；

(2)本身有着具体的或人为定义的含义；

(3)能与其他节点形成一定的逻辑关系；

因此，区分不同类别的节点、对节点进行具体的类别定义不仅可以明确节点的含义，同时也增强了模型的语义。在这里，我们赋予节点以如下的几种类型定义：活动、子过程、开始与结束标记、同步节点。

一、活动

活动是指在一段不间断的时间间隔内为实现某一目标由人工或自动完成的一个企业行为，是组成业务流程的最基本单元。一个企业的所有活动的集合覆盖了企业中各类业务流程的全部细节。虽然企业中的活动多种多样、千差万别，但是却可以用一个统一的结构化框架来描述它。

（1）输入：活动的输入部分是保证活动开始的物质条件，通常包括企业资源与信息对象。如原始物料、各类电子化文档，包括数据表格、图形文档、文本文档、电子邮件及 Web资源等。

（2）输出：活动的输出部分是活动的结果。活动的输入与输出构成了每一个基本活动单元与外部(其他活动单元)之间的接口，封装了内部具体的任务处理过程，包括角色与约束；而活动输入与输出在内容上的一致也为实现工作流模型的重用提供了保证，通过建立相应的输出→输入映射机制，一个活动单元便可以同多个活动单元进行组合，出现在不同的业务流程当中。

（3）角色：角色是指企业中以一定的技能要求为前提、能够完成某项专职工作的企业人员的集合，它与企业的组织模型紧密相关。活动所需的角色包括执行者与负责人两类，二者在活动中形成了上、下级的关系，下级负责活动的执行，上级则负责监督、检查与异常情况的处理。

（4）约束：活动在执行的过程中总是有一定的约束条件，这也体现了竞争激烈的市场与独立自主的客户对企业所施加的压力。最主要的一个约束就是对活动的时间要求，即活动应该在有限的时间内必须完成。第二个约束是活动的优先级，赋予活动不同的优先级将区别了不同活动对企业的重要程度。优先级

越高的活动在工作流实施运转的过程中将享有更为优先的申请企业资源与人员的权利。除了时间和优先级以外，根据企业本身的特点，还将会有许多其他方面的约束。

在统一的活动描述框架下，我们可以定义多种不同类型的活动。一方面可以方便用户建模，使用户更直观地理解具体活动的含义；另一方面，工作流机针对不同类型的活动，在实施过程中可以做出不同的处理，使系统更灵活、更高效。活动分类的标准有很多，可以根据企业的实际情况来确定。比较基本的一种分类是将活动类型确定为人工型与自动型两种。人工型的活动是通过工作表的生成来通知相关的人员，依靠人员以手工或启动应用的方式来完成；自动型活动则是在工作流机的驱动下直接启动应用或利用自动化设备来完成的活动，例如工作流机自动启动某台计算机上的绘图应用程序并打印一份图纸。这种自动型活动充分体现了工作流管理系统所实现的企业内部不同应用间的过程集成。

二、子过程

作为组成业务流程的最基本单元，活动是不能被进一步分解的。一旦我们的流程比较复杂，涉及的环节比较多，那么活动的数量也将大大增加，有向图中的节点数也必然会不断膨胀。这首先影

响了用户对流程中各主要环节的把握和理解。事实上，我们可以把某些关系紧密的活动集合起来，在图上以一个节点表示，这就形成了子过程的概念。

子过程是一类能够分解的节点类型，它的内部可以包含组成工作流模型的所有元素类型，实质上就是一个子工作流。子过程的引入大大增强了模型的表达能力，使模型具有了层次化的概念，并支持自顶向下的建模过程。我们规定，子过程可以出现在模型的任何层次，即允许子过程内部再次嵌入子过程。通常，用户可以在模型的最顶层全部用于过程来表示，这样即确定了模型的总体逻辑结构，进而再在每个子过程中详细地布置活动及其他模型元素，直到完成最底层的基本活动的建模。

三、开始与结束标记

由于有向图本身是一种非线性的数据模型结构，与线性结构不同的是，它可能具有多个入口节点(即只有连接弧由该节点发出，而没有连接弧指向该节点)，这就给用户与工作流机正确理解流程的逻辑顺序带来了不便，甚至会发生疏漏与错误。因此，我们又人为地定义了一类具有特定含义的标志性节点 开始

标记 *。我们规定，开始标记为一个工作流模型(或子过程)的唯一入口点，它无前趋节点，即Pre(*)=Φ。

对于一个实际的业务流程，可能会由于不同的执行情况而出现不同的结果。对应于有向图，这种情况就表现为一次不能遍历图中的全部节点，只有部分路径被选择执行，图中会出现多个出口节点，它们标志着流程的结束。为了清晰地表达流程的结束状态，并与开始标记相对应，我们同样引入结束标记 *，同时规定，结束标记 *为一个工作流模型(或子过程)的唯一出口点，它无后继节点，即Post(*)=Φ。

四、同步节点

在我们把一个实际的业务流程映射成为工作流模型时，根重要的一点，就是要保证活动间的逻辑关系不变。“与”和“或”是两类最基本的逻辑关系，它是表达各种复杂关系的基础，工作流模型必须具备表达“与”和“或”关系的能力。

我们在前面已经定义了模型的演进规则，对于任意一个处于非执行状态的节点n，只要有一条输入连接弧发生了转移，那么该节点即可被执行，这实际上就表达了“或”的关系，即∪{n′ ∈Pre(n)，l=(n′ ，n)∈L，Trans(l)=1且l发生转移}。

对于“与”的关系，我们通过增加一类新的节点 同步节点S来表达，它对活动起协调、同步的作用。我们规定，同步节点S的动态行为完全遵循演进规则，所不同的是，当S处于执行状态时(State(S)=1)，将判断它的所有输入连接弧是否已经全部发生转移：若是，则S的状态就出1变为0，即S执行完毕；否则，S仍处于执行状态，并将继续判断，直至满足上面的条件后S才执行完毕，State(S)=1→0。这意味着同步节点将使它的所有前趋节点都执行完毕后才继续推进流程，表达了“与”的关系，即∩{n′ ∈Pre(n)，l=(n′ ，n)∈L，Trans(l)=1且l发生转移}。

同步节点的设置不仅区分了有向图中节点的多条输入连接弧之间的“与”、“或”关系，而且可以使用户从图中清楚、直观地理解流程的执行过程，既丰富了模型的语义，也方便了用户建模。

2. 连接弧

作为有向图中的另一类组成元素，连接弧表达了图中不同节点元素之间的逻辑顺序关系。它从前趋节点指向后继节点，体现了节点状态的转移与有向图

的演进。

连接弧发生转移是有条件的，因此我们在前面的定义中为每一条连接弧都绑定了一个二值的布尔型转移函数Trans(l)。对于转移函数的组成，我们有如下定义：

转移函数是由一系列条件经过“与”、“或”组合而成的，其中每一个条件就是一个谓词逻辑，它的结果也是“真”、“假”二值的，这些逻辑的最终组合结果决定了转移函数的取值[18] 。

根据连接弧转移条件的特点，我们把在工作流模型中所应具体表达的连接弧类型分为如下两类：

(1)永真型：即连接弧的转移函数值永远为“真”，Trans(l)＝1。这体现了一种顺序关系，不需要经过任何条件的判断，只要前趋节点执行完毕，即可激活后继节点。

(2)不定型：即转移函数的取值是需要在具体的工作流执行过程当中由工作流机或人加以判断来确定的。这种判断实际上体现了一种选择关系，即根据不同的情况，通过满足条件的连接弧的转移，实现对某一节点的多个后继节点的选择性激活。

从直观和方便的角度出发，我们把“永真型”的连接弧称为“无条件连接弧”，而把“不定型”的连接弧称为“有条件连接弧”。这两种连接弧在描述活动节点间的时序关系与逻辑顺序的同时，也隐含了交换数据的内容，即前趋节点的输出可以做为后继节点的输入。但这种表达方式处理数据流的能力是根有限的，比如，对于那些不具备前趋／后继关系的节点就不能显式地定义。为此，我们引入另外一种连接弧—— “数据连接弧”。这种连接弧可以连接于同一层次中的任意两个有数据交换的节点之间来定义数据流，数据流动的方向就是连接弧箭头所指的方向。

有以上这些定义，我们可以针对各种实际情况建立一种简单、直观，具有较强描述能力的面向企业应用的工作流模型 。

2.2 基于MVC的应用开发框架

通用媒体增值业务管理系统的业务层主要采用MVC模式进行业务处理，其三层结构为：

Model(模型)	对应业务模型,负责具体业务逻辑的实现。主要由 spring + javabean 实现。
View(视图)	对应用户界面，负责与用户的交互。一般由 xwork + velocity+javascript 实现。
Controller(控制)	负责对视图和业务模型对象进行统一的调度和控制，是应用系统处理具体流程和导向的核心部分，它为视图和模型对象之间的通信提供统一的接口，并且处理系统流程的走向。由 webwork 的 Action 实现。

表 2-1 MVC 三层结构

其示意图如下所示：

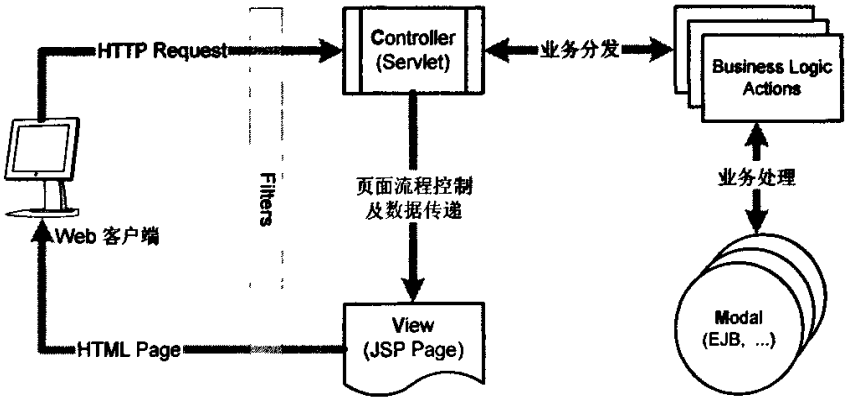


图 2-4 MVC 框架示意图

通过这种设计模式,实现了业务逻辑(商业模型)，界面设计，以及应用系统结构设计之间的独立性，保证了应用系统各个组成部分的灵活性和可扩展性,并在此基础上实现了统一的权限，区域控制和页面流程控制。

在开发阶段,MVC 模式定义了各类开发人员的任务和实现规范，并通过内置的调试支持机制，使用户界面和业务逻辑的设计，编码和测试能够独立和并行进行. 例如，基于同样一份系统设计，页面设计人员可以在业务逻辑尚未实现时就进行页面设计和测试。

MVC 框架充分利用了 J2EE 平台的各种功能，通过 Filter, Event Listener,以及自定义的业务模型接口，实现了模块化和可插入式组件系统，为应用系统的维护，修改和扩展留下了空间. 例如，当某项业务需求改变时，系统维护人员可以通过配置工具为其指定符合需求的业务处理程序，而不需要修改页面代码。

2.3 基于 Hibernate 的 ORM 技术[8][13]

通用媒体增值业务管理系统采用了 Hibernate 的 ORM 技术，Hibernate 是一种实现对象和关系之间映射（Object Relation Mapping）的框架。它对 JDBC 进行了轻量级的对象封装[9]，使得 Java 程序员可以使用面向对象编程思维来操作关系数据库。在多层结构的应用中，业务层和数据层之间存在一个持久层，它负责应用到数据库的数据存储，数据的检索和更新。持久层的实现技术包括 JDBC、实体 Beans、JDO 以及 Hibernate 等，但是 JDBC 中数据访问对象和 SQL 语句直接绑定在一起降低了可维护性，且不支持继承和多态。EJB 不支持继承和多态而且还需要额外的 EJB 容器。相比之下 Hibernate 则是一个非常好的选择，目前的应用系统大多使用关系数据库，在做设计和开发时又是面向对象的方式，这时就可以选用 Hibernate 来实现对象、关系之间的映射和数据的持久化。而且 Hibernate 拥有一种功能非常强大的查询语言（HQL），这种语言与 SQL 非常相似，便于掌握[10]。

在应用 Hiberante 框架时，首先编写 O/R 映射描述文件，完成对象、关系数据库之间的映射。持久对象可以根据映射文件生成，然后编写业务逻辑类。这些 JavaBean 实现了具体的业务逻辑，也封装了对 Hibernate 的访问。Hiberante 利用数据库以及其他一些配置文件如 hibernate.properties，XML Mapping 等，为应用程序提供数据持久服务。

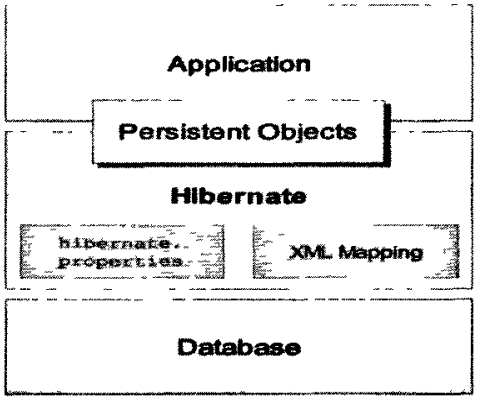


图 2-5 Hibernate 所处位置

图 2-5 展示了 Hibernate 使用数据库和配置文件数据来为应用程序提供持久化服务（和持久化的对象）

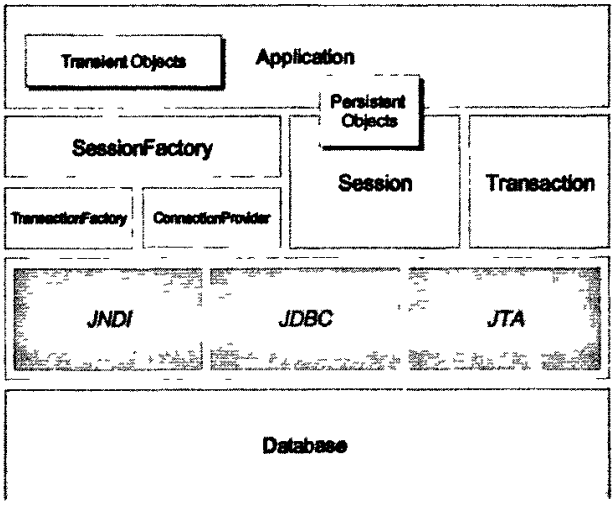


图 2-6 Hibernate 体系结构

让我们更细致地观察一下运行时的体系结构。 挺不幸的，Hibernate 是比较复杂的，提供了好几种不同的运行方式。我们展示一下两种极端情况。轻型体系中，应用程序自己提供 JDBC 连接，并且自行管理事务。这种方式使用了 Hibernate API 的一个最小子集。全面解决体系中，对于应用程序来说，所有的底层 JDBC/JTA API 都被抽象了，Hibernate 会替你照管所有的细节[12]。

SessionFactory：对属于单一数据库的编译过的映射文件的一个线程安全的，不可变的缓存快照。它是 Session 的工厂，是 ConnectionProvider 的客户。可能持有一个可选的（第二级）数据缓存，可以在进程级别或集群级别保存可以在事物中重用的数据。 可能持有事务之间重用的数据的缓存。

会话，Session：单线程，生命期短促的对象，代表应用程序和持久化层之间的一次对话。封装了一个 JDBC 连接。也是 Transaction 的工厂。保存有必需的（第一级）持久化对象的缓存，用于遍历对象图，或者通过标识符查找对象。持有持久化对象的缓存。

持久化对象（Persistent Object)及其集合(Collection)：生命期短促的单线程的对象，包含了持久化状态和商业功能。它们可能是普通的 JavaBeans/POJOs，唯一特别的是他们现在从属于且仅从属于一个 Session。一旦 Session 被关闭，他们都将从 Session 中取消联系，可以在任何程序层自由使用（比如，直接作为传送到表现层的 DTO,数据传输对象）。

临时对象(Transient Object)及其集合（Collection）：目前没有从属于

一个 Session 的持久化类的实例。他们可能是刚刚被程序实例化，还没有来得及被持久化，或者是被一个已经关闭的 Session 所实例化的。

事务： 单线程，生命期短促的对象，应用程序用它来表示一批工作的原子操作。是底层的 JDBC, JTA 或者 CORBA 事务的抽象。一个 Session 某些情况下可能跨越多个 Transaction 事务。

ConnectionProvider：JDBC 连接的工厂和池。从底层的 Datasource 或者 DriverManager 抽象而来。对应用程序不可见，但可以被开发者扩展/实现[11]。

TransactionFactory：事务实例的工厂。对应用程序不可见，但可以被开发者扩展/实现。

2.4 基于 webwork 的表示层技术[14][17]

WebWork 是由 OpenSymphony 组织开发的，致力于组件化和代码重用的拉出式 MVC 模式 J2EE Web 框架。WebWork 目前最新版本是 2.1，现在的 WebWork2.x 前身是 Rickard Oberg 开发的 WebWork，但现在 WebWork 已经被拆分成了 Xwork1 和 WebWork2 两个项目。 Xwork 简洁、灵活功能强大，它是一个标准的 Command 模式实现，并且完全从 web 层脱离出来。 Xwork 提供了很多核心功能：前端拦截机(interceptor)，运行时表单属性验证，类型转换，强大的表达式语言(OGNL - the Object Graph Notation Language)，IoC (Inversion of Control 倒置控制)容器等。WebWork2 建立在 Xwork 之上，处理 HTTP 的响应和请求。WebWork2 使用 ServletDispatcher 将 HTTP 请求的变成 Action(业务层 Action 类)，session (会话) application (应用程序) 范围的映射，request 请求参数映射。WebWork2 支持多视图表示，视图部分可以使用 JSP, Velocity, FreeMarker, JasperReports, XML 等。在 WebWork2.2 中添加了对 AJAX 的支持，这支持是构建在 DWR 与 Dojo 这两个框架的基础之上。

Web 应用程序的设计开发是复杂并且费时的。然而，你能够通过运用一种框架处理常见的 Web 应用程序来简化开发流程。许多开源 Web 应用框架能够做到这一点甚至更好一些。这些开发框架中最好的一个就是 WebWork，是开源项目中 OpenSymphony 组的一个 Web 应用开发框架。

WebWork 的最大优点是它的简单性和灵活性。WebWork 有一个很小的 API，它使开发者可以迅速进行开发工作。WebWork 是许多特性和适用性的组合，包括

使用variour view技术，例如JavaServer Pages(JSP),Velocity,Extensible Stylesheet Language Transformations Specification(XSLT)和JasperReporters。WebWork拥有一个活跃的社区，有许多文章、开发者和用户。

WebWork是建立在称为XWork的Command模式框架之上的强大的基于Web的MVC框架。

WebWork2的特性包括

1. 灵活的Validation框架，允许你在XML文件中定义验证内容，在运行时通过Interceptor自动应用，因此完全脱离Action类。新版支持客户端验证。

2. Type conversion 允许你在类之间很容易转换对象。

3. OGNL(Object Graphical Navigation Language)表达式语言： 允许动态对象图表遍历和方法执行，使用ValueStack透明访问多Beans的属性。WW2也具有使用JSTL的能力。

4. IoC(Inversion of Control)容器，管理组件的生命周期，使客户获得组件实例不需要创建注册类（与容器环境无关）。

5. Velocity Templates，使UI组件可重用，从而允许开发者容易定制Web页面的look & feel[18]。

6. Interceptors，在Actions处理的前后动态拦截，以简单化Action代码，增加减少代码的机会。

7. 支持I18N。

8. 容易和第三方软件集成，包括Hibernate, Spring, Pico, Sitemesh等。

9. 支持多种视图技术，如JSP, Velocity, FreeMarker, JasperReports, XML等。

10. 支持Packages和Namespaces，来管理Actions。

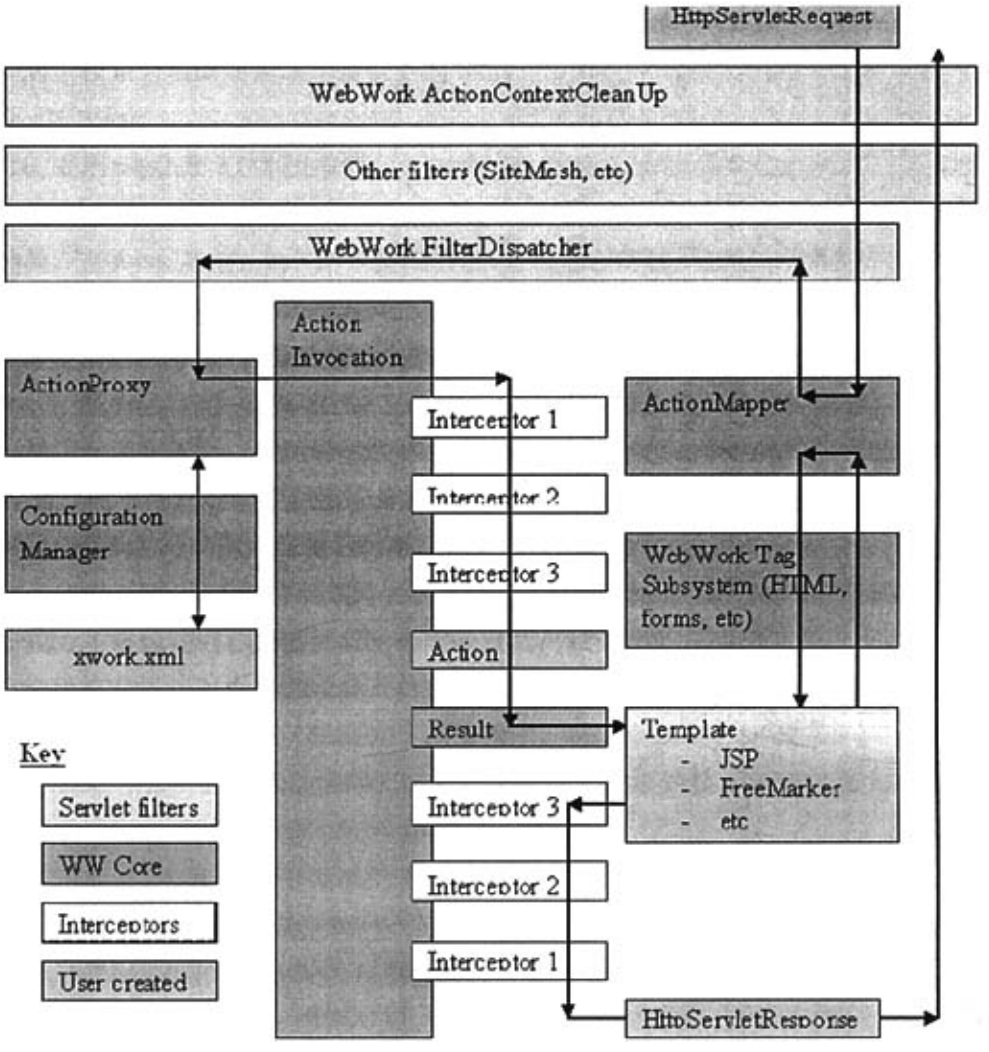


图 2-7 webwork 体系结构

一个初始的请求被发送到 servlet 容器，如果 ActionMapper 决定应该调用一个 action，FilterDispatcher 就把请求委托给 ActionProxy，ActionProxy 通过 WebWork 的配置管理器读取 xwork.xml 文件里的配置信息。然后创建一个实现了命令模式的 ActionInvocation，这一过程包括在调用 action 本身之前调用所有的 interceptor，一旦 action 方法返回，actionInvocation 就要查找 xwork.xml 文件中这个 action 的结果码所对应的 result，通常情况下 result 会调用 JSP 或 velocity、freemarker 等模版来呈现页面结果给用户。当呈现页

面时，可以使用webwork提供的一些标签，其中一些组件可以和ActionMapper一起工作来为后面的请求呈现恰当的URL。

2.5 采用web2.0 AJAX技术[19]

术语Ajax用来描述一组技术，它使浏览器可以为用户提供更为自然的浏览体验。在Ajax之前，Web站点强制用户进入提交/等待/重新显示范例，用户的动作总是与服务器的“思考时间”同步。Ajax提供与服务器异步通信的能力，从而使用户从请求/响应的循环中解脱出来。借助于Ajax，可以在用户单击按钮时，使用JavaScript和DHTML立即更新UI，并向服务器发出异步请求，以执行更新或查询数据库。当请求返回时，就可以使用JavaScript和CSS来相应地更新UI，而不是刷新整个页面。最重要的是，用户甚至不知道浏览器正在与服务器通信：Web站点看起来是即时响应的。

虽然Ajax所需的基础架构已经出现了一段时间，但直到最近异步请求的真正威力才得到利用。能够拥有一个响应极其灵敏的Web站点确实激动人心，因为它最终允许开发人员和设计人员使用标准的HTML/CSS/JavaScript堆栈创建“桌面风格的（desktop-like）”应用程序。

Ajax不是一种技术。实际上，它由几种蓬勃发展的技术以新的强大方式组合而成，Ajax包含：

- 基于XHTML和CSS标准的表示。
- 使用Document Object Model进行动态显示和交互。
- 使用XMLHttpRequest与服务器进行异步通信。
- 使用JavaScript绑定一切。

Ajax的工作原理

Ajax的核心是JavaScript对象XmlHttpRequest。该对象在Internet Explorer 5中首次引入，它是一种支持异步请求的技术。简而言之，XmlHttpRequest使您可以使用JavaScript向服务器提出请求并处理响应，而不阻塞用户。

在创建Web站点时，在客户端执行屏幕更新为用户提供了很大的灵活性。下面是使用Ajax可以完成的功能：

- 动态更新数据，无需用户单击刷新并等待服务器重新发送整个页面。

- 提升站点的性能，这是通过减少从服务器下载的数据量而实现的。例如，每次刷新直播页面时，需要下载整个直播页面，包括框架页面等，但用了 ajax 技术以后，只要下载更新直播内的短信和彩信数据即可！消除了每次用户输入时的页面刷新。
- 直接编辑表格数据，而不是要求用户导航到新的页面来编辑数据。对于 Ajax，当用户单击 Edit 时，可以将静态表格刷新为内容可编辑的表格。用户单击 Done 之后，就可以发出一个 Ajax 请求来更新服务器，并刷新表格，使其包含静态、只读的数据。

在通用媒体增值业务管理系统当中部分采用了 AJAX 技术，使用户有了桌面应用程序的体验。

第 3 章 系统需求分析

3.1 通用媒体增值业务管理系统目标和功能

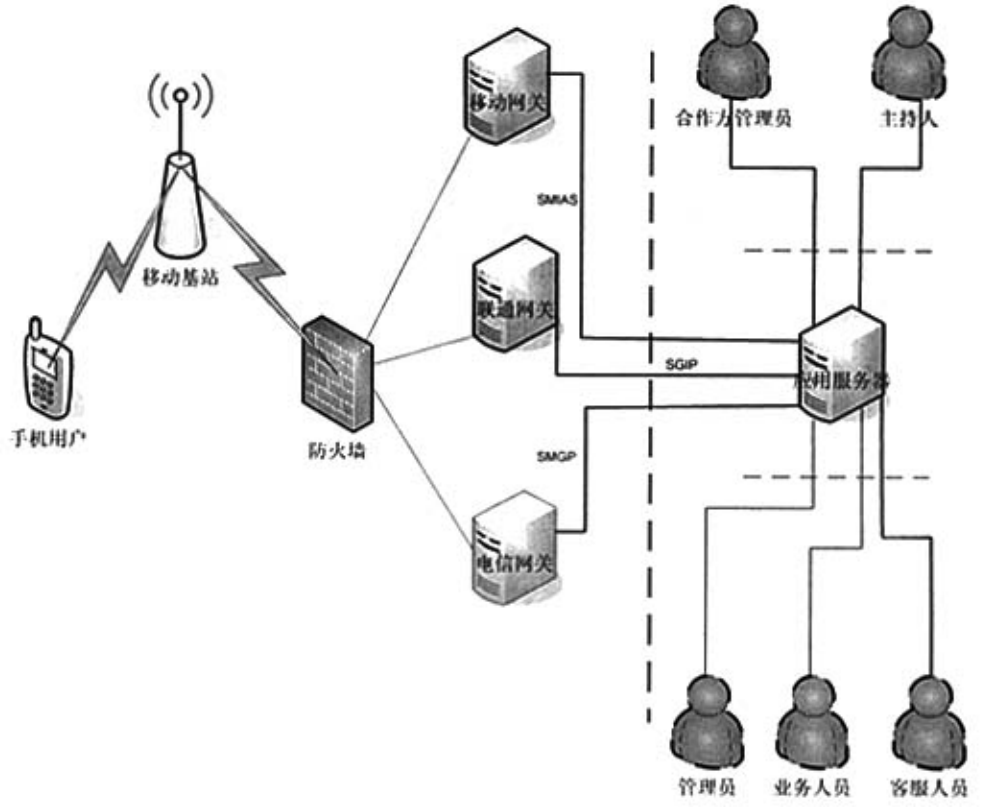


图 3-1 应用场景

通用媒体增值业务管理系统的目标是研究和设计一套适合媒体使用的、集成短信、彩信等多种增值业务类型的管理系统，其目标包括：

(1)遵循国家信息产业部以及移动、联通、电信等运营商的规划，促进增值业务的发展和市场规范[5]。

(2)使电台、电视台、酒吧、学校等行业和机构都能方便快捷的操作和使用增值业务管理系统，开展自己的业务。

(3)促进用户和媒体之间的交流。

(4)方便按不同角色进行管理，方便信息检索。

通用媒体增值业务管理系统的功能见图3-1所示，主要是提供信息检索和后台管理的一个通用的移动增值业务管理系统，该系统的功能主要包括：

(1)系统内部信息实时发布，并可以按用户进行选择性发布。

(2)用户分不同角色提供不同的功能，即可分配不同的模块。

(3) 提供一个直播平台，主持人可以方便查看用户的短信和彩信[7]，并可以按时间、按手机号码、按短信内容、等各种条件进行信息检索。

(4)按短信票数的统计功能，分组统计。

(5)抽奖功能，主持人可以按一定条件进行抽奖。

(6)点播功能，点播包括文字点播、彩信点播、交友点播以及wap push点播。

(7)系统扩展升级必须非常方便。

(8)方便开发插件。

3.2 系统用例概况

3.2.1 系统角色

1. 系统管理员。系统管理员主要负责管理整个系统，包括用户管理，日志管理，新闻管理以及权限管理等等。
2. 业务人员。业务人员负责管理属于自己业务范围内的主持人用户，因此需要管理主持人用户以及这些用户所有模块的配置，并回复这些主持人用户的意见和建议。
3. 合作方管理员。合作方管理员主要是负责管理属于自己的用户，比如某个电台的所有主持人用户，那么这个合作方管理员可以是电台的台长。合作方管理员主要是需要查看帐单，因此只考虑帐单浏览功能。
4. 主持人用户。该系统使用最多的是主持人用户，主持人用户可以操作直播、交友、统计、抽奖、答题、点播、帐单统计等等模块。

3.2.2 系统总用例

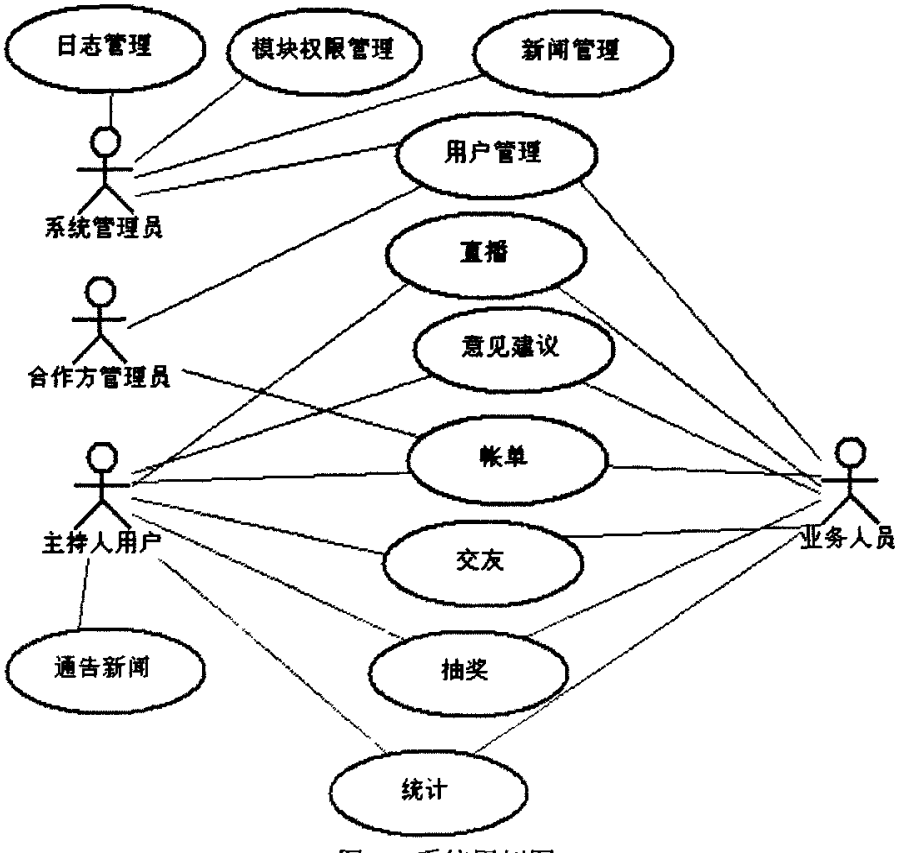


图 3-2 系统用例图

用例图 3-2 描述了系统管理员用户、合作方管理员用户、业务人员以及主持人用户的各个用例。以下是各个用例的简单描述：

1. 日志管理：主要是整个系统的日志管理，包括查看用户的登录日志以及各种操作日志，以及系统消息队列数量等等，通过查看这些日志，可以了解系统的运行状况，确保系统安全稳定运行。

2. 模块权限管理：主要是划分、添加、删除、修改系统的各个权限组（比如直播模块、统计模块、抽奖模块等等），当划分好模块权限以后，可以设置某用户是否具有该模块的操作权限。

3. 新闻管理：主要是管理系统的公告、新闻等信息，发布一些全体或者是个体的新闻类信息。

4. 用户管理：系统管理员可以管理系统当中的所有用户，包括添加、删除、修改用户等操作，可以给用户分配适合的权限和模块，如果是合作方或者是业务人员可以分配该用户的受管用户（即哪些主持人用户受管）；合作方管理员

只能管理属于自己管理的主持人用户。

5. 意见建议：业务人员有权限操作主持人的意见和建议，给予适当的回复信息，而主持人主要是提出意见和建议。

6. 交友:业务人员可以对属于自己管理的主持人用户的交友模块进行管理，比如分配交友号码等等，而主持人用户则可以浏览交友用户等信息。

7. 帐单管理：业务人员可以配置属于自己的主持人用户的帐单模块，包括结算比率等等，而合作方或主持人用户可以浏览自己的帐单统计，包括收发数量，根据结算比率查看自己的每天以及每个月的收入情况。

8. 直播：业务人员可以配置某主持人用户的直播模块，主持人则可以按时间查看或搜索短信和彩信的接收记录。

10. 交友：业务人员可以配置主持人的交友模块，交友模块主要提供一个平台让主持人可以和用户进行交流，提供一个用户交友的平台，比如主持人可以根据交友用户的信息在节目当中进行播报。

11. 统计：主持人有时可能会开展一些投票活动类的节目，比如唱歌比赛，用户则可以通过编号给选手投票，主持人则可以随时查看统计结果。

12. 抽奖：主要是提供一个抽奖平台，按要求抽出中奖用户，并可下发中奖通知短信。

13. 新闻通告：该用例主要是下发一些通知（公开或只针对某主持人）。

3.3 主要用例分析

3.3.1 直播用例

直播用例主要包括 MO 消息处理子用例、直播管理子用例以及直播浏览子用例。

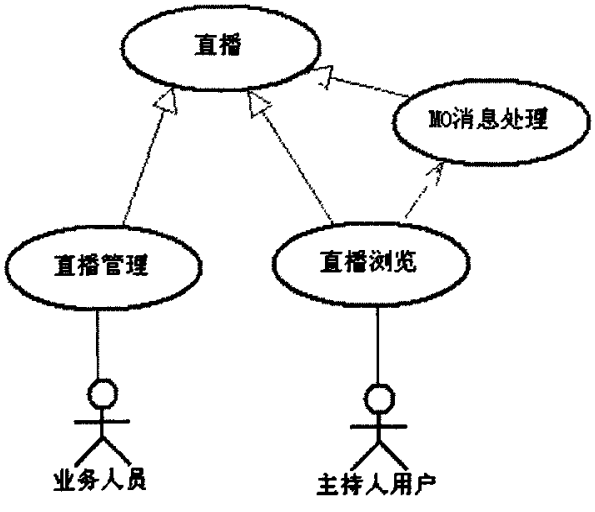


图 3-3 直播子用例图

直播用例可以泛化成直播管理、直播浏览以及MO消息处理三个子用例，同时直播浏览子用例又依赖于MO消息处理子用例。

3.3.1.1 MO消息处理子用例

MO消息处理是系统接收到短信或彩信以后经过一系列处理写入直播表的整个过程，其事件流程如图3-4所示：

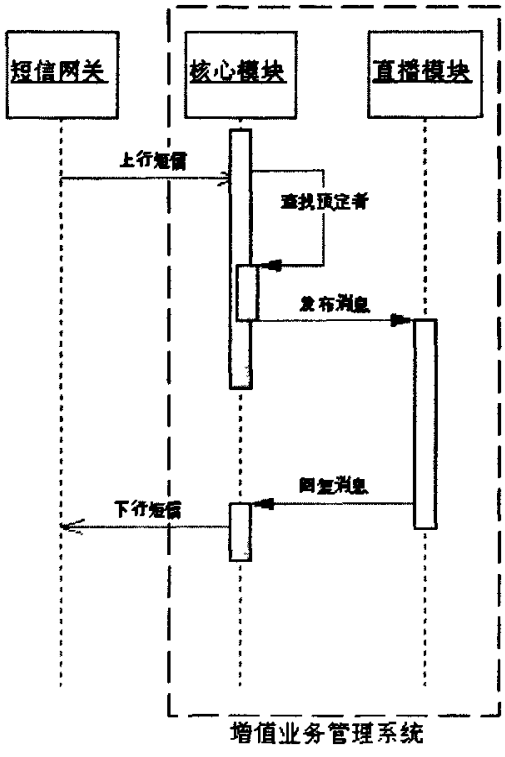


图3-4 MO消息处理时序图

- 当网关接收到一条MO以后，路由给媒体增值业务管理系统。
- 根据消息预定发布模式把消息路由给直播模块。
- 直播模块完成对MO的业务处理，最终写入消息直播接收表，然后读取移动、联通小灵通的回复内容（由管理员事先配置），自动给用户回复一条消息。

3.3.1.2 直播管理子用例

- 管理员可以配置直播的移动联通和小灵通的各种回复语，移动、联通和小灵通必须要有不同的回复内容配置。
- 可以配置显示过滤器，对直播内容进行过滤，这样主持人浏览直播内容

的时候可以过滤掉一些不必要的数据。

3.3.1.3 直播浏览子用例

主持人用户进入直播页面以后，自动按时间倒序显示当天的所有短信和彩信记录。

- 如果在节目当中阅读了某条短信，可以单击短信做阅读标记，阅读标记为在文字上划一条删除线。
- 如果需要删除某条短信或彩信记录，则可以单击该条信息后的删除连接。
- 如果需要回复用户短信或彩信，则可以单击某手机号码后的回复连接，单击回复以后，将弹出一个回复对话框页面，输入需要回复的内容，其内容可以在收藏夹或者在常用语里选择。
- 在直播页面主要的搜索功能，可以选择某个连续的时间段，也可以选择某个时间段内的片段时间（比如搜索这个月的 1 号到 15 号的每天下午 13：00 到 14：00 的短信），搜索也可以输入手机号码以及内容。
- 对搜索出来的直播记录可以导出来，生成 EXCEL 文件。

3.3.2 统计用例

统计用例主要包括 MO 消息处理子用例、统计管理子用例和统计浏览子用例，子用例图如下：

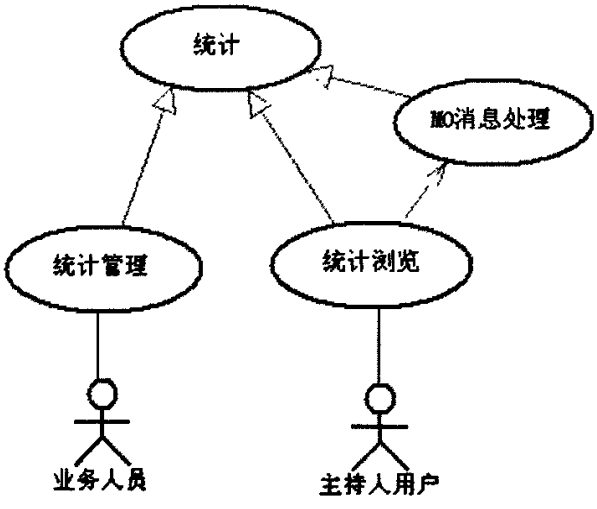


图 3-5 统计子用例图

3.3.2.1 MO消息处理子用例

当网关接收到短信以后，会经过一系列的处理最终写入统计表其具体流程大致如下：

- 当网关接收到短信以后，首先提交给媒体增值业务管理系统的核心处理模块；
- 媒体增值业务管理系统核心模块接收到短信以后，会根据消息发布预定模式将消息路由给统计模块；
- 统计模块根据配置的正则表达式决定该消息是否属于自己能处理的范畴，如果是统计模块需要处理的消息，则查找对应短信内容的统计项，如果能找到，则该统计项票数加 1，否则读取管理员事先配置的错误回复内容，自动给用户回复该错误信息，提醒用户如何参与，如果不是统计模块需要处理的短信则丢弃该短信或彩信，并返回 false。

统计模块分统计组和统计项，统计项是指各个统计编号，把各个统计编号归成一组，形成一个统计组，每个统计组可以有不同的回复语，统计项的匹配规则按长优先的原则模糊匹配，比如有一个统计项为"12" 和一个统计项为"122"，如果用户发送短信"122 你好",则应该匹配到 122 这个统计项。另外有些时候需要对票数进行作弊，因此必须提供作弊功能，可以手工增加票数。

3.3.2.2 统计管理子用例

统计管理主要提供业务人员对统计的管理功能，可以根据主持人节目需求进行有效的统计配置，以便开展一些投票类的业务，业务人员可以进行的操作大概有：

- 可以配置统计模块正则表达式，只有符合正则表达式的短信才能进入统计模块，可以配置当匹配不到统计项的时候的回复语。
- 可以添加、删除、修改统计组，并设置统计组的未开始统计、正在统计和统计结束时的各个回复语。统计组的类别包括每天统计、每月统计和某个开始时间到某个结束时间的统计三个类型。统计组的投票限制分每天限制、总体限制和每天总体都要限制。
- 可以针对某统计组添加、删除和修改统计项，统计项的个数不受限制，禁止统计内容一样的统计项。
- 可以添加特殊票数，主要是作弊时用。

3.3.2.3 统计浏览子用例

主持人可以浏览查询统计结果，主要操作有：

- 可以按时间段进行票数统计。
- 可以查看某统计项的详细接收记录。
- 并可以针对某个手机号码进行单独的短信或彩信回复。
- 统计某时间段内容，发送短信或彩信数量最多的前几名。
- 可以按手机号码、短信内容、时间进行搜索统计模块里的数据。

3.3.3 抽奖用例

抽奖主要分web方式直接抽奖以及windows桌面程序方式，可以对直播数据、某统计组或统计项进行抽奖。有时为了节目需要，可以随时掺入作弊号码。对已经产生的抽奖结果，主持人可以随时查询。对中奖的手机号码，可以选择是否立即发送中奖通知。因此抽奖用例分抽奖管理子用例、web方式抽奖子用例以及桌面应用程序抽奖子用例，用例图如图3-6所示。

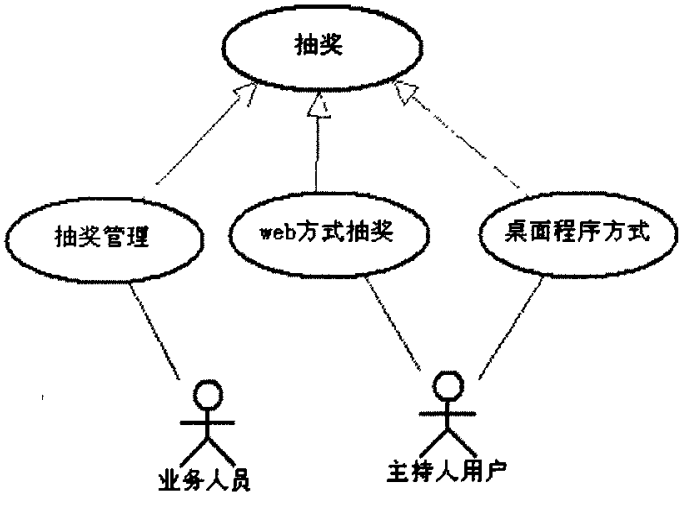


图3-6 抽奖子用例图

3.3.3.1 抽奖管理子用例

业务人员可以对抽奖进行必要的配置管理，具体操作有：

- 业务人员进入抽奖模块管理页面，可以设置中奖回复语。
- 可以设置是否需要作弊，并设置作弊条件，最多作弊次数，和每天最多作弊次数。

3.3.3.2 web方式抽奖子用例

主持人可以直接通过web方式，主持人进入抽奖页面以后，可以选择某个时间段内的数据进行抽奖，可以选择对直播所有的手机号码进行抽奖，可以针对某个统计项的手机号码进行抽奖。

3.3.3.2 桌面程序方式抽奖子用例

主持人也可以下载桌面应用程序，通过桌面应用程序来进行抽奖，抽奖数据通过http方式下载，这种抽奖方式适合现场活动或电视直播。

3.3.4 问答用例

问答用例主要包括MO信息处理子用例、问答管理子用例和问答浏览子用例，子用例图如图3-7所示。

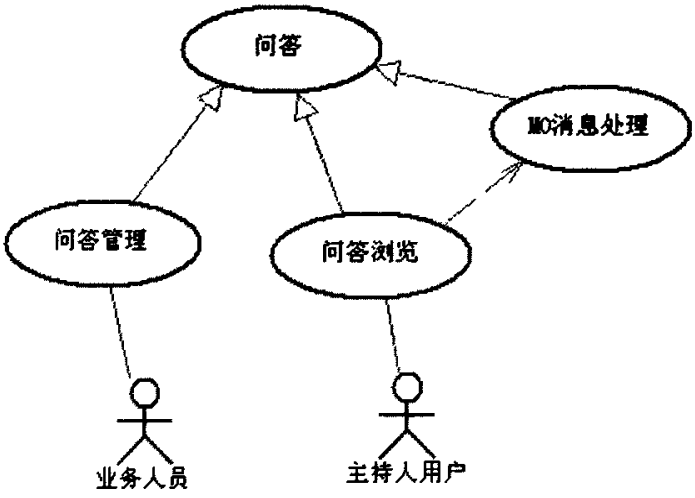


图3-7 问答子用例图

3.3.4.1 MO消息处理子用例

跟直播和统计一样，网关接收到消息以后，首先提交给媒体增值业务管理系统的核心模块，由核心模块根据发布预定模式把消息转发给问答模块，具体时序图如图3-8所示：

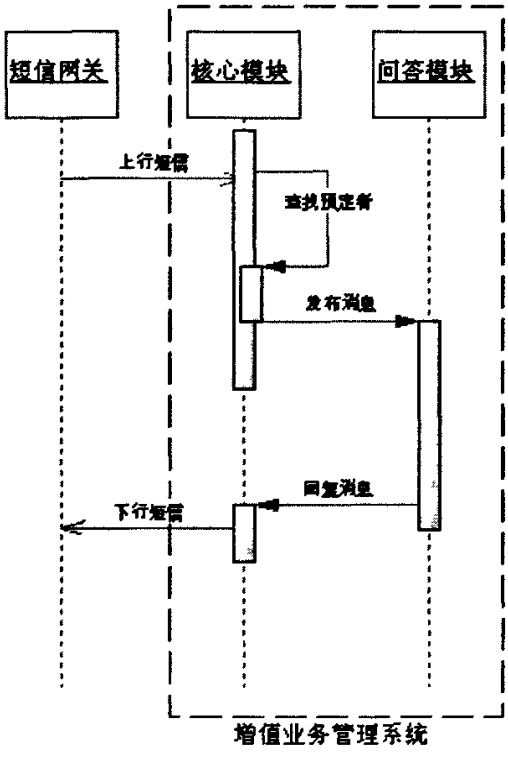


图 3-8 问答时序图

当系统接收到 MO 信息以后，将路由给问答模块，问答模块则根据事先配置好的正则表达式决定是否处理该 MO 消息，如果不是问答模块处理的 MO 消息则丢弃不处理，如果是问答模块需要处理的 MO 消息，则进入问答模块业务处理，并根据配置信息回复用户，其流程如图 3-9 所示：

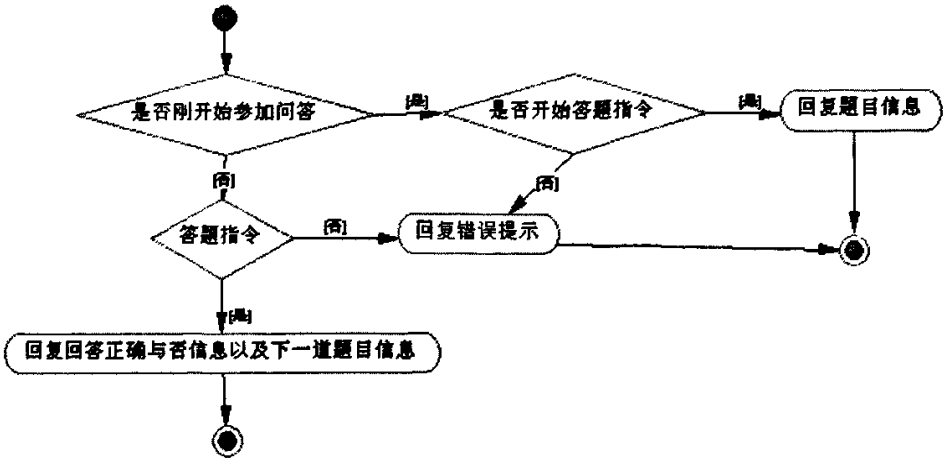


图 3-9 答题活动图

3.3.4.2 问答管理子用例

问答主要用于电视台答题节目，业务人员可以配置答题的指令，答题的题目，用户发送答题指令后，系统需要判断答题的正确与否。

3.3.4.3 问答浏览子用例

主持人用户可以查看有多少用户参与答题，以及每个用户答对和错的数量以及当前积分。

3.3.5 交友用例

交友用例主要包括 MO 消息处理子用例、交友管理子用例以及交友浏览子用例，跟直播、统计和问答的划分类似。

3.3.5.1 MO 消息处理子用例

当系统接收到 MO 消息以后，经过系统核心模块的处理然后路由给交友模块，由交友模块完成整个 MO 的处理流程，交友主要使用在电台或酒吧，主要实现线下的手机用户可以彼此匿名聊天，交友的流程大致如图 3-10 所示：

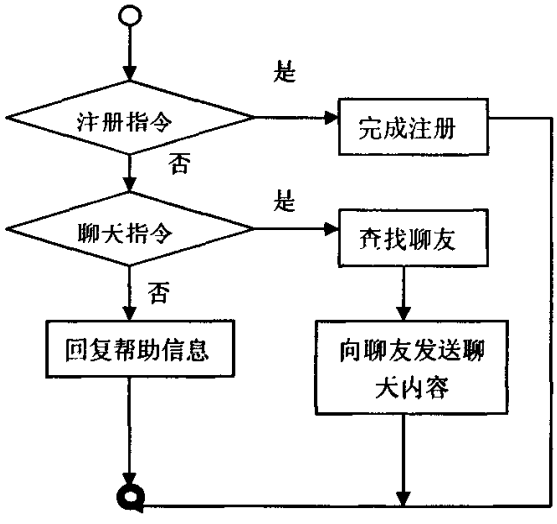


图 3-10 交友流程图

交友的所有指令见表 3-1：

操作	指令	发送号码	指令说明

定制交友	T	066880	成功定制后进入“缘来是你”手机短信聊天交友
搜索在线聊友	M	0668802	接收到在线聊友的爱情观点，直接回复信息可以开始聊天
游戏主菜单	V	0668802	查看社区主菜单
进入聊天室	L	0668802	查找网恋情真等不同主题的聊天室的聊友
都市情	D	0668802	搜索不同地区的聊友
查找地区聊友	C+区号	0668802	寻找指定地区的聊友
方言情	F	0668802	找寻有共同语言的聊天对象
同城同乡缘	Z	0668802	输入自己的家乡区号，系统会为您速配同乡聊友
激情论坛	J	0668802	可查看别人留下的贴子，也可以自己发贴子或者回贴子
同城约会	T	0668802	可以发布约会，取消约会或者响应别的聊友邀请的约会
梦想成真	X	0668802	参与不同项目的游戏，让你梦想成真
个人资料	I	0668802	修改个人资料，查看自己的魅力指数
查看开心宝典	K	0668802	查看幽默短语、甜言蜜语、俏皮情话、妙语如珠等多种多样的短信
修改性别	XB	0668802	修改自己的性别。
修改昵称	NC	0668802	修改自己的昵称
修改爱情观点	ZZ	0668802	修改自己的爱情观点
查询帮助	H	0668802	获取帮助信息

表 3-1 交友指令表

3.3.5.2 交友管理子用例

交友管理主要是提供业务人员方面为主持人用户管理配置交友模块，比如分配交友特服号，配置交友指令等。

3.3.5.3 交友浏览子用例

主持人可以查看当前参与该节目的交友用户，浏览交友用户的详细交友信息，并可向听众推荐交友用户。

3.3.6 帐单用例

帐单用例主要包括帐单管理、帐单浏览、帐单统计三个子用例，帐单管理主

要由业务人员操作完成，帐单浏览则可以提供给合作方管理员和主持人用户查看对帐，子用例图如图3-11所示：

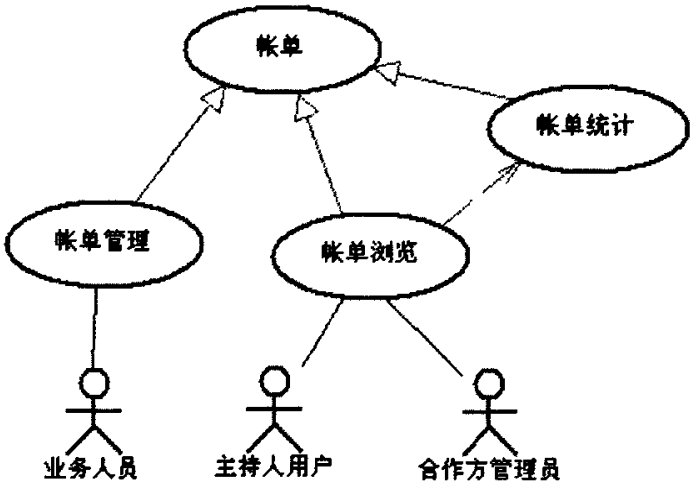


图3-11 帐单子用例图

3.3.6.1 帐单管理子用例

业务人员对帐单的管理主要是为主持人或合作方管理员配置帐单模块，主要操作是配置某主持人用户的结算比率和坏帐率等。

3.3.6.2 帐单浏览子用例

合作方管理员和主持人用户对帐单的浏览主要有：

- 必须按每天以及按每月显示短信量，且可以按移动、联通和小灵通分类，包月的要显示包月数量；
- 此外根据每个主持人用户的结算比率统计出帐单结果。
- 如果是合作方管理员，那么要统计出该管理员用户下面所有受管用户的数量统计和费用统计，可以按每天以及按每月查询。

3.3.6.3 帐单统计子用例

帐单统计主要是每天根据昨天的上行MO数量以及下行MT数量进行统计，并依照结算比率计算出帐单结果提供合作方管理员和主持人用户查看。

3.3.7 点播用例

点播用例主要包括交友点播子用例、文字点播子用例、彩信点播子用例、

WAP-PUASH 点播子用例、点播管理子用例、MO 消息处理子用例，如图 3-12 所示：

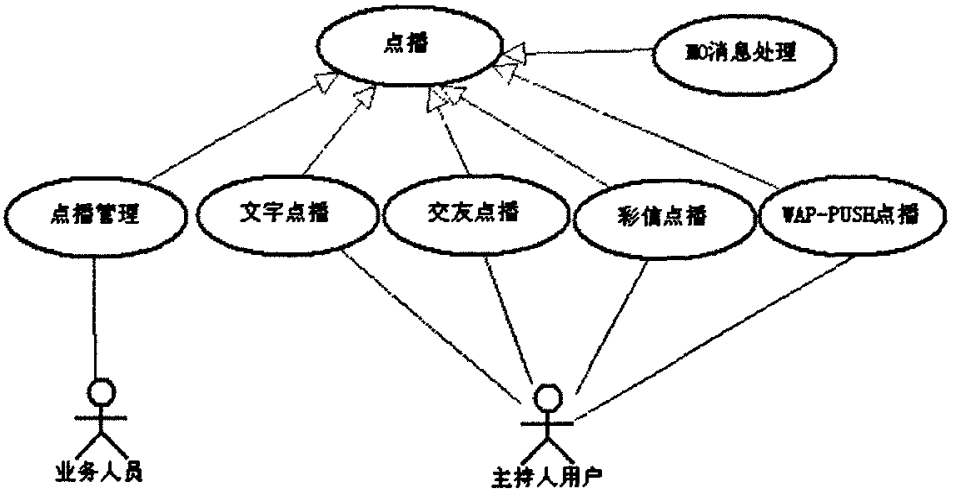


图 3-12 点播子用例图

3.3.7.1 点播管理子用例

业务人员进入某主持人用户的点播管理页面，则可以配置主持人用户对应的点播模块，设置点播模块能处理的各种指令。点播管理主要是业务人员负责替主持人事先配置好各个点播指令等内容，主持人自己比较难以制作的某些点播内容业务人员可以替他们制作。

3.3.7.2 文字点播子用例

主持人如果在直播里看到某条短信比较经典，希望让更多的听众一起来分享，以往是主持人一条条转发给听众，现在主持人只要把该条短信加入到点播模块，报一个编号即可让更多正在收听节目的朋友来索取该条短信！主持人进入点播页面以后，将显示交友点播、文字点播、彩信或 WAPPUSH 点播内容，并显示用户点播的次数，主持人也可以自己添加新的点播内容。

3.3.7.3 交友点播子用例

主持人如果在直播里面看到某个听众希望交友，那么可以将该听众的号码加入到点播模块，让正在收听的听众发送某个指令来索取该号码，主持人报交友的编号即可！

3.3.7.4 彩信点播子用例

我们会根据电台和主持人的需要，把电台的一些资源做成彩信的形式，让听众来下载，比如电台主持人的照片，宣传动画，宣传语，节目介绍等等信息，

更好的实现和听众的交流。

3.3.7.5 WAP-PUSH 点播子用例

由于彩信目前浙江移动支持最大的大小为 100k，因此，如果电台提供的多媒体文件过大，可以采用 WAP PUSH 的方式让听众来下载（大小不受限制）。

3.3.7.6 MO 消息处理子用例

点播模块 MO 处理流程：

- 当系统核心模块接收到 MO 消息以后，核心模块把该消息路由给点播模块。
- 点播模块根据事先配置好的正则表达式决定是否处理该 MO 消息，如果不是该模块处理的信息，则丢弃不处理，如果是该点播模块处理的短信，则进如点播模块业务处理，一般情况下，点播指令为数字，因此当匹配到某个数字编号以后，则回复用户该数字编号对应的信息（可以是交友的内容、文字内容、彩信或 WAP PUSH）。
- 点播模块完成 MO 处理以后，回复消息给用户，从而完成整个 MO 处理流程。

3.4 非功能性要求

3.4.1通用媒体增值业务管理系统的系统要求

通用媒体增值业务管理系统通用构建了全新媒体增值业务管理环境，通过强大的信息检索和交互功能为媒体和用户消费者之间提供了一个良好的互动的交流平台。

1、良好的用户交互功能。部分采用ajax技术，实现用户良好的web交互操作。

2、直观的内容搜索查询，使通过该系统检索内容更加容易和迅速。

3、成熟的内容时效功能。确保最新的、有用的信息才会出现在该系统上。

5、高效的保密机制。确保整个网站和信息的完整性。

6、完整的网站管理。提供权限的管理配置功能，管理员可以快速配置和开展新的业务。

3.4.2 通用媒体增值业务管理系统的软件要求

1、软件上必须确保系统安全稳定可靠，网站响应能力必须达到每秒 50 个请

求。

2、网站开发当中使用了很多新的技术，必须保证技术之间的兼容性和稳定性，避免出现许多技术运用不合理导致的系统不够稳定。

第 4 章 系统设计与实现

4.1 系统结构

4.1.1 网络拓扑

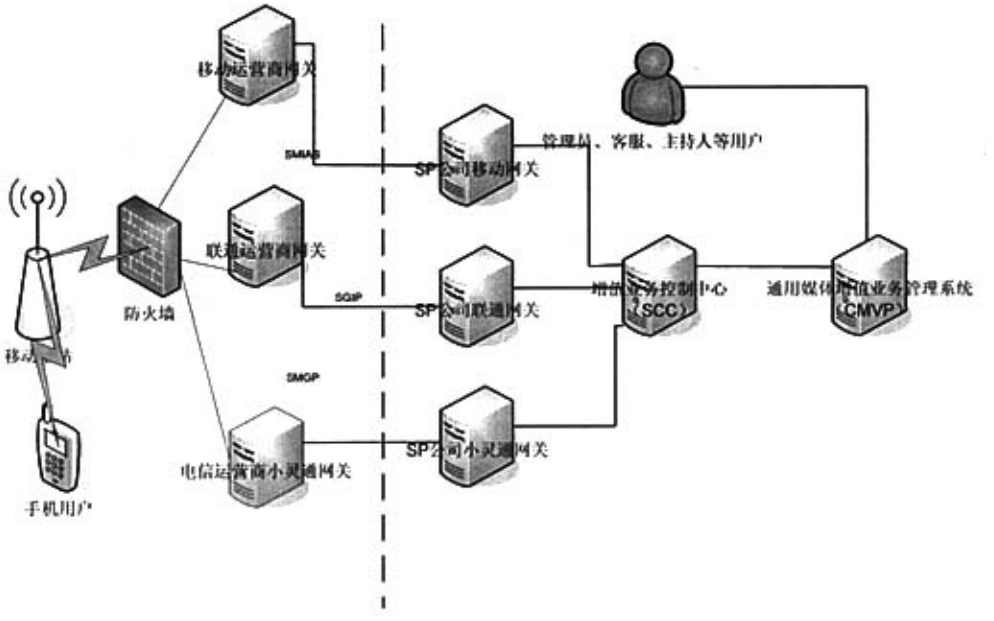


图 4-1 网络拓扑结构图

MO 流程：网络拓扑结构如图 4-1，首先手机用户发送短信以后，经过移动的无线网络基站，将 MO 消息转发到移动运营商网关，然后再由运营商的短信或彩信网关将 MO 消息转发给 SP 的短信或彩信网关[1]，当接收到短信或彩信以后，SP 网关程序并不是立即交给通用媒体增值业务管理系统来处理，而是先路由给增值业务控制中心（简称 SCC），先由 SCC 处理决定是否路由给通用媒体增值业务管理系统还是其他应用[2]。

MT 流程：当通用媒体增值业务管理系统需要下发短信或彩信给用户时，首先需要把短信或彩信提交给 SCC，由 SCC 决定路由给哪个 SP 网关（可以是其他 SP 的短信或彩信网关）[3]，SP 公司的网关接收到下发的 MT 消息以后，将该

条 MT 提交给运营商的短信或彩信网关，运营商然后再将该消息通过无线网络基站发送到用户的手机上，从而完成整个业务处理流程[4]。

通过图 4-1 可以清楚的了解到通用媒体增值业务管理系统在整个系统拓扑结构当中所处的位置。

4.1.2 设计原则

通用媒体增值业务管理系统在设计上了采用了以下策略：

- 分层原则。短信服务系统采用了分层策略，将整个系统分为三层，业务层，服务层和短信层。
- 可扩展原则。这在整个系统的上中下三层都得到了体现，比如在网关代理层，通过统一的接口和配置，短信服务系统可以方便地扩展网关代理实现与各短信网关的通信，在核心层，可以方便扩展监视的对象，在业务层，短信服务系统可以方便地增加模块，通过注册，实现与核心层的通信。比如发送短信，接收短信，接收短信发送状态等。
- 可配置原则。短信服务系统有许许多多的配置参数，可以方便地通过页面设置，改变和优化系统的特性和性能。
- 监控原则。短信服务系统对有些关键数据进行实施监控，及时反映数据的变化。

4.1.3 软件开发构架

通用媒体增值业务管理系统采用 MVC 架构，如图 4-2：

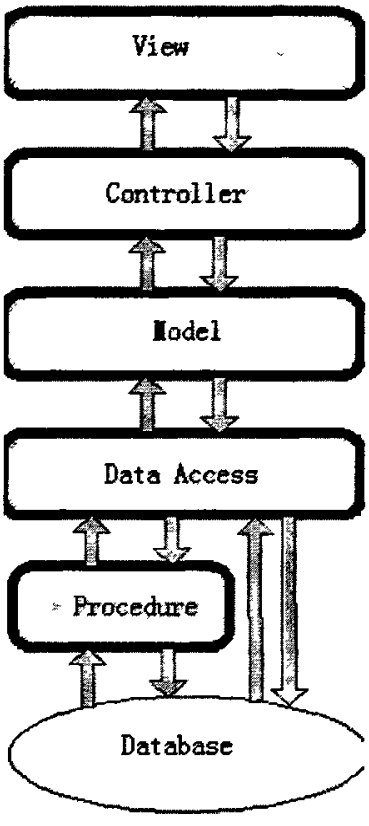


图 4-2 MVC 架构图

1. 表示层：负责处理页面显示逻辑，包括 html、java web start、applet 等，该层负责响应用户的请求，并返回处理结果。

2. 控制层：负责业务逻辑处理。

3. MODEL 层：数据库服务层主要是通过一台或一组数据库服务层为整个应用系统提供支持。在业务层通过相应的接口访问数据层。

4.1.4 采用 spring 来整合数据库层以及业务层、表示层[15]

Spring 不重新开发已有的东西。因此，在 Spring 中你将发现没有日志记录的包，没有连接池，没有分布事务调度。这些均有开源项目提供(例如 Commons Logging 用来做所有的日志输出，或 Commons DBCP 用来做数据库连接池)，或由你的应用程序服务器提供。因为同样的原因，我们没有提供 O/R mapping

层，对此，好的解决办法是采用Hibernate或JDO等。Spring的目标是使已存在的技术更加易用。例如,尽管我们没有底层事务协调处理，但我们提供了一个抽象层覆盖了JTA或任何其他的事务策略。

Spring没有直接和其他的开源项目竞争，除非我们感到我们能提供新的一些东西。例如，像许多开发人员，我们从来没有为Struts高兴过，并且感到在MVC web framework中还有改进的余地。在某些领域，例如轻量级的IoC容器和AOP框架，Spring有直接的竞争，但是在这些领域还没有已经较为流行的解决方案。(Spring在这些区域是开路先锋。) Spring也得益于内在的一致性。Spring在应用服务器之间是可移植的。

Spring的设计核心是org.springframework.beans包，为与JavaBeans一起工作而设计。这个包一般不直接被用户使用，但作为基础为更多的其他功能服务. 下一个较高层面的抽象是"Bean Factory"。Spring bean factory是一个普通的Factory，它使对象能够按名称获取，并且能管理对象之间的关系。Bean factories支持两种对象模式:

- Singleton：在此模式中，有一个具有特定名称的共享对象实例，它在查找时被获取。这是默认的，而且是最为经常使用的。它对于无状态对象是一种理想的模式。
- Prototype：在此模式中，每次获取将创建一个独立的对象。例如，这可以被用于允许用户拥有他们自己的对象。由于BeanFactory是一个简单的接口，它能被为了底层存储方法而实现。你能够方便地实现你自己的BeanFactory，尽管很少用户需要。最为常用的定义是：XmlBeanFactory：可解析简单直观的定义类和命名对象属性的XML结构。我们提供了一个DTD来使编写更容易。ListableBeanFactoryImpl：可提供解析存放在属性文件中的bean定义，和可通过编程创建BeanFactories。

每个bean定义可能是一个POJO(通过类名和JavaBean初始属性定义)，或是一个FactoryBean。FactoryBean接口添加了一个间接层。通常，这用于使用AOP或其他方法来创建代理对象，例如添加声明性事务管理的代理(这在概念上和EJB侦听相似，但在实践中实现更简单)。

BeanFactories能在一个层次结构中可选择性的参与，根据来自祖先的继承定义。这使在整个应用中公共配置的共享成为可能，虽然个别资源，如

controller servlets，也拥有他们自己的独立的对象集合。

倒置控制的几个重要好处。如：

1. 因为组件不需要在运行时间寻找合作者，所以他们可以更简单的编写和维护。在 Spring 的 IoC 版本里，组件通过暴露 JavaBean 的 setter 方法表达他们依赖的其他组件。这相当于 EJB 通过 JNDI 来查找，EJB 查找需要开发人员编写代码。

2. 同样原因，应用代码更容易测试。JavaBean 属性是简单的，Java 核心的，并且容易测试：仅编写一个包含自身的 Junit 测试方法用来创建对象和设置相关属性即可。

3. 一个好的 IoC 实现隐藏了强类型。如果你使用一个普通的 factory 来寻找合作者，你必须通过类型转换将返回结果转变为想要的类型。这不是一个主要问题，但是不雅观。使用 IoC，你在你的代码中表达强类型依赖，框架将负责类型转换。这意味着在框架配置应用时，类型不匹配将导致错误；在你的代码中，你无需担心类型转换异常。

4. 大部分业务对象不依赖于 IoC 容器的 APIs。这使得很容易使用遗留下来的代码，且很容易的使用对象无论在容器内或不在容器内。例如，Spring 用户经常配置 Jakarta Commons DBCP 数据源为一个 Spring bean：不需要些任何定制代码去做这件事。我们说一个 IoC 容器不是侵入性的：使用它并不会使你的代码依赖于它的 APIs。任何 JavaBena 在 Spring bean factory 中都能成为一个组件。

最后应该强调的是，IoC 不同于传统的容器的体系结构（如 EJB），应用代码最小程度的依靠于容器。这意味着你的业务对象可以潜在的被运行在不同的 IoC 框架上-或者在任何框架之外-不需要任何代码改[16]。

在本系统里，跟 hibernate 的结合例如：

```
<bean id="txManager2" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
<property name="dataSource">
  <ref local="dataSource"/>
  </property>
</bean>
<bean id="sessionFactory" class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">
   <property name="mappingResources">
```

```
<list>
<value>/com/sourceware/cmvp/platform/user/dao/hibernate3/T_cmvpUserInfo.hbm.xml</value>
<value>/com/sourceware/cmvp/platform/user/dao/hibernate3/T_cmvpUserAdmin.hbm.xml</value>
<value>/com/sourceware/cmvp/platform/module/T_cmvpModuleInfo.hbm.xml</value>
...
```

通过 spring 的 bean 工厂机制配置 hibernate 的映射关系，数据源等等。而在表示层，webwork 跟 spring 的结合也是非常的简单方便，只要在 webwork.properties 里配置成：webwork.objectFactory=spring 即可！

4.2 数据库设计与实现

4.2.1 实体关系模型

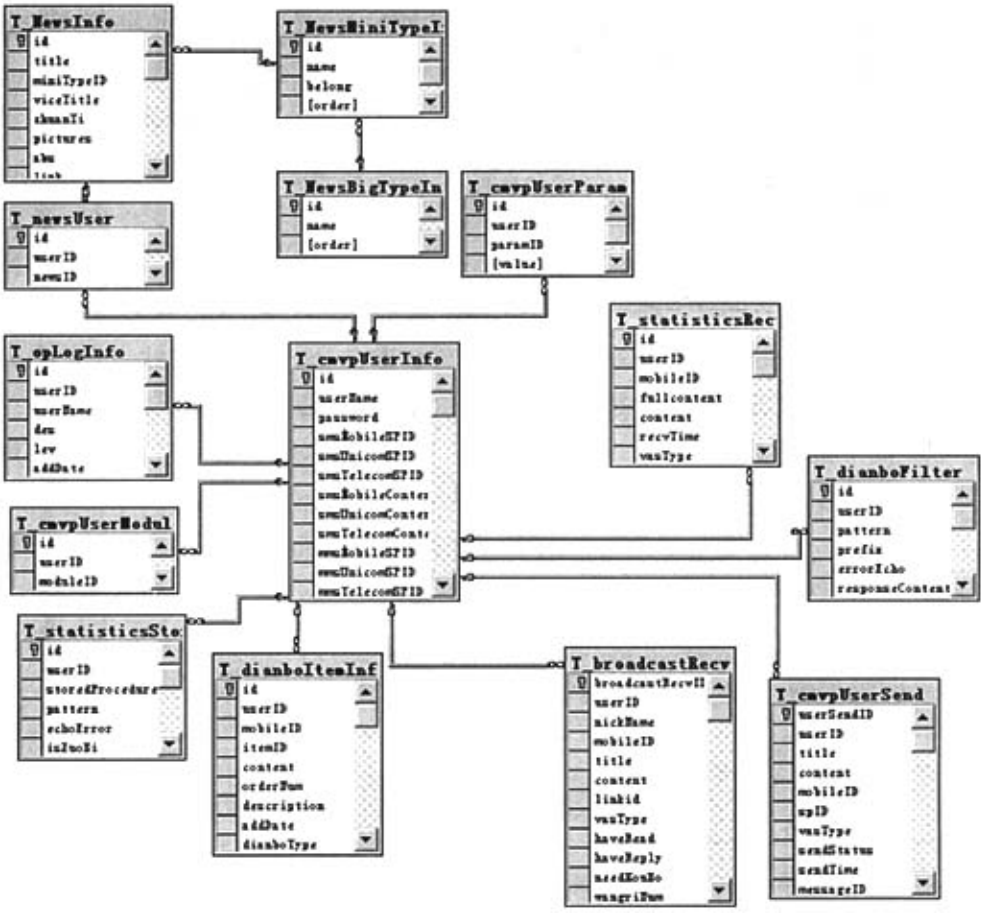


图 4-3 数据表关系图

图 4-3 是系统主要表结构的一个关系图，系统有各种不同的用户，因此需要有一张用户表、用户权限表等，直播模块必须要有直播接收表，直播过滤表等。而点播模块则不要有点播信息表，点播用户配置表，统计模块则要有统计用户配置表。

主要的数据库表结构设计如下：

T_broadcastEchoInfo 表是直播模块短信的回复语存储在该表，通过 userid 和 T_userInfo 表进行关联，即每个主持人用户可以对应一个直播回复语记录。

T_broadcastFilter 表则是直播过滤表，当主持人在浏览直播记录时，有写信息可能需要进行过滤，也通过 userid 和 T_userInfo 进行关联。

T_broadcastRecv 表则是直播记录表，所有属于某个主持人的短信或彩信都会写入该表，这样主持人就可以在直播模块浏览属于自己的短信或彩信，通用该表也通过 userid 和 T_userInfo 进行关联。

T_cmvpMOCache 表：有些主持人用户需要根据短信或彩信的状态报告来决定是否显示该条短信，因此需要有一张表来暂时缓存该条短信，直到收到状态报告，该条记录才被删除。通过 userid 和 T_userinfo 关联。

T_cmvpSend 表：系统发送表，写入该表的短信或彩信都将被取走提交给运营商网关，没有关联关系。

T_cmvpUserInfo 表：是整个系统的用户表，存储所有的管理员用户、客服人员用户、业务人员用户、合作方管理员用户以及主持人用户。

T_cmvpUserSend 表记录主持人的回复信息，和 T_userInfo 通过 userid 关联。

T_statisticsChannelInfo 表记录统计组的信息，统计组应该是属于某个主持人的，因此必须和 T_userInfo 通过 userid 进行关联。

T_statisticsItemInfo 表则是存储了某个统计组下面的所有统计项，因此必须和 T_statisticsChannelInfo 关联。

T_statisticsRecvInfo 表是统计接受记录表，和 T_userInfo 通过 userid 进行关联。

T_statisticsStoredProcedureInfo 表保存统计模块所有可用的存储过程，统计模块可以由存储过程进行处理，没有关联。

T_statisticsStoreUserRelation 表是统计模块用户关联表，通过此表可以查出某主持人用户对应的统计模块处理存储过程是哪个，因此和 T_userInfo 关

联。

4.2.2 主要数据库表结构描述

1. 用户表，表名是 T_cmvpUserInfo，见表 4-1。

表名	T_ cmvpUserInfo		
字段名	字段属性	默认值	描述
id	Int（主键）		自增
userName	Varchar(20)		用户名，最短 3 位最长 20 位
password	Varchar(20)		密码，最短 3 位最长 20 位
smsMobileSPID	Varchar(25)		移动特服号
smsUnicomSPID	Varchar(25)		联通特服号
smsTelecomSPID	Varchar(25)		电信特服号
smsMobileContent	Varchar(50)		移动短信内容
smsUnicomContent	Varchar(50)		联通短信内容
smsTelecomContent	Varchar(50)		电信短信内容
mmsMobileSPID	Varchar(50)		彩信移动特服号
mmsUnicomSPID	Varchar(50)		彩信联通特服号
mmsTelecomSPID	Varchar(50)		彩信小灵通特服号
template	Varchar(50)		用户页面摸板
des	Varchar(50)		描述，用户名称
userType	Int	1	用户类型 0:管理员 1:主持人用户
isLocked	Int	0	该用户是否被锁定 0:false 1:true
[language]	Varchar(10)	zh_CN	该用户所使用的语言
defaultFee	Int	0	资费信息，主要用于联通，默认为 0
addDate	dateTime		创建时间
用途：	用户表，用于记录系统所有的用户和帐号		
缺点：			
优点：			

表 4-1 T_cmvpUserInfo

2. 直播接收记录表：T_broadcastRecvInfo，见表 4-2。

表名	T_ cmvpUserInfo		
字段名	字段属性	默认值	描述
broadcastRecvID	Int（主键）		自增
userID	Int		对应用户的 ID
nickName	Varchar(20)		对应手机用户的昵称
mobileID	Varchar(12)		手机号码
title	Varchar(100)		彩信标题
content	Varchar(5000)		短信或彩信的内容
linkid	Varchar(500)		
vasType	int	0	增值业务类型 0:短信 1:彩信
haveRead	Int	0	标记是否已读 0:false 1:true
haveReply	int	0	标记是否已回复 0:false 1:true
needKouBo	int		标记是否属于口播记录 0:false 1:true
wangriNum	Int		该手机号码往日发送条数
todayIndex	Int		该条信息今日的序号
todayNum	Int		该手机号码今日发送条数
recvTime	datetime	Getdate()	该条短信的接收时间
用途：	直播模块短信和彩信接收记录表		
缺点：			
优点：			

表 4-2 T_broadcastRecvInfo

3. 发送表：T_cmvpSend，见表 4-3。

表名	T_ cmvpUserInfo		
字段名	字段属性	默认值	描述
id	Int（主键）		自增
srcMobileID	Varchar(24)		发送发起方号码
destMobileID	Varchar(24)		信息接收方号码

content	Varchar(5000)		信息内容
serviceCode	Varchar(100)		业务代码
fee	Int		资费
msgType	Int		消息类型
linkID	Varchar(100)		
vasType	Int		增值业务类型 0:短信 1:彩信
priority	int		发送优先级，从高到低 9-0
sheduleTime	datetime		定时发送的时间
用途：	系统短信或彩信的发送表，只要写入该表的记录都将被取走发送给用户。		
缺点：			
优点：			

表 4-3 T_ cmvpSend

4. 统计模块统计组表：T_statisticsChannelInfo，见表 4-4。

表名	T_ cmvpUserInfo		
字段名	字段属性	默认值	描述
statChannelID	Int（主键）		自增
userID	Int		用户 ID
description	Varchar(100)		统计组名称
echoStart_mobile	Varchar(250)		统计进行当中的移动回复
echoNoStart_mobile	Varchar(250)		统计组没有开始时的移动回复
echoEnd_mobile	Varchar(250)		统计组已经结束时的移动回复
echoStart_unicom	Varchar(250)		统计组进行当中联通的回复
echoNoStart_unicom	Varchar(250)		统计组没有开始时的联通的回复
echoEnd_unicom	Varchar(250)		统计组已经结束时的联通的回复
echoStart_telecom	Varchar(250)		统计组进行当中小灵通的回复
echoNoStart_telecom	Varchar(250)		统计组没有开始时的小灵通的回复
echoEnd_telecom	Varchar(250)		统计组已经结束时小

			灵通的回复
startTime	Datetime		统计开始时间
endTime	Datetime		统计结束时间
limitNumber	Int		限制票数，0 不限制
echoLimit	Varchar(250)		限制时的回复信息
limitType	Int		显示票数类型 0:不限制 1：每天限制 2：总体限制 3：每天总体都限制
timeType	Int		统计时间类型， 0：开始到结束统计时间类型 1：每天统计时间类型
用途：	用户的统计组配置表		
缺点：			
优点：			

表 4-4 T_ statisticsChannelInfo

4.3 基于工作流的模型设计

媒体增值业务管理系统的工作流环境是基于工作流技术系统模型的核心部分，它集中解决媒体用户办公环境的协作问题。根据WFMC工作流参考模型，我们建立了如图4-4所示的媒体增值业务系统工作流的系统模型。模型中各模块的主要功能如下：

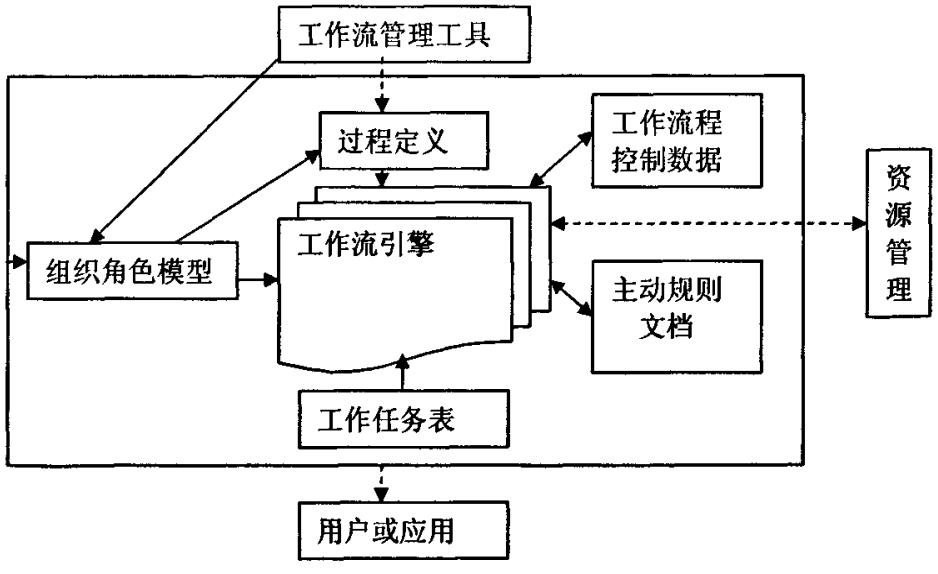


图4-4 工作流系统模型

工作流管理工具：提供流程、建模工具，通常以图形化方式定义、监控流程。

组织/角色模型：通过管理工具建立或从人力资源库中提取的基于职能的部门和基于职务的人员信息模型。它将为模型定义及生成和管理过程实例提供参考。

工作流引擎：解释过程定义，创建过程实例，并依据流程控制数据控制其执行；按照调度规则对任务进行调度；生成任务项并填写到有关用户的任务表上生成工作任务表；给管理人员提供控制、管理、监督功能。

过程定义：由计划人员通过工作流管理工具建立的提供给工作流引擎解释执行的工作流程模型定义。

工作流程控制数据：指工作流管理环境的系统信息和过程实例的动态信息。根据控制数据可以得到过程的执行轨迹，以支持管理员的监督、查询。如每个过程实例的状态信息(初始、运行、活动…)；每个活动实例的状态信息(准备、活动、挂起…)。

工作任务表：是工作流引擎执行过程实例中生成并分配给用户的任务项队列表。其中每个任务项包含任务名称、任务内容，完成期限和其它为准确完成操作所必备的数据。

主动规则文档：实时监控过程实例的执行，并根据执行情况主动向管理人员或用户发出信息，如新任务的到达，过程执行异常报告等。

4.3.1 信息发布流程设计

系统的信息发布采用平行发布与管理模式，管理员可以采用专业html编辑工具比如dreamweaver等制作和发布站点的信息。信息发布工作流程如图4-5所示：

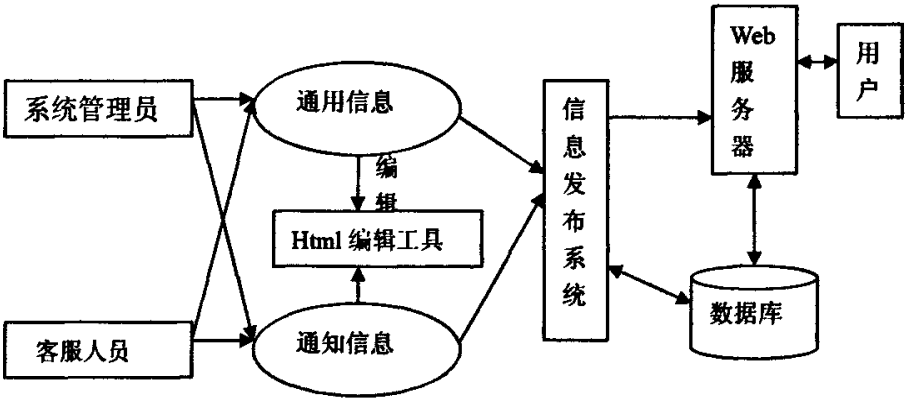


图 4-5 信息发布工作流程图

4.3.2 增值业务信息流

系统的增值业务信息流采用发布预定的模式，具体如图4-6所示：

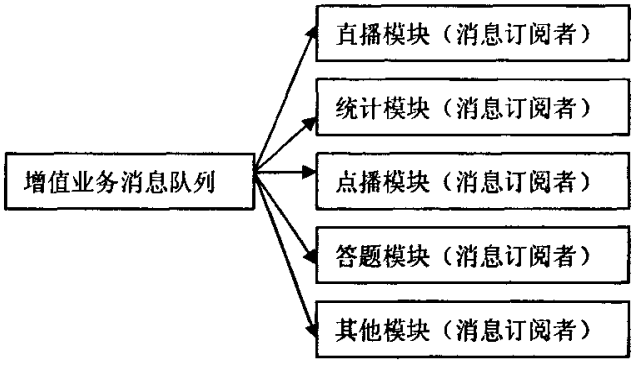


图4-6 发布预定处理模式

系统接收到增值业务消息以后，将放入消息接收队列，每一条消息将触发各个模块的isAcceptThisMessage方法，如果是该模块处理的消息，则调用onMessage方法，完成业务逻辑处理。

4.3.3 核心模块 MO 处理

MO 处理主要涉及以下几个类（如图 4-7 所示）：

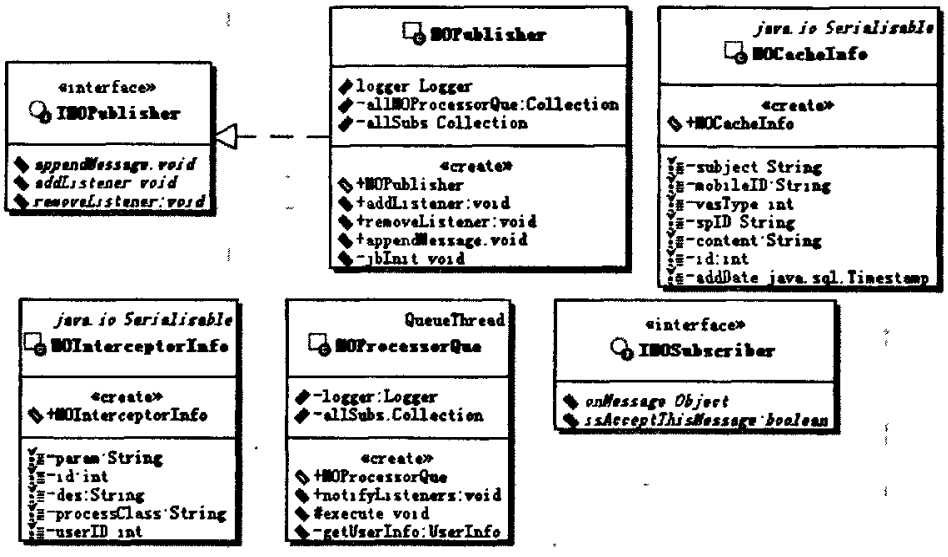


图 4-7 核心 MO 处理模块类图

1. com.sourceware.cmvp.platform.mo.wf.servlet.MOServlet:该 servlet 用来接收所有的 MO 消息。在 web.xml 里配置该 servlet。
2. com.sourceware.cmvp.platform.mo.MOPublisher:当 MOServlet 接收到 MO 以后，将调用 MOPublisher 的 appendMessage 方法，MOPublisher 的构造方法里将初始化 N 个队列线程：

```
int moProcessorQueNum =
Integer.parseInt(ConfigProperties.getProperty("mo.processor.count"));
for (int i = 0; i < moProcessorQueNum; i++) {
        MOProcessorQue que = new MOProcessorQue(allSubs);
        que.start();
        allMOProcessorQue.add(que);
}
```

3. MOProcessorQue：该类是系统MO处理队列类，为了实现MO的异步处理，设计了MOProcessorQue类，遵循先进先处理的原则，完成所有MO的处理。
4. IMOSubscriber：所有的MO消费者都必须实现IMOSubscriber接口，该接口有两个方法，一个是：isAcceptThisMessage(Message msg)：用来判断该模块是否能处理该条MO消息，如果能处理则返回true，否则返回false;一个是onMessage(MessageAdapter msg)方法：用来完成该模块的业务处理.
5. MOInterceptorInfo：拦截类，主要是实现动态配置拦截MO，以完成一些特殊处理，比如可能要拦截所有移动的短信，看是不是以数字或字母开头的，如果不是，则提醒用户要在短信前加上数字或字母。

4.3.4 重要模块概况

通用媒体增值业务管理系统按模块进行划分，开发人员只专注系统当中部分模块，模块和模块之间尽量低偶合，模块内部则高内聚，系统内每个模块一般能完成一个比较独立的功能，比如直播模块将完成对MO消息的处理，能让业务人员方面管理配置以及主持人能实时查看浏览用户发上来的短信或彩信。

4.3.5 直播模块

直播模块主要功能包括：

- 直播模块负责显示用户发上来的短信和彩信，并且主持人可以回复、修改、做标记、按时间内容和手机号码进行查询、对直播的数据可以导出等操作。
- 业务人员可以配置直播的移动联通和小灵通的各种回复语，可以配置显示过滤器，对直播内容进行过滤。
- 在页面上，直播页面主要包括顶上的新闻和通知、搜索查询控制面板和短信彩信列表组成。新闻和通知以滚动形式醒目的提醒主持人用户。搜索查询条件有：按时间搜索，包括连续的时间，和片段的时间；按内容模糊搜索，以及按手机号码模糊搜索。
- 直播数据可以导出。

- 短信和彩信列表每条信息可以做修改、删除、收藏、复制等操作，可以做已阅读标记，可以做口播标记。
- 另外每条信息可以通过右键菜单被添加到点播或者交友模块。

在MO消息处理上，所有接收到的短信和彩信都在后台处理并写入直播数据表，前台只负责从直播数据表当中查找记录，并不包含其他运算。

直播模块业务处理流程大致为：

先查找该MO消息属于哪个主持人用户，取出用户ID，然后查找该手机号码对应的往日发送数量以及今日参与数量，然后查找该手机号码是否存在昵称，如果存在则把昵称也一并写入直播数据表，然后判断该手机号码是移动、连通还是小灵通，并查找对应的回复语进行回复，具体步骤如图4-8所示的直播模块MO消息处理流程。

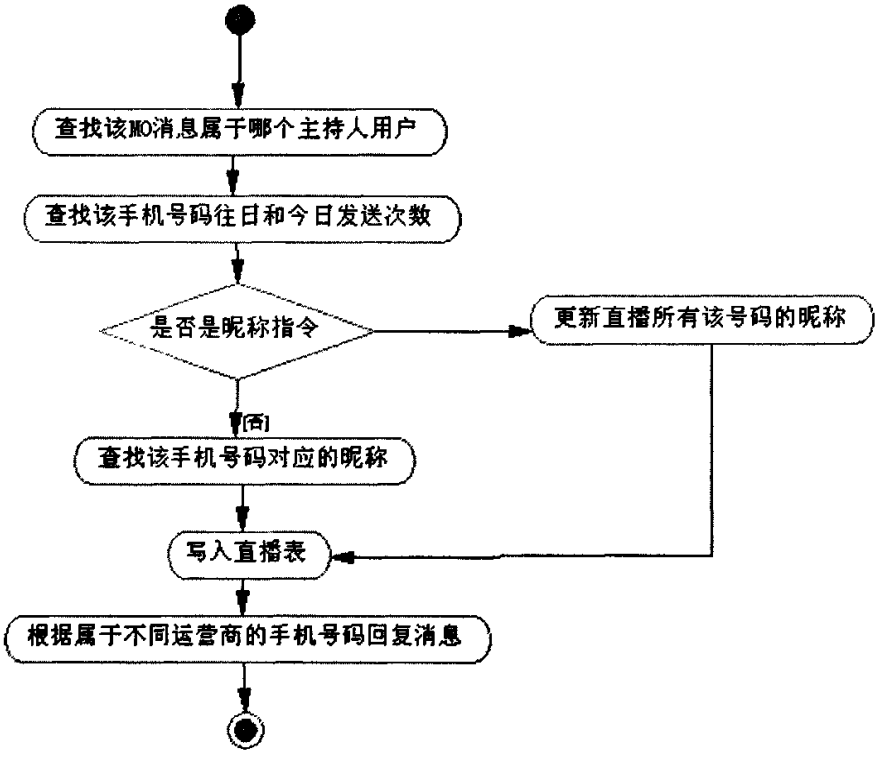


图4-8 直播模块MO消息处理流程

4.3.6 统计模块

统计模块主要功能包括：

- 负责票数统计，主持人可以按时间段进行票数统计，可以查看某统计项的详细接收记录，并可以针对某个手机号码进行单独的短信或彩信回复。
- 统计模块还必须统计某时间段内容，发送短信或彩信数量最多的前几名。直播页面显示所有统计组的每个统计项的编号、名字以及票数（百分比）情况。
- 跟直播模块类似，统计当中的按时间查询也包括连续的时间查询和片段的时间查询。
- 某个统计组内各个统计项的回复是统一的，也就是只配置统计组的移动、联通、小灵通的回复，统计项的匹配按模糊匹配原则进行匹配，并且长优先，先匹配到长的再匹配到短的，比如我配置了“1”号和“11”号两个统计项，用户发送“11号”将先匹配到11而不是同时匹配到1，如果用户发送“1”，则匹配到1号。
- 统计组的类型分：一种是从某开始时间到某结束时间的统计；一种是每天进行统计的统计组。前一种用在活动类的，比如唱歌投票有一个投票开始时间和结束时间，后一种用在电台统计，比如每天统计听众发送Y的短信数量。

统计模块的MO处理流程如图4-9所示：

- 先查找是否有对应的统计项，如果没有则直接回复错误信息。
- 如果有则判断该统计项属于的统计组当前状态，如果是未开始则回复未开始的信息回复，如果正在进行当中则对该统计项票数加一，并回复投票成功的信息；如果已经结束则回复已经结束的信息。

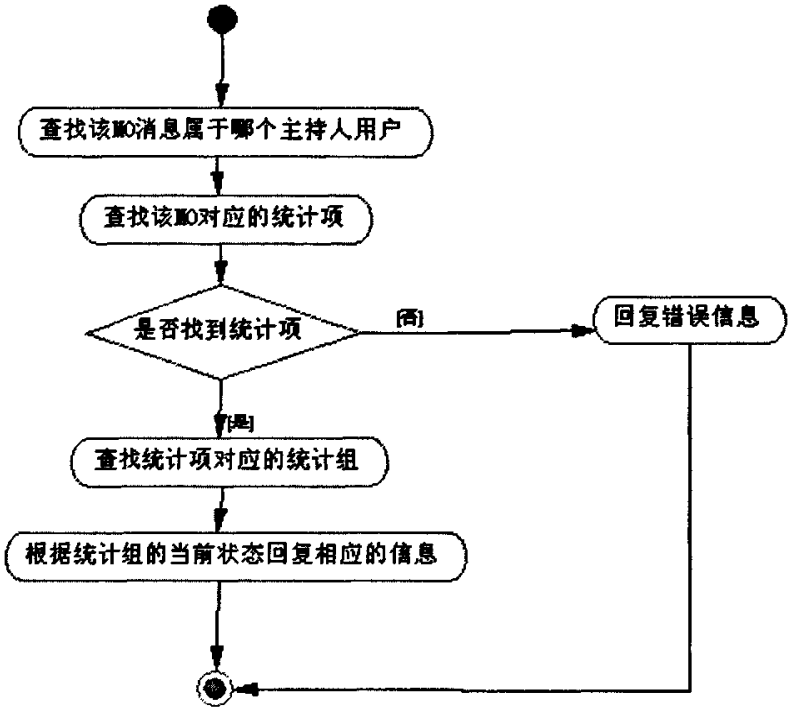


图4-9统计MO处理流程图

4.3.7 点播模块

点播类型分交友点播、文字点播、彩信点播和WAP-PUSH点播，点播通过统一的配置，决定各个点播类型的开头匹配指令，比如字符串：5,6,7,8则表示交友点播指令为5开头后面跟数字，文字点播指令为6开头后面跟数字，彩信点播指令则是7开头后面跟数字，WAP-PUSH点播指令是8开头后面加上数字编号。

- 交友点播是允许用户按编号点播想交友的用户资料，用户可以下载该编号对应的交友用户的资料。
- 而文字点播是允许用户点播下载一些比较好的短语等。
- 彩信点播内容比较丰富，允许用户点播一些多媒体信息，多媒体信息可以是图片文字和声音等。
- 而WAP-PUSH点播则是短信的一种特殊格式，用户会收到一个下载的连接

地址，通过该地址，用户可以下载多媒体文件等，因为彩信有大小限制，因此，通过WAP-PUSH的方式提供用户下载将不受限制。

点播模块MO消息处理流程如图4-10所示：

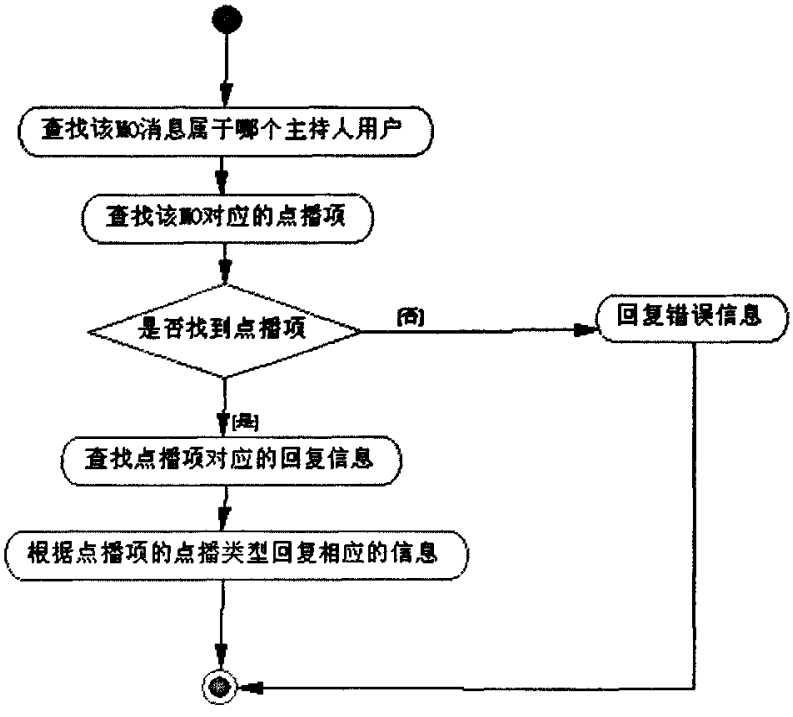


图4-10点播模块MO处理流程

4.3.8 新闻模块

业务人员可以编辑发布新闻或通知信息，其优势是所见即所得的编辑方式，使编辑人员非常直观方便。该编辑器提供了类似office的编辑风格，用户使用起来容易上手，使用熟悉的方式进行信息编辑，另外如果编辑人员熟悉html标签，还可以切换到html书写方式，使编辑人员更加方便自由，这个在线的新闻和通知编辑系统，充分实现网站页面编辑的灵活性的同时，最大限度的降低了工作量和对专业编辑人员的依赖。

4.4 系统安全设计

4.4.1 基于角色管理的权限控制

1. 通用媒体增值业务管理信息管理系统是一个大型的分布式数据资源管理系统，它包括信息量巨大以及不同程度的信息敏感度，各种有访问需求的用户，使得其安全管理非常复杂。基于角色的系统安全控制模型是目前国际上流行的先进的安全管理控制方法。我们通过分配和撤消角色来完成用户权限的授予和撤消，安全管理人员根据需要定义各种角色，并设置合适的访问权限，而用户根据其责任和资历再被指派为不同的角色。这样，整个访问控制过程就分成两个部分，即访问权限与角色相关联，角色再与用户相关联，从而实现了用户与访问权限的逻辑分离，如下图4-11所示，角色可以看成是一个表达访问控制策略的语义结构，它可以表示承担特定工作的资格。

图4-11 基于角色控制的策略

2. 通用媒体增值业务管理系统由于大部分使用者为电台、电视台等用户，需要尽可能的减少输入；另外系统的短信彩信数据需要考虑是不是能更加的安全，防止数据被窃听，因此，系统采用了CA认证系统，每个电台或电视台用户可以领取一张属于自己的CA证书，只要安装了该证书（如图4-12），就可以通过ssl连接，认证该客户端的连接是否安全。

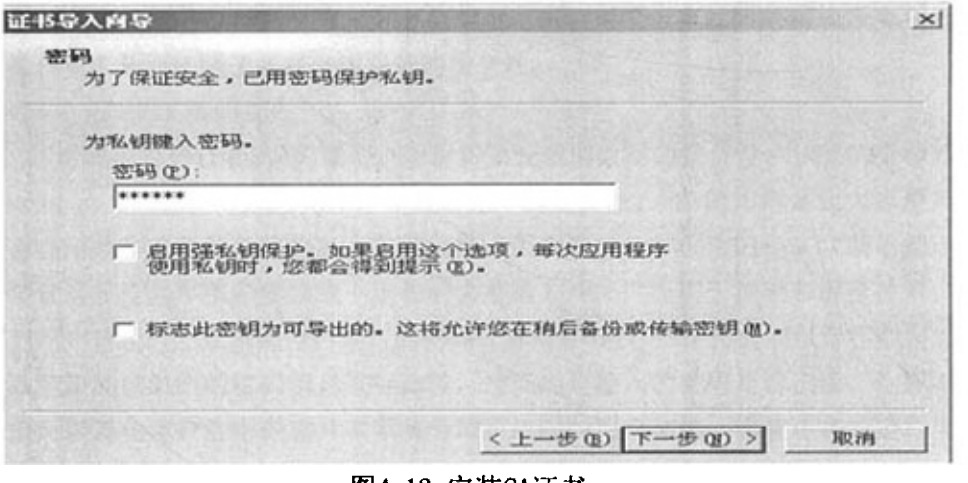


图4-12 安装CA证书

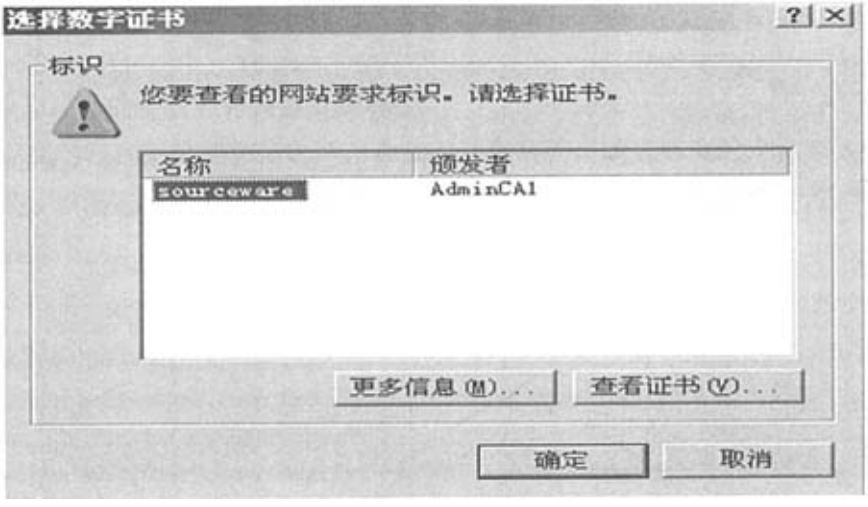


图4-13 登录时选择证书

一旦安装了CA证书，就可以通用该证书登录连接到服务器（如图4-13所示选择已经安装的证书），而不用再输入用户名和密码。同时，用户也可以直接输入用户名和密码登录系统，密码通过md5加密，确保密码安全，登录时，系统会进行用户帐号的认证操作，如果帐号正确将分配此用户的相关权限及所在部分的相关权限，并进行权限的整合。并将此登录用户的相关用户信息、权限信息存在session中，以便之后的一些管理操作工作权限的正确分配。起到不同人员分管其中一部分，而且权限只到各自管理范围，对于没有权限的模块没有相应操作权。从而加强了系统分级管理的安全性。

4.4.2 密码MD5编码加密

密码信息进行MD5编码加密，以保护系统敏感数据的安全性： MD5的全称是Message Digest Algorithm5，即为信息—摘要算法。MD5的作用是让大容量信息在用数字签名软件签署私人密匙前被“压缩”成一种保密的格式（就是把一个任意长度的字节串变换成一定长的大整数）。MD5广泛用于加密和解密技术，当用户登录时，系统把用户输入的密码计算成MD5值，然后再去和保存在数据库系统中的已编码的密码信息进行比较，进而确定输入的密码是否正确。这样不但可以避免用户的密码被具有系统管理员权限的用户知道，而且还在一定程度上增加了密码被破解的难度。

4.4.3 用户超时退出系统

每个请求都会经过一个UserLoginFilter，如果发现该用户的session超时，

那么将自动返回到登录首页。

4.5 发布预定的增值业务消息处理模型

系统接收到短信或彩信以后，会放入消息队列，消息队列的个数是系统配置好的，当消息进入消息队列以后，将会把消息传递给各个消息预定者。Xml 配置文件如下：

```
<beans>
  <bean id="moPublisher" class="com.sourceware.cmvp.platform.mo.MOPublisher"/>
  <bean id="broadcastSubscriber"
class="com.sourceware.cmvp.broadcast.BroadcastSubscriber"/>
  <bean id="statisticsSubscriber"
class="com.sourceware.cmvp.statistics.StatisticsSubscriber"/>
  <bean id="questionSubscriber"
class="com.sourceware.cmvp.quiz.QuestionSubscriber"/>
  <bean id="dianboSubscriber"
class="com.sourceware.cmvp.dianbo.DianboSubscriber"/>
  <!-- register broadcast subscriber. -->
  <bean id="registerSubscriber1"
class="org.springframework.beans.factory.config.MethodInvokingFactoryBean">
    <property name="targetObject">
      <ref local="moPublisher"/>
    </property>
    <property name="targetMethod">
      <value>addListener</value>
    </property>
    <property name="arguments">
      <list>
        <ref bean="broadcastSubscriber"/>
      </list>
    </property>
  </bean>
  <!-- register statistics subscriber. -->
  <bean id="registerSubscriber2"
class="org.springframework.beans.factory.config.MethodInvokingFactoryBean">
    <property name="targetObject">
      <ref local="moPublisher"/>
    </property>
    <property name="targetMethod">
```

```
      <value>addListener</value>
    </property>
    <property name="arguments">
      <list>
        <ref bean="statisticsSubscriber"/>
      </list>
    </property>
  </bean>
  <!-- register quiz subscriber. -->
  <bean id="registerSubscriber3"
class="org.springframework.beans.factory.config.MethodInvokingFactoryBean">
    <property name="targetObject">
      <ref local="moPublisher"/>
    </property>
    <property name="targetMethod">
      <value>addListener</value>
    </property>
    <property name="arguments">
      <list>
        <ref bean="questionSubscriber"/>
      </list>
    </property>
  </bean>
  <!-- register dianbo subscriber. -->
  <bean id="registerSubscriber4"
class="org.springframework.beans.factory.config.MethodInvokingFactoryBean">
    <property name="targetObject">
      <ref local="moPublisher"/>
    </property>
    <property name="targetMethod">
      <value>addListener</value>
    </property>
    <property name="arguments">
      <list>
        <ref bean="dianboSubscriber"/>
      </list>
    </property>
  </bean>
</beans>
```

通过 spring 的 bean 工厂机制，可以实现动态注入消息预定者。

第 5 章　典型业务系统实现

5.1 直播序列

主持人浏览直播信息的时序图如图 5-1 所示：

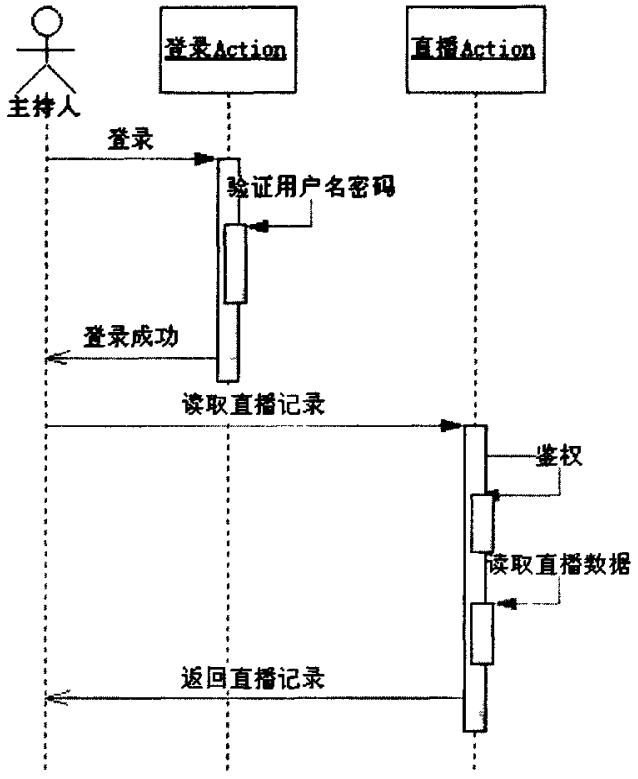


图 5-1 直播操作时序图

主持人看到的首先是登录页面，然后输入用户名和密码向登录 ACTION 发出登录请求，如果用户名和密码正确，主持人将进入直播页面，并向直播 ACTION 发出读取直播记录的请求，直播 ACTION 将对 session 当中存储的用户进行鉴权，读取该主持人用户能看到的直播数据返回给用户。

5.2 登录直播系统

主持人用户打开系统，首先出现的是登录页面 index.ww,如图 5-2 所示。

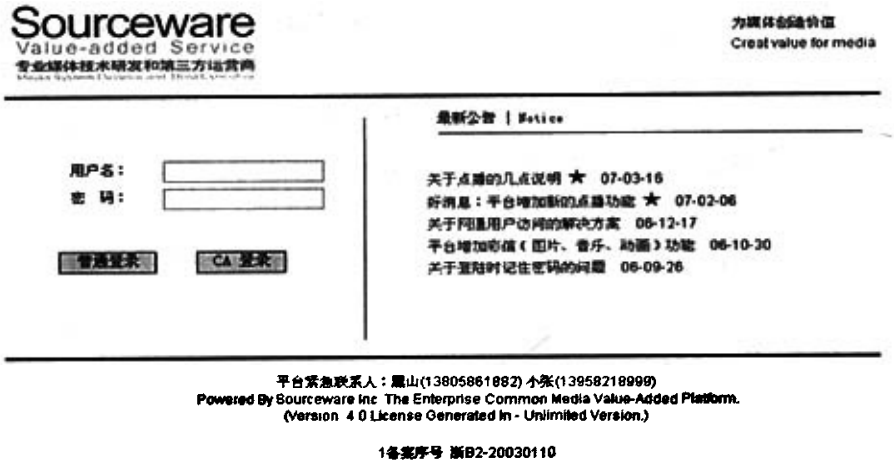


图 5-2 登录页面

主持人用户输入用户名和密码，点击普通登录，将向登录 ACTION 发出登录请求，

1. 负责显示登录页面的 Action 类为：

com.sourceware.cmvp.platform.user.wf.action.IndexAction

该 action 类的主要代码片段：

```
NewsDAO nd = (NewsDAO) PlatformFacade.getInstance().getBean(
        "newsDAOProxy");
    int miniTypeID = Integer.parseInt(ConfigProperties.getProperty(
        "news.news.minitypeid"));
    this.allLatesNews = nd.getAllLatestNewsInfoByMiniTypeID(miniTypeID, 1,
10);//取出最新的 10 条新闻
```

IndexAction 将调用新闻 model 层接口，取出最新的 10 条新闻，然后通过 webwork+velocity 把最新的 10 条新闻格式化输出。

在 webwork 的 xwork.xml 当中的 action 配置为：

```
    <action name="index"
class="com.sourceware.cmvp.platform.user.wf.action.IndexAction">
      <result name="success" type="redirect">
        <param name="location">/${userInfo.template}/defaultpage.ww</param>
      </result>
```

```
    <result name="login" type="velocity">
      <param name="location">index.cfm</param>
    </result>
    <result name="error" type="redirect">
      <param name="location">error.popup.back.ww</param>
    </result>
    <interceptor-ref name="params"/>
  </action>
```

2. 负责接受该登录请求并处理的 Action 类为：

com.sourceware.cmvp.platform.user.wf.action.LoginAction

主要代码片段为：

```
userInfo = ( (IUserDAO) platformFacade.getBean("userDAOProxy")).
```

authenticate(userName, password);//调用 MODEL 层的 authenticate 方法验证用户名和密码，返回用户信息。

在 xwork.xml 当中的 action 配置为：

```
  <action name="login"
class="com.sourceware.cmvp.platform.user.wf.action.LoginAction">
    <result name="success" type="redirect">
      <param name="location">/${userInfo.template}/defaultpage.ww</param>
    </result>
    <result name="error" type="redirect">
      <param name="location">error.popup.back.ww</param>
    </result>
    <interceptor-ref name="params"/>
  </action>
```

5.3 浏览直播记录

成功登录以后，首先进入的是直播页面 broadFrame.ww，这是一个框架页面，里面包含的主页面即直播记录页面为 broadList.ww，如图 5-3 所示：

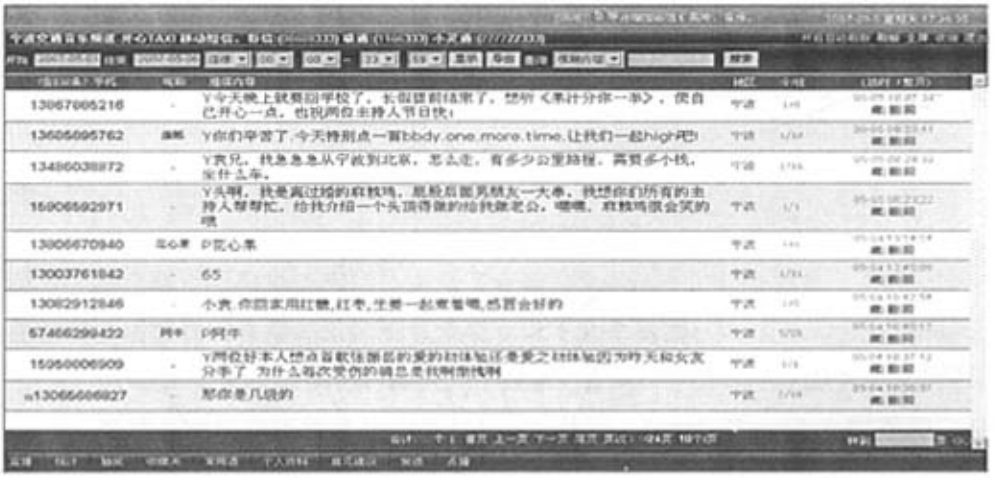


图 5-3 直播页面

直播浏览页面和 ACTION 的关系可以见 xwork 的配置文件：

```
<action name="broadList"
class="com.sourceware.cmvp.broadcast.wf.action.GetAllBroadCastRecvAction">
      <result name="success" type="velocity">
        <param name="location">broadList.cfm</param>
      </result>
      <result name="error" type="redirect">
        <param name="location">error.popup.back.ww</param>
      </result>
      <interceptor-ref name="params"/>
</action>
```

com.sourceware.cmvp.broadcast.wf.action.GetAllBroadCastRecvAction 这个 action 类接收用户提交的查询数据，并搜索出查询结果，通过 velocity 对结果格式化输出，如果发生错误，则重定向到 error.popup.back 这个 action。

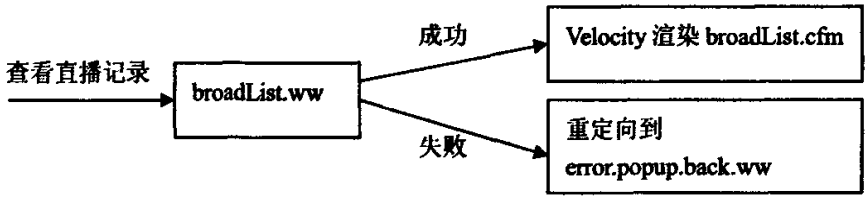


图 5-4 直播页面渲染流程

图 5-4 是直播页面渲染流程图，表示了页面和 action 之间的关系。

在实现直播模块的直播记录的浏览主要有以下几个页面：

1. broadFrame.cfm 页面：提供一个搜索、查询、导出等功能的 frame 页面，在该页面里面将包含 broadList.cfm 页面。Xwork 里 action 配置为：

```
<action name="broadmainAction"
class="com.sourceware.cmvp.broadcast.wf.action.BroadCastMainAction">
    <result name="success" type="velocity">
      <param name="location">broadFrame.cfm</param>
    </result>
    <result name="error" type="redirect">
      <param name="location">error.popup.back.ww</param>
    </result>
    <interceptor-ref name="params"/>
  </action>
```

2. broadList.cfm 页面：主要是列出所有搜索结果， 被包含在 broadFrame.cfm 页面里。

3. broadList4ajax.cfm 页面：该页面是了实现无刷新效果的记录页面，只显示结果集合。该 action 在 xwork 里的配置为：

```
<action name="broadList4ajax"
class="com.sourceware.cmvp.broadcast.wf.action.GetAllBroadCastRecvAction">
    <result name="success" type="velocity">
      <param name="location">broadList4ajax.cfm</param>
    </result>
    <result name="error" type="redirect">
      <param name="location">error.popup.back.ww</param>
    </result>
    <interceptor-ref name="params"/>
  </action>
```

4. error.popup.back.ww：该action负责弹出方式显示错误信息，action在xwork当中的配置为：

```
<action name="error.popup.back"
class="com.sourceware.wf.action.ErrorAction">
    <result name="success" type="velocity">
      <param name="location">/msg.popup.back.cfm</param>
    </result>
  </action>
```

com.sourceware.cmvp.broadcast.wf.action.BroadCastMainAction 该action类主要是收集查询条件等信息。

com.sourceware.cmvp.broadcast.wf.action.GetAllBroadCastRecvAction 该action类主要是查询条件搜索出全部符合条件的直播记录，并按页输出。

```
IBroadcastDAO broadDao = (IBroadcastDAO) PlatformFacade.getInstance().
    getBean("broadcastProxy");
if (this.getEnable() == 1) { // 是否是时间片段 (Y
  this.pageUtil = broadDao.getAllBroadCastRecvByTimeSegment(this.
      getUserID(), this.getQuery_condition(), this.getQuery_content(),
      DateTime.toTimestamp(startTime),
      DateTime.toTimestamp(endTime),
      DateTime.toTimestamp(this.getStartTimeSeg()),
      DateTime.toTimestamp(this.getEndTimeSeg()), needKouBo == 1,
      this.getPage(),
      this.getRowsPerPage()
      );
}
else {
      this.pageUtil = broadDao.getAllBroadCastRecv(this.getUserID(),
      this.getQuery_condition(), this.getQuery_content(),
      DateTime.toTimestamp(startTime),
      DateTime.toTimestamp(endTime), needKouBo == 1, this.getPage(),
      this.getRowsPerPage());
}
```

该action将调用DAO层getAllBroadCastRecv或getAllBroadCastRecvByTimeSegment方法，取出直播记录。

MODEL层主要涉及到以下几个类：

IBroadcastDAO：直播模块 DAO 接口类，定义了 DAO 方法。

BroadcastDAO：直播模块 DAO 实现类，实现了 IBroadcastDAO 接口类定义的所有方法。

BroadcastRecvInfo：直播记录的属性类。

HIBERNATE 实现 DAO 层通过 spring 来封装如下：

```
<beans>
  <bean id="broadcastDAO"
class="com.sourceware.cmvp.broadcast.dao.hibernate3.BroadcastDAO">
    <property name="sessionFactory">
      <ref bean="sessionFactory"/>
    </property>
  </bean>
  <bean id="broadcastProxy"
class="org.springframework.transaction.interceptor.TransactionProxyFactoryBean
">
    <property name="transactionManager">
      <ref bean="transactionManager"/>
    </property>
    <property name="target">
      <ref local="broadcastDAO"/>
    </property>
    <property name="transactionAttributes">
      <props>
        <prop key="insert*">PROPAGATION_REQUIRED,-Exception</prop>
        <prop key="create*">PROPAGATION_REQUIRED,-Exception</prop>
        <prop key="delete*">PROPAGATION_REQUIRED,-Exception</prop>
        <prop key="*">PROPAGATION_REQUIRED</prop>
      </props>
    </property>
  </bean>
</beans>
```

通过 spring 来管理 DAO，broadcastProxy 是一个代理类，实际上动态继承了 broadcastDAO，且通过 spring 的 bean 属性动态注入机制，实现注入事务管理对象：transactionManager，从而实现事务的可配置化。

第6章 结论与展望

6.1 结论

通用媒体增值业务管理系统从2006年下半年开始做需求分析和设计到开发完成，目前已经在浙江省里广泛使用了将近一年的时间，期间也倾注了我和我的同事们的心血，在此对已经取得的工作做一个总结：

1. 解决了多种增值业务集成的问题，在浙江省内第一家可以让电台、电视台同时开展短信和彩信业务。丰富了媒体机构的业务类型以及和用户之间的沟通的手段。

2. 提供了二次开发接口，可以新编写拦截器，完成新的业务逻辑，方便开展新业务以及新需求的变化。

3. 技术上推陈出新，使用了目前比较流行的J2EE技术，以及部分采用WEB2.0技术实现前台页面。使用户在操作使用上更加方便快捷，另外考虑了换肤技术，是不用的用户可以有不同的页面皮肤。

4. 另外在安全设计上更加可靠、稳定，可以选择CA认证，确保数据安全！

5. 彻底解决了帐单问题，使 SP 和合作方的结算更加高效直观，省去在帐单问题上出现的纠纷。

6. 在短信和彩信消息处理上，采用了消息发布预定的处理模式，可以大大提升系统性能，另外还做了消息缓存，针对状态报告可以生成不同的消息类型，这种全新的设计方法，实践证明是行之有效的，可以解决状态报告成功失败而导致的帐单数据不一致的问题。

7. 解决了国际化问题，可以根据不同的用户，选择不同的语言。

8. 应用了数据CACHE和页面的CACHE技术，大大提供了访问速度。

6.2 进一步工作的方向

本文的研究虽然取得了初步的成功，但依然任重道远，尚有许多有待进一步深入进行的研究工作，这里择其要者简要讨论如下：

1. 系统目前还没有考虑未来 3G 应用[6]，虽然支持彩信这种多媒体应用，但在彩信多媒体资源的安全上目前仍然没有考虑，比如用户的视频、相片等资料，应该有良好的安全措施，避免被任意下载，这是接下去需要

考虑的事情。

2. 信息智能化，目前短信和彩信的回复都是事先由管理员或客服人员配置好的，但往往有很多用户会把短信或彩信发送到错误的号码，比如某听众想参加 06688136 这个号码的某电台夜话类节目，但他误发到066881136，这个时候系统应该能智能的提示用户应该发送到哪个号码，而不是现在的直接一个错误提示。
3. 进一步完善工作流在本系统当中的使用，是信息发布和浏览更加合理，并提供多种渠道浏览方式，包括手机、PDA 等。

以上的不足和改进之处是经过此次设计给我带来的收获，指明今后学习探索的方向，我将继续努力，不断完善，希望不久的将来，自己的系统能被更多的电台、电视台、学校、酒吧使用，使我的人生目标得以实现。

致谢

衷心感谢我的导师金伟祖教授，在我硕士学位论文阶段对我的悉心关怀和耐心指导，我的每一点进步都和金老师的言传身教分不开的，他思维活跃，平易近人，对学术问题一丝不苟和广博的学识将使我受益终身。同时他的敬业精神也为我树立了人生的典范。从论文的开题到结束都离不开导师的关心和指导。

同时也感谢我的校外导师李进良教授，他渊博的知识，丰富的经验对我顺利完成论文和项目起了重要作用。在我课题研究的各个阶段都给予了悉心指导。

导师严谨的治学态度和一丝不苟的工作作风为我树立了学习的榜样，在导师无微不至的关怀下，论文得以顺利完成，在此向我的校内以及校外导师致以深深的谢意！

感谢父母在我成长道路上付出的心血和精力;感谢妻子在事业和生活中给予了无私的关爱和理解；在亲人们的大力支持下，我的课题及论文得以顺利进行。感谢我学习和工作过的单位，是她们提供了我工作的机会、提供项目实践机会，让我的课题实践得以顺利完成。

参考文献

[1] 互联网短信网关接口协议（V2.0.0），中国移动通信集团发布,2002年4月

[2] 互联网短信网关接口协议（V3.0.0），中国移动通信集团发布,2004年8月

[3] 短消息业务联网协议1.2,中国联合通信公司，2001年,10月

[4] 中国电信集团公司企业标准（SMGP协议）（v1.38）中国电信集团公司,2003年5月30号

[5] SMAIS系统ICP接口使用手册(ANSI C),中国移动通信集团公司，2001年7月

[6] 小灵通短消息网关API开发手册(JAVA)

[7] 中兴MM7 API用户手册，2004年4月9号

[8] 孙卫琴编著，《精通Hibernate:Java对象持久化技术详解》,,电子工业出版社，2005

[9] Art Taylor 等《JDBC 数据库编程与 J2EE》，电子工业出版社，2004.4

[10]Subrahmanyam Allamaraju 等《J2EE 服务器端高级编程》机械工业出版社 2001.9

[11]Danny Eyers 等《JAVA 数据编程指南》，电子工业出版社，2002.1

[12]萧仁惠 陈锦辉等《JDBC 数据库程序设计》，中国铁道出版社，2004.3

[13]夏昕，曹晓钢，唐勇 编著《深入浅出 Hibernate》，电子工业出版社，2005.6

[14]谭颖华，张云飞，唐勇 译《WebWork in Action 中文版》，电子工业出版社，2006.11

[15]林信良 著《Spring 技术手册》，电子工业出版社，2006.6

[16]Spring Reference Manual:
http://static.springframework.org/spring/docs/1.2.x/reference/index.html

[17]Patrick Lightbody, Jason Carreira, 《Webwork in action》, 2005.9

[18]Velocity Developer Guide:
http://jakarta.apache.org/velocity/docs/developer-guide.html

[19]Dave Crane, Eric Pascarello, Darren James, 《Ajax In Action》, 2006.

[20] Christian Bauer and Gavin King《Hibernate In Action》,2004.