

# 北京交通大学

专业硕士学位论文

基于 IBM MDM Web Services 的病人主索引预约就诊  
系统的设计与实现

The Design and Implementation of the Master Patient  
Index based Doctor Appointment System using IBM  
MDM Web Services

作者：周亚卫

导师：赵 宏

北京交通大学

2010 年 6 月





## 学位论文版权使用授权书

本学位论文作者完全了解北京交通大学有关保留、使用学位论文的规定。特授权北京交通大学可以将学位论文的全部或部分内容编入有关数据库进行检索,提供阅览服务,并采用影印、缩印或扫描等复制手段保存、汇编以供查阅和借阅。同意学校向国家有关部门或机构送交论文的复印件和磁盘。

(保密的学位论文在解密后适用本授权说明)

学位论文作者签名: 周亚卫

导师签名: 赵冬

签字日期: 2010年06月23日

签字日期: 2010年6月23日



中图分类号: TP311.52

UDC: 004.41

学校代码: 10004

密级: 公开

# 北京交通大学

## 专业硕士学位论文

基于 IBM MDM Web Services 的病人主索引预约就诊系统  
的设计与实现

The Design and Implementation of the Master Patient Index based  
Doctor Appointment System using IBM MDM Web Services

作者姓名: 周亚卫

学 号: 08122450

导师姓名: 赵宏

职 称: 副教授

工程领域: 软件工程

学位级别: 硕士

北京交通大学

2010 年 6 月



## 致谢

本论文的工作是在我的导师赵宏教授的悉心指导下完成的，赵宏教授严谨的治学态度和科学的工作方法给了我极大的帮助和影响。在此衷心感谢二年来赵宏老师对我的关心和指导。

赵宏教授悉心指导我们完成了实验室的科研工作，在学习上和生活上都给予了我很大的关心和帮助，在此向赵宏老师表示衷心的感谢。

赵宏教授对于我的工作和论文都提出了许多的宝贵意见，在此表示衷心的感谢。

在 IBM 公司工作及撰写论文期间，王勇、郑伟（音译 Wei Zheng）等同事对我论文中的 MDM 调研工作给予了热情帮助，在此向他们表达我的感激之情。

另外也感谢我的家人，他们的理解和支持使我能够在学校专心完成我的学业。





## 中文摘要

信息孤岛问题是一个我国医院信息化建设的普遍现象。在早期的信息系统建设中,大部分医院由于缺乏经验、缺乏协同管理和长期发展大思路,常常是为应付单一应用而上马建设,基本没有考虑到信息共享的问题。后来各类系统发展多了,孤岛问题也就显现了出来。消除信息孤岛的问题在国外已经提出了很多年。解决医疗信息孤岛的问题,最主要的是要以病人信息为中心,建立起全局的病人基本信息的索引。一旦建立起全局的病人主索引系统,并且提供该索引系统的 Web Services,各个医疗机构异构的信息系统之间或者同一医疗机构的不同信息系统之间,就可以通过访问病人索引系统的 Web Services 来扩展遗留的信息系统,从而既保留了原先的投资,又实现了病人基本信息的共享。通过病人索引系统来维护全局的病人基本信息,来保证病人信息的一致性和完整性,从而解决了各个不同的医院信息系统的重复冗余的病人基本信息,作为各个医院信息系统构建数据仓库阶段的一个补充。

本文对病人主索引预约就诊系统进行了需求分析、体系结构设计、详细设计,并使用 J2EE 架构实现了病人主索引预约就诊系统。本系统的实现采用了 B/S 模式,系统服务组件的实现使用 EJB,并采用 Web Services 适配器类对 EJB 服务进行封装,Web 应用层采用 Strut1 实现,Web 层调用使用服务器端的 Web Services,从而实现病人主索引功能和预约就诊功能。

该系统目的主要是通过构建一个以病人基本信息为主的索引管理系统,并且通过该系统实现由社区医院到三级医院的就诊预约,同时实现社区医院查询病人转诊到三级医院的就诊病历,以方便病人就近的选择医疗服务,病人索引管理子系统通过录入不同的医疗系统的病人基本数据,来维护全局的病人基本数据。

**关键词:** 病人索引; 就诊预约; 全局病人基本数据; Web Services

**分类号:** TP311.52

## ABSTRACT

**ABSTRACT:** To solve the problem of medical information island, it is important to build a global patient index system whose main job is to manage the master patient index. To deal with the platform dependant problems of the traditional Enterprise Application Integration that usually cost the customer a big expenditure, we can use the web services to encapsulate the business services. By using web services, not only can preserve the old heterogeneous hospital information system, it can also provide additional business value provided by the new developed services. You can access patient index system by Web Services to extend the legacy information system, which not only retains the original investment, but also realize the sharing of basic information of patients. Using patient index system whose main job is to manage the master data of patient, to ensure the consistency and integrity of patient information, thus solve the redundant duplication of basic information for patients which reside in various hospital information systems. The patient index system can be a supplementation before the Medical Enterprise Service Bus is built. The Master Patient Index based Hospital Reservation System is a medical information system whose job is not only managing the master patient data, but also realize the hospital reservation process. By using this system, the general practitioner in a community medical center can help a patient to appoint a day with a specialist physician in a hospital, after the patient visit, the specialist physician record the diagnosis in the system, the general practitioner in a community medical center then can query for the medical record for the treatment of patient. The system facilitate the patient, the patient can chose a nearby community medical center for basic treatment.

This thesis embodies the analysis, design and implementation of the the Master Patient Index based Dcotor Appointment System. The realization of the system is based on IBM MDM Server Web Services, and Struts1 framework.

**KEYWORDS:** Patient Index; Doctor Appointment; Master Patient Data; Web Services

**CLASSNO:** TP311.52

## 序

本论文的选题源于在 IBM 公司实习时参与的一个项目。在该医疗系统正式开发之前,我曾经在 IBM 公司就 MDM Server 进行了相关的研究和学习,并通过做相应的实验验证了项目实现所需的 MDM Server 的特性,尤其是其 Web Services 接口的使用和基于 MDM 的软件开发,通过一个多月的调研工作以后,基本掌握了基于 MDM 进行软件开发的流程,从而降低了项目开发的风险性。

整个项目主要是能够构建一个以病人基本信息为主的索引管理系统,并且通过该系统实现由社区医院到二三级医院的就诊预约,同时实现社区医院查询病人转诊到二三级医院的就诊病历,以给病人提供就近医疗服务,方便病人就近选择医疗服务,系统同时提供了查询病人在健康中心的体检信息的功能,尽管该功能是可选的。病人索引管理子系统可以维护全局的病人数据,该子系统的服务是实时的,用以在某一时刻内提供病人准确的核心的病人数据。系统服务是通过 EJB 实现的,并且通过实现以供 Web Services Adapter Web 应用程序,实现外部用户通过 Web Services 来调用系统服务。客户端通过 Web 应用程序调用 Web Services 的方式,提供系统功能实现。

在该系统开发的过程中,通过销售人员获得了加拿大 MDM Server 架构组成员 Wei Zheng 的联系方式,在遇到 MDM 方面的技术问题时,我通过电子邮件向他进行咨询,他都很热情的给予了回复和专业的帮助,项目组的组长王勇在技术上面同样给予了我极大地帮助。在需求方面,在项目组长的组织下,和需求方进行了若干次正式的交流,在交流结束以后,我们进行软件的原型开发,整个项目的完成经历了十几次的迭代开发才得以完成。

该系统只是 IBM 智慧的医疗商业解决方案的冰山一角,要实现医疗行业的病人数据在真正意义上的共享,不仅需要医疗标准的统一,以达到数据格式的统一,而且在技术上还要实现医疗行业的 Enterprise Service Bus,以此来达到真正意义上的医疗行业的数据共享,给病人就医提供优质的服务。

## 目录

中文摘要.....	iii
ABSTRACT.....	iv
序.....	v
1 绪论.....	1
1.1 项目背景.....	1
1.2 技术选择的原因.....	2
1.2.1 基于传统数据集成的解决方案.....	2
1.2.2 基于 Web Services 的解决方案.....	3
1.2.3 IBM MDM Server.....	5
1.2.4 Web 层开源框架 Struts.....	5
1.3 项目应用价值.....	7
1.4 论文主要内容.....	8
2 需求分析.....	10
2.1 项目介绍.....	10
2.2 功能性需求.....	11
2.2.1 用例识别.....	11
2.2.2 用例图.....	12
2.2.3 用例描述.....	12
2.3 非功能性需求.....	17
3 系统分析.....	20
3.1 领域模型.....	20
3.1.1 Party 包结构.....	20
3.1.2 Identifier 包结构.....	21
3.1.3 Location 包结构.....	21
3.1.4 Link 包结构.....	22
3.1.5 Suspect 包结构.....	23
3.2 系统顺序图 (SSD) .....	23
3.2.1 注册病人.....	24
3.2.2 查询病人基本信息.....	24
3.2.3 记录健康信息.....	25

3.2.4	修改病人基本信息 .....	25
3.2.5	删除病人基本信息 .....	26
3.2.6	合并病人基本信息 .....	26
3.2.7	转诊病人 .....	27
3.2.8	添加转诊病历信息 .....	27
3.2.9	查看转诊病历信息 .....	28
3.3	操作契约 (Operation Contracts) .....	28
3.3.1	addPerson .....	28
3.3.2	searchPerson .....	29
3.3.3	modifyPerson .....	29
3.3.4	deletePerson .....	30
3.3.5	searchPersonWithSimilarRecords .....	31
3.3.6	getMergePersons .....	31
3.3.7	mergePerson .....	32
3.3.8	hospitalRegister .....	32
3.3.9	writeCase .....	33
4	体系结构设计 .....	34
4.1	逻辑视图(Logical View) .....	34
4.1.1	表示层 (Presentation Layer) .....	35
4.1.2	应用控制层 (Application Controller Layer) .....	36
4.1.3	服务层 (Services Layer) .....	37
4.1.4	领域模型层 (Domain Model) .....	38
4.2	流程视图(Process View) .....	39
4.3	部署视图(Deployment View) .....	40
5	详细设计 .....	42
5.1	组件设计 .....	42
5.2	组件 Web Services 适配设计 .....	49
5.3	Web 应用设计 .....	52
5.3.1	注册病人功能设计 .....	53
5.3.2	查询病人功能设计 .....	54
5.3.3	合并相似病人功能设计 .....	55
5.3.4	病人预约就诊功能设计 .....	57
5.3.5	记录预约病历功能设计 .....	59
6	系统实现 .....	61

6.1	病人基本信息管理.....	61
6.1.1	注册病人.....	61
6.1.2	合并病人.....	63
6.1.3	病人基本信息的查看、修改与删除.....	64
6.2	病人预约就诊.....	65
6.2.1	预约请求.....	66
6.2.2	查询预约记录.....	66
6.2.3	查询预约病历.....	67
7	结 论.....	69
7.1	完成工作.....	69
7.2	工作体会.....	69
7.3	后续工作.....	70
	参考文献.....	71
	作者简介.....	72
	独创性声明.....	73
	学位论文数据集.....	74

# 1 绪论

## 1.1 项目背景

随着世界信息技术的迅速发展与我国医疗卫生事业的深化改革,以及卫生部主持的国家金卫工程的实施使我国医院管理信息化的进程大大加快,我国医疗行业的信息化建设近年来也取得了较大进展。但是,随着老百姓对医疗质量要求的不断提高,医疗机构需要为病患者提供更人性化、更合理的服务,所以,实现全面信息共享已是大势所趋,越来越多的医院认识到,只有通过信息化建设,逐步建立信息化医院,才能支持医院的可持续发展,从而大力提高医院综合效益和运行效率。医疗卫生事业关乎国计民生,而医疗信息化则是医疗卫生事业能否在信息时代更好地服务于患者、服务于社会的重要支撑,正受到政府部门及越来越多的相关企业及组织的关注。在新一轮的医改浪潮中,医疗服务信息化将是必然的趋势。新医改方案中提出了通过信息化手段,建立医院间的资源共享,从而实现医疗服务资源的最优整合和最大协同效应<sup>[1]</sup>。

新医改方案的颁布,为医疗信息化建设的推进提供了强大的支持力量。在新医改投入的 8500 亿元中,仅信息化 IT 建设就将分得近千亿的大蛋糕。而信息孤岛、标准不统一等问题,还在困扰着很多医院的信息化推进进程<sup>[2]</sup>。

信息孤岛问题是一个我国医院信息化建设的普遍现象。在早期的信息系统建设中,大部分医院由于缺乏经验、缺乏协同管理和长期发展大思路,常常是为应付单一应用而上马建设,基本没有考虑到信息共享的问题。后来各类系统发展多了,孤岛问题也就显现了出来。目前国内的 HIS 厂商有 500 多家,各个 HIS 系统的命名和数据结构均不相同,造成了各个医院使用不同的 HIS 系统。多种系统平台和非标准信息格式阻碍了同行间及行业间的信息交流。消除信息孤岛的问题在国外已经提出了很多年。由于我国医疗行业的信息化进程较为落后,因此表现为一个部门的系统可能是医院的信息孤岛,一家医院的系统可能是地区的信息孤岛,甚至一个地区的系统可能是全国的信息孤岛<sup>[3]</sup>。

解决医疗信息孤岛的问题,最主要的是要以病人信息为中心,建立起全局的病人基本信息的索引。一旦建立起全局的病人主索引系统,并且提供该索引系统的 Web Services,各个医疗机构异构的信息系统之间或者同一医疗机构的不同信息系统之间,就可以通过访问病人索引系统的 Web Services 来扩展遗留的信息系统,从而既保留了原先的投资,又实现了病人基本信息的共享。通过病人索引系

统来维护全局的病人基本信息, 来保证病人信息的一致性和完整性, 从而解决了各个不同的医院信息系统的重复冗余的病人基本信息, 作为各个医院信息系统构建数据仓库阶段的一个补充。

病人主索引系统的基本思想是: 建立一个以病人为主题的数据库, 在这个数据库中, 每个病人都对应唯一的一个标识符。跟 EMPI 通过注册关联的企业可以包括医疗中心、门诊部、医学部和康复中心等。所有的注册系统可以通过一些标识符从 EMPI 中检索出病人的信息。这一检索过程可以有前台的记录员通过 EMPI 的搜索功能来完成, 也可以在病人注册的过程中由系统来完成。EMPI 将具有确定性索引, 这样可以通过姓名、社会保险号、出生日期和性别的组合来进行准确匹配查找。其他的检索条件可以是姓名的前四个字符或者其他的关键标识符。最好的搜索机制是通过概率公式来进行概率搜索。通过这种方法改善了匹配条件。通过明确地标识每一个病人, 来改善对病人的护理。如果有临床资料库, 身份证可以用来唯一标识一个人, 它是非常有价值的信息的一部分。在治疗的过程中, 能得到病人的历史病例信息是十分有价值的, 历史病例信息可以包含任何诊疗, 康复, 或住院的记录信息<sup>[4]</sup>。

本系统就是首先建立病人的主索引, 并且利用主索引的 Web Service 来实现预约就诊流程。

## 1.2 技术选择的原因

医疗行业的信息系统软件有一个特点, 就是需要不断升级, 就是需要不断适应我国医疗制度改革的变化, 不断的适应国家和地方政府的法规变化。这种不断增加的需求和各种管理规定的更新要求系统具有更加灵活的应变能力。现阶段, 最重要的是实现对病人数据的共享, 这可通过对病人基本信息进行集成来实现。

实现不同系统的数据集成共享, 可采用传统的 EAI 方式, 也可采用基于 Web Services 的集成技术实现。

### 1.2.1 基于传统数据集成的解决方案

数据集成发生在企业内的数据库和数据源级别。通过从一个数据源将数据移植到另外一个数据源来完成数据集成。数据集成是现有 EAI 解决方案中最普遍的一个形式。然而, 数据集成的一个最大的问题是商业逻辑常常只存在于主系统中, 无法在数据库层次去响应商业流程的处理, 因此这限制了实时处理的能力。

并且, 传统的 EAI 都是通过编制接口程序来实现, 但是这只是涉及简单的两



个系统相连。假设有  $N$  个系统，如果这些系统都需要互联，那么需要  $N \times (N-1)$  个接口，每增加一个系统，理论上接口是成倍增加的，这是不可想象的。接口的设计和维持需要耗费大量的时间和人力成本，既不能保证接口间信息互传时没有丢失，也无法做到实时监控。当一个参数发生变化的时候，相应的接口程序就必然要做相应的修改，所以使用这种方法来实现信息共享所带来的难度可想而知<sup>[3]</sup>。

最重要的一点是，客户希望保留原有的医院信息系统，只以扩展的形式来改变原有系统，实现病人基本信息的共享，以实现预约流程处理。

### 1.2.2 基于 Web Services 的解决方案

Web 服务以面向对象技术为基础，对数据和编程元素进行封装，以便不同的基于 Web 的应用程序能够访问，利用 Web 服务，比如 SOAP，浏览者可以从其它同样基于 SOAP 的站点获取价格信息，并且传送给客户进行比较。Web 服务利用不同目标系统，触发不同的行为和信息来提供后台服务，SOAP 以及其他 Web 服务采用 RPC 和其他类似的技术。SOAP 同样也基于 XML。

由于 Web 服务是由一系列标准所组成的，所以 Web 服务集成各种应用的方法是标准化的，具有较好的通用性和兼容性，同时面向对象和 XML 等相关技术的采用，使得 Web 服务具有更好的跨平台性，可以更好的满足分布式集成的要求。Web 服务的集成结果是一种松耦合的集成模式。它通过建立涵盖服务通信 (SOAP)、服务描述 (WSDL) 和服务发现 (UDDI) 等标准实现应用集成的框架。另外，由于和 Web 技术的结合，使得 Web 服务对 Web 应用具有很好的协调性。再者，Web 服务能够更好的满足企业以客户为中心的协同工作<sup>[6]</sup>。

Web 服务最大的特点就是跨平台，这是 COM, CORBA 都没有很好解决的问题。java 和 .net，甚至其他开发工具的交互，在 Web 服务中得到了解决，而且移动设备也可以使用，所以说 Web 服务非常适合应用于松耦合分散式的跨平台跨技术的场合。

Web Services 的 SOAP-EJB 调用过程<sup>[6]</sup>如下所示：

- SOAP 协议处理程序接受到一个外来的 SOAP 消息。SOAP 协议处理程序解析消息并决定需要调用的 EJB 实例。这个过程可以有选择的是否现在 JNDI 查找。处理程序中可能已经缓冲了相关 EJB 实例的引用。
- SOAP 处理程序以合适的输入参数调用相应的 EJB 实例。EJB 可以是 Stateless Session, Stateful Session, 或者实体 Bean。如果 EJB 是有状态会话 Bean 或者是实体 Bean，则实体 Bean 必须以一定的方式保存会话

标志或者实体 Bean 的主键。

- EJB 调用任意的后台资源, 后台资源可以使其他的 EJB, 数据库, 或者是 JMS 描述。EJB 中包括了 Web Service 的业务实现。
- EJB 将相应消息发送给 SOAP 处理程序。SOAP 处理程序将 EJB 响应消息中转换成基于 XML 格式的响应消息, 如果 EJB 抛出了异常, 那么 SOAP 处理程序产生一个 SOAP 错误消息并发送给客户端。
- SOAP 处理程序为响应消息产生一个 SOAP Envelope 并把它发送给客户端。

#### 两种方案的比较

下面是传统的 EAI 解决方案和 Web 服务之间的一些基本的不同点:

- 简单性: 毫无疑问, 相对于典型的 EAI 解决方案(也许包括分布式技术如 DCOM 和 CORBA), Web 服务更便于设计、开发、维护和使用。既然开发和使用 Web 服务的平台框架已经准备好了, 创建跨越多个应用程序的商务流程处理将变得相对简单。
- 开放标准: 不像有所有权的 EAI 解决方案, Web 服务是基于开放标准诸如 UDDI、SOAP、HTTP 的。这个可能是导致 Web 服务被广泛接受的最重要的因素。事实上基于现存的开放标准消除了企业潜在地为了支持新出现的 Web 技术的投资的需要。
- 灵活性: 既然 EAI 解决方案需要点对点集成, 一端的改变必须告知另外一端, 这自然使集成变得非常的生硬, 同时也是浪费开发人员的时间的。基于 Web 服务的集成是非常灵活的, 因为他是建立在发布服务的应用程序和使用服务的应用程序之间的松散耦合。
- 便宜: EAI 解决方案, 诸如消息中介, 其实施是非常昂贵的。而 Web 服务的实施则会变得便宜而快速。
- 范围: EAI 解决方案, 诸如消息中介, 把应用程序作为一个单个的实体来集成。然而 Web 服务允许企业把大的应用划分为小的独立的逻辑实体并且包装他们。举例来说, 企业可以为一个 ERP 应用的不同的商业组件进行包装。如订单管理、接受购买订单、订单情况、订单确认、帐户接收、帐户支付等等。
- 动态: Web 服务通过提供动态的服务接口来实施一个动态的集成。然而传统的 EAI 解决方案都是静态处理的。

总之, 传统的 EAI 是一种紧耦合集成模式, 比较适用于那些对性能要求较高的、需要多种层次集成的应用集成系统。Web 服务是一种标准化的松耦合集成模式, 比较适用于那些需要更大的灵活性, 改动频繁的应用集成系统<sup>[7]</sup>。

### 1.2.3 IBM MDM Server

IBM Master Data Management Server 作为企业应用。IBM Master Data Management Server 是采用 J2EE 架构的企业应用,它旨在给企业的客户数据、产品数据和合同信息提供一个统一的操作视图,并且提供一个可以通过多种方式处理更新的应用环境。它使得那些前台服务系统和后台服务系统实时的结合起来,从而保证了主数据的唯一性和完整性。它主要采用基于组件的 XML 和 J2EE 架构来快速的和其它系统进行集成,以提高灵活性和可扩展性。

IBM MDM Server 作为开发平台。IBM Master Data Management Server 同时提供了可以集成到 Rational Software Architecture IDE 的插件工具,使用这个插件,可以进行基于组件的可视化开发,从而使 IBM MDM Server 成为一种开发平台。使用该平台,可以生产对 MDM Server 的添加和扩展,平台会自动生成相应的 EJBs、Java 代码、PureQuery Java 类、Web Services、属性文件、sql 查询语句、dtd 和 xsd 文件。然后开发人员只需根据业务逻辑重新定制和编写自动生产的代码,以实现基于组件式的开发,并实现最大限度的软件复用。

IBM MDM Server 的业务逻辑层主要采用 EJB 来实现。因为 EJB 是用于开发安全、可扩展、事务型和多用户组件的一种分布式组件模型。EJB 可以部署在分布式多层体系结构环境中的服务器端的软件组件,它允许把应用程序逻辑与系统级服务分开,使系统开发人员集中精力编写具体的业务处理问题,而不必关心系统的编程。每个 EJB 可由一个或多个 Java 对象组成,每个 EJB 是按照相同的规范开发的,Bean 之间可以相互调用,实现任务的分解和进行业务处理,具有非常高的扩展性。

在一个公司或者行业中的业务实体中的关键因子,它们会被各种系统(CRM、ERP 等)用到。

MDM Server 提供了一下几种方式进行编程访问<sup>[9]</sup>:

- RMI
- JMS
- Batch
- Web Services

根据项目需要,为了保留各个医院遗留的医疗信息系统,并且提供跨平台和跨语言的特性,本系统将采用 MDM Server 的 Web Services 调用方式。

### 1.2.4 Web 层开源框架 Struts

Struts 是一种开源框架，可用来构建 Web 应用程序，它基于流行的 Model-View-Controller (MVC) 设计范型。该框架构建在一些标准的技术之上，比如 Java Servlets、JavaBeans、ResourceBundles 和 XML，并且可提供灵活和可扩展的组件。Struts 以 ActionServlet 的形式实现了 Controller 层，并建议使用 JSP 标记库构建 View 层。Struts 通过 Action 类提供了围绕 Model 层的包装器。图 1-1 展示了基于 Model-View-Controller 设计的 Struts 框架。

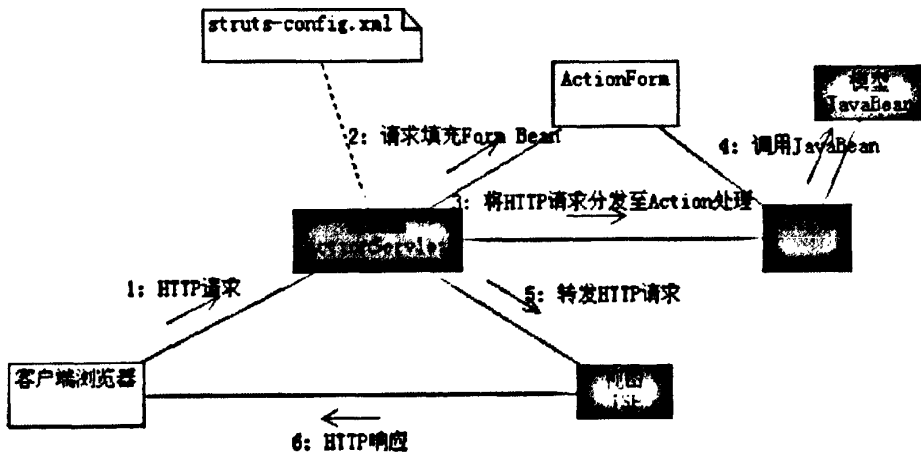


图 1-1 Struts 和 MVC

图解说明：其中不同颜色代表 MVC 的不同部分：红色（控制器）、紫色（模型）和绿色（视图）。

Struts 组件：

- **Action**。应用程序的每个 Action 都会扩展 Struts 的 `org.apache.struts.action.Action` 类。这些 Action 类为应用程序的 Model 层提供了一个接口，充当围绕业务逻辑的包装器。每个 Action 类都必须向 `perform()` 方法提供其特定于用例的实现。`perform()` 方法经常返回类型 `ActionForward` 的一个值。
- **ActionForm**。应用程序的 `ActionForm` 扩展了 Struts 的 `org.apache.struts.action.ActionForm` 类。`ActionForm` 是一些封装和验证请求参数的简单 `JavaBean`。要验证请求数据，`ActionForm` 的 `validate()` 方法必须给出一个特定于该情况的实现。`ActionForm` 作为运载工具，向 Action 类提供请求数据。一个 JSP 对象与各自的 `ActionForm` 对象相结合，构成应用程序的 View 层。在该层，几乎 JSP 对象的每个表单字段都映射到相应的 `ActionForm` 的属性。
- **JSP 定制标记库**。JSP 定制标记库是用标记表示的一组行为的集合。这是

JSP Specification 1.1 的一个强大特性；它将其他应用程序层的表示区别了开来。这些库易于使用，而且可以以一种类似 XML 的方式来读取。只要尽量少地在其中使用 Java scriptlet，就可以轻松维护 JSP 组件。Struts 提供的 JSP 标记包括 HTML、逻辑和 bean 标记。

- ActionErrors. 可以使用 ActionError 来支持异常处理。ActionError 捕捉应用程序异常，并将其传送给 View 层。每个异常都是一个 ActionError 实例的集合。ActionError 可以封装错误消息，而 Presentation 层中的 `</html:errors>` 可以呈现 ActionError 集合内的所有错误消息。

首先，控制器（ActionServlet）进行初始化工作，读取配置文件（struts-config.xml），为不同的 Struts 模块初始化相应的 ModuleConfig 对象。比如配置文件中的 Action 映射定义都保存在 ActionConfig 集合中。相应地有 ControlConfig 集合、FormBeanConfig 集合、ForwardConfig 集合和 MessageResourcesConfig 集合等。

控制器接收 HTTP 请求，并从 ActionConfig 中找出对应于该请求的 Action 子类，如果没有对应的 Action，控制器直接将请求转发给 JSP 或者静态页面。否则控制器将请求分发至具体 Action 类进行处理。

在控制器调用具体 Action 的 execute 方法之前，ActionForm 对象将利用 HTTP 请求中的参数来填充自己（可选步骤，需要在配置文件中指定）。具体的 ActionForm 对象应该是 ActionForm 的子类对象，它其实就是一个 JavaBean。此外，还可以在 ActionForm 类中调用 validate 方法来检查请求参数的合法性，并且可以返回一个包含所有错误信息的 ActionErrors 对象。如果执行成功，ActionForm 自动将这些参数信息以 JavaBean（一般称之为 form bean）的方式保存在 Servlet Context 中，这样它们就可以被其它 Action 对象或者 JSP 调用。

Struts 将这些 ActionForm 的配置信息都放在 FormBeanConfig 集合中，通过它们 Struts 能够知道针对某个客户请求是否需要创建相应的 ActionForm 实例。

Action 很简单，一般只包含一个 execute 方法，它负责执行相应的业务逻辑，如果需要，它也进行相应的数据检查。执行完成之后，返回一个 ActionForward 对象，控制器通过该 ActionForward 对象来进行转发工作。我们主张将获取数据和执行业务逻辑的功能放到具体的 JavaBean 当中，而 Action 只负责完成与控制有关的功能。遵循该原则，所以在上图中将 Action 对象归为控制器部分<sup>[9]</sup>。

### 1.3 项目应用价值

在一个复杂的医疗服务体系内,企业级的患者身份识别方法通过医疗服务机构(医院、主治医生、专家、流动保健中心等)将患者的临床和管理信息结合,进行动态和持续的交流,最终将信息提供给为患者服务的医护人员,使他们为患者提供相应的医疗保健<sup>[10]</sup>。正确执行识别管理-识别患者并使之与其医疗信息对应起来是处理病人信息及应用电子病历的先决条件。该过程类似于在银行通过正确的账号进入银行账户<sup>[11]</sup>。不过,当信息来源于已经建立很长时间的多级信息系统时,就会产生存在多个患者身份标识以及同一患者的既往病历信息分散在许多医疗机构的复杂问题。

在这种多极信息系统中对患者身份进行管理是所有患者统一身份识别管理系统的重要能力之一。在复杂医疗服务系统内进行患者统一身份识别是所有患者身份识别管理的关键业务需求。每个人都可以有许多不同类型的识别方式,比如个人医院账户、病历号、药房信息系统产生的识别码等。在患者登录过程中(预约就医、注册、管理等),患者的身份通常通过一系列识别符或者关键信息(姓名、地址、出生日期等)得以确认,匹配的结果可能是唯一的,也可能不是唯一的。

因此,医院的病历管理部门面临艰巨的挑战。他们必须通过合并或不合并患者记录以及建立或者删除数据库之间的患者信息识别符不匹配的问题。所有这些工作都不允许有错误,因为错误的患者身份识别可能会导致灾难性的后果。

通过本系统,可以管理病人基本信息。在提取数据到本系统的过程中,系统会自动应用匹配规则,产生相似的病人记录,供管理人员选择合并或者删除重复记录。本系统提供了病人基本信息的全局视图,并可以实现跨医疗机构的预约申请,并且提供查询预约病历记录,从而是病人享受多个不同医疗机构的服务,提供服务质量。

本系统采用基于组件的开发技术,大大提高了开发效率,提高了可重用性和灵活性。系统采用 Web Services 实现,从而满足了跨平台、跨语言的特殊需求。客户需求在将来发生变化时,可在不影响遗留系统原有功能的基础之上进行功能的扩展开发。

## 1.4 论文主要内容

第一章 绪论。该部分分析了项目的背景和应用意义,并对项目使用的技术进行分析比较和概述,最后阐述了项目的应用价值。

第二章 需求分析。该部分对系统的需求进行了详细的阐述,主要包括项目介绍、功能性需求和非功能性需求。功能性需求部分采用了用例进行了详细的描述。

第三章 系统分析。该部分是需求分析到系统设计的过渡。从用例中提取概念

类，构建领域模型以对业务领域的概念类进行可视化表示；从用例中提取系统事件构建和处理事件的系统行为，构建系统顺序图；结合领域模型中的概念类、系统顺序图中的系统行为和用例中的前置条件与后置条件，对系统行为进行详细描述，构建操作契约。

第四章 体系结构设计。构建逻辑视图，对系统中最重要类、服务包和子系统的划分进行描述；构建流程视图，描述系统中执行的任务和任务之间的交互；构建部署视图，描述部署的物理节点，以及物理节点任务分配的情况。

第五章 详细设计。对系统的服务组件进行详细设计，包括类的详细设计、动态顺序图的设计和 EJB 的详细设计；对系统 Web 应用进行功能的详细设计，实现用例。

第六章 系统实现。根据系统的详细设计进行编码实现，提供功能的具体实现效果。

第七章 结论。陈述实现系统和编写论文所完成的工作，以及工作体会和后续的工作说明。

## 2 需求分析

### 2.1 项目介绍

现在,不同地区的医疗部门,或者同一地区的不同医疗部门,甚至是同一医疗单位内部,往往存在着异构的医疗信息系统。并且,因为各个医院的医疗需求各不相同,即便是在将来,要想给各个不同的医疗部门提供一个统一的软件系统简直是一场噩梦。所以,对于一个唯一的一名病人,能够在不同的医疗组织之间共享、交换彼此的病人数据更显得意义重大。

不采用软件系统的帮助,手工的核对不同的医疗信息系统中的病人数据是非常费时的,从而提高了医疗部门的费用。但是,如果不去共享病人数据,在极短的时间间隔之内,病人不得不在不同的专科医院之间进行相同的医疗检查科目。这不仅给病人带来了不便,提高了病人的就医成本,这也是病人所不能容忍的。所以,将不同的医疗信息系统如何集成从而形成全局的电子健康档案(Electronic Health Record)过程中,如何显示地唯一表示一个病人同样是很必要的。

与采用相同的医院信息系统(Hospital Information System)相比,基于病人全局标识符的病人视图更加合理,因为这样既保留了各个医疗单位的遗留系统,又解决了医院信息孤岛的难题<sup>[12]</sup>。

EMPI(Enterprise Master Patient Index)是一个这样的数据库,它包括一个以病人为主题的数据库,在这个数据库中,每个病人都对应唯一的一个标识符。那些跟 EMPI 通过注册关联的企业包括医疗中心、门诊部、医学部和康复中心等。所有的注册系统可以通过一些病人标识从 EMPI 中检索出病人的信息。这一检索过程可以有前台的记录员通过 EMPI 的搜索功能来完成,也可以在病人注册的过程中由系统来完成。EMPI 具有确定性的索引,这样可以通过姓名、社会保险号、出生日期和性别的组合来进行准确匹配查找。其他的检索机制可以通过姓名的前四个字符或者其他的关键标识符来进行检索。最好的搜索机制是通过概率公式来进行概率搜索。通过这种方法改善了匹配条件。将这些匹配条件统一的进行集中存储能带来很多好处。通过明确地标识每一个病人,能够改善对病人的护理水平,提供医院的服务质量。如果有临床资料库,身份证可以用来唯一标识一个人,它是非常有价值的信息的一部分。在治疗的过程中,能得到病人的历史病例信息是十分有价值的,历史病例信息可以包含任何诊疗,康复,或住院的记录信息。由集中管理的 EMPI 带来的好处是巨大的。关于 EMPI 值得注意的一点是,它需要一个维



护到位的管理机制、工作人员和流程的保障。通过每年审核 EMPI 来保证数据质量是一个不错的方法。

IBM Infosphere MDM Server 是 IBM 为企业提供一种通用的基于组件的 XML 和 J2EE 技术的主数据管理的框架与应用。它主要给其他系统提供统一的数据视图管理功能,以达到快速地与其他系统集成目的,从而使企业各部门在不必修改和抛弃原有系统的情况下,实现跨部门的数据共享与管理。

该项目以 MDM 为开发平台,并且利用 MDM 提供的 Web Services 来实现医院病人的主索引管理系统,并且提供和电子健康档案系统结合的方案。主要目标是把各个医院应用系统中的病人核心信息进行集成和统一管理,并为各个医院的子系统提供有关病人的核心的、准确的、统一的数据视图,以达到数据统一共享的最终目标。

## 2.2 功能性需求

### 2.2.1 用例识别

系统的上下文描述:体检中心可以利用病人主索引系统记录体检信息。管理员可以检索、添加、修改、删除病人基本信息,合并相似的病人基本信息(当确认数据库中的两个病人是同一个人时)。社区医院可以利用病人主索引系统进行转诊预约,如转诊到某个专科医院某个科室,并提供转诊单。专科医院可以利用病人主索引系统查询转诊预约,并记录预约病历。社区医院可以利用病人主索引系统查询专科医院的预约诊断病历,并提供治疗服务,方便病人就近接受治疗。

采用如下步骤进行用例识别:确定参与者;确定用例;确定参与者和用例的关系;描述单个用例;细化用例。

确定参与者。从系统的上下文中,提取用例的参与者:护士,管理员、体检中心、社区医院和专科医院。

确定用例。从系统的上下文中,确定用例:注册病人(包括添加病人用例和扩展用例合并病人信息)、查询病人、记录健康记录、转诊病人、查看转诊病例、管理病人基本信息和管理预约信息。

确定参与者和用例的关系。参加 2.2 用例图一节。

描述单个用例和细化用例参加 2.3 用例描述部分。

2.2.2 用例图

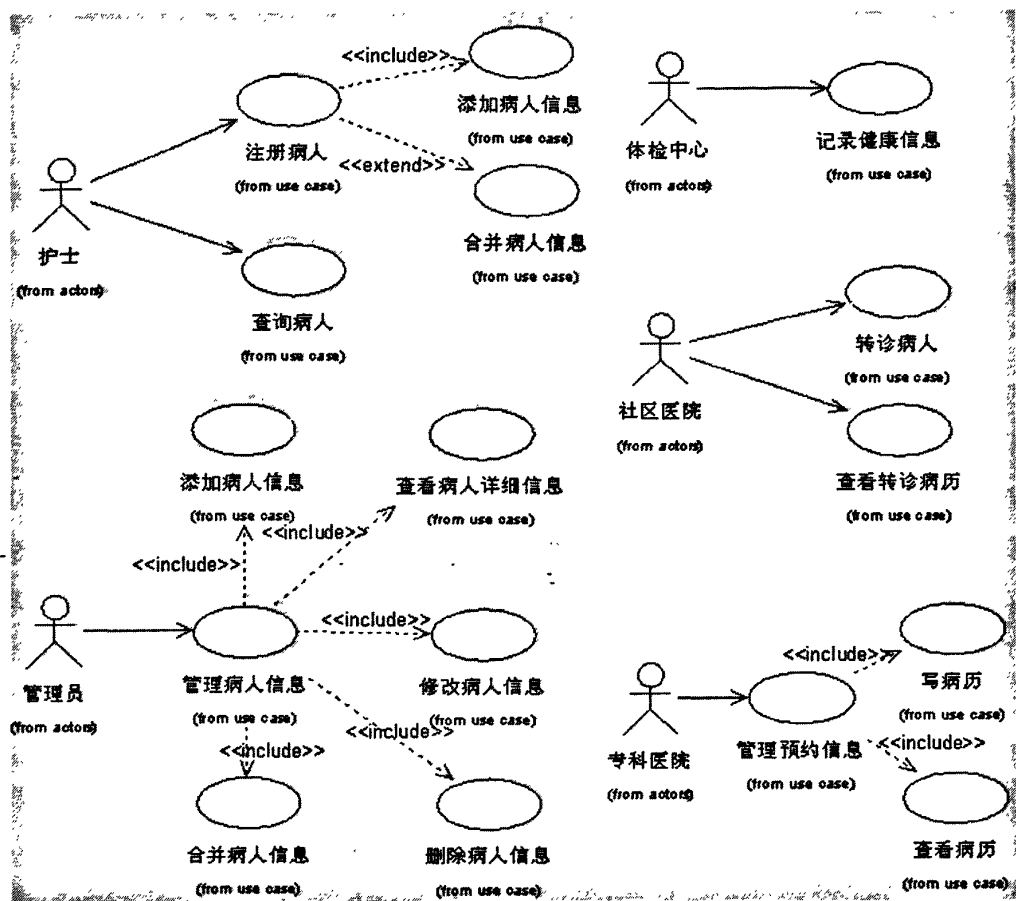


图 2-1 系统用例图

在系统的功能性分析中，采用了用例分析的方式和手段。本部分使用用例视图来捕获用户如何使用系统、子系统和组件的行为<sup>[13]</sup>。

注册病人的用例描述见表 2-1，查询病人的用例描述见表 2-2，记录健康记录的用例描述见表 2-3，转诊病人的用例描述见

表 2-4，管理病人基本信息的用例描述见表 2-5，管理预约信息的用例描述见表 2-6，查看转诊病例的用例描述见表 2-7。

2.2.3 用例描述

表 2-1 用例描述-注册病人

用例名:	注册病人	用例层次:	用户目标
用例 ID:	C-1	主要参与者:	护士

描述:	该用例描述护士当接收到病人的挂号时注册病人信息
前置条件:	用户已经登录到系统
后置条件:	成功注册之后, 创建一条病人信息记录。如果病人信息完全重复, 创建失败; 如果病人信息类似, 创建新的病人记录, 并提示有相似病人信息。
主事件流:	<div>1. 病人填写基本信息表</div> <div>2. 病人提交病人基本信息表给挂号护士</div> <div>3. 护士输入病人信息</div> <div>4. 系统记录病人信息</div> <div>5. 系统查询相似病人信息</div> <div>6. 系统显示添加成功后的病人信息</div>
扩展流:	<div>5a. 相似病人记录</div> <div>存在相似的病人信息时, 当相似度比较低时, 先添加一条新的记录, 该记录存储新添加的病人信息, 并且产生一个相似关联记录, 相似关联记录存储着两个相似病人的相似关联; 如果相似度比较高, 则直接更新相似的病人记录, 而不添加新的病人记录。添加新记录成功以后, 显示相似的病人信息, 由用户选择是否合并病人相关的记录信息。</div>

表 2-2 用例描述-查询病人基本信息

用例名:	查询病人基本信息	用例层次:	用户目标
用例 ID:	C-2	主要参与者:	护士
描述:	该用例描述护士通过查询条件查询病人信息		
前置条件:	用户已经登录到系统		
后置条件:	查询不到用户时, 提示用户查询无此人; 查询成功时, 显示病人记录, 供用户选择查看病人详细基本信息。		
主事件流:	<div>1. 用户输入查询条件</div> <div>2. 用户提交查询条件</div> <div>3. 系统验证查询条件的合法性</div> <div>4. 系统执行不同的搜索规则</div> <div>5. 系统显示查询结果</div>		
扩展流	<div>3a. 条件非法</div> <div>不进行查询, 并提示用户重新输入查询条件</div>		

表 2-3 用例描述-记录健康信息

用例名:	记录健康信息	用例层次:	用户目标
用例 ID:	C-3	主要参与者:	体检中心
描述:	该用例描述体检中心记录病人的体检结果信息		
前置条件:	用户已经登录到系统		
后置条件:	首先建立病人的基本信息,并将病人对应的体检信息进行记录。		
主事件流:	<div>1. 健康中心记录人员输入病人的基本信息和体检结果。</div> <div>2. 用户提交病人的基本信息和体检结果给系统。</div> <div>3. 系统检查数据的有效性。</div> <div>4. 系统添加一条病人的基本信息记录,并关联病人的体检结果记录。</div> <div>5. 系统显示添加成功以后的病人基本信息和病人体检结果。</div>		
扩展流:	<div>3a. 数据无效</div> <div>当健康中心记录人员输入的数据经验证无效时,系统提示数据的有效性,并提示用户重新输入。</div>		

表 2-4 用例描述-转诊病人

用例名:	转诊病人	用例层次:	用户目标
用例 ID:	C-4	主要参与者:	社区医院
描述:	该用例描述社区医院对病人进行转诊的操作		
前置条件:	用户已经登录到系统		
后置条件:	转诊成功之后,给出转诊单,其中信息要包括转诊医院和转诊科室、其预约时间		
主事件流:	<div>1. 社区医院的全科医师查找要进行转诊的病人的基本信息(C-2)</div> <div>2. 查到病人以后列出病人记录,并提示进行转诊操作。</div> <div>3. 全科医师进入转诊操作,系统显示出病人的基本信息和需要填写的转诊医院和对应得相应科室、转诊时间。</div>		

	<div>4. 全科医师输入病人要转诊的医院名称和科室，以及预约时间。</div> <div>5. 系统验证预约时间的合法性。</div> <div>6. 系统进行预约申请。</div> <div>7. 系统显示预约信息，必须包括预约号。</div>
扩展流:	<div>5a. 预约时间非法</div> <div>当预约时间非法时，系统提示用户错误信息，并提示用户重新输入预约时间。</div>

表 2- 5 用例描述-管理病人信息

用例名:	管理病人信息	用例层次:	用户目标
用例 ID:	C-5	主要参与者:	管理员
描述:	该用例描述管理员管理病人基本信息的操作		
前置条件:	用户已经登录到系统		
后置条件:	执行相应的操作，并更新病人的基本数据		
主事件流:	<div>1. 管理员查询病人的基本信息（C-2）</div> <div>2. 系统列出查询的病人记录，并显示操作类型（查看、修改、删除）。</div> <div>3. 系统执行相应的操作，并保证病人的基本信息的一致性和完整性。</div>		
子事件流:	<div>S1: 查看病人详细信息。</div> <div>S1-1. 用户选择查看操作。</div> <div>S1-2. 系统显示病人的基本信息。</div> <div>S2: 修改病人基本信息。</div> <div>S2-1. 用户选择修改操作。</div> <div>S2-2. 系统显示病人的基本信息。</div> <div>S2-3. 用户修改病人的基本信息。</div> <div>S2-4. 用户提交修改后的病人基本信息。</div> <div>S2-4a. 用户提交的数据非法</div> <div>1. 系统提示用户重新输入。</div> <div>2. 系统显示病人的基本信息。</div> <div>S2-5. 系统更新病人的基本信息。</div> <div>S2-6. 系统显示更新后的病人基本信息</div> <div>S3: 删除病人基本信息。</div>		

	<div>S3-1. 用户选择删除操作。</div> <div>S3-2. 系统提示用户进行删除确认。</div> <div>S3-2-1. 用户选择删除操作。</div> <div>系统删除病人的基本信息。并更新搜索结果。</div> <div>S3-2-2. 用户选择取消删除操作</div> <div>系统取消删除操作。并显示操作列表。</div> <div>S4: 合并病人基本信息。</div> <div>S4-1 用户选择合并病人操作</div> <div>S4-2 系统查询相似关联记录</div> <div>S4-3 系统列出相似记录</div> <div>S4-4 用户修改相似记录的相似项，统一条目</div> <div>S4-5 用户选择合并操作</div> <div>S4-6 系统合并两天相似的病人记录为一条</div> <div>S4-7 显示合并后的相似项列表</div>
--	---

表 2-6 用例描述-管理预约信息

用例名:	管理预约信息	用例层次:	用户目标
用例 ID:	C-6	主要参与者:	专科医院
描述:	该用例描述专科医院的专科医师管理预约信息，包括查看病历信息和写病历信息。		
前置条件:	用户已经登录到系统		
后置条件:	专科医师执行相关操作成功，并更新了病人的病历信息。		
主事件流:	<div>1. 专科医师输入病人的预约号。</div> <div>2. 专科医师提交预约号给系统。</div> <div>3. 系统进行查询。</div> <div>3a. 系统查无此预约号</div> <div>1. 提示预约号不存在。</div> <div>2. 回到流程 3</div> <div>3b. 系统查询到此预约号（S4）</div>		
子事件流:	<div>S4. 处理预约信息</div> <div>S4-1 显示预约病人的基本信息</div> <div>S4-2. 查看预约病历信息。</div> <div>1. 用户选择查看病历信息</div>		

	<div>2. 系统显示病历记录详细信息</div> <div>S4-3. 写病历操作</div> <div>1. 用户选择写新病历操作</div> <div>2. 系统显示新病历列表</div> <div>3. 用户填写新病历</div> <div>4. 用户提交病历信息</div> <div>5. 系统保存病历信息</div>
--	--

表 2-7 用例描述-查看转诊病历

用例名:	查看转诊病历	用例层次:	用户目标
用例 ID:	C-7	主要参与者:	社区医院
描述:	该用例描述社区医院的全科医师查看在专科医院开具的转诊病历		
前置条件:	用户已经登录到系统		
后置条件:	全科医师执行相关操作成功，并查询到病人的转诊病历信息。		
主事件流:	<div>1. 全科医师输入病人的预约号。</div> <div>2. 全科医师提交预约号给系统。</div> <div>3. 系统进行查询。</div> <div>3a. 系统查无此预约号</div> <div>1. 提示预约号不存在。</div> <div>2. 回到流程 3</div> <div>3b. 系统查询到此预约号（S4）</div> <div>4. 系统列出转诊病历</div> <div>5. 用户选择查看转诊病历</div> <div>6. 系统执行查询，列出转诊病历详细条目</div>		

2.3 非功能性需求

项目	描述
功能性	<div>日志记录和错误处理。对于所有的错误，都对其进行持久化处理，以分析错误原因。</div> <div>外部规则。对于可变的业务规则，采用可配置的方式，可以在某一业务点上选择不同的业务规则，并且提供业务规则的可定</div>

	<p>制性。从而实现业务的灵活可变性。</p> <p>安全性。用户的操作要进行身份验证，以保证系统数据的安全授权。</p>
可用性	<p>系统实现应采用英文显示，显示界面具有简洁性，具有菜单列表，尽量避免 web 弹出窗口，页面中要给出合适的标题和子标题，避免较长的载入时间，避免用户等待，尽量将重要的信息一次显示给用户，尽量避免滚动条的出现，除非它有必要存在。确保良好的可读性，用清晰的字体并且限制数字用不同的字体出现，采用简单主义，避免额外的信息和视觉元素干扰用户的阅读。</p>
可靠性	<p>设备的硬件故障可能造成本软件不能运行或不能正常进行输入 / 输出等后果，系统的资源不足及网络传输通道阻塞可能造成本软件不能正常运行，并有可能造成机器“死机”，上述故障的处理由用户自行解决。</p> <p>软件在运行过程中产生的数据库错误，将由系统自动记入错误日志，非网络传输引起的错误将由系统管理员或软件开发解决。</p> <p>软件在运行过程中产生的其他错误，将根据情况由软件开发者或软件开发者协助系统管理员解决。</p>
性能	<p>软件应保证系统运行稳定，避免出现系统崩溃；软件必须保证有数据的一致性和完整性，不影响正常业务；软件应尽量做到响应快速、操作简便。</p>
可支持性	<p>不同的医疗信息系统在使用由系统提供的服务时，业务规则也该是可定制的，比如提供查找病人规章的可定制性，以方便用户进行搜索规则的定制功能。</p> <p>不同的客户可能使用不同的医疗信息系统来调用系统所提供的服务，所以要求服务实现采用 Web Services 的方式。</p>
实现约束	<p>系统要求采用 Java 技术实现，服务组件采用 EJB 实现，以提高系统的可移植性和可配置性。</p> <p>为了简化系统开发,采用 IBM Master Data Management Server 进行开发,并且要求采用 Struts 来实现 Web 层,以提高软件的后期可维护性。</p> <p>Web Services 运行时环境应采用 IBM WebSphere JAX-RPC, 而不是采用 Apache Axis。</p>
接口	<p>系统运行的硬件接口如下:</p>



	<div>1. 服务器端： P4 2.8 主频、4096M 内存、40G 硬盘、10 / 100M 网卡以上配置</div> <div>2. 客户端： 最低：PIII400 主频上、512M 内存、10G 硬盘、10 / 100M 网卡以上配置</div> <div>3. 一台路由器</div> <div>4. 远程拨入设备</div> <div>系统运行软件：</div> <div>1. IBM WebSphere Server 6.0</div> <div>2. DB2 9.0</div> <div>3. RedHat Enterprise Linux 5</div>
--	---

### 3 系统分析

#### 3.1 领域模型

系统领域模型的包结构如图 3-1 所示：

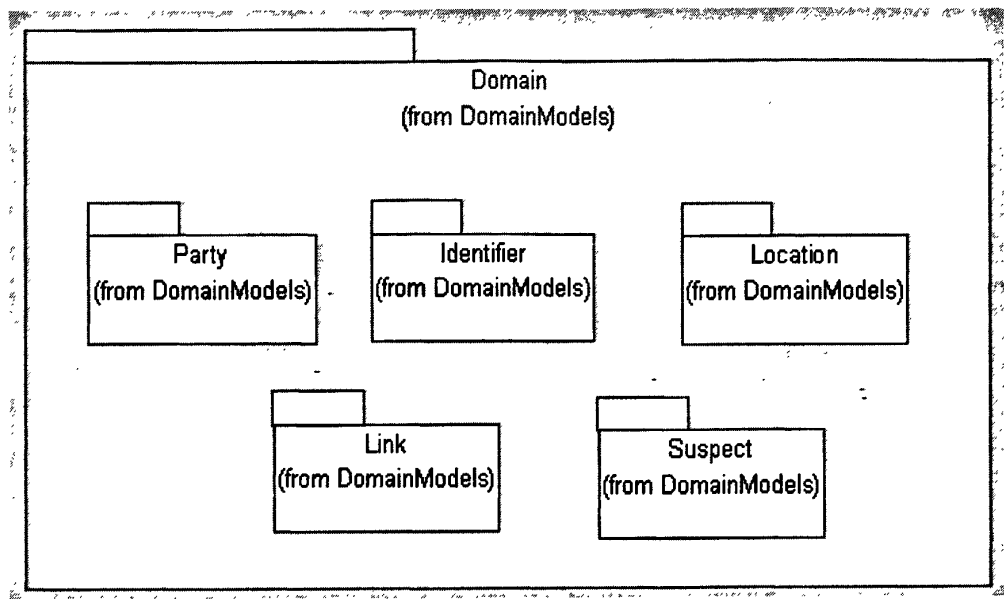


图 3-1 领域模型包图

##### 3.1.1 Party 包结构

单位（或者组织）和人在许多方面存在相似性。它们都有一些共同的属性用来描述各自的特性，比如：地址、电话号码、电子邮件等等。从商业领域的角度，比如在商业合同中，它们统一地被称作“方”，如“甲方”、“乙方”等。

如果建模时将它们看成毫无联系的概念类，当同时涉及到双方时，将使得模型更加复杂。比如：在一个医院科室结中，一个大的科室可能包括专家组，同时包括直接附属医师，在开具病人病历时，病历中的开具方不可能同时是某个医师和专家组。如果将人和组织统一的建模为“Party（某一方）”时，就可以简单的表示这一问题<sup>[14]</sup>。

在图 3-2 中，Person 代表对人的抽象。Organization 表示组织，比如医院、企事业单位。Person Name 表示人的姓名。Organization Name 表示组织名称。Medical Case 表示医疗病历。Health Record 表示体检记录。

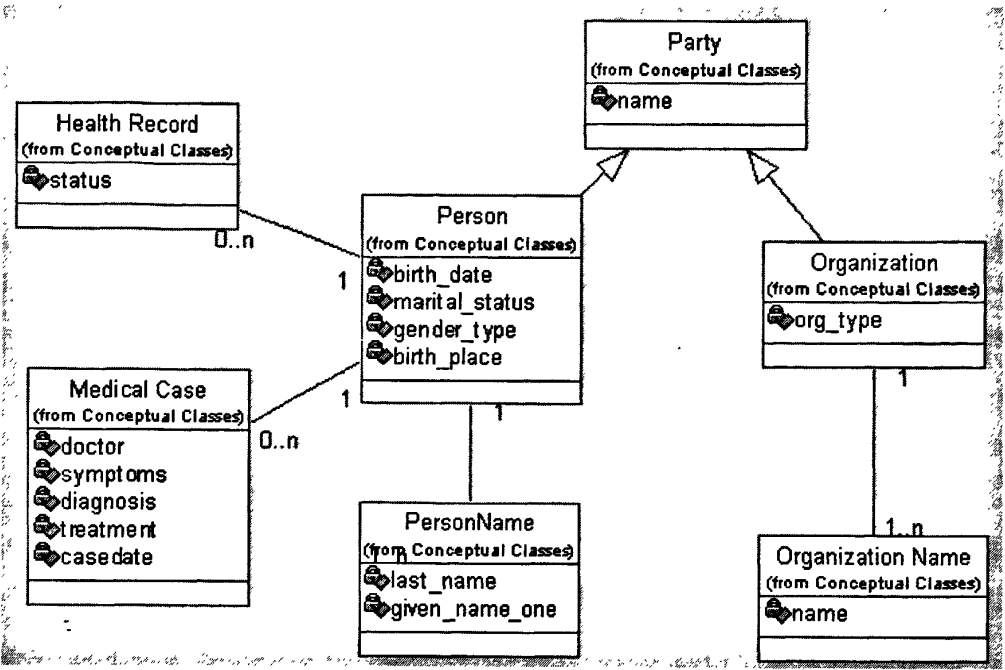


图 3-2 Party 包结构

3.1.2 Identifier 包结构

在图 3-3 中，Identifier 表示 Party 的标识符。比如一个人可以有不同的标识符：身份证号，社会保险号，医保号等。Identification Type 表示标识号码的类型。比如身份证，保险号类型，医保号类型。

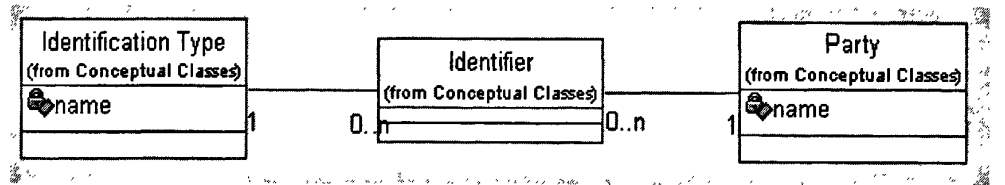


图 3-3 Identifier 包结构

3.1.3 Location 包结构

在图 3-4 中，Location 表示 Party 的地址结构。它是联系方式 Contact Method Location 和住址 Address Location 的父类。Contact Method Location 表示联系方式，主要将特定的联系方式与特定的 Party 联系起来。Contact Method 表示特

定的联系方式，比如手机号码，或者办公邮件。Contact Method Type 表示联系方式的一级种类，比如联系电话，电子邮件等，每一种联系方式可以有不同的子类，比如：联系电话可以有手机号、办公电话、家庭电话等。Contact Method Location Type 表示联系方式的二级子类，比如办公电子邮件，私人电子邮件等。Address Location 表示住址，主要将特定的住址与特定的 Party 联系起来。Address 表示特定的地址，比如：中国河北省石家庄市和平东路 16 号。Country 表示国家，比如 CN 表示中国。Residence 表示地址类型，比如可以是宿舍、办公室、公寓、邮寄信箱等。Province State Type 表示州，或者省份，比如河北省。

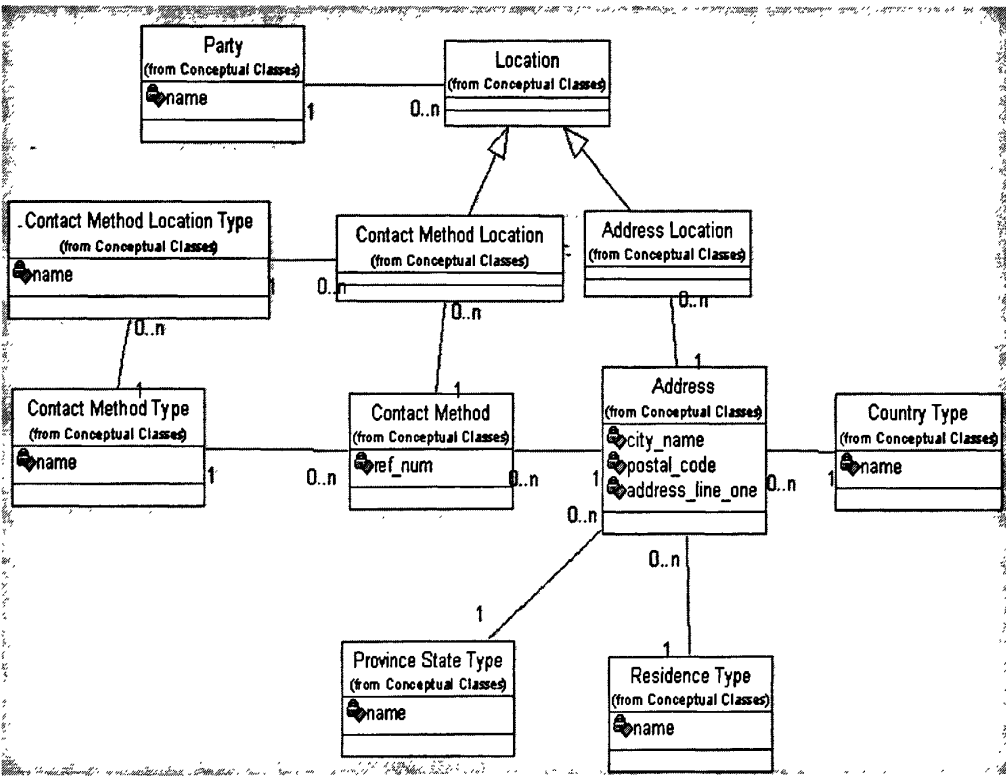


图 3-4 Location 包结构

3.1.4 Link 包结构

在图 3-5 中，Party Equivalency 表示 MDM 系统中某个特定的 Party 与外部系统中某个 Party 的对应关系。例如：MDM 系统中记录着一个人，姓名为 John Frank，外部有一个医院的电子健康档案系统，记录着一个姓名 John Frank 的人，经验证，这两个人属于同一个人。这两个记录就称为等价物（equivalency）。External System Type 表示外部系统，外部是指相对于 MDM 系统而言，比如：在 MDM 系统之外存一个医院的电子健康档案系统，该系统因为使用到了 MDM 的 Web Services，

与MDM系统发生了关联，则成该电子健康档案系统为一种External System。

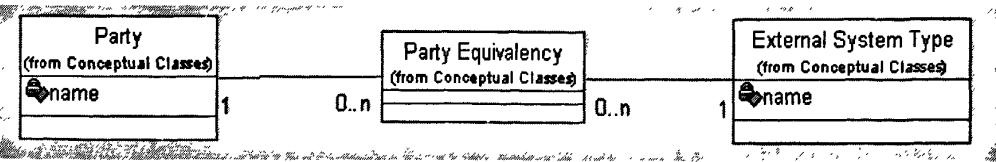


图 3-5 Link 包结构

3.1.5 Suspect 包结构

在图 3-6 中，Suspect 表示相似项，当两个特定的 Party 某些属性相同时，则称这两个 Party 可能相似。Math Relevancy Type 表示在比较两个 Party 时，匹配关联条目的种类，比如：如果两个人的名字中的姓是匹配的，则另其关联类型值是 1；如果两个人的名字中的名是相同的，则另其关联类型值是 2。Suspect Status Type 表示两个不同的 Party 相似的程度，比如：两个 Party 是可疑的重复项，或者是确定的重复项。Suspect Type 表示相似类型的编码，默认分为四个等级：A1，A2，B，C，A1 表示两个 Party 肯定是重复项，A2 表示两个 Party 可能是重复项（可能性比较大），B 表示两个 Party 可能不是重复项（可能性比较小），C 表示两个 Party 不是重复项。Suspect Reason Type 表示相似的缘由类型，比如：两个人相似，可能是因为 SSN 相同，或者是因为姓名和出生日期相同。

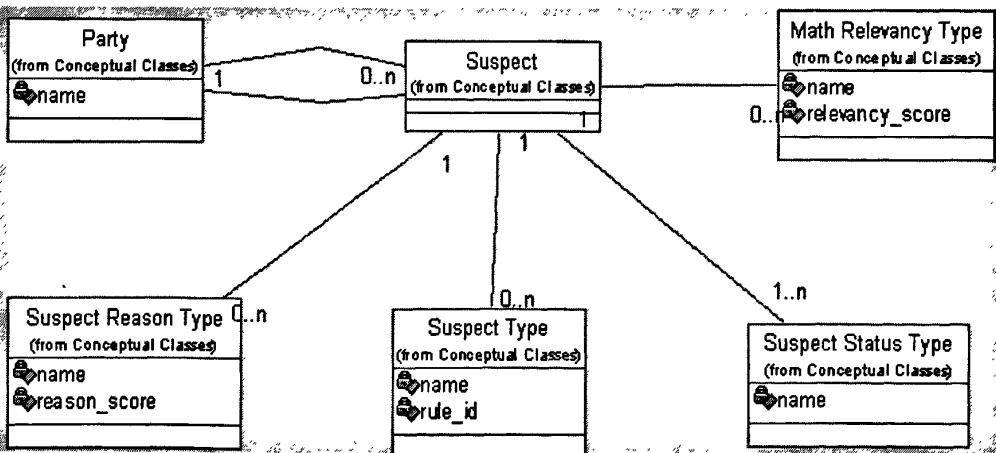


图 3-6 Suspect 包结构

3.2 系统顺序图（SSD）

从用例中提取系统事件和处理事件的系统行为。主要地系统顺序图如下所示：

3.2.1 注册病人

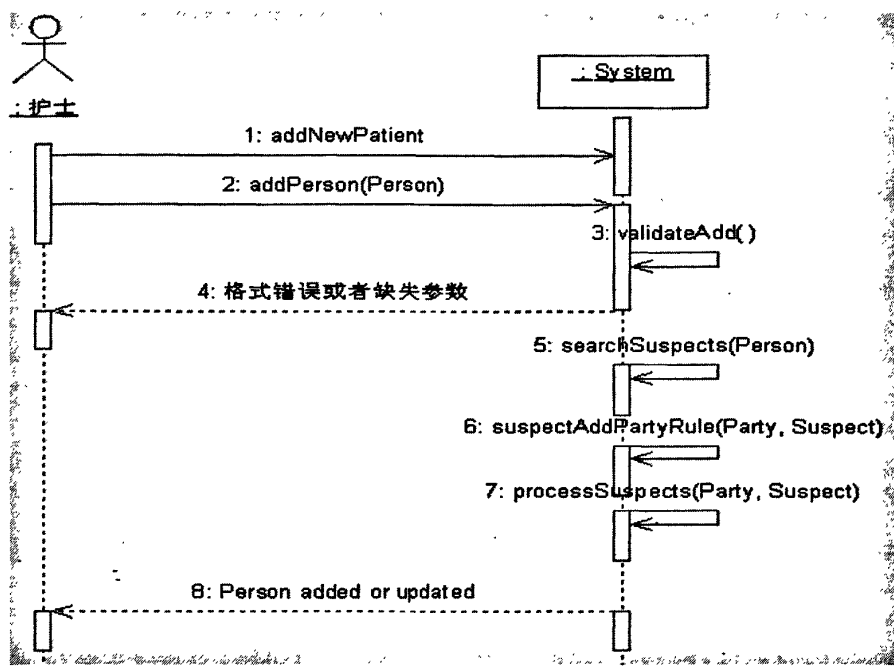


图 3-7 注册病人 SSD

从注册病人的用例描述中提取系统行为 addPerson，关于 addPerson 的详细描述见表 3-1。

3.2.2 查询病人基本信息

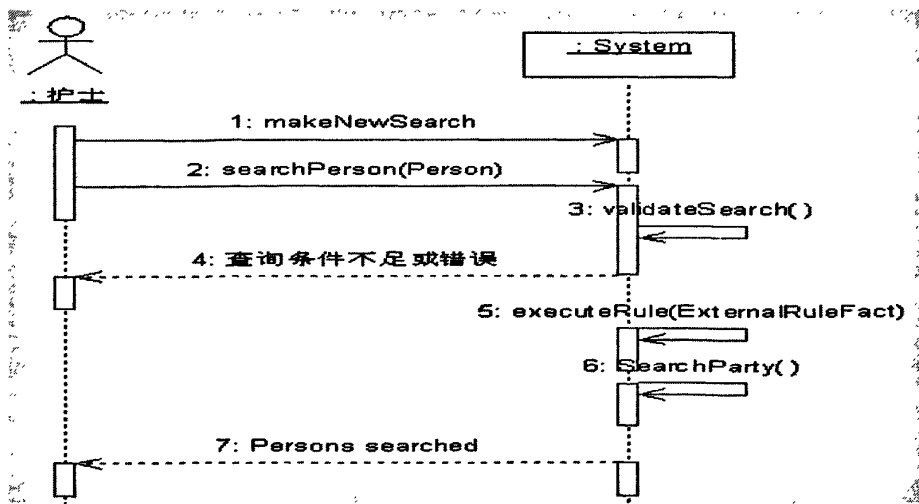


图 3-8 查询病人基本信息 SSD

从查询病人基本信息的用例描述中提取系统行为 searchPerson，关于

searchPerson 的详细描述见表 3-2。

3.2.3 记录健康信息

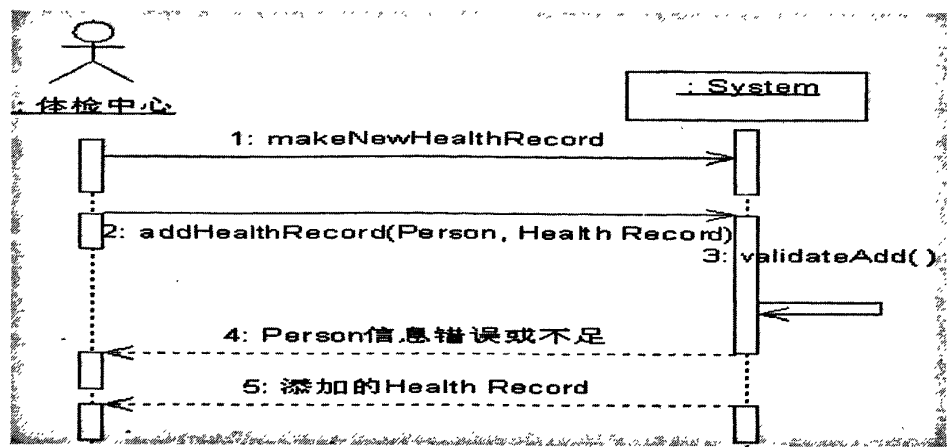


图 3-9 记录健康记录 SSD

记录健康记录是体检中心出发的事件，其系统行为是 addHealthRecord，这将作为 Web 应用程序的系统方法。

3.2.4 修改病人基本信息

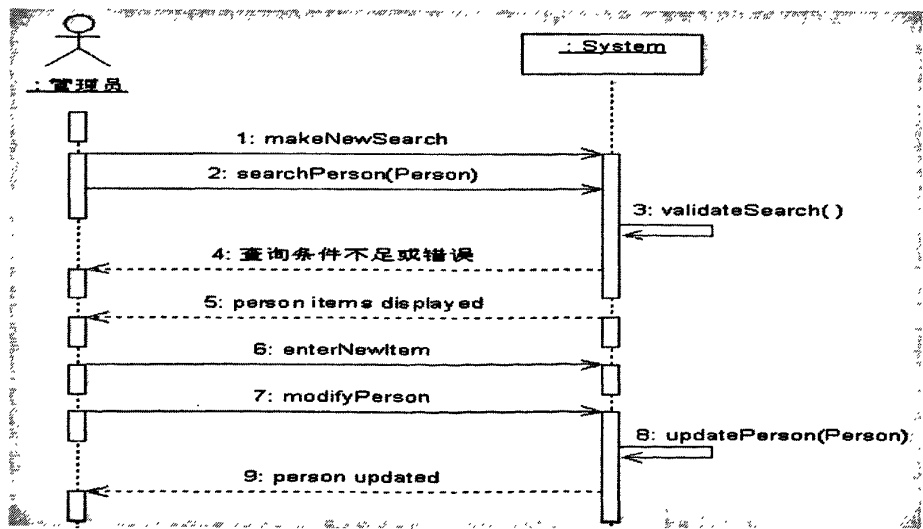


图 3-10 修改病人基本信息 SSD

从管理病人信息的用例描述中提取系统行为 modifyPerson，关于 modifyPerson 的详细描述见表 3-3。

3.2.5 删除病人基本信息

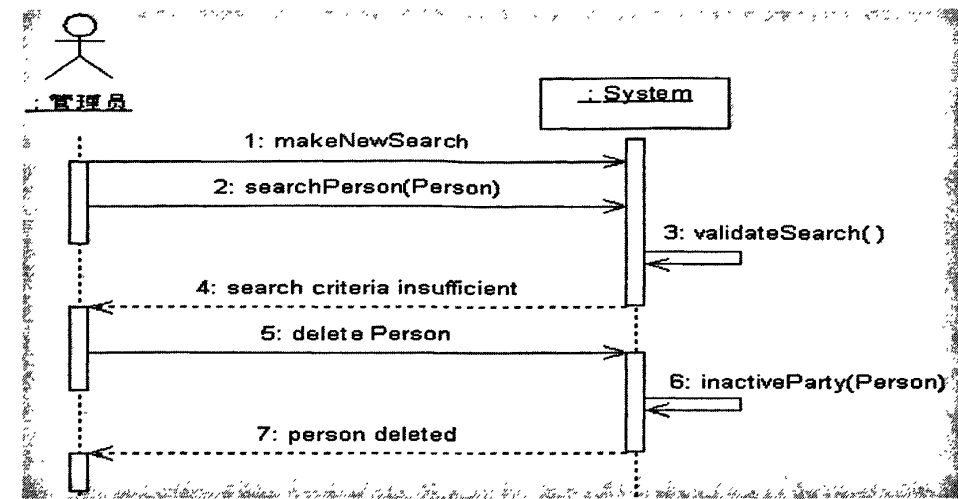


图 3-11 删除病人基本信息 SSD

从管理病人信息的用例描述中提取系统行为 deletePerson，关于 deletePerson 的详细描述见表 3-4。

3.2.6 合并病人基本信息

从管理病人信息的用例描述中提取系统行为 mergePerson，关于 mergePerson 的详细描述见表 3-7。

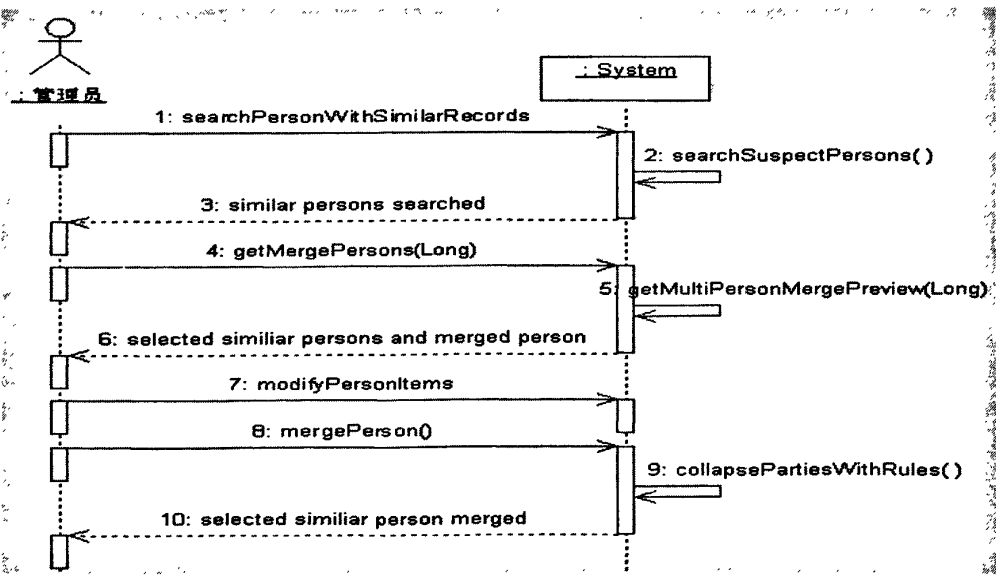


图 3-12 合并病人基本信息 SSD



3.2.7 转诊病人

从转诊病人的用例描述中提取系统行为 hospitalRegister，关于 hospitalRegister 的详细描述见表 3- 8。

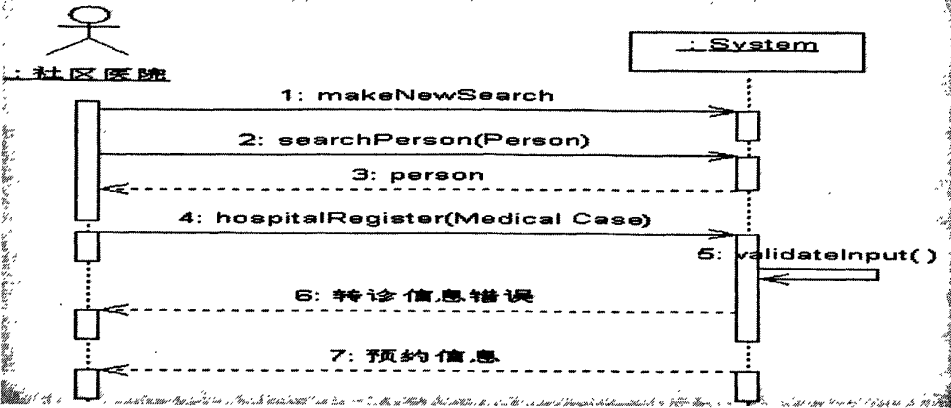


图 3- 13 转诊病人 SSD

3.2.8 添加转诊病历信息

从管理预约信息的用例描述中提取系统行为 writeCase，关于 writeCase 的详细描述见表 3- 9。

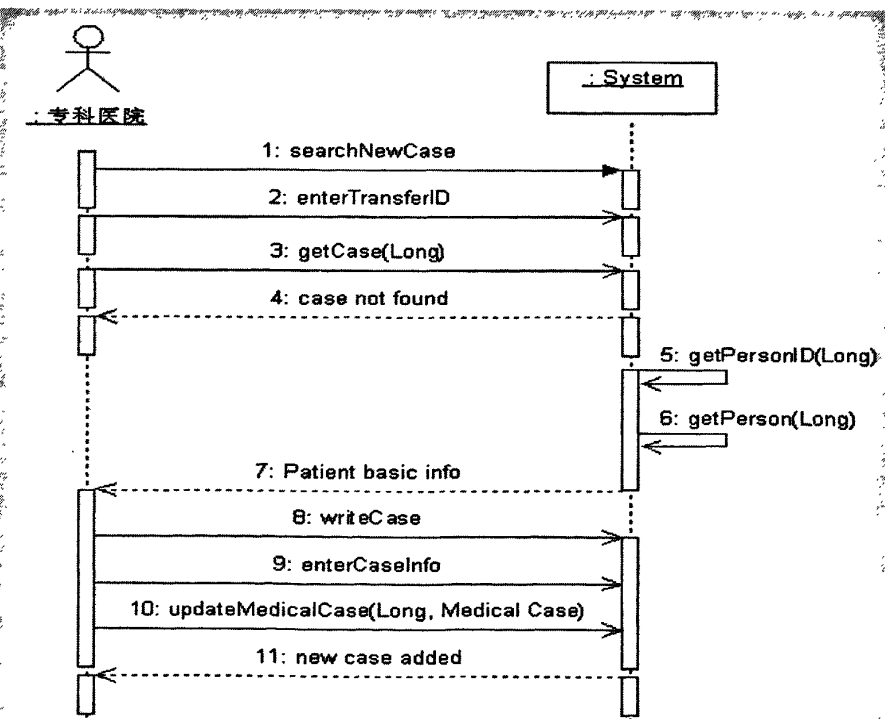


图 3- 14 添加转诊病例信息 SSD

3.2.9 查看转诊病历信息

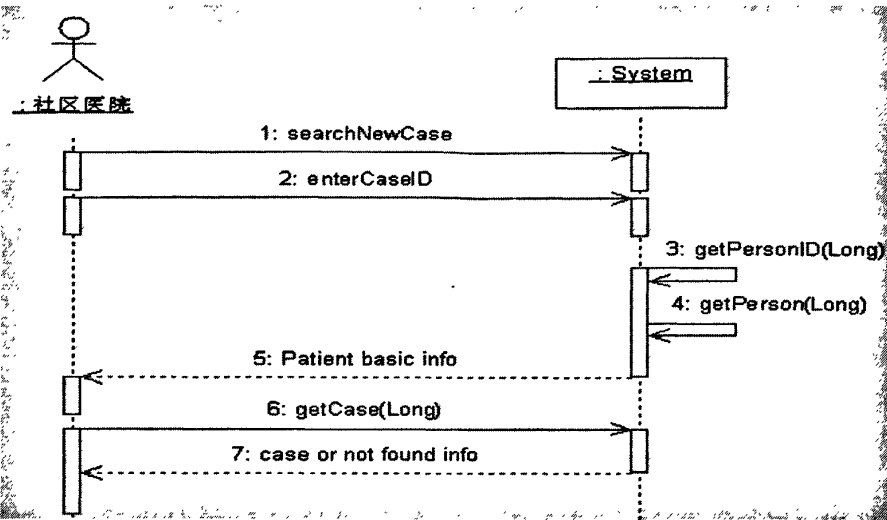


图 3-15 查看转诊病历信息 SSD

在专科医院添加病例信息成功以后，社区医院可以查询转诊病例，见转诊病例用例描述，该用例对应系统行为 `getCase`。

3.3 操作契约（Operation Contracts）

系统主要地操作契约 (Operation Contracts) 如下所示：

3.3.1 addPerson

表 3-1 addPerson

操作 (Operation):	addPerson(personInfo: Person)
相关引用 (Cross Reference):	用例：注册病人
前置条件 (Preconditions):	注册病人事件触发
后置条件 (Post conditions):	1. 一个 Person 实例 person 被创建。 2. 一个 PersonName 实例 personName 被创建并且其属性 lastName, firstName 被 personInfo 的相应属性赋值。 3. 一个 Identifier 实例 identifier 被创建并且其

	<p>属性 type, number 被 personInfo 的相应属性赋值。</p> <p>4. 一个 Address 实例 address 被创建并且其属性 countryType, provinceStateType, residenceType, cityName, postalCode, addressLineOne 被 personInfo 的相应属性赋值。</p> <p>5. 一个 ContactMethod 实例 contactMethod 被创建并且其属性 contactMethodType, refNumber 被 personInfo 的相应属性赋值。</p> <p>6. 基于 PartyID, person 实例和 identifier、address、contactMethod、personName 实例被关联在一起。</p>
--	--

3.3.2 searchPerson

表 3- 2 searchPerson

操作 (Operation):	searchPerson(personInfo: Person)
相关引用 (Cross Reference):	用例：查询病人基本信息
前置条件 (Preconditions):	查询病人事件触发
后置条件 (Post conditions):	<p>1. 一个 PersonSearch 实例 personSearch 被创建。</p> <p>2. personSearch 的属性 lastName, firstName, dateOfBirth, socialID, contactMethod 被 personInfo 的相应属性赋值。</p> <p>3. Party. SearchPary 事件被触发。</p>

3.3.3 modifyPerson

表 3- 3 modifyPerson

操作 (Operation):	modifyPerson (personInfo: Person)
相关引用 (Cross Reference):	用例：管理病人信息

前置条件 (Preconditions):	修改病人信息事件触发
后置条件 (Post conditions):	<div><div>1.    personInfo.personID    被    赋    值    被</div><div>Party.getPerson 事件的 partyID 参数。</div><div>2.    一个 Person 实例 person 被创建。</div><div>3.    Party.getPerson 的返回值被付给 person 实例。</div><div>4.    一个 person 实例的 personName, dateOfBirth,</div><div>gender 等属性被 personInfo 的相应属性赋值修</div><div>改。</div><div>5.    person 实例被赋值给 Party.updateParty 事件,</div><div>并且该事件被触发。</div></div>

3.3.4    deletePerson

表 3- 4 deletePerson

操作 (Operation):	deletePerson (personInfo: Person)
相关引用 (Cross Reference):	用例：管理病人信息
前置条件 (Preconditions):	修改病人信息事件触发
后置条件 (Post conditions):	<div><div>1.    一个 Party 实例 party 被创建。</div><div>2.    Party.getParty 事件被触发， pesonInfo 的</div><div>personID 被赋值给 getParty 事件的 partyID 参</div><div>数。</div><div>3.    Party 实例被 getParty 事件的返回值赋值。</div><div>4.    一个 Suspect 实例 vectorSuspect 被创建。</div><div>5.    getAllSuspectsForParty 被触发，其返回值被</div><div>赋值被 vectorSuspect。</div><div>6.    vectorSuspect 的迭代实例变量 suspect 被赋值</div><div>给 Party. updateSuspectStatus 事件，用户更</div><div>新的病人的 suspect 实例的状态全部被更改为 C</div><div>级。</div><div>7.    party 被赋值给 Party.inactivateParty 事件的</div><div>inactivatedParty        参        数        ,</div></div>

	Party.inactivateParty 事件被触发。
--	------------------------------

3.3.5 searchPersonWithSimilarRecords

表 3- 5 searchPersonWithSimiliarRecords

操作 (Operation):	searchPersonWithSimilarRecords ()
相关引用 (Cross Reference):	用例：管理病人信息
前置条件 (Preconditions):	查找所有相似病人事件触发
后置条件 (Post conditions):	<div>1. 一个 SuspectPersonSearch 实例 suspectPersonSearch 被创建。</div> <div>2. suspectPersonSearch 的 inquiryLevel 被设置成 0，表示值查询 Party 数据。</div> <div>3. 一个 Party 实例 party 被创建。</div> <div>4. Party. searchSuspectPersons 事件的参数 suspectPersonSearch 被 1 中的 suspectPersonSearch 赋值，并且 Party. searchSuspectPersons 被触发。</div>

3.3.6 getMergePersons

表 3- 6 getMergePersons

操作 (Operation):	getMergePersons(ids: integer[])
相关引用 (Cross Reference):	用例：管理病人信息
前置条件 (Preconditions):	查找某个病人的合并病人信息的事件触发
后置条件 (Post conditions):	<div>1. 两个 Party 实例被创建 parties[2]。</div> <div>2. parties [0].partyID 被 ids[0] 初始化， parties [1].partyID 被 ids[1]初始化，</div> <div>3. parties 数组变量被传递给 Party. comparativePreviewCollapseParties 事件。</div> <div>4. Party. comparativePreviewCollapseParties</div>

	事件返回对象 mergedParty 5. 一个 PersonInfo 实例 personInfo 被创建。 6. personInfo 被 mergedParty 赋值。返回 personInfo。
--	--

3.3.7 mergePerson

表 3- 7 mergePerson

操作 (Operation):	mergePerson (ids : integer[], mergedPerson:PersonInfo)
相关引用 (Cross Reference):	用例：管理病人信息
前置条件 (Preconditions):	合并病人信息的事件触发
后置条件 (Post conditions):	1. 三个 Party 实例被创建 parties[3]。 2. parties [0].partyID 被 ids[0] 初始化， parties [1].partyID 被 ids[1]初始化。 3. party[2]被 mergedPerson 实例化。 4. parties 被 赋 值 给 Party.collapsePartyWithRules 事件参数。 5. Party.collapsePartyWithRules 执行合并操作。

3.3.8 hospitalRegister

表 3- 8 hospitalRegister

操作 (Operation):	hospitalRegister(case: MedicalCase)
相关引用 (Cross Reference):	用例：转诊病人
前置条件 (Preconditions):	病人转诊事件触发
后置条件 (Post conditions):	1. 一个 PartyEquivalency 实例 partyAdminSysKey 被创建。 2. 一个 Integer 实例 caseID 被创建。 3. caseID 被 MedicalCase.addCase(case) 初始化。

	<ol style="list-style-type: none"> <li>4. partyAdminSysKey.partyID 的值被改为 case.partyID。</li> <li>5. partyAdminSysKey.adminSysPartyID 的值被改为 caseID。</li> <li>6. partyAdminSysKey.adminSystemType 的值被改为 “HER”。</li> <li>7.       方                                  法                                  Party. addPartyAdminSysKey(partyAdminSysKey) 被触发，并建立一条病人转诊记录。</li> </ol>
--	--

### 3.3.9 writeCase

### 表 3-9 writeCase

操作 (Operation):	writeCase(caseModified: MedicalCase)
相关引用 (Cross Reference):	用例: 管理预约信息
前置条件 (Preconditions):	写病历事件触发
后置条件 (Post conditions):	<ol style="list-style-type: none"> <li>1. 一个MedicalCase的实例originalCase被创建。</li> <li>2. 方 法 MedicalCase.getMedicalCase(caseModified.caseID) 被 执 行, 返回值赋给 originalCase。</li> <li>3. 将 caseModified 的属性 caseDate, doctor, symptom, diagnosis, treatment 属性分别赋值给 originalCase 的相应属性。</li> <li>4. 方 法 MedicalCase.updateMedicalCase(originalCase)被执行, 写病历完成。</li> </ol>

## 4 体系结构设计

本系统基于如下应用场景：

1. 小王在公司组织的年度体检中，发现血压有些高，体检中心建议小王去医院复诊一下。
2. 于是，小王周一去了就近的社区医院。护士接待了他，先在本地系统查询，发现没有就诊记录，于是在系统中新建了一条病人信息，全科医师对他进行了初步诊断，发现社区的条件有限，无法很好的确认病情和病因，于是利用病人主索引系统给小王开出了人民医院的转诊单。
3. 小王周一去了人民医院，医师从病人主索引系统中获取病人的基本信息，进行了充分的检查，确认了其病因，并开具治疗方案。
4. 小王为了治疗方便，去了社区医院，全科医师利用病人主索引系统查询小王预约就诊的治疗方案，并进行相应的治疗。

本系统主要功能：

1. 记录体检信息。体检中心可以利用病人主索引系统记录体检信息。
2. 管理各个医院的病人基本信息，给各个医院提供全局的一致的病人视图——病人主索引系统。管理功能包括检索、添加、修改、删除病人基本信息，合并相似的病人基本信息（当确认数据库中的两个病人是同一个人时）。
3. 预约处理。社区医院可以利用病人主索引系统进行转诊预约，如转诊到某个专科医院某个科室，并提供转诊单。
4. 预约病历记录。专科医院可以利用病人主索引系统查询转诊预约，并记录预约病历。
5. 预约病历查询。社区医院可以利用病人主索引系统查询专科医院的预约诊断病历，并提供治疗服务，方便病人就近接受治疗。

### 4.1 逻辑视图(Logical View)

本系统采用四层结构，如图 4-1 所示：

系统分为四层：表示层、应用层、服务层和领域层<sup>[15]</sup>。表示层主要有各个主要功能模块的 JSP、Struts 标签、JavaScript 构成。应用控制层是基于开源框架 Struts 实现的，主要包括如下几部分：负责处理主索引系统功能请求的 Actions、负责处理预约功能请求的 Actions、以及存储 HTTP 请求与响应数据的 FormBean、Struts 的配置文件。服务层主要是为应用控制层提供合成业务服务的，主要包括



两部分：合成服务类和 Web Services 代理类。领域模型层主要包括 MDM Server 的原有部分和扩展部分，扩展部分包括病历信息和健康记录，该层包括核心领域模型服务，服务的控制层和服务代理层。每一次都包括核心类以及类之间的关联，如下几节将详细的阐述各层的核心类的构成和关联关系。

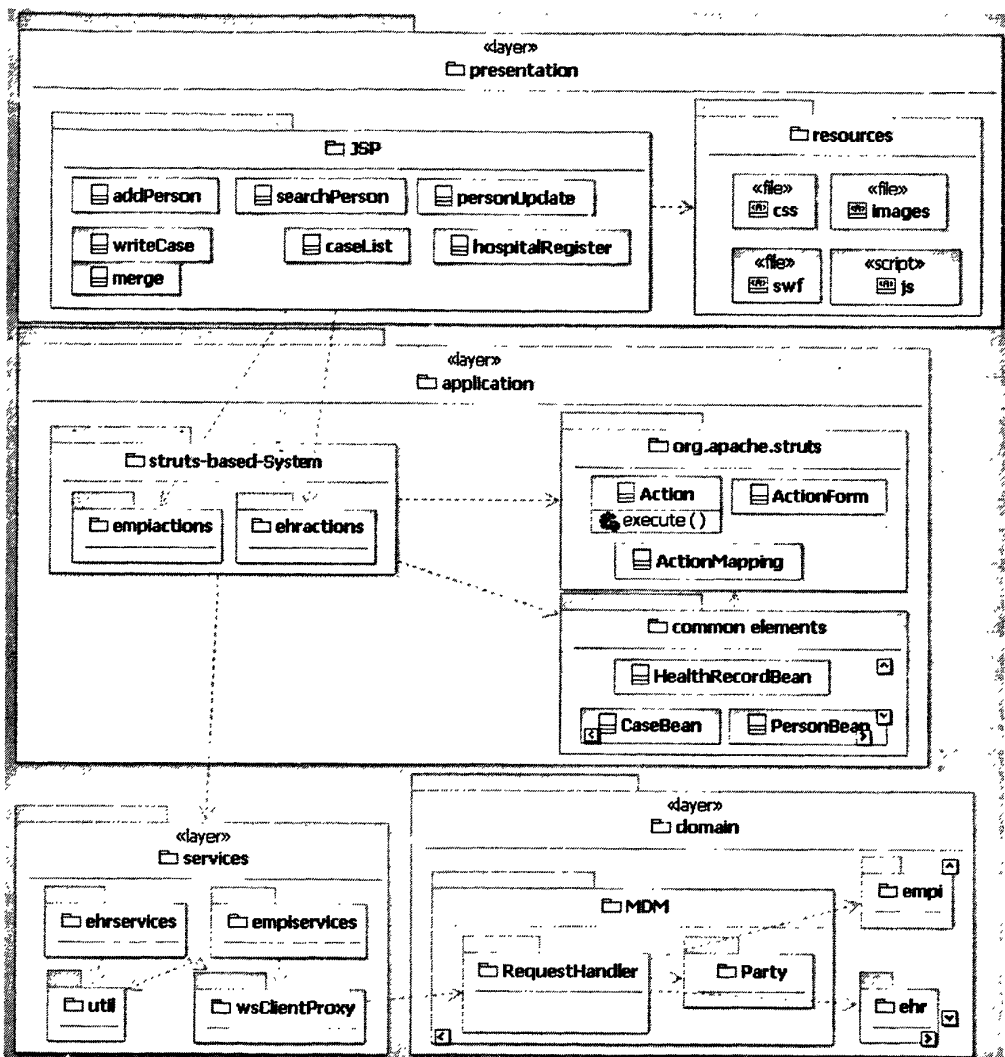


图 4-1 系统逻辑分层包图

### 4.1.1 表示层（Presentation Layer）

在表示层中，如图 4- 2 所示,主要的 JSP 应具有如下功能：添加病人基本信息、修改病人基本信息、更新病人基本信息、合并病人基本信息、预约处理、写预约病历和读取预约病历。JSP 需要访问的资源文件有 CSS、images 文件、swf 文件和 JavaScript 文件。

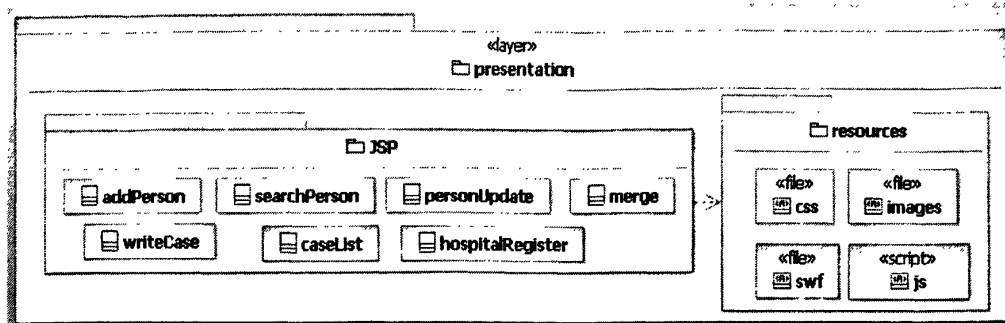


图 4-2 表示层逻辑视图

#### 4.1.2 应用控制层 (Application Controller Layer)

应用控制层，如图 4-3 所示，在整个系统中主要起到前端控制器的作用。该层主要负责接收用户的请求；并且根据请求信息来决定用户请求的行为，也就是应该决定对何种事件采取何种相应；如果需要的话，将模型中的数据提取处理交给视图显示；最后选择相应的视图回发给用户。

触发事件与系统行为类的对应关系为：

- addPerson→AddPersonAction
- serarchPerson→SearchPersonAction
- deletePerson→DeletePersonAction
- modifyPerson→ModifyPersonAction
- mergePerson→MergePersonAction
- hospitalRegester→RegisterAction
- addHealthRecord→AddPersonPhisicalRecordAction
- addCase→WriteCaseAction
- getCase→GetCaseInfoAction

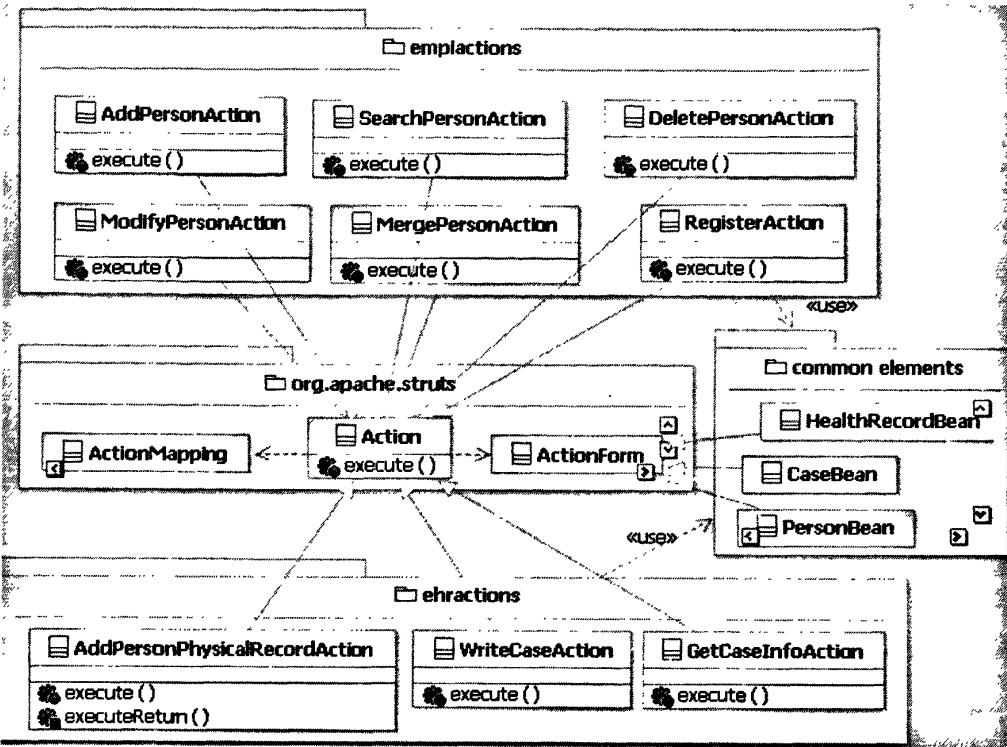


图 4-3 应用控制层逻辑视图

### 4.1.3 服务层（Services Layer）

服务层，如图 4-4 所示，在整个系统中主要起到对业务组件的 Web Services 调用重组，以满足业务需求的作用，该层为应用控制层提供业务服务。

各个子层的职责和核心类：

- 子系统 Services 层的主要功能是负责初始化 Web Services 参数，并按照业务逻辑编 Web Services，来完成相应的业务。
- util 服务工具层主要对 Web Services 代理层进行参数封装，以方便子系统 Services 层的使用。
- Web Services 代理子层主要是为服务器业务组件的 Web Services 提供客户端代理，以便客户端的 Web Services 访问。业务组件的 Web Services 主要有粗粒度和细粒度的服务，例如 AddPerson 服务为一粗粒度服务，它包含 SearchSuspects、processSuspects、AddPersonName 等细粒度的服务。该层主要采用 IBM WebSphere JAX-RPC Web Services 运行时环境运行 Web Services 代理。

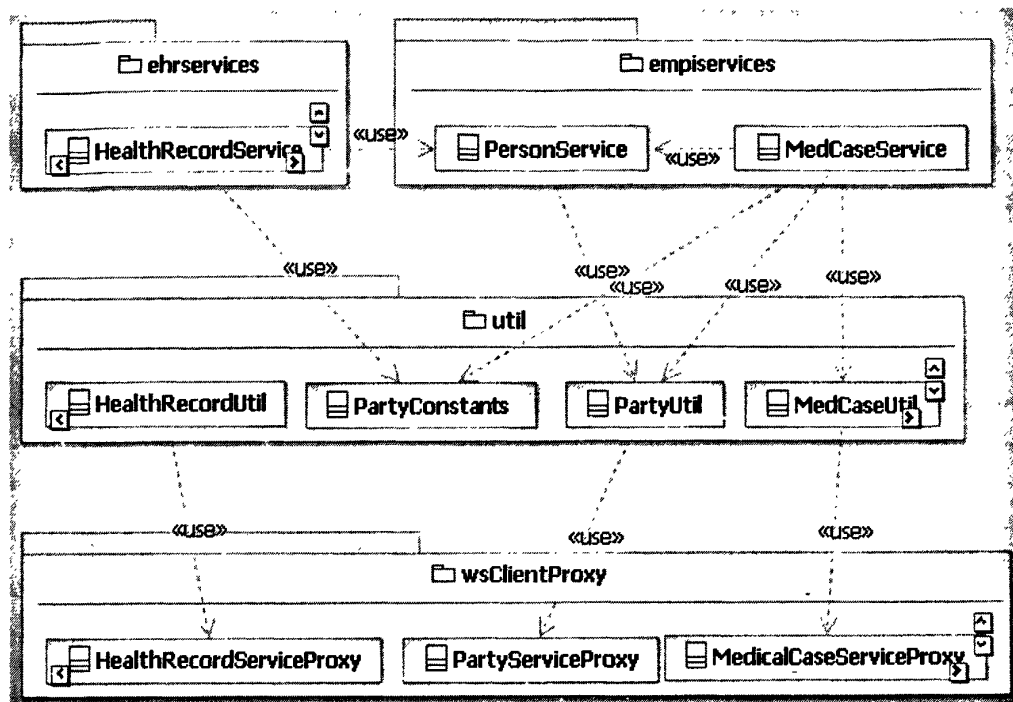


图 4-4 服务层逻辑结构图

#### 4.1.4 领域模型层 (Domain Model)

领域模型层，如图 4-5 所示，在整个系统中主要起到对业务进行组件封装的作用，业务组件采用 EJB 进行实现。该层为服务层提供业务组件服务。

MDM Server 中的 Party 业务模型中包括了对 Person 业务服务的封装，本系统主要使用到了 IBM MDM Server Party 域中的 Person Web Services。在 MDM 中，一个主要地框架是 RequestHandler 框架，Request/Response 框架主要服务处理 MDM 外部的请求和响应信息。Request Handler 最终将 Web Services 的 XML 文件中的参数转换成 Java 类，并根据请求服务信息查找相应的服务组件控制器类，由服务组件控制器选择相应的服务组件执行客户端的服务请求。

本系统中的业务组件开发部分采用了与 MDM 组件结构相同的架构，如上图所示，主要由两部分构成：业务组件控制器和业务组件。扩展的病历组件类和健康记录组件类详细说明如下：

- 读取操作的控制器类负责读取服务的控制，诸如：GetXXX, SearchXXX 之类的操作由 MDM 的 Request Handler 转发给此类服务控制组件。MedicalCaseFinderImpl 类是病历组件读取操作控制器类，HealthRecordFinderImpl 类是健康记录组件读取操作控制器类。

- 写操作的控制器类负责写服务的控制，诸如：  
addXXX, updateXXX, deleteXXX, setXXX, modifyXXX 的操作由 MDM 的 Request Handler 转发给此类服务控制组件。MedicalCaseTxnBean 类是病历组件写操作控制器类，HealthRecordTxnBean 类是健康记录组件写操作控制器类。
- 业务组件类实现核心业务，并负责与数据库实体进行交换。  
MedicalCaseComponent 是病历组件类，HealthRecordComponent 是健康记录组件类。

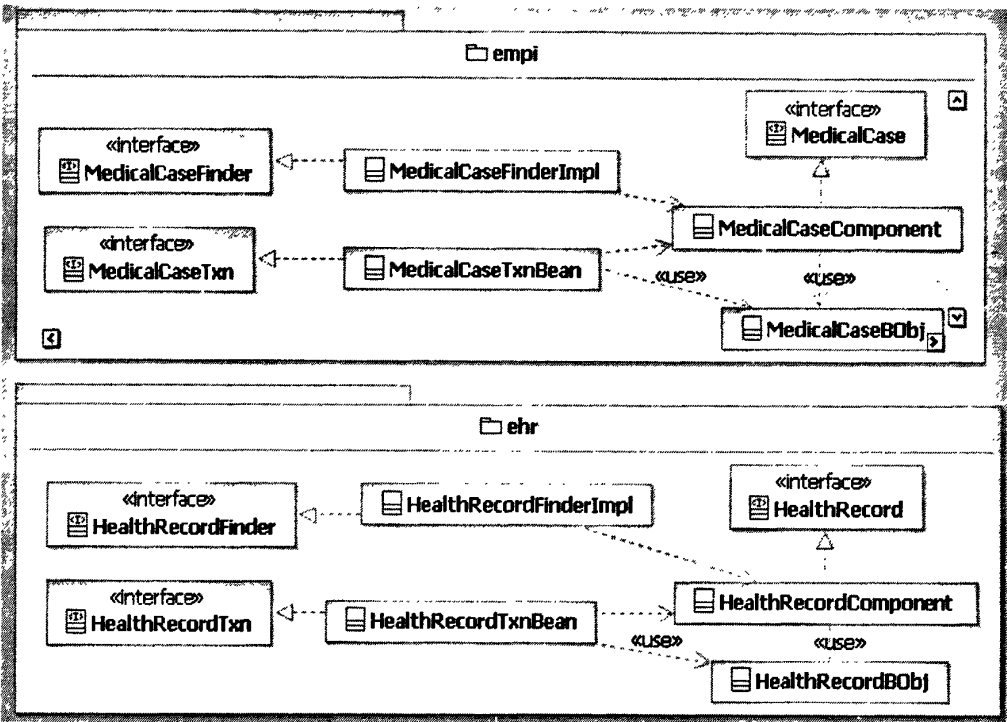


图 4-5 领域模型层逻辑视图

## 4.2 流程视图(Process View)

系统的主要进程:病人主数据管理与预约 Web 主进程(EMPI Web Application)、预约病历处理 Web 主进程 (HER Web Application)、MDM Party 业务服务 (Patient Service)、病历业务服务 (MedicalCase Service) 和健康记录服务 (HealthRecord Service), 如图 4-6 所示。

病人主数据管理与预约 Web 主进程 (EMPI Web Application) 集成了病人基本信息 Web 服务代理 patinetProxy 和病历 Web 服务代理 MedicalCaseProxy, 预约

病历处理 Web 主进程（HER Web Application）集成了病历 Web 服务代理 MedicalCaseProxy 和健康记录 Web 服务代理 HealthRecordProxy。

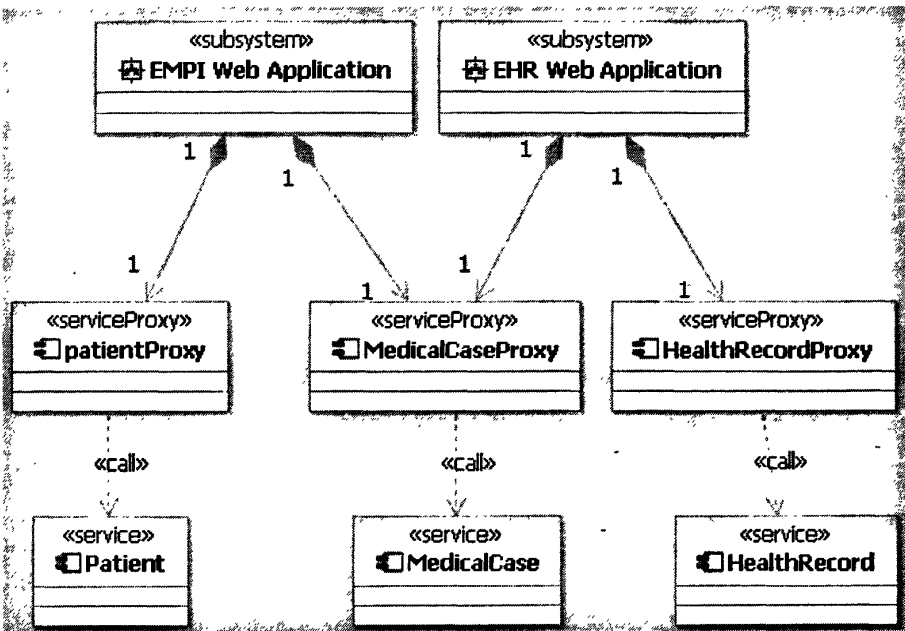


图 4-6 流程视图

4.3 部署视图(Deployment View)

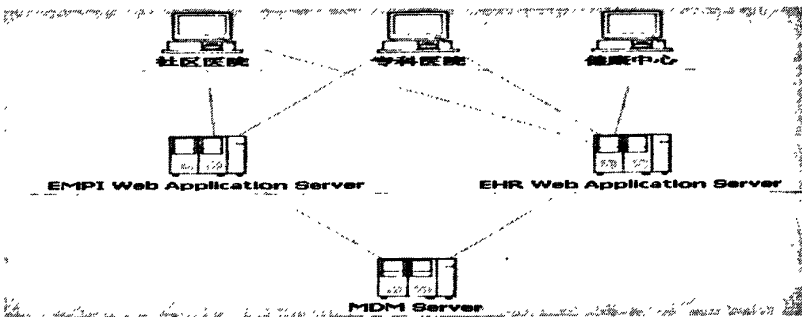


图 4-7 部署视图

病人主数据管理与预约 Web 主进程（EMPI Web Application）和预约病历处理 Web 主进程（HER Web Application）分别部署在不同的 IBM WebSphere 服务器上，MDM Party 业务服务（Patient Service）、病历业务服务（MedicalCase Service）和健康记录服务（HealthRecord Service）部署在 IBM MDM 服务器上。社区医院、专科医院和健康中心通过 Internet 来访问病人主数据管理服务和预约处理系统。如图 4- 7 所示。其中不同的 Server 均部署在 IBM WebSphere Server 6.0 服务器

上, IBM WebSphere Server 6.0 运行的操作系统环境为 RedHat Enterprise Linux 5, MDM 采用其集成的数据库 DB29.0。在系统的部署中, 按照非功能性需求中的接口约束进行安装和测试。

## 5 详细设计

系统的类设计可分为服务器端的应用类设计和 Web 端的应用程序的类设计。服务器端的类主要分为服务组件类和服务组件 Web Services 转换类, 服务组件类用来实现核心业务功能, 服务组件 Web Services 转换类提供 EJB 向 Web Services 转换功能。Web 应用程序类主要包括 Action 类, Services 类和 Web Services Proxy 类。

在系统的服务组件类设计中, 通过 EJB 中的 Stateless Session Bean 和 Java 类来设计实现病历 (MedicalCase) 业务组件和健康信息 (HealthRecord) 业务组件, 而 Party 中的其它部分、Location、Identifier、Link 和 Suspect 主题则采用 IBM MDM Server 中的现有主题业务。

按照非功能性需求中各种约束, 进行如下设计约束: 服务器组件的日志记录工具采用 MDM 框架的 LoggerManager 提供的 Logger 接口类, 异常处理采用 MDM 框架的 ExceptionUtils 类来实现异常的重定向; 通过将 EJB 组件的业务方法暴露为 Web Services, 满足了非功能性需求设计中的可支持性和实现约束中的条目; 系统采用 MDM 框架实现核心服务类, 因为 MDM 对核心服务所需的数据进行了缓存, 有效的提供了系统数据检索的性能, 经验证, 其性能表现完全满足了客户的性能要求; 在 Web 层的错误处理中, 采用将具体的错误信息转化为 struts 的用户可以识别的错误信息, 提供了系统的友好性和可用性。

如下各节中对系统的核心类进行了描述。其中结构相似的类只列举了其中的一个作为代表详细阐述。

### 5.1 组件设计

自定义的服务组件包括病历组件 MedicalCase 和健康信息组件 HealthRecord。健康信息组件 HealthRecord 与病历组件 MedicalCase 的结构相似, 图 5-1 是病历组件 MedicalCase 的类图表示。



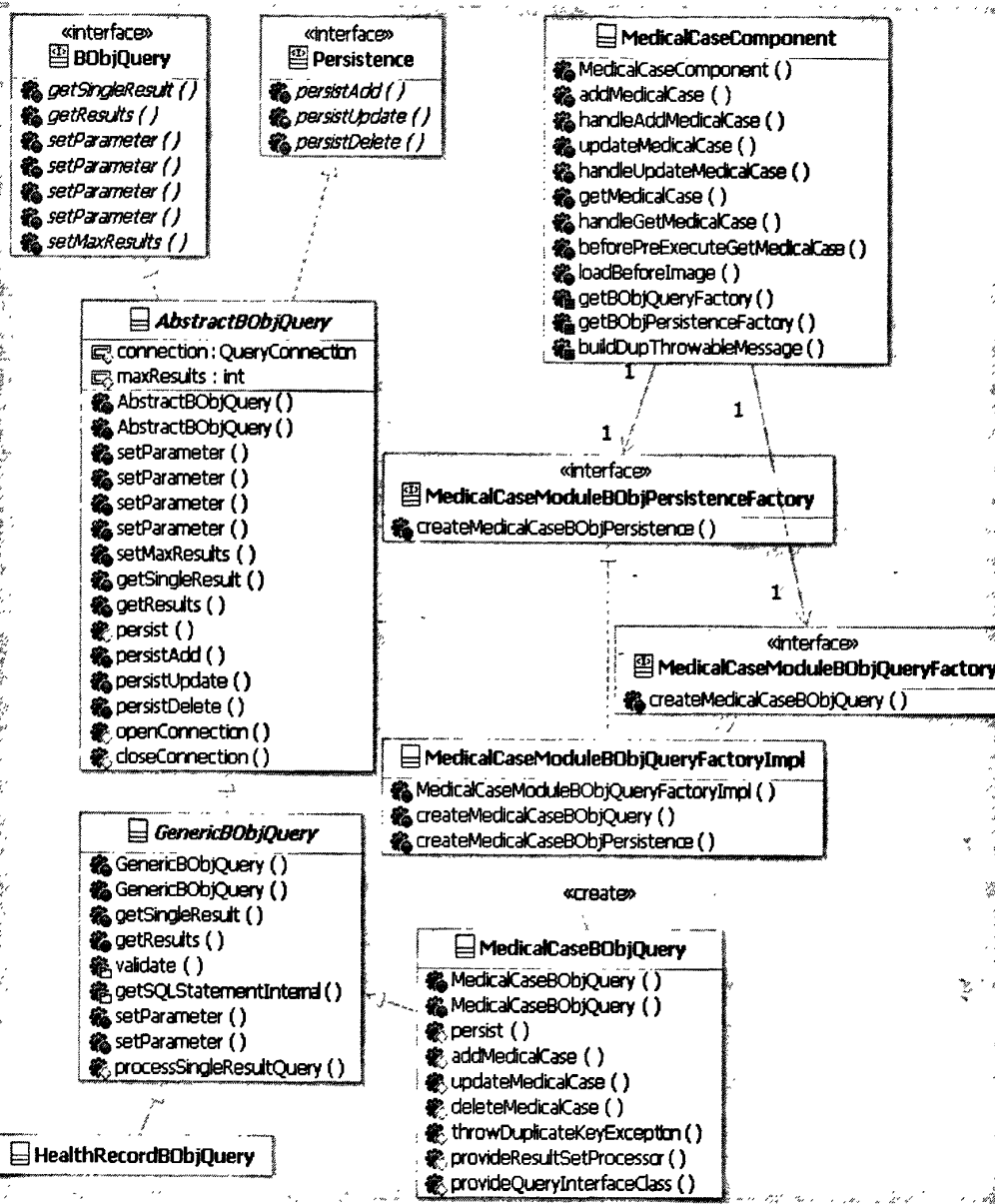


图 5-1 病历组件类图

HealthRecordBObjQuery 类、MedicalCaseBObjQuery 类负责调用数据访问层的实体类的方法,并且实现数据属性的格式转换和提取,它们都继承了 MDM 的 GenericBObjQuery 类,在这个类中封装了接口 BObjQuery 和接口 Persistence 的方法,其中接口 BObjQuery 的方法主要是对数据查询类服务的封装,而接口 Persistence 主要是对持久化数据服务的封装。在核心组件类 MedicalCaseComponent 中使用到了 HealthRecordBObjQuery 类和 MedicalCaseBObjQuery 类,为了使类 MedicalCaseComponent 独立于

HealthRecordBObjQuery 类和 MedicalCaseBObjQuery 类的创建、组合和表示，因为 HealthRecordBObjQuery 类和 MedicalCaseBObjQuery 类中分别包含了 BObjQuery 接口方法与 Persistence 接口方法的实现，因此为了联合使用 BObjQuery 接口方法与 Persistence 接口方法，在类的设计上采用了抽象工厂模式。

addMedicalCase(business object)方法的调用顺序：

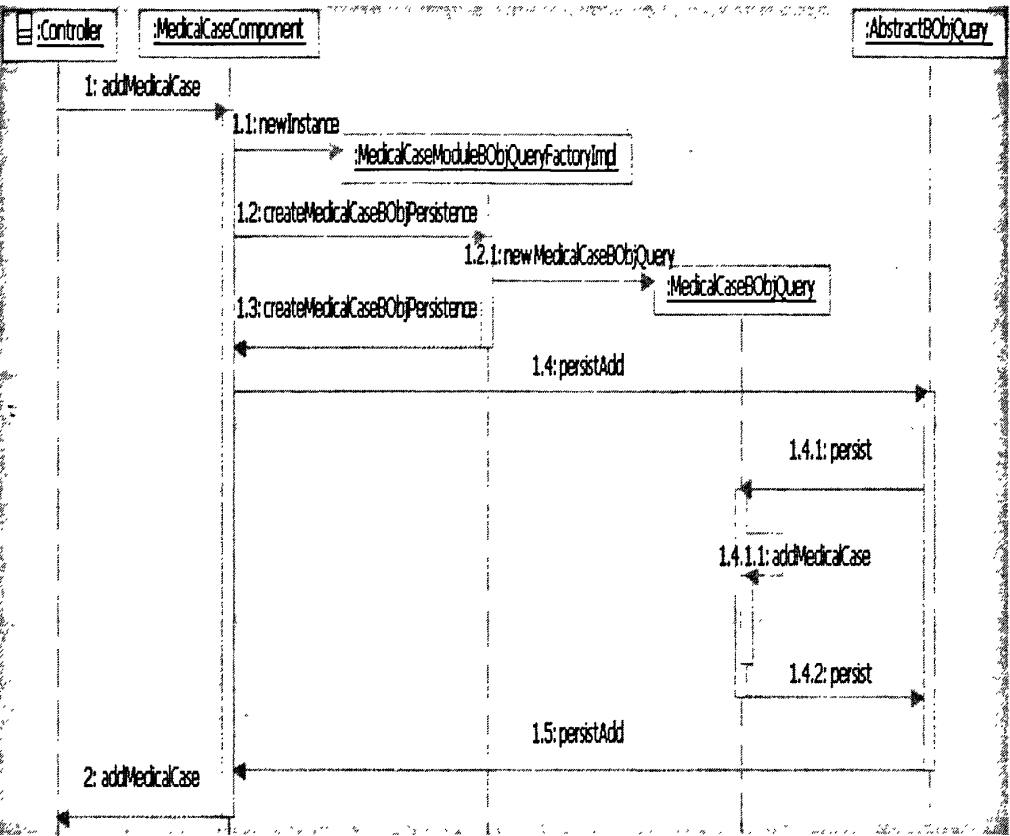


图 5-2 addMedicalCase 顺序图

- 1 addMedicalCase(business object)
- 组件服务控制器调用组件服务 addMedicalCase (business object)，MedicalCaseTxnBean 组件服务控制器首先构建 MedicalCaseComponent 对象，并调用 MedicalCaseComponent 对象的 addMedicalCase 方法。
- 1.1 newInstance 方法
- 病历组件 MedicalCaseComponent 通过读取属性配置文件的工厂类的名称，通过 Java 反射创建 MedicalCaseModuleBObjQueryFactory 的实现类对象。
- 1.2 createMedicalCaseBObjPersistence (add operation tag)
- 方法调用 MedicalCaseBObjQuery 类的带操作标记参数的构造函数。
- 1.2.1 new MedicalCaseBObjQuery (add operation tag)
- 调用 MedicalCaseBObjQuery 构造函数构造对象。

1.3 createMedicalCaseBobjPersistence (add operation tag) 调用返回  
返回生成的 MedicalCaseBobjQuery 对象，并将对象赋给 Persistence 接口。

1.4 persistAdd ()

用返回的 Persistence 对象接口调用 AbstractBobjQuery 类的 PersistAdd() 方法。PersistAdd () 方法是 AbstractBobjQuery 默认适配器类提供实现的方法。

1.4.1 Persist ()

Persist () 方法是 AbstractBobjQuery 默认适配器类提供的模板方法，该方法最终由具体的 HealthRecordBobjQuery 子类和 MedicalCaseBobjQuery 子类提供选择不同类型的持久化方法。

1.4.1.1 addMedicalCase()

MedicalCaseBobjQuery 类中的 persist () 方法根据 operation tag 标识的操作类型，选择不同的策略，在这里 operation tag 标识的是 add 操作，所以选择调用 MedicalCaseBobjQuery 类的 addMedicalCase () 方法。

updateMedicalCase(business object)方法的调用顺序:

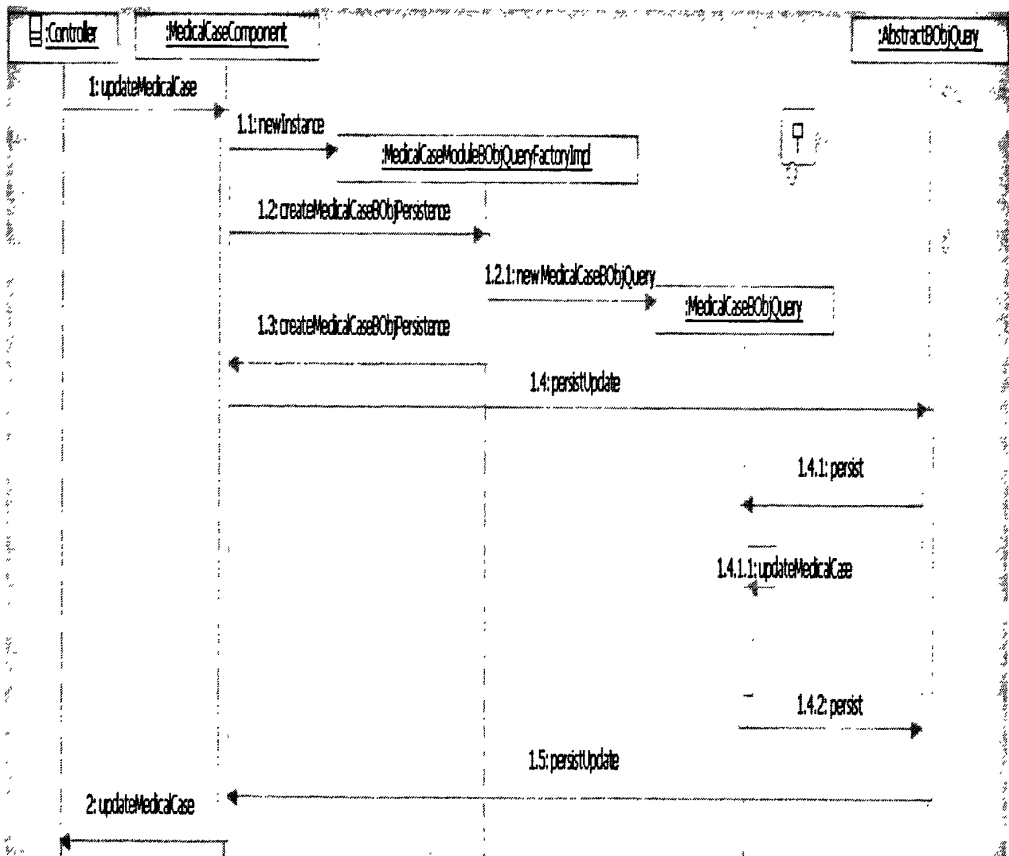


图 5- 3 updateMedicalCase 顺序图

### 1 updateMedicalCase(business object)

组件服务控制器调用组件服务 updateMedicalCase (business object), MedicalCaseTxnBean 组件服务控制器首先构建 MedicalCaseComponent 对象, 并调用 MedicalCaseComponent 对象的 updateMedicalCase 方法。

#### 1.1 newInstance 方法

病历组件 MedicalCaseComponent 通过读取属性配置文件的工厂类的名称, 通过 Java 反射创建 MedicalCaseModuleBObjQueryFactory 的实现类对象。

#### 1.2 createMedicalCaseBObjPersistence (update operation tag)

方法调用 MedicalCaseBObjQuery 类的带操作标记参数的构造函数。

##### 1.2.1 new MedicalCaseBObjQuery (update operation tag)

调用 MedicalCaseBObjQuery 构造函数构造对象。

1.3 createMedicalCaseBObjPersistence (update operation tag) 调用返回

返回生成的 MedicalCaseBObjQuery 对象, 并将对象赋给 Persistence 接口。

#### 1.4 persistUpdate ()

用返回的 Persistence 对象接口调用 AbstractBObjQuery 类的 PersistUpdate () 方法。PersistUpdate () 方法是 AbstractBObjQuery 默认适配器类提供实现的方法。

##### 1.4.1 Persist ()

Persist () 方法是 AbstractBObjQuery 默认适配器类提供的模板方法, 该方法最终由具体的 HealthRecordBObjQuery 子类和 MedicalCaseBObjQuery 子类提供选择不同类型的持久化方法。

##### 1.4.1.1 addMedicalCase()

MedicalCaseBObjQuery 类中的 persist () 方法根据 operation tag 标识的操作类型, 选择不同的策略, 在这里 operation tag 标识的是 update 操作, 所以选择调用 MedicalCaseBObjQuery 类的 updateMedicalCase () 方法。

图 5-4 是 MedicalCase 实体、业务对象、数据库访问类的类图:

数据库访问类的接口分为查询接口 MedicalCaseInquiryData 和持久化接口 EObjMedicalCaseData。业务对象 MedicalCaseBObj 中引用了实体对象 EObjMedicalCase。

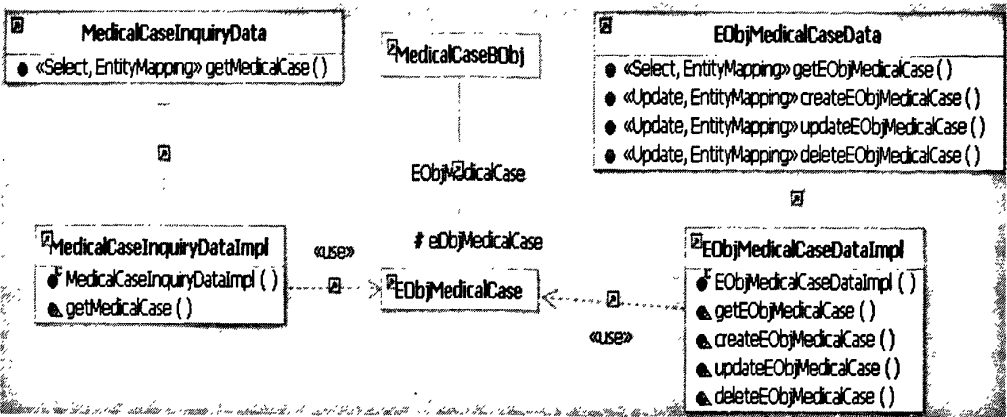


图 5- 4 MedicalCase Data Access Object  
getMedicalCase(medical case id) 方法的调用顺序:

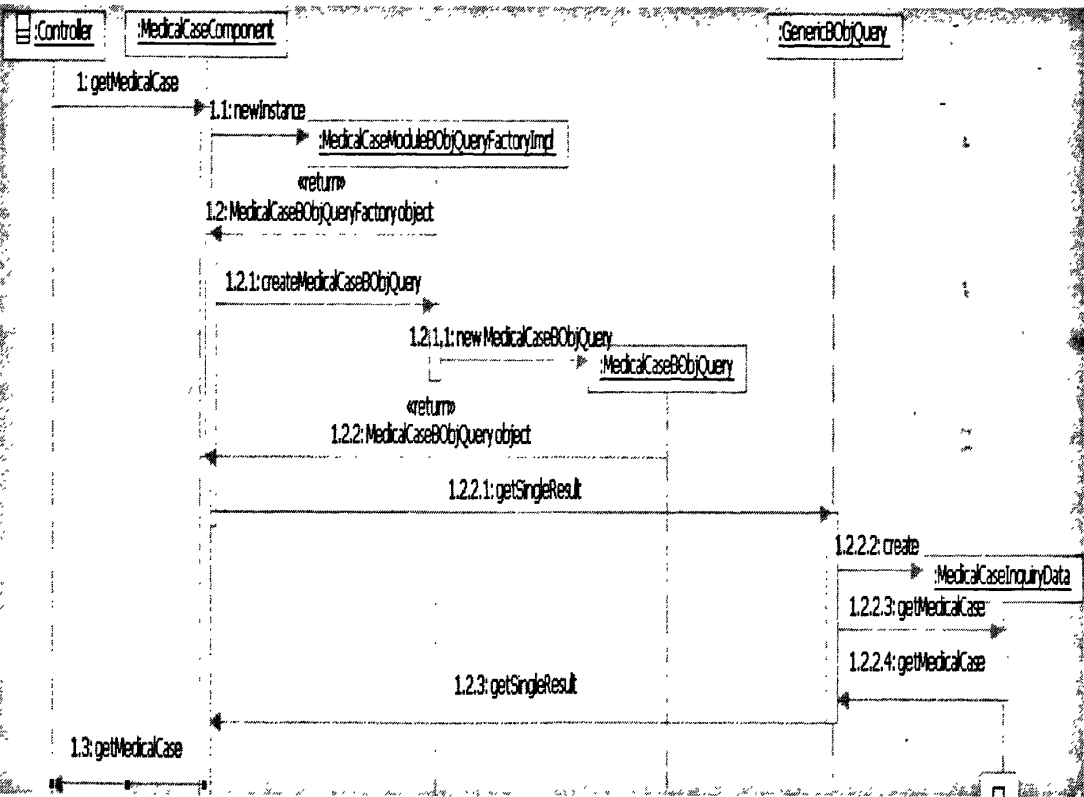


图 5- 5 getMedicalCase 顺序图

1 getMedicalCase(medical case id)  
组件服务控制器调用组件服务 getMedicalCase (medical case id),  
MedicalCaseTxnBean 组件服务控制器首先构建 MedicalCaseComponent 对象, 并调用 MedicalCaseComponent 对象的 getMedicalCase 方法。

1.1 newInstance 方法

病历组件 MedicalCaseComponent 通过读取属性配置文件的工厂类的名称, 通

过 Java 反射创建 MedicalCaseModuleBObjQueryFactory 的实现类对象。

1.2 return MedicalCaseBObjQueryFactory object

返回调用生产的 MedicalCaseModuleBObjQueryFactory 的实现类对象。

1.2.1 createMedicalCaseBObjQuery (query operation tag)

方法调用 MedicalCaseBObjQuery 类的带操作标记参数的构造函数。

1.2.1.1 new MedicalCaseBObjQuery (query operation tag)

调用 MedicalCaseBObjQuery 构造函数构造对象。

1.3 createMedicalCaseBObjPersistence (query operation tag) 调用返回  
返回生成的 MedicalCaseBObjQuery 对象，并将对象赋给 BObjQuery 接口。

1.3.1 getSingleResult (medical case id)

用返回的 BObjQuery 对象接口调用 GenericBObjQuery 类的 getSingleResult (medical case id) 方法。

1.3.1.1 create MedicalCaseInquiryData 对象

1.3.1.2 getMedicalCase(medical case id)

调用 MedicalCaseInquiryData 对象的数据层访问方法  
getMedicalCase(medical case id)。

1.3.1.2 返回 getMedicalCase(medical case id) 实体数据

1.3.1 返回 getSingleResult (medical case id)

在返回之前，将获得的实体数据属性转换为业务对象属性。并返回业务对象。

2 返回 getMedicalCase(medical case id)结果

将业务对象转换成 MDM 的 Response 对象，并返回。

病历控制器类采用会话 Bean 实现，其类结构如图 5-6 所示：

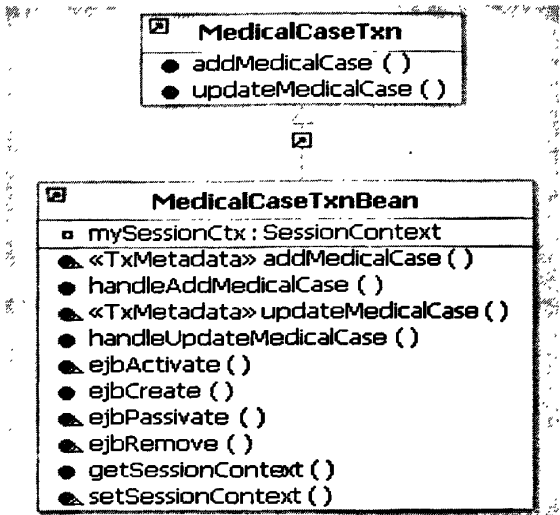


图 5- 6 MedicalCaseTxnBean 类图

在 MedicalCaseTxnBean 会话 Bean 中, 引用到了数据库数据源, 以供数据库访问层的类实体。会话 Bean 的详细描述如下所示:

```
<session id="Session_MedicalCaseTxn">
    <ejb-name>MedicalCaseTxn</ejb-name>
    <local-home>com.ibm.ehr.medcase.controller.MedicalCaseTxnLocalHome
</local-home>

<local>com.ibm.ehr.medcase.controller.MedicalCaseTxnLocal</local>

<ejb-class>com.ibm.ehr.medcase.controller.MedicalCaseTxnBean</ejb-class>

    <session-type>Stateless</session-type>
    <transaction-type>Container</transaction-type>

    <resource-ref id="ResRef_1">
        <res-ref-name>jdbc/DWLCustomer</res-ref-name>
        <res-type>javax.sql.DataSource</res-type>
        <res-auth>Container</res-auth>
        <res-sharing-scope>Shareable</res-sharing-scope>
    </resource-ref>
    <resource-ref id="ResRef_2">
        <res-ref-name>jdbc/DWLConfig</res-ref-name>
        <res-type>javax.sql.DataSource</res-type>
        <res-auth>Container</res-auth>
        <res-sharing-scope>Shareable</res-sharing-scope>
    </resource-ref>
</session>
```

## 5.2 组件 Web Services 适配设计

由于 SOAP 协议是无状态和无连接的, 因此采用无状态会话 Bean 包装 Web 服务。组件 Web Services 适配类的设计类图如图 5-7 所示。

主接口 MedicalCaseServiceRemoteHome 负责控制一个 Bean 的生命周期 (生产、删除和查找 Bean), 该接口扩展了 javax.ejb.EJBHome 接口, 在 EJBHome 接口

中：方法 `getEJBMetaData()` 返回 `EJBMetaData` 接口的引用，取得 Bean 的信息，`EJBMetaData` 不是远程接口，这个类扩展了 `java.io.Serializable`，所以可序列化，具有序列化的特性；方法 `getHomeHandle()` 返回主对象的句柄，句柄是主接口 `MedicalCaseServiceRemoteHome` 的持久性引用，这个类扩展了 `java.io.Serializable`，所以可序列化，具有序列化的特性，`HomeHandle` 对象可以传递给另一个 JVM，且不传递安全信息，这样新的应用可以不使用 JNDI 来查找对象既可以获得这个主接口，并来创建和获得 Bean 实例；方法 `remove()` 用来删除一个 Bean 的实例，对于一个会话 Bean，执行 `Remove` 操作将引用的 Bean 返回到池中，由池来管理其生命周期。

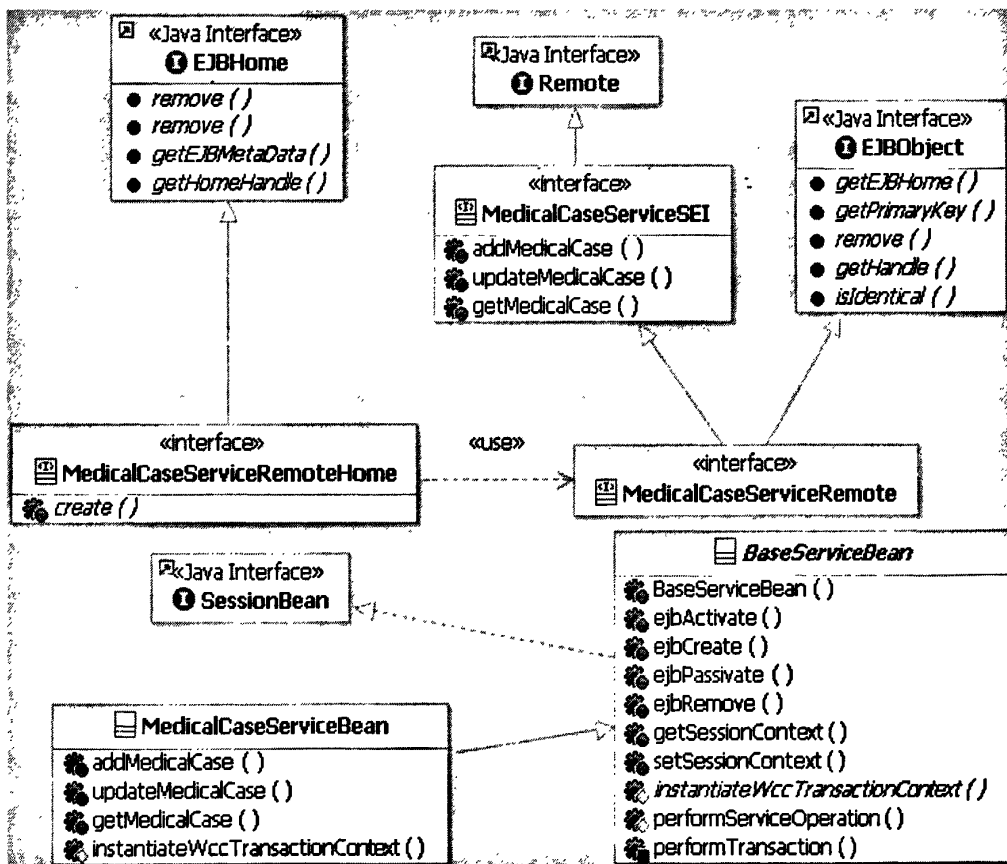


图 5-7 MedicalCase Web Services Adapter 类图

远程接口 `MedicalCaseServiceRemote` 当远程用户调用主接口类生成方法 (`create()`) 时，给客户生成一个组件的远程引用。远程接口 `MedicalCaseServiceRemote` 扩展了 `MedicalCaseServiceSEI` 接口，`MedicalCaseServiceSEI` 接口定义病历组件 `MedicalCaseComponent` 中的业务服务，以实现对组件 `MedicalCaseComponent` 方法的适配作用。在 `EJBObject` 中，方法 `getEJBHome()` 返回远程主接口对象的引用；方法 `getHandle()` 返回当前组件接口



对象的句柄, 和远程主接口的句柄 HomeHandle 一样, 这个对象是被序列化的, 所以可以保存到本地或通过 RMI/IIOP 协议传输给其他 JVM 上的客户使用, 而免去 JNDI 查找和调用主接口的 create 方法, 只要执行 Handle.getEJBObject() 方法即可取得这个 Bean 实例的引用; getPrimaryKey() 方法一般用于 Entity Bean, 如果在 Session Bean 中调用, 抛出 java.rmi.RemoteException; 方法 isIdentical() 用于对当前引用的 Bean 实例和另一 Bean 实例进行比较, 因为即便是 Bean 实例相同但有可能不是来自同一个引用, 不能使用 equals() 方法; 方法 remove() 删除当前引用的 Bean 实例, 由容器来决定是否真的释放内存, 通常会返换到组件池中, 删除之后要将对象的引用指向为 null。

类 MedicalCaseServiceBean 是 Bean 实现类, 它继承了抽象类 BaseServiceBean, 抽象类 BaseServiceBean 为默认适配器类, 它不仅实现了 SessionBean 接口, 以供 EJB 容器管理 Bean 使用, 并且它通过方法 performTransaction(Request request) 访问 MDM 请求处理框架中的服务控制器服务, 使用服务控制器的 processRequest(context, request) 实现对病历组件 MedicalCaseComponent 中业务服务的调用。instantiateWccTransactionContext(control, serviceName) 是 BaseServiceBean 类中提供的模板方法, 在其子类 MedicalCaseServiceBean 中提供了实现, 主要是为方法 processRequest(context, request) 提供调用服务的参数 context, 参数 context 主要包括服务类的名称。

MedicalCaseServiceBean 的详细描述如下所示:

```
<enterprise-beans>
```

```
    <session id="Session_1">
```

```
        <ejb-name>MedicalCaseService</ejb-name>
```

```
        <home>com. ibm. ehr. medcase. medicalca  
se. service. MedicalCaseSe  
rviceRemoteHome</home>
```

```
        <remote>com. ibm. ehr. medcase. medicalcase. s  
ervice. MedicalCaseSe rviceRemote</remote>
```

```
        <service-endpoint>com. ibm. ehr. medcase. med  
icalcase. service. Medi  
calCaseServiceSEI</service-endpoint>
```

```
<ejb-class>com.ibm.ehr.medcase.medicalcas  
e.service.MedicalCase  
ServiceBean</ejb-class>  
<session-type>Stateless</session-type>  
<transaction-type>Container</transaction-type>  
<ejb-local-ref id="EJBLocalRef_1">  
    <ejb-ref-name>ejb/com/dwl/base/requestHandler/bea  
        ns/DWLServiceController</ejb-ref-name>  
    <ejb-ref-type>Session</ejb-ref-type>  
  
    <local-home>com.dwl.base.requestHandler.bean  
        s.DWLServ iceControllerLocalHome</local-home>  
    <local>com.dwl.base.requestHandler.beans.DWLService  
        ControllerLocal</local>  
  
    <ejb-link>DWLCommonServicesEJB.jar#DWLServi  
        ceControl ler</ejb-link>  
    </ejb-local-ref>  
</session>  
</enterprise-beans>
```

### 5.3 Web 应用设计

在系统的 Web 应用中, 主要分为 Action 的设计和 Service 类的设计, action 起到前端控制器的作用, 控制器负责处理客户的请求, 并且通过数据验证以后执行服务类的 Web 应用逻辑。Service 核心类为 MedicalCaseService 类和 PersonService 类, 如图 5-8 所示:

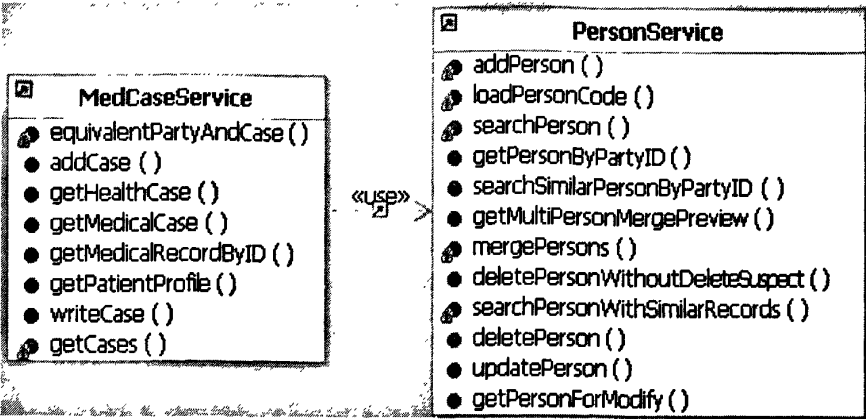


图 5-8 服务层的核心类类图

5.3.1 注册病人功能设计

添加病人的顺序图如图 5-9 所示：

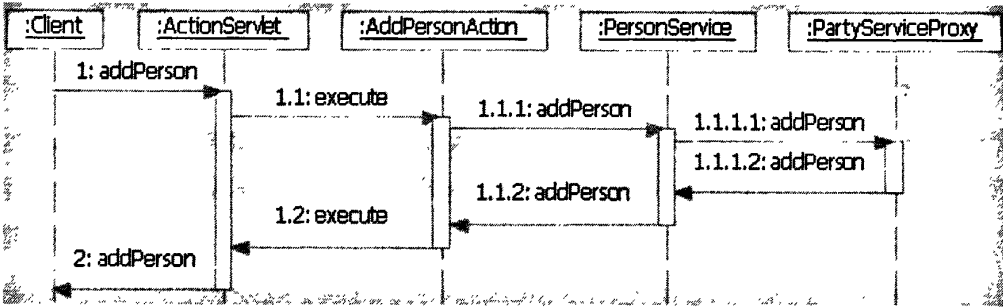


图 5-9 添加病人时序图

1 addPerson 事件触发

客户通过输入病人参数后，提交添加病历按钮，触发添加病人的事件 `addPerson`，事件经由 Web 容器首先到达前端控制器类 `ActionServlet`，前端控制器类 `ActionServlet` 根据事件类型，触发 `AddPersonAction` 行为。

1.1execute (mapping, form, request, response)

`AddPersonAction` 执行行为方法 `execute(mapping, form, request, response)`，在方法中，验证接受到的病人数据的有效性，并调用组件服务的客户端接口方法。

1.1.1 addPerson (person)

在 `addPerson (person)` 方法中，首先将传入的 `person Form` 对象的数据按照 `MDM addPerson` 方法的参数格式进行对象构造和添加，并调用 `PartyServiceProxy` 的 `addPerson` 方法。

1.1.1.1 addPerson (control, person)

调用 MDM 的 `addPerson` 方法, 参数 `control` 为方法的一些控制信息, 包括请求 ID, 请求语言和请求用户, 参数 `person` 为服务组件实体类, 与 form bean 的 `person` 参数不同。

1.1.1.2 返回 `PersonResponse` 对象

1.1.2 返回新添加病人的 ID

1.2 返回添加结果类型

添加操作如果成功, 触发 `getPerson` 事件; 操作如果失败, 返回添加病人界面。

2 返回新添加的病人基本信息或者返回错误信息

### 5.3.2 查询病人功能设计

图 5-10 中阐述了查询某个病人是的活动图, 在整个过程开始, 调用者首先触发了 `getPerson` 的事件, 事件经由 Web 容器首先到达前端控制器类 `ActionServlet`, 前端控制器类 `ActionServlet` 根据事件类型将处理转发给 `AddPersonAction` 行为, `AddPersonAction` 行为调用 `PersonService` 的 `getPersonByPartyID(party id)` 方法后返回查询的病人信息。

返回病人信息后对 `getPerson` 事件的子事件进行如下分析:

- 子事件为查询某个病人的基本信息, 则以只读的方式显示病人详细的基本信息。
- 子事件为修改某个病人的基本信息, 则以可编辑的方式显示病人的基本信息, 以供用户修改病人基本信息。
- 子事件为查询某个病人以后, 对其进行医院预约申请, 则显示病人基本信息与预约医院列表。
- 子事件为查询跟某个病人基本信息相似的病人, 则转发事件 `getSimilarPerson` 给核心控制器 `ActionServlet` 处理。

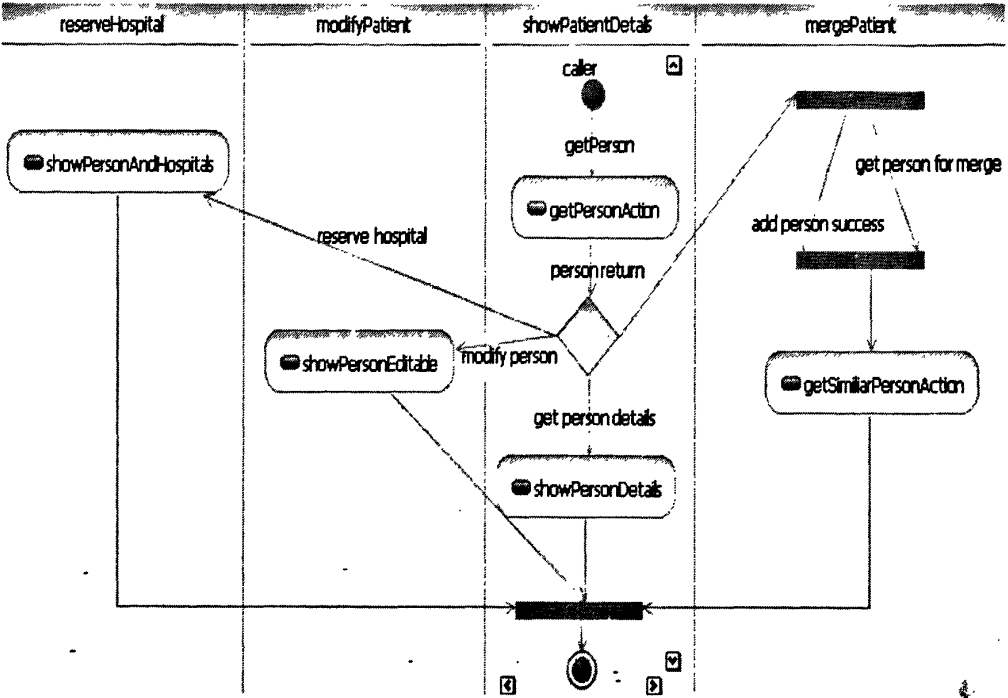


图 5-10 查询病人活动图

5.3.3 合并相似病人功能设计

合并相似病人的顺序图设计如图 5-11:

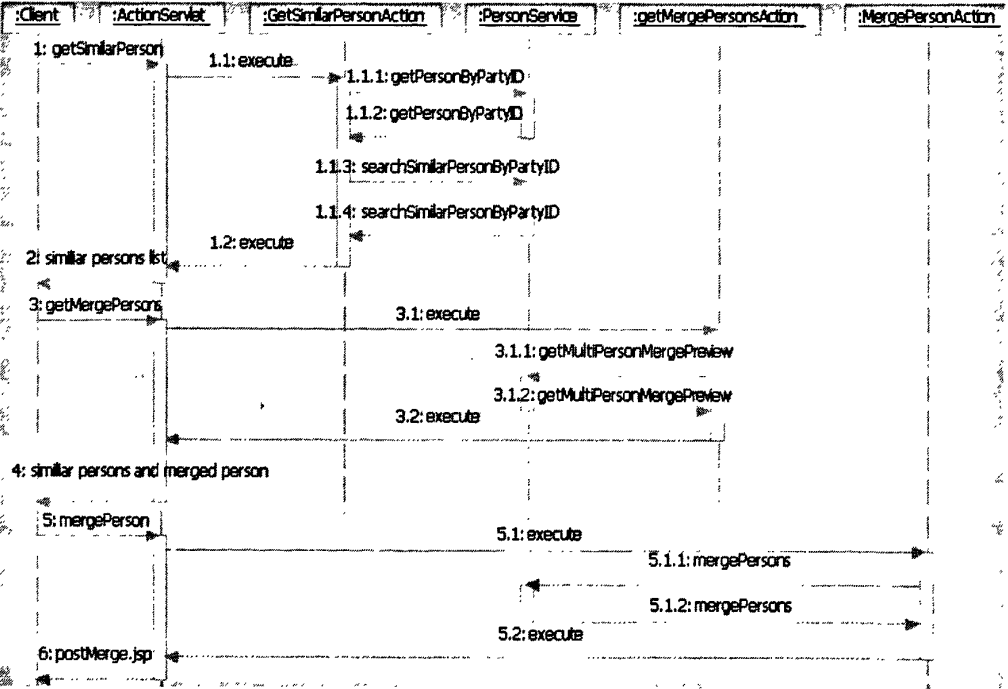


图 5-11 合并病人时序图

### 1 getSimiliarPerson 事件

用户选择某一个病人以后,选择合并操作,触发合并事件,事件经由 Web 容器首先到达前端控制器类 `ActionServlet`,前端控制器类 `ActionServlet` 根据事件类型,触发 `GetSimiliarPerson` 行为。

#### 1.1 execute(mapping, form, request, response)

`GetSimiliarPerson` 行为执行事件处理方法 `execute`,`execute` 方法验证病人 Bean 的 ID 不能为空,并调用 `PersonService` 提供的方法 `getPersonByPartyID`。

##### 1.1.1 getPersonByPartyID (personID)

方法通过调用 `Party` 类的 `getParty` 方法,`getParty` 方法返回 `response` 对象,其中包含了查询结果。

##### 1.1.2 getPersonByPartyID 返回 personBean 对象

从 `response` 对象中得到查询目标病人的基本信息,并且转换成 `personBean` 对象,最为目标病人的基本信息。

##### 1.1.3 searchSimiliarPersonByPartyID(personID)

通过目标病人的 ID 查询相似病人记录,该方法调用 `MDM PartyProxy` 类的 `getAllPartySuspects` 方法,查询相似病人记录信息。

##### 1.1.4 返回查询到的所有相似病人记录

在 `getAllPartySuspects` 返回的 `Response` 对象中,通过 `get` 方法获得相似病人 ID,并通过 ID 查询到病人详细记录以后,转化成 Bean 格式,以便界面显示。最后返回包含所有相似病人记录的 `List` 对象。

#### 1.2 execute 方法返回操作类型标记

`execute` 方法最后判断 `List` 的内容,如果不为空,这返回成功标记,否则返回失败标记。

### 2 返回包括所有相似病人的列表

#### 3 getMergePersons 事件

用户根据查询返回的目标病人的相似病人类别,选择合并操作,触发 `getMergePersons` 事件。事件经由 Web 容器首先到达前端控制器类 `ActionServlet`,前端控制器类 `ActionServlet` 根据事件类型,触发 `getMergePersons` 行为。

#### 3.1 execute(mapping, form, request, response)

`Execute` 方法从 Web 服务器端缓冲的 `Session` 中获得目标病人和其相似病人的 `JavaBean`,并将他们的所有 ID 记录在一个 ID 字符数组中 `IDs`,将其作为参数调用 `PersonService` 的 `getMultiPersonMergePreview` 方法。

##### 3.1.1 getMultiPersonMergePreview (IDs)

方法调用 `PartyProxy` 的 `comparativePreviewCollapseParties` 方法,其中的

参数为 Party 对象数组，数组中的每个 Party 对象的 ID 对象方法 getMultiPersonMergePreview 的参数 IDs 中的 ID。

3.1.1 返回 getMultiPersonMergePreview 查询到目标病人和相似病人 Preview JavaBean 对象

方法 getMultiPersonMergePreview 返回 Response 对象，从 Response 对象通过 get 方法返回包含相似信息记录的 Party 对象数组，从中取出 PartyID 不为空的 Party 对象，并且将其数据格式转换成 Person 的 FormBean 格式并返回。

3.2 execute 方法返回操作类型标记

方法返回操作类型标记，并且将查询到的 Preview 病人对象、目标病人对象，目标对象的相似对象转换为相应的 FormBean，以进行显示。

4 显示目标病人、相似病人和合并病人预览信息。

5 mergePerson 事件

用户修改合并后的病人信息，提交合并操作，触发 mergePerson 事件。事件经由 Web 容器首先到达前端控制器类 ActionServlet，前端控制器类 ActionServlet 根据事件类型，触发 MergePerson 行为。

5.1 execute(mapping, form, request, response)

form 参数是 Preview 病人 Bean，从目标病人对象和目标对象的相似对象中获得他们的 ID，存入 List，调用 PersonService 的 MergePersons 方法。

5.1.1 mergePersons (ids, modifiedBean)

构造 Party 对象数组 parties，数组大小为 ids.size 加 1，将所有相似的病人记录和 modifiedBean 代表的修改后的病人记录存入 Party 对象数组 parties，调用 PartyProxy 的 collapsePartiesWithRules(control info, parties) 方法。

5.1.2 mergePersons 方法返回

5.2 execute 方法返回操作类型标记

6 转向到合并病人页面

### 5.3.4 病人预约就诊功能设计

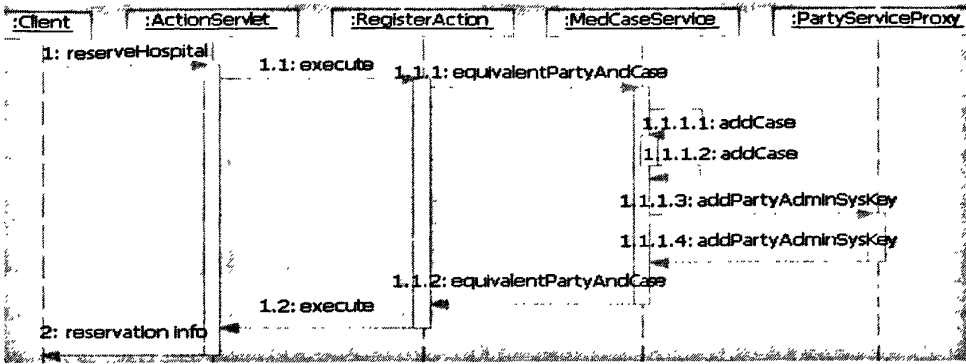


图 5-12 转诊病人时序图

1 触发 reserveHospital 事件

用户在查询病人后所得的病人列表中，选择医院预约操作，触发了 reserveHospital 事件,事件经由 Web 容器首先到达前端控制器类 ActionServlet，前端控制器类 ActionServlet 根据事件类型，触发 Register 行为。

1.1 execute(mapping, form, request, response)

参数 form 中包含了 PatientID，execute 方法验证 PatientID 的有效性，并构造 MedCaseService 对象,调用 MedCaseService 对象的 equivalentPartyAndCase 方法。

1.1.1 equivalentPartyAndCase (partyID)

方法因为要执行两个不同的原子服务，并且要求服务执行的完整性，所以构造事务 UserTransaction 对象，以采用事务操作。构造 PartyAdminSysKey 对象，该对象表示病人和病历的关系，将 partyID 赋值给 PartyAdminSysKey 对象的 PartyID 属性。调用 addCase 方法。

1.1.1.1 addCase ()

方法 addCase 依次构造 MedicalCase 业务对象和 MedicalCaseServiceProxy 对象，并调用 MedicalCaseServiceProxy 对象的 addMedicalCase (medicalCase) 方法添加新的病历记录。

1.1.1.2 addCase () 返回新添加的病历 ID

1.1.1.3 addPartyAdminSysKey (control, partyAdminSysKey)

构造 PartyProxy 对象,将 addCase()返回的病历 ID 赋值给 partyAdminSysKey 的 AdminSysPartyID，调用 PartyProxy 对象的 addPartyAdminSysKey 方法。

1.1.1.4 addPartyAdminSysKey (control, partyAdminSysKey) 返回结果值 addPartyAdminSysKey 返回 Response 对象。

1.1.2 equivalentPartyAndCase 返回新添加的病历 ID

1.2 execute 方法后续处理



将 equivalentPartyAndCase 方法返回的 ID 赋值给 caseBean 的 ID 属性，将 caseBean 设置在 Request 对象，以便提交给用户。

2 reservation Info

返回病人预约医院的操作信息，如果预约成功，则显示预约信息，预约信息主要包括预约病历 ID（作为预约号使用），预约医院，预约医院的门诊、预约日期和预约期限提示信息，同时包括病人的基本信息。如果预约失败，这提示用户预约失败，并尽量显示操作失败的原因，以提高操作的人性化功能。

5.3.5 记录预约病历功能设计

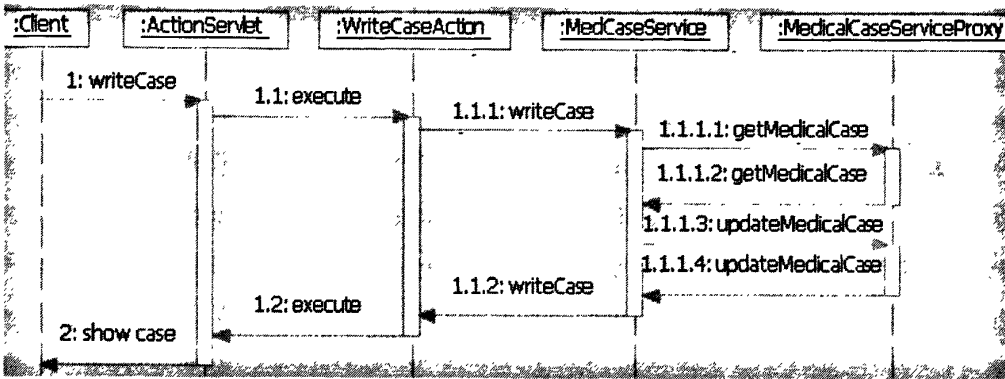


图 5- 13 记录预约病历时序图

1 触发 writeCase 事件

用户通过预约号码查询病人的基本信息和新添加的病历成功后，用户选择写病历操作，触发了 writeCase 事件。事件经由 Web 容器首先到达前端控制器类 ActionServlet，前端控制器类 ActionServlet 根据事件类型，触发 WriteCase 行为。

1.1 execute(mapping, form, request, response)

将参数 form 转换成 CaseBean 对象，CaseBean 对象中包含了预约号，预约号同时又是病历号 caseID，execute 方法验证 caseID 的有效性，并构造 MedCaseService 对象，调用 MedCaseService 对象的 writeCase 方法。

1.1.1 writeCase (caseBean)

方法因为要执行两个不同的原子服务：查询病历服务和更新病历服务，从而要求服务执行要具有完整性，所以构造事务 UserTransaction 对象，以采用事务操作。构造 MedicalCaseServiceProxy 对象，然后调用 MedicalCaseServiceProxy 对象的 getMedicalCase 方法。

1.1.1.1 getMedicalCase (caseBean)

方法 `getMedicalCase` 调用 `MedicalCase` 组件的方法获得病人在预约时创建的病历对象。

#### 1.1.1.2 `getMedicalCase (caseBean)` 返回查询到的 `Response` 对象

#### 1.1.1.3 `updateMedicalCase (control, medicalCase)`

从 `Response` 对象中获得查询到病历对象，并将病历对象赋值给新构造的 `MedicalCase` 引用，将 `caseBean` 中的属性（病历修改日期、主治医师姓名、病人症状信息、诊断记录信息和治疗方案信息）赋值给 `medicalCase` 对象，最后调用 `MedicalCaseServiceProxy` 对象的 `updateMedicalCase` 方法。

#### 1.1.1.4 `updateMedicalCase (control, medicalCase)` 执行后返回

方法 `updateMedicalCase` 若能够成功执行，则不抛出异常，否则抛出处理异常 `ProcessingException`。

#### 1.1.2 方法 `writeCase (caseBean)` 执行后返回

#### 1.2 `execute` 方法后续处理

如果 `writeCase` 方法执行成功，则返回更新后的病历信息，否则，返回错误信息。

#### 2 `show case` 显示更新病历操作后的信息

## 6 系统实现

系统的主要功能实现包括：病人基本信息管理、病人预约就诊管理。根据非功能性需求中的可用性中的要求，设计界面元素并实现系统对于功能。

### 6.1 病人基本信息管理

病人基本信息管理的功能为：注册病人、检索病人、合并病人、查看病人基本信息、修改病人基本信息和删除病人记录。每一个功能都对应不同的操作界面，其中合并病人和管理病人基本信息功能模块的实现基于检索病人功能模块，即检索病人操作界面是合并病人操作界面与管理病人基本信息操作界面的入口界面。进入该子系统以后默认的操作界面是检索病人，如图 6-1 所示：

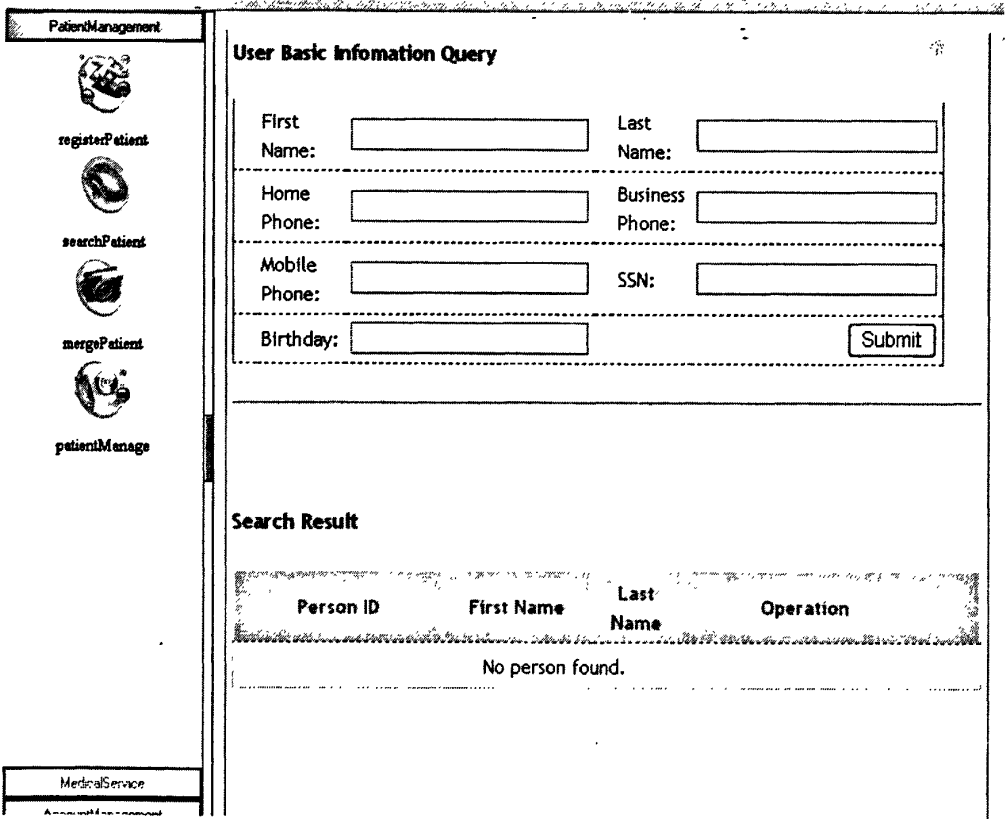


图 6-1 病人基本信息管理主界面

#### 6.1.1 注册病人

通过位于病人基本信息管理主界面左侧的工具栏，选择 registerPatient 操

作标签，可以进入注册病人操作界面，如图 6-2 所示：

Basic Information

First Name:

mik

Last Name:

green

Gender:

☒ Male ☐ Female

Birthday:

02/10/1984

Marital:

☐ Married ☒ Single ☐ Divorced

SSN:

13012511112223

Insurance No.:

1235646

Birth Place:

China

Contact Message

Home Phone

86 - 010 - 45896325

Bussiness Phone

86 - 010 - 45628456

Mobile Phone

156698584752

Address Detail

Home District:

Country 

China

 Province 

Beijing

 City 

Beijing

Home Address:

NO.28 Dong Bei wang

Zipcode:

10023

Submit

图 6-2 注册病人操作界面

添加病人成功以后，显示添加成功以后的病人基本信息。如果有相似的病人记录，则将相似病人搜索出来，供用户选择合并操作，如图 6-3 所示：

Personal Information

First Name:

mike

Last Name:

green

SSN:

13012511112222

Gender:

M

Birthday:

02/10/1984

Insurance No:

1235643

Birthday Place:

China

Home Phone:

86 - 010 - 59648956

Bussiness Phone:

86 - 010 - 2659483 -

Mobile Phone:

15364856248

Home Address:

NO.28 Dong Bei Wang , Beijing , AK , China

Zipcode:

10042

Similar Person List

☐

Person ID

SSN

First Name

Last Name

☐

552127389035440687

13012511112222

mike

green

Submit

图 6-3 添加成功后搜索相似的病人记录

6.1.2 合并病人

通过位于病人基本信息管理主界面左侧的工具栏，选择 mergePatient 操作标签，可以进入相似病人操作界面，如图 6- 4 所示：

The following patients have possible similar records

Person ID	First Name	Last Name	Operation
256127389431257892	mike	green	<div>Similar</div>
456127391865665518	mke	green	<div>Similar</div>

图 6- 4 搜索所有存在相似记录的病人

通过 Similar 操作，可以选择其中的某个病人的相似记录，进而进入 Merge 操作界面：如图 6- 5 所示：

The following patients will be merged this time

Patient ID: 234127389068303118

First name:mke

Last name:green

Gender:M

Birth day:02/10/1984

SSN:13012511112222

Insurance ID:1235643

Birth Country:China

MobilePhone:15364856248

Address:

Country:China

Province:AK

City:Beijing

zipcode:10042

Address:NO.28 Dong Bei Wang

Patient ID: 552127389035440687

First name:mike

Last name:green

Gender:M

Birth day:03/12/1984

SSN:13012511112222

Insurance ID:1235642

Birth Country:China

Mobile Phone:

Address:

Country:China

Province:AK

City:Beijing

Zipcode:

Address:NO.18 Jiao Da Dong lu

Basic Information

First Name

mke

Last Name

green

Gender

☒ Male ☐ Female

Birth day

02/10/1984

SSN

13012511112222

Insurance ID

1235643

BirthPlace

China

Contact Method

Home Phone

86

- 010

-

59648956

Bussiness Phone

86

- 010

-

2659483

-

Mobile Phone

15364856248

Address Detail

HomeDistrict

CountryChina

Beijing

provinceBeijing

- city

HomeAddress

NO.28 Dong Bei Wang

Zipcode

10042

Submit

图 6-5 合并病人操作界面

6.1.3 病人基本信息的查看、修改与删除

通过位于病人基本信息管理主界面左侧的工具栏，选择 PatientManage 操作标签，进入病人基本信息管理操作界面，通过查询以后，可以进行病人基本信息

的查看、修改和删除操作，如图 6- 6 所示：

User Basic Infomation Query

First Name:	<input type="text"/>	Last Name:	<input type="text" value="green"/>
Home Phone:	<input type="text"/>	Business Phone:	<input type="text"/>
Mobile Phone:	<input type="text"/>	SSN:	<input type="text"/>
Birthday:	<input type="text"/>	<input type="button" value="Submit"/>	

Search Result

Person ID	First Name	Last Name	Operation
256127389431257892	mike	green	<input type="button" value="detail"/> <input type="button" value="delete"/> <input type="button" value="modify"/>
456127391865665518	mke	green	<input type="button" value="detail"/> <input type="button" value="delete"/> <input type="button" value="modify"/>

图 6- 6 病人基本信息的查看、修改与删除操作界面

6.2 病人预约就诊

全科医师通过工具栏选项 MedicalService 下的 HospitalRegister 操作，可以对病人进行转诊处理，此过程同样基于查询服务，通过查询进入预约就诊操作界面，如图 6- 7 所示：

PatentManagement

MedicalService

HospitalRegister

PersonalMedicalRecords

User Basic Infomation Query

First Name:	<input type="text"/>	Last Name:	<input type="text" value="green"/>
Home Phone:	<input type="text"/>	Business Phone:	<input type="text"/>
Mobile Phone:	<input type="text"/>	Social ID:	<input type="text"/>
Birthday:	<input type="text"/>	<input type="button" value="Submit"/>	

Search Result

Person ID	First Name	Last Name	Register
256127389431257892	mike	green	

图 6- 7 病人预约操作界面

6.2.1 预约请求

全科医师选择 Register 操作以后，进入预约处理操作界面，如图 6- 8 所示：

<b>Personal Basic Infomation</b>	
Name: mike green	SSN: 13012511112222
InsuranceID: 1235643	
<b>Hospital Register</b>	
Hospital: People's First Hospital	Department: Internal Medicine
Appointment Date: 06/20/2010	Submit

图 6- 8 预约请求操作界面

预约成功以后的详细信息如图 6- 9 所示：

<b>Hospital Register Sheet</b>	
No:210981761543	
<b>People's First Hospital</b>	
Register ID:	992127389463042120
Department:	Internal Medicine
Register Date:	06/20/2010
Remark:	Registered only from the community hospital, Haidian District, Beijing, is valid for 2 days

图 6- 9 预约详细信息界面

6.2.2 查询预约记录

通过 Register ID 查询预约记录信息，进入预约病历操作界面，在病历操作界面可以选择 Write Case 操作进行预约病历的记录操作，如图 6- 10 所示：



Appointment

Search Criteria

App. ID: 992127389463042120

Search

Reset

Patient Profile

Basic Information

First name	mike	Last name	green
Gender	M	Birthday	03/12/1984
Social Security No.	13012511112222	Insurance No.	1235643
Birth Place	36		

Contact Information

Home Tel.	86 - 010 - 59648956
Biz Tel.	86 - 010 - 2659483 -
Mobile	15364856248

Address Information

Country	China	State/Province	AK
City	Beijing	Zipcode	100044
Home address	NO.18 Jiao Da Dong Lu		

VIEW CASES

WRITE CASE

图 6- 10 查询预约信息操作界面

选择写病历操作以后，进入写病历主界面，如图 6- 11 所示：

Write Medical Case

Basic Information

Patient ID	992127389463042120
Doctor	John Frank

Symptom

Loss of appetite ,Diaphoresis ,Abnormal sense of taste,Vomiting Cough

Diagnosis

Flu

Treatment

stay at home ;Get plenty of rest ;Drink a lot of liquids ;Do not smoke or drink alcohol.Amantadine.

WRITE

Reset

图 6- 11 写病历操作界面

6.2.3 查询预约病历

全科医师通过工具栏选项 MedicalService 下的 PersonalMedicalRecord 操作，可以查询病人的预约就诊病历，此过程同样基于查询服务，通过查询进入查

看预约病历操作界面，如图 6- 12 所示：

User Basic Infomation Query

First Name:

Last Name:

green

Home Phone:

Business Phone:

Mobile Phone:

Social ID:

Birthday:

Submit

Search Result


Person ID	First Name	Last Name	Medical Record
256127389431257892	mike	green	

图 6- 12 查询病历操作界面

## 7 结论

### 7.1 完成工作

本文主要阐述了以下内容:

1. 对全局病人索引预约就诊系统进行需求分析,并且解释全局病人索引预约就诊系统在不同医疗机构中所起到的作用,以及它和其他医疗系统之间的关系,充分说明了该系统的应用价值。
2. 基于系统需求用例,采用面向对象的分析方法,进行系统分析,建立领域模型,发现系统行为,最终确定系统的功能,为系统设计提供铺垫。
3. 以用例视图为中心,进行系统体系结构设计,分别就系统的逻辑、运行和物理架构进行了详细阐述,为系统的详细设计奠定基础。
4. 分别从客户端和服务端,阐述了系统的详细设计。在系统的详细设计中,采用了 UML 面向对象的建模与设计技术,对系统的功能实现进行建模。
5. 介绍了系统的开发技术、开发工具和开发平台。

系统实现采用 Web Services 调用的方式,采用 JavaEE 技术进行实现,实现了服务的灵活性和跨平台性。

### 7.2 工作体会

在系统的设计和开发的过程中,实际的体会到了迭代式的软件开发方法,以及团队合作的重要性。在软件开发的工程中,应该对需求有准确的定位和把握,对需求中可能发生变化的部分,在不会影响设计约束的条件下,应该采用支持可配置的灵活的编程方式,适当的采用设计模式可以大大的提高系统开发的灵活性和并行开发的效率。基于组件式的软件开发方法,需要基于组件式的开发工具,IBM Rational Software Architect 集成了软件分析、设计和开发的许多功能,大大方便了软件的实施工程。

通过在公司参加该系统的项目开发,使我在实践中深刻的体会到了软件开发的流程和软件工程的思想,并认识到了理论指导的重要性。在需求分析的过程中,采用用例分析的方式对用户的功能性需求进行描述,既直观,有详细。重复的发现用户操作的业务流程,为系统的功能性设计和实现奠定了基础。对于以业务处理为主应用场景,采用 UML 面向对象的分析与设计的方法,可以很好的对系统的

实现提供素材,实现需求到实现的无缝连接和平滑过渡。在系统设计的过程中,以用例为中心,分别从逻辑、开发、物理和部署的不同视图进行建模,可以为后期项目的开发、维护和部署人员提供素材和指导,从而避免设计的不全面性。

作为一个软件从业者,应该具有良好的沟通能力和应变能力,英语交流的能力很重要,尤其在商业活动日益国际性的大环境下,具备良好的英语交流能力可以应对许多机遇和挑战。

### 7.3 后续工作

本系统的服务实现虽然具有跨平台的特性,但是并没有采用 ESB 总线技术,ESB 总线技术是 EAI 技术的一个延伸。就好像一条大河,每个系统里的数据好像小河里的水,都汇集到大河里,再由大河管理疏导这些水,应该流到哪里就输送到哪里。ESB 总线技术的关键是不论系统承载的数据是什么样子,但是流到大河里的水必须是标准格式,是可以任意交换的。这个标准,使得数据流都能以服务的形式提供出来,非常灵活,也不会产生数据丢失的问题。这样做的好处是一劳永逸的解决了一个医院内部信息共享的问题,还同时解决了医院和外界信息交换的问题,它搭起了一座桥梁,把医院内的信息孤岛连通,也同时开启了医院与外界交流的大门。

本系统实现了跨医院的预约就诊,同时提供了全局病人的索引管理功能,但是本系统开发并没有基于具体的医疗标准,这对系统的长期发展提供了挑战,尽管系统采用 Web Services 对服务进行了封装。系统的后期工作为结合 Health Level 7 卫生信息交换标准,进行系统服务组件的二次开发。

## 参考文献

- [1] 陶晓慧, 马晓昱. 浅析我国医疗信息行业发展策略. [EB/OL]. [http://cio.ccw.com.cn/research/hangye/htm2008/20080710\\_462598.asp](http://cio.ccw.com.cn/research/hangye/htm2008/20080710_462598.asp). 2008-07-10/2010-02-30
- [2] 吴勇毅. 医疗信息化市场 8500 亿商机诱人. [EB/OL]. <http://news.cnfol.com/090403/101,1280,5691412,00.shtml>. 2009-04-03/2010-02-30
- [3] 韩立场. 关于医院信息化建设中的信息孤岛问题的几点思考. [EB/OL]. <http://beijhello.blog.sohu.com/104959027.html>. 2008-11-22/2010-02-30
- [4] Soloman I. Appavu. Analysis of unique patient identifier options final report.Part3 Unique Patient Identifier. [EB/OL]. <http://www.ncvhs.hhs.gov/app0.htm>. 1997-12-24/2010-02-30
- [5] W3C Working Group. Web Services Architecture. [EB/OL]. <http://www.w3.org/TR/ws-arch>. 2004-02-11/2010-02-30
- [6] David Chappell, Tyler Jewell. Java Web Services. First Edition.US. O'Reilly.2002-03.172 页至 173 页.
- [7] 柴晓路, 梁宇奇. Web 服务技术、架构和应用.第一版.北京.电子工业出版社.2003-01. 18 页至 25 页.
- [8] IBM corporation.IBM InfoSphere Master Data Management Server v8.0.1 information center.[EB/OL].<http://publib.boulder.ibm.com/infocenter/mdm/v8r0m0/index.jsp>.20089-12/2010-02-30
- [9] 王和全.深入 Struts 1.1.[EB/OL].<http://www.ibm.com/developerworks/cn/java/l-struts1-1/>. 2003-08-02/2010-02-30
- [10] John Quinn.张琨. 电子病历系统中患者统一身份识别管理:美国的一些经验. 2006 中华医院信息网络大会暨中美医院信息化论坛论文集. 西安建国饭店.2006. 836 页至 841 页.
- [11] Dirk Krechel The LENUS Master Patient Index: Combining Hospital Content Management with a Healthcare Service Bus [Dissertation]. IEEE Xplore.2008.page 170-page172
- [12] PeterM. Yellowlees,ShaynaL.Marks,MichaelHogarth,StuartTurner.Standards-based,open-source electronic health record systems: a desirable future for the U.S. health industry.Telemedicine Journal and e-Health. 2008.vol.14.no.3 . page 284-page288
- [13] Grady Booch,James Rumbaugh,Ivar Jacobson.The Unified Modeling Language Reference Manual. SECOND EDITION.US. Addison Wesley Professional.2004-07.page 77-page80.
- [14] Len Silverston.The Data Model Resource Book Revised Edition Volume1-A Library of Universal Data Modelsfor All Enterprises.first edition.NEW YORK. Wiley Computer.2001-01.page 29-page57
- [15] Craig Larman.Applying UML and Patterns.third edition. 北京. 机械工业出版社.2006-01.page 29-page57

## 作者简历

- 教育经历

2004 年 9 月---2008 年 7 月 河北科技大学 信息管理与信息系统专业

2008 年 9 月至今 北京交通大学 软件学院

- 工作经历

2008 年 5 月至今 国际商业机器（中国）投资有限公司

## 独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作和取得的研究成果，除了文中特别加以标注和致谢之处外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得北京交通大学或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

学位论文作者签名：周亚卫      签字日期：2010年06月23日

学位论文数据集

表 1.1： 数据集页

关键词*	密级*	中图分类号*	UDC	论文资助
病人索引；就诊 预约；全局病人 基本数据；Web Services	公开	TP311.52	004.41	
学位授予单位名称*		学位授予单位代码*	学位类别*	学位级别*
北京交通大学		10004	工程	硕士
论文题名*		并列题名		论文语种*
基于 IBM MDM Web Services 的 病人主索引预约就诊系统的设计 与实现				中文
作者姓名*	周亚卫		学号*	08122450
培养单位名称*		培养单位代码*	培养单位地址	邮编
北京交通大学		10004	北京市海淀区西 直门外上园村 3 号	100044
工程领域*		研究方向*	学制*	学位授予年*
软件工程		软件工程	2 年	2010 年
论文提交日期*	2010 年 6 月			
导师姓名*	赵宏		职称*	副教授
评阅人	答辩委员会主席*		答辩委员会成员	
	张遂征			
电子版论文提交格式 文本 (√) 图像 ( ) 视频 ( ) 音频 ( ) 多媒体 ( ) 其他 ( ) 推荐格式: application/msword; application/pdf				
电子版论文出版 (发布) 者		电子版论文出版 (发布) 地		权限声明
论文总页数*	72			
共 33 项, 其中带*为必填数据, 为 22 项。				