火龙果•整理
uml.org.cn

华中科技大学

硕士学位论文

Android平台环境下应用程序架构的设计与实现

姓名：罗震

申请学位级别：硕士

专业：工业工程

指导教师：黄金国

20090523

火龙果•整理
uml.org.cn

华 中 科 技 大 学 硕 士 学 位 论 文

摘 要

无线通信业和因特网的迅猛发展和融合直接导致了智能手机需求的大幅增长，智能手机去年在全球的销量将近 2 亿，并且发展势头日益猛烈，市场前景一片大好。随着 3G 技术的发展，手机的功能越来越强大，消费者对智能手机的要求也越来越高，这也直接导致了手机软件设计的质量和效率将变得越来越重要。而智能手机软件复杂度与开发周期的矛盾，手机系统资源有限与功能众多的矛盾，网络下载与安全隐患的矛盾，使传统软件开发模式成为智能手机软件开发的严重障碍。

为了提高手机软件开发的质量和效率，本文提出了在底层平台与上层应用之间设计一个中间架构层，建立一个较为稳定的软件开发框架的思想。本文首先对软件架构理论进行了分析，在对架构设计的条理性原则和可靠性原则进行了充分权衡之后，将该架构分为四层，从上到下依次是：应用层，安全层，业务层和适配层，各层相对独立。应用层负责手机应用的初始化、关闭以及相关控件的工作；安全层负责保护数据，防止病毒木马等恶意攻击；业务层负责包装各类手机应用业务，并向上提供相关服务给应用层调用；适配层则负责与协议栈的数据交互。

本文的试验选用 Android 平台。Android 是 Google 开发的基于 Linux 平台的开源手机平台，为我们提供了一系列的 API 和开发工具包，它包括操作系统、用户界面和应用程序 ——移动电话工作所需的全部软件，而且不存在任何以往阻碍移动产业创新的专有权障碍。

在本文的最后，应用这个架构，我们在 Android 平台上开发出 GTalk 这款即时聊天软件，证实了该架构的可应用性。

关键词：Android 平台　软件架构　分层模式　消息映射　业务代理对象

火龙果•整理
uml.org.cn

Abstract

The rapid development of wireless communication and internet technology, as well as their fusion directly result in the rapid demand increase of smart phone. In the past year, the global sales volumes of smart phone reach nearly 200,000,000, and the trend of development is increasingly evident and the foreground of market is bright. With the development of 3G technology, the functions of smart phone become more and more strong, and customers' desire to smart phone becomes more and more high, which directly result in the necessary that the quality and efficiency of software development on smart phones should get more importance. However, the contradictions between software complexity and development cycle, between limitation of system source and the diversification of functions and between download from net and security cause the traditional mode to become a serious obstacle of software development on smart phones.

In order to improve the development quality and efficiency of software on smart phone, this thesis proposes a way that designs a middle architecture between the upper application and the lower platform so as to form a comparatively stable framework of software development. This article first make a analysis of the software architecture theories, then make a serious consideration in the principle of coherence and the reliability, based on which we divide the whole architecture into four layers, and they are application layer, security layer, business layer and adaptive layer in the order that from up to down, which are respectively independent to other layers. The application layer is responsible for the initialization, close of application and the task of related controllers. The security layer is responsible for protecting data so as to keep virus and trojan from attacking the system. The business layer is responsible for the packaging of various application businesses, and provides the service for the application layer. The adaptive layer is responsible for the data interaction with protocol stack.

In this thesis, the test chooses Android platform. Android is an open source cell phone platform based on Linux platform developed by Google, which provides us a series of API and development toolkits. It contains operating system, user interface and applications—all the software the cell phone needs during operation, and there's no obstacle in exclusive rights that counteracts the innovation of mobile industry.

火龙果•整理
uml.org.cn

In the end of this thesis, applying to architecture, an instant messaging software named Gtalk is developed, and the applicability of the framework is confirmed.

Key Words: Android platform; software architecture; layered mode; message mapping; business-broker object

火龙果•整理
uml.org.cn

华 中 科 技 大 学 硕 士 学 位 论 文

独创性声明

本人声明所呈交的学位论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除文中已经标明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的研究成果。对本文的研究做出贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：

日期：　　年　　月　　日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，即：学校有权保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权华中科技大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

本论文属于　保密□，在_____年解密后适用本授权书。

不保密□。

（请在以上方框内打“√”）

学位论文作者签名：　　指导教师签名：

日期：　　年　　月　　日　　日期：　　年　　月　　日

火龙果•整理
uml.org.cn

1 绪论

1.1 智能手机软件研发现状

所谓智能手机，是指具有操作系统的手机[1]。从内部来说，它支持应用程序的轮转调度和多线程的切换。从外部来看，它能够支持第三方软件的安装和应用。从这个意义上来看，智能手机完全可以被称为一部小型电脑。

1.1.1 乐观的一面

在移动市场激烈的争夺中，手机的发展已然经过了模拟通讯时代、数字通讯时代、过渡时代、多媒体时代的演变，现在，已经进入 3G 时代[2]，所谓的 4G 超宽频时代也已经初见端倪。人们对智能手机提出越来越新的要求，智能手机的功能越来越多，现在的移动设备在功能上与 PC 机的界限已经越来越模糊。实现智能移动已经不仅仅是手机厂商们的梦，而是大势所趋，人心所向，谁也无法阻挡。手机的功能越来越强大，消费群体越来越庞大，无线因特网也在呈现日新月异的发展，两者的结合正是珠联璧合般的完美，这正是智能手机产生的良好背景[3]。

开源是智能手机发展的一个新趋势[4]，目前，智能手机厂商和运营商都宣布了自己的开源战略或产品。一年前，Google 推出了 Android 开源移动平台计划，并于今年 9 月推出了安装 Android 操作系统的第一款手机 G1。开源意味着向全世界公布源代码，人人都可以对系统进行修改，从而使系统变得更加完善。

IDC（Internet Data Center，互联网数据中心）每年都会对手机市场的发货量进行报导[4]。根据这几年的报导数据，我们可以看出，全球手机的发货量呈现飞速增长的趋势，而增长最为强劲的是智能手机，它的发货量同比增长率每年都会超过 100%。可以预见，智能手机发展的势头必将有增无减，这充分证明智能手机的研发前途无量。

1.1.2 面临的挑战

任何事物都有两面性，主要表现在：智能手机销量的疯狂增长意味着必须提高开发效率，缩短开发周期，而消费者对移动设备的功能需求越来越高，这意味着手机软件的开发过程将更为复杂和耗时。这正是矛盾的所在。我们必须找到一个途径：在提高产品性能的过程中，既可以保证质量，又可以缩短开发周期。

要达到这个面的，有两个途径：

（1）从硬件上考虑，优化手机硬件系统架构[5]。方法之一是将传统的既处理通信协议又实现应用功能的单一高性能内核处理器系统架构和基带处理器+应用处理器的系统架构优化为采用多个不同的处理器内核的系统架构。随着集成技术的发展和处理器成本的下降，这种架构必将成为一种趋势。

（2）从软件上考虑。方法之一是扬弃传统的面向对象的编程思想，采用构件技术[6]。文献 6 中阐述的“和欣”技术就是这样的例子。

我们对这两种方案进行权衡，得出的结论是：硬件方案行不通。原因是，就算我们在手机里面采用多处理器内核，让它能处理一切事务，但是移动设备不能不考虑功耗方面的问题，有哪个用户能忍受每天为手机充电呢？更何况主要的先进芯片技术都掌握在国外几大厂家手中[7]。因此，从硬件着手提高手机功能既不明智，又不符合中国的国情。我们只能从软件方面着手解决这个矛盾。

1.2 研究的目的和意义

通过对智能手机研发现状的分析，我们知道，解决智能手机开发问题最有效的途径就是改进软件的开发思路。就如生产关系的发展推动生产力的不断进步一样，需求的不断快速增长同样刺激了软件开发方式的不断改良，软件开发先后经历了随意编程→函数的使用→结构化→模块化→面向对象→构件化等阶段。到了现在这个时代，耦合程度依然较高的面向对象的软件开发技术已经成为快速高涨的软件需求的阻碍因素，这样的开发方式有一些固有的问题[7]：(1）代码重用率很低，开发周期长（2）界面代码与逻辑功能代码混在一起，结构不够清晰，难于维护（3）所有代码支付给用户，知识产权易受侵犯。

这样的软件开发模式在现实中已经引起了很多的问题，轻则导致功能异常，重则引起频繁死机[8]。另外，智能手机的开发是以手机平台为基础的，当前存在的大部分手智能手机开发平台允许用户在平台上开发第三方软件，创建新的应用，充分发挥自己的聪明才智，打造属于自己的个性化手机。但是，情况并不简单，不同手机平台一般拥有不同的规范[9]，对于上层应用软件的开发就没有一个统一的设计和架构，这样一个百花齐放的环境对于软件业来说是一个严重的阻碍。这种情况造成的后果是，开发一套新软件，就设计一个开发新架构，开发者们将大量的时间花费在对新应用的结构设计上，代码复用率极低，不利于缩短开发周期和保证软件质量。在这样的情况下，我们考虑对软件本身的开发手段进行改进，方法之一是对手机软件进行合理的架构设

火龙果•整理
uml.org.cn

计[10]，使所有的软件开发都基于这样的架构，让开发者将更多的精力专注于软件本身性能的研究上来，而不用将太多心思花费在整体架构和设计模式上来。

基于这样的考虑，本文选用 Android 手机平台作为实验平台，在这个平台之上设计了一个中间架构层。对上层软件的开发（GTalk 即时聊天软件的开发）进行架构设计，建立了一个应用开发框架。程序员的所有的应用开发工作都基于这样一个架构层，从而大大的提升开发效率和产品性能。最后，我们在 Android 手机平台上应用该架构实现了 Gtalk 即时聊天软件的开发，验证了该架构的成功性。

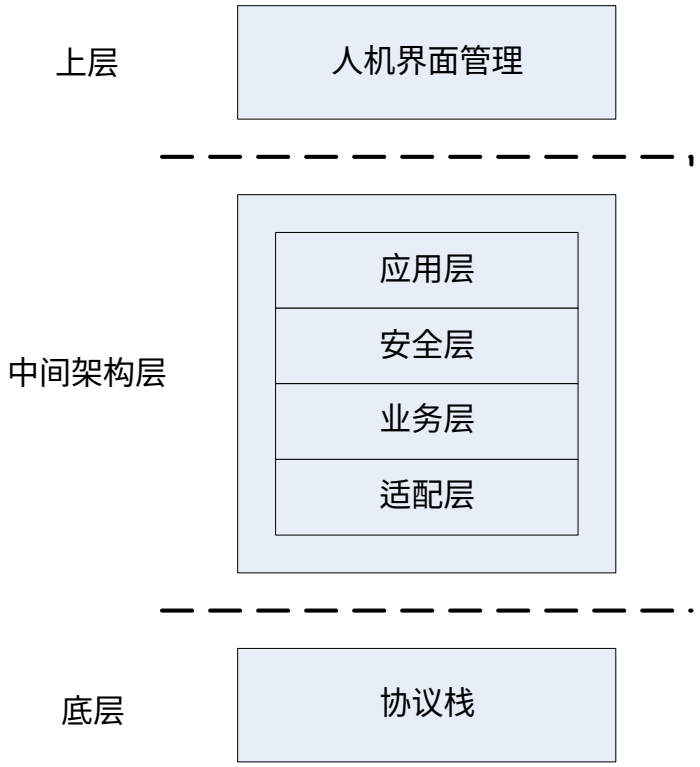

图 1-1 中间架构层的结构

1.3 本文的研究内容

本文的内容主要包括以下几个方面：

(1)课题背景和研究意义；

(2)应用程序架构的相关理论和国内外研究现状；

(3)平台之上的中间层的设计，由四个层次组成——应用层、安全层、业务层、适配层；

火龙果•整理
uml.org.cn

(4)基于中间层架构的 Google Talk 即时聊天软件的设计和实现；

(5)今后工作的展望。

1.4 论文的组织结构

本文后续章节的内容作如下安排：

第二章：对应用程序架构理论进行综述，包括应用程序架构的模式和目标，总结出了软件架构的两个要素和必备的六个组件，并对中间架构层的设计给出了大致思路。

第三章：阐述本文对中间架构层的设计和各层的功能，具体包括应用层、安全层、业务层和适配层。在设计过程中，给出了架构层的可靠性估计模型，并且给出了一种适合手机应用的消息传递机制。

第四章：是架构设计的具体应用实例，应用这个架构，我们在 Android 平台上成功的开发出 Gtalk 这款即时聊天软件。本章对各个模块应用基于架构层的开发进行了详细的描述，充分验证了本文设计的中间架构层的成功性。

第五章：归纳了本文的研究成果，提出了现在仍然存在的不足之处，并提出了改进的方法。

火龙果•整理
uml.org.cn

2 应用程序架构理论

2.1 应用程序架构理论综述

2.1.1 应用程序设计的模式

从抽象到具体的区分来看，应用程序设计一般划分为三种模式[16]:架构模式、设计模式、和代码模式。

架构模式是一种高级别的设计，它事先定义好子系统，并规定它们之间的相互关系。它就好比建筑师对房屋的整体规划，是抽象的而不是具体的，其好坏可以影响到总体布局和框架性结构。设计模式是中等规模的模式。它们在规模上比架构模式小，但又独立于特定编程语言和编程惯例。它好比房屋内各个造型之间的建筑关系，例如，或附属，或相邻，它是介于具体和抽象之间的层次，其好坏不会影响到系统的总体布局和总体框架，但可能对其中的子系统有较大影响。代码模式是特定的范例和与特定语言有关的编程技巧。它好比建筑工人的具体施工，比如卧室的油漆应该怎样刷，客厅的吊顶如何悬挂，它是真正具体实施的层次，其好坏会影响到一个中等尺度组件的内部、外部的结构或行为的底层细节，但不会影响到一个部件或子系统的中等尺度的结构，更不会影响到系统的总体布局和大尺度框架。

三个模式各具特色，对于一个真正的软件设计师来说，它们都是必不可少的。基于本文研究内容，这里只对架构模式理论稍作介绍。架构模式[16]描述软件系统里的所有的子系统和组件以及它们之间的协调关系。它也叫做系统模式，它是位于抽象层的，起到统筹全局作用的模式，也是软件设计最重要的一步。

架构模式常常划分成如下的几种[16]:

(1)从多用户层面到结构型：包括 Layers mode、Blackboard mode、Pipes/Filters mode 等。其中分层模式已广受程序员的喜爱，大家熟悉的服务器-客户端（Server-Client）模式就是分层模式的典型代表。

(2) 适配系统型：是一种灵活的架构型，它的优点在于可以根据应用平台协议栈的不同而自动的调整系统结构，能够适应技术和需求的不断变化。包括 Reflection mode、Micro kernel mode 等。

(3) SOA（面向服务）型：近几年大行其道的SOA 架构方法主要是以服务定义为中心，其主要应用方向就是从服务集成角度设计开发企业应用软件，把各种应用转

变为标准服务形成资源共享。典型的SOA 实现是通过合理化现有企业应用的层次来实现企业服务，即一种具有多层结构的架构。而这种层次又可分为逻辑层次与物理层次。

(4)人机互动型，它在最上层提供界面，用来与用户进行交流。包括MVC模式、PAC模式等。MVC(Model - View- Controller)是一种典型的面向模式的架构方法，MVC 架构的软件会通常包含了这样三种组件:Model、View、Controller。

本文的中间架构层的设计总体上采取现时流行的分层模式思想，从上层应用到底层的数据传输分别用不同级别的各层实现。另外，本文的架构设计也吸取了对于适配系统型、面向服务型和人机互动型的优点，体现在具体的层次设计中，后文将有详细描述。

2.1.2 应用程序架构的目标

一般而言，应用程序架构设计要达到如下的目标[18]:

(a)可重用性:为了避免重复劳动，为了降低成本，我们希望能够重用之前的代码、之前的设计。可重用性是程序员们梦寐以求追求的目标，他们已经在架构重用上做了很多的工作，工作的成果称为框架，比如说 Windows 的窗口机制、JZEE 平台等。

(b)高效性:不论是什么系统，我们都希望架构是高效的。这一点对于一些特定的系统来说尤其重要。例如嵌入式实时系统、高访问量的网站。

(c)可定制性。同样的一套软件，可以根据客户群的不同和市场需求的变化进行调整。现在很多软件的安装程序都会配备好几个安装包，安装时会让用户自己选择适合自己的版本（例如，典型安装，完全安装和简单安装）

(d)可扩展性。在新技术出现的时候，一个软件系统应当允许导入新技术，从而对现有系统进行功能和性能的扩展。这要求软件设计师要用长远的眼光来思考问题，为软件预先编制空闲的应用程序接口以准备增加以后可能会导入的新功能。

(e)可维护性。软件系统的维护包括两方面，一是排除现有的错误，二是将新的软件需求反映到现有系统中去。一个易于维护的系统可以有效地降低技术支持的花费。

(f)可靠性。软件系统对于用户的商业经营和管理来说极为重要，因此软件系统必须非常可靠。

(g)可移植性。我们希望架构放之四海而皆准，即使不能这样也能在尽量大的范围内使用，而不是仅仅局限于少数操作系统下的软件开发。

(h)安全性。架构设计作为统筹全局的抽象层次，安全问题不得不考虑。

火龙果•整理
uml.org.cn

2.1.3 应用程序架构设计的研究现状

2001年，德国科学家Sandra Haseloff认为[19]标准的三层架构模式不能为移动用户服务的发展提供一个充分的基础，这些类型应用的特征需要不同的方法去进行应用设计。在这样的形势下，有必要继续为移动设备发展特定的应用程序架构。他们的方法是通过为中间层提供一个特定的分层设计和服务，这个中间层的宿主可以是移动应用的，也可以是被重利用的组件。这篇论文为解决处理多应用和静态应用上的多信息资源提供了一个设计方法，那就是创建一个通讯服务层，他们通过一个移动文档管理的应用领域范例阐述了自己的结论。他们的通讯服务层架构模型如下[19]：

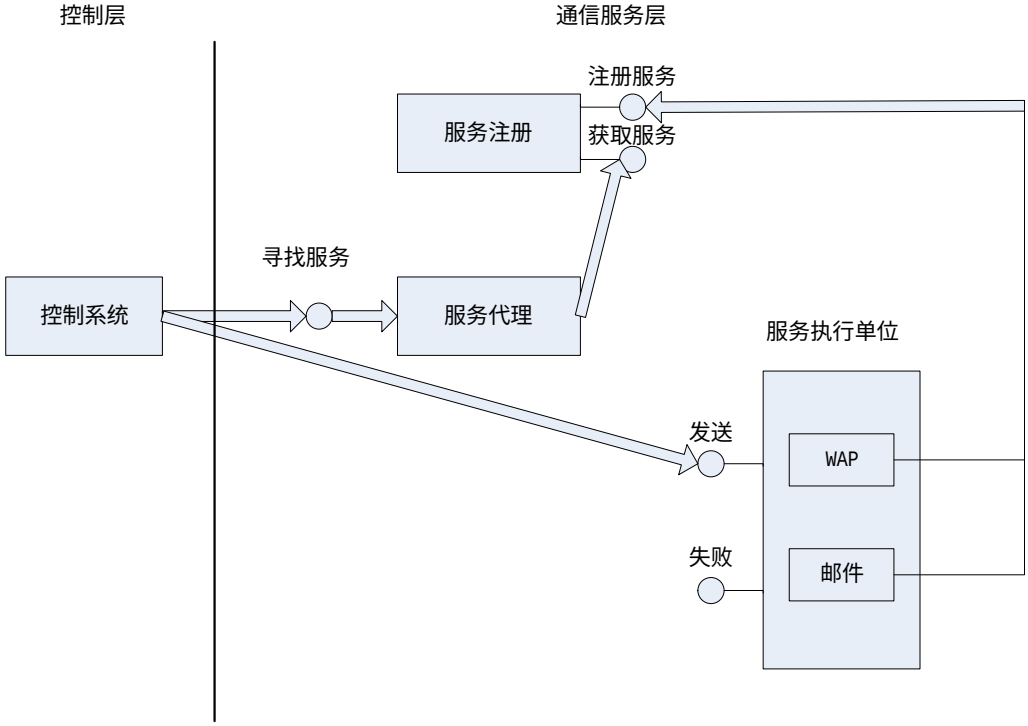

图2-1 通信服务层的架构设计

每个服务都应该有一个分布移动应用去执行这些任务：

(1) 对可用组件以及他们的服务和内容进行管理。

(2) 为用户需求和信息服务进行代理。

(3) 动态选择合适的服务。

因此，每个组件层都必须被重新定义。在这个图中，他们阐述了通信服务层的开发方法。当这个应用必须将信息传输给用户时，那么下一步就是寻找一个合适的设备去接受数据。为了做到这一点，控制子系统利用一个来自通信服务层的服务代理接口。这个控制器随后初始化信息对象传输过程。在这一点上，必须强调为一个特定通讯通道的通

火龙果•整理
uml.org.cn

信服务除了服务执行单位外，不能被其它任何组件。这个服务执行单位的作用是对设备和个人通讯组件的具体协议项目接口进行封装。从通讯层的视角来看，有必要设计一个内部不可见的服务注册。这个组件负责持续跟踪可用通讯通道的路径。它为通讯服务的服务执行单位的注册和注销提供接口。

这个开发架构是利用了一个叫做DONDE的系统。在这个应用中，通讯通道不是必须被动态选择。所以，在通讯服务之中不需要代理组件。然而，信息资源确是必不可少的。既然DONDE系统是用来为将来的服务而不是文档需求提供支持的，那么，就必然有一个负责创建合适类型服务的服务代理层。

这个在服务执行单位之上的移动文档管理对象是从基类的服务对象继承而来的。它也包含两个子服务。这个相同的管理可用文档系统和传真系统的注册机制支持一个具体的用来提供唯一接口的执行单位和一个具体的用来创建合适类型对象的代理。

2007年，北京大学的方娟教授提出了一个多层的网格入口架构[20]，她在传统的架构之上，增加了一个移动代理层。引入移动代理的目的是监视和获取网格资源信息。它可以减少网络负担，并且能够及时的获取网格资源信息。在Multi-layer Grid Protal Architecture Based on Mobile Agent这篇论文中，她分析了这个多层网格架构，提出了相应的安全策略，并且介绍了各个移动代理的功能。这篇论文给出的架构模型如图2-2[20]：

用户层：最高层是用户层，它就是通过Web浏览器与系统进行交互的终端用户。它包含运行Web浏览器的终端工作站。

入口层：在入口层他们用GPDK作为开发主要功能的工具。GPDK是一个用来开发网格用户的工具包，它提供了进入网格服务的一系列核心组件。这些组件是来自JavaBeans的。GPDK JavaBeans分为五个种类：安全，用户模型，文件传输，工作提交和信息服务。

安全层：在信息安全领域，三个核心原则是机密性、完整性和可用性。虽然入口安全涉及到许多方面，但是在这个架构里，对于不同网格实现的安全特征，主要考虑四个方面：认证、授权、审查和会话管理。

移动代理层：在这个架构里这是一个特殊的层。它在这里实际上负责获取不同动态资源和网格节点的状态。

资源层：这些在网格里的资源是按位置分配的，并且各隶属于不同的组织，各自有自己的资源管理机制。它们可以为需要计算支持的用户收取费用。

火龙果•整理
uml.org.cn

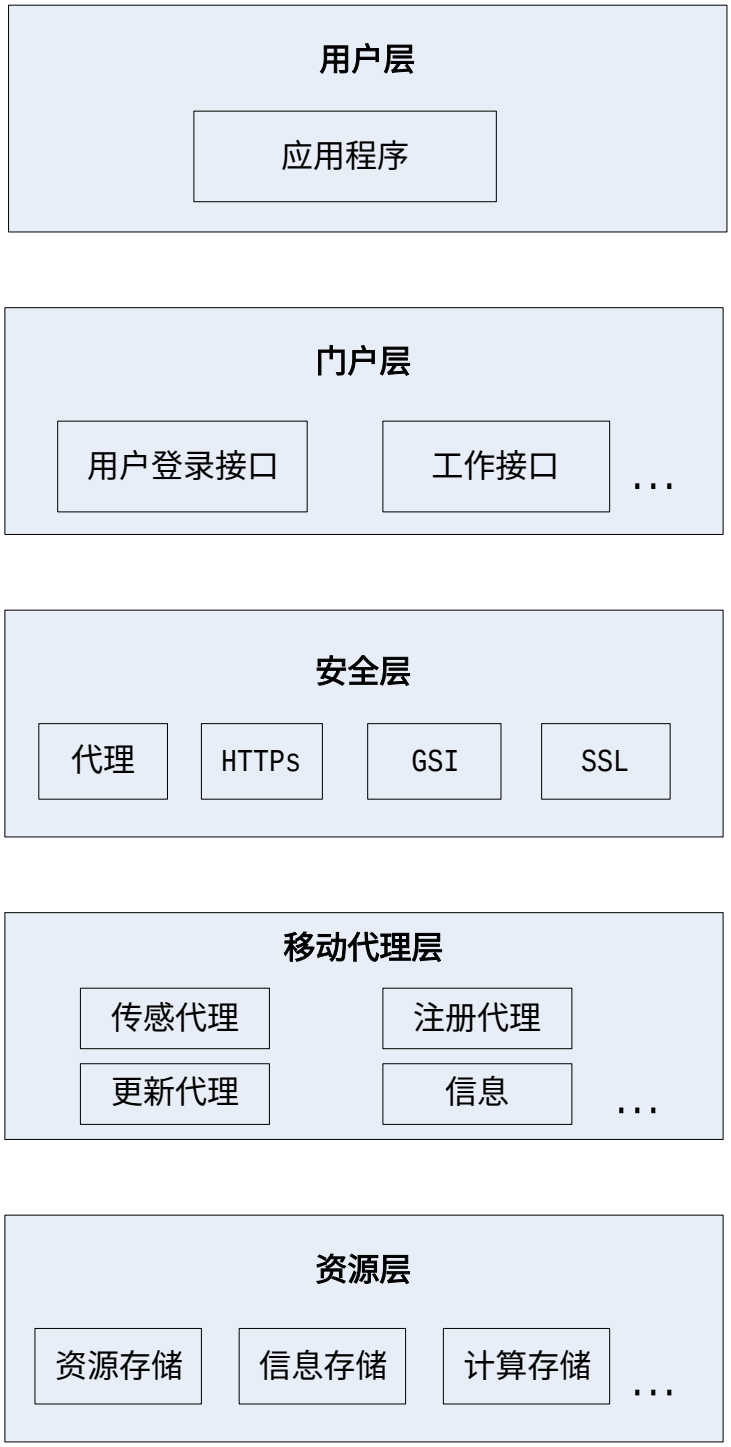

图2-2 网格入口体系的五层架构

移动代理为支持计算网格的动态服务提供一个重要的模式。一个移动代理是一个软件模块，它根据执行的具体任务，可以在网络主机之间移动。IBM的Aglet软件包被用作

火龙果•整理
uml.org.cn

移动代理的运行时环境，在运行过程中，它可以创造许多静态的和移动代理实例，随后它便可以达到网格环境的节点状态。

两种方法可以定义这些代理。第一种方法是将这些代理分成静态代理和移动代理两类。另一种方法是将这些代理按功能划分，例如信息代理、计算代理和语义代理。这些主要的代理和它们的系统功能如下：

传感代理：监视本地资源的状态信息。消息对象的传输时通过大量传输模式实现的。它没有运行正确的信息命令，但是可以调用基类的方法来返回一个结果。传感代理作为一个静态代理处在一个代理运行时环境，等待更新代理去调用自己。传感代理包括许多种，可以分别获取不同的资源状态信息。

更新代理：调用传感代理去更新节点状态信息，随后获取所有的实时数据和状态信息。它也可以作为静态代理去处在代理运行时环境从而去运行更新。

注册代理：网格节点可以用它去注册和注销。如果节点的静态信息发生改变，它可以被注册代理更新。本地节点网络信息可以通过传感代理得到，然后实现增加节点信息的工作。注销是实现删除节点信息的工作。

信息代理：对应于信息资源，这种代理可以获得资源参数和资源领域等等。

计算代理：对应于资源计算，这种代理可以监视CPU负荷，计算速度，总内存和可用内存，还有一些其他的与主机相关的参数。

存储代理：对应于资源存储，这种代理可以为大量数据存储系统测量典型的输入/输出参数，例如读/写吞吐量，访问时间等。

语义代理：对应于语义资源，这种代理的目的是在文档集里找到自然语言问题的精确答案。

移动代理可以通过位置信息移动到多网格资源。在状态时段，系统会聚集这些移动代理带来的网格资源信息，并把对应的状态信息交给入口。这个不可逆的移动代理会被收回；用户代理最终会被销毁。

该论文认为，在现有成果的基础上，将来的工作可以通过引入多层架构去进一步结合其他领域，并根据标准来评估这个系统。

2003年，太原重机研究所的Guoyou Zhang和Yinzhang Guo两位学者在传统的三层架构模式的基础上，设计出了一套适应性网格架构系统。模型如下[21]：

火龙果•整理
uml.org.cn

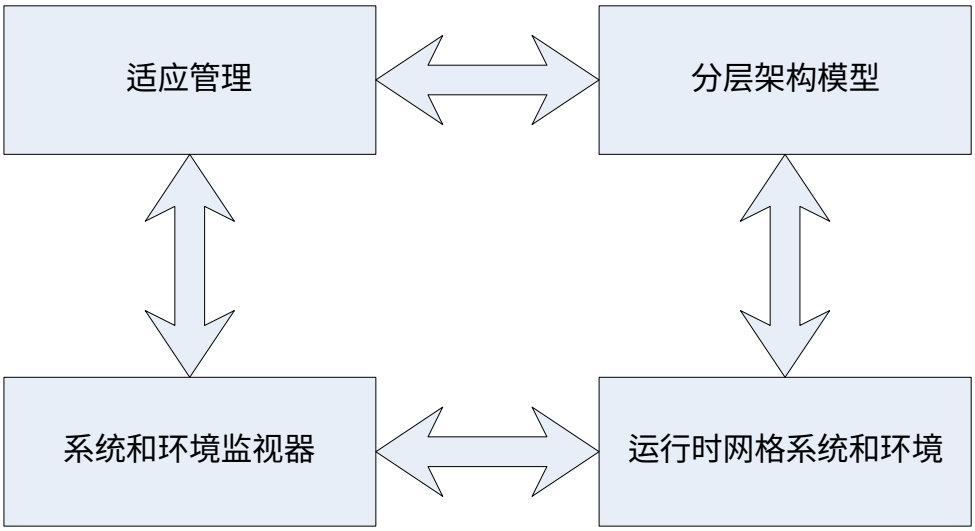

图2-3 自适应网格系统的架构模型

自适应性软件可以动态的决定和适应它的输出，并且根据交互环境停留在相应的状态。Laddaga 将自适应软件定义为："当评价显示它没有完成应该做的任务，或者可能有更好的功能性能时评价和改变它自身的行为的软件"

自适应系统可以评估它的行为，并决定系统状态的变化。对于网格计算，它为应用程序提供一个大范围的分布式资源。所以这个适应管理是网格应用程序的一个重要要求。例如，网格应用程序应该可以在运行时适应它们自己，从而处理诸如资源变化之类和系统错误的问题，并且应该可以在最小的人工干预的情况下自动适应。

软件架构将系统看作组件组合和结构中的连接件集合。它可以被用来从架构模型中构造系统。许多架构类型被归纳为管道漏斗、黑板和共享信息系统。对于适应系统，架构模型的用户可以有很多好处：

（1）一个在系统上的全局视角可以被架构建造，且系统可以决定一些全局变化，例如增加或改变组件从而达到某些属性。

（2）自适应可以在运行时动态的改变它的行为。架构模型可以阐明限制条件从而帮助确保任何变化的有效性。

（3）使用结构模型并分离系统的适应性管理，可以使适应性机构从特定系统中分离，从而实现再利用。

在这个模型中，我们将适应管理从网格应用中分离出来。这个运行时网格系统和环境块包括应用程序在操作环境下本身的集合。这个运行时和环境监视块将对适应有用的信息收集起来。这个信息是经过过滤的并且对适应管理是即时的。分层架构模型与运行时网格系统交互，并且能够区分适应和适应管理。适应管理接收从系统和环境监视器到

火龙果•整理
uml.org.cn

来的信息，并且为架构模型做决定以适应这个系统运行时行为。

为说明这个结构模型，该论文为适应性实验开发了一个叫做 GCWC 的绘图编辑器。对于适应性网格应用程序的实验主要聚焦在远程数据储存和传输带宽上。这个绘图编辑器可以从客户系统上绘制基本形状，储存到远程服务器或者其它类似系统上。这样的系统可以根据传输带宽改变它的服务器到网络上可用的服务器上，这样可使传输数据尽可能的便利和安全。在我们的实验中，网格应用程序的安全是被考虑的。用户登录到服务器可以浏览整个组，共享网格资源。

为了达到这个目标，一些额外的操作被引入。对于系统和环境的监控器，需要带宽的测量装置和滤波功能。对于分层的结构模型，有必要进行应用程序限制的分析和系统的调整。对于适应性管理，关于引导分层结构的模型的说明也应该被提示。

综上所述，笔者认为，具体来说，一个中间层架构需要具备以下六种组件[19]：

（1）分布式移动应用的基本服务。

在这一层中，有各种与分布式应用广泛相关的服务，例如：

(a)系统管理的服务。这些服务负责启动，停止和配置组件，并且具备错误事件管理功能。

(b)分布式系统的监督服务。

(c)为用户和组件进行验证和授权的安全服务。

（2）能为一个业务对象提供具体环境的信息服务。

一个信息服务组件包装一个具体环境的业务对象。它代表传递给用户的内容，同时可以包括选择合适信息的过滤机制。信息服务组件可以是网络内容，ERP 系统，CSCW 系统，企业信息网格和许多其他的内容。每个信息服务都应该有自己的接口和数据。

（3）一个能掌握应用层所有组件执行并且能优化系统的控制单位。

架构需要控制机制去协调所有组件之间的交互。这个组件管理所有应用程序的执行。更重要的是，它负责应用程序内部的信息处理的优化。

（4）一个能确定用户和可用程序的本地组件。

本地服务提供确定一个用户和当前可执行应用程序的一种方式，它还有信息传输功能。

（5）传递和接受数据的通信组件。

这个通信组件层直接与用于进行交互。通信服务负责与一个具体设备的双向数据传输，这个设备可以是无线连接或者通过局域网连接。

（6）将信息对象转化到精确形式的传输组件。

火龙果•整理
uml.org.cn

传输服务将信息对象转化为一个能让接收设备进行处理的格式。在转化过程中，传输组件必须提供服务质量参数。

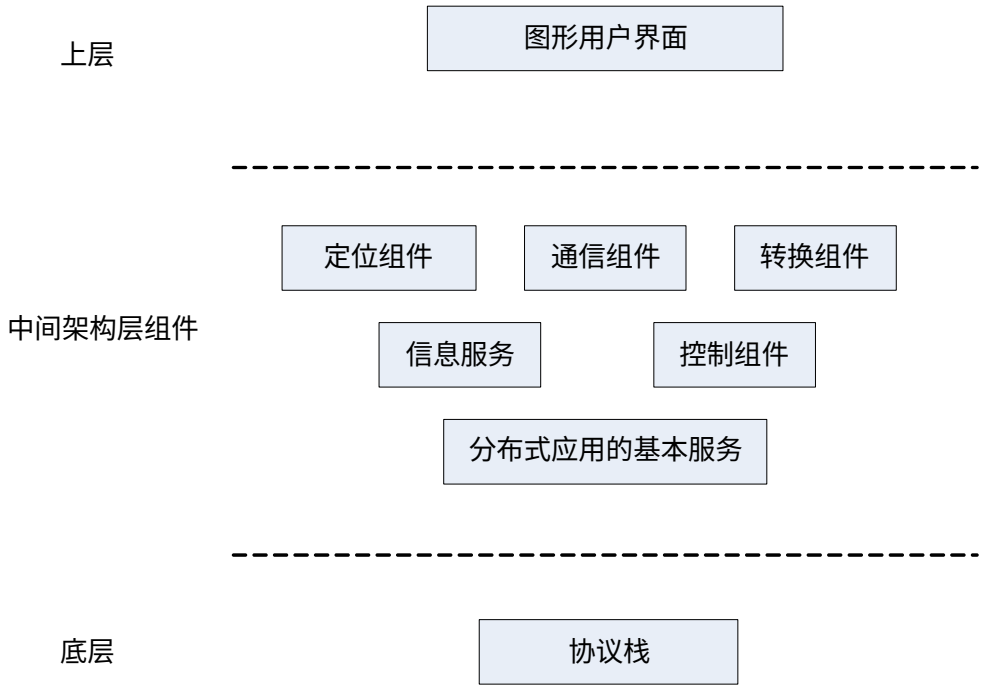

图 2-4 中间架构层组件

根据软件开发的具体情况，中间架构层的设计方法灵活多变，并没有一个统一的模型。但是，无论软件工程师如何设计他们的架构，都需要保证至少能实现上述六种组件的功能。

另外，通过对综述的分析，本文的架构层设计至少能从以下 3 个方面受到启发：

（1）服务的思想。德国科学家 Sabdra Haseloff 提出了通讯服务层的思想，中间层为宿主提供通讯服务，宿主可以是不同的应用或者组件，也就是说一个服务项可以被多个宿主调用。我们的架构设计可以采纳类似这样的方法，分出一个专门提供服务的业务层，系统根据应用需要调用相应的服务，这样做至少有两个好处：一是可以大大节省系统资源，因为一个服务项可以被多个应用调用；二是使系统对资源的管理更加条理化，因为在启动前系统就能预先知道有多少待调用的服务，可以预先为它们创建内存空间，而不是等到应用需要的时候再去动态创建。

（2）继承的思想。在 DONDE 系统中，提到移动管理对象是从基类的服务对象继承而来。本文受此启发，中间架构层的创建中，各个层次的创建思想为：首先设计一个基类，不过这个基类是一个完全的抽象类，它与 DONDE 系统的服务机制不同，

火龙果•整理
uml.org.cn

因为抽象类是不能定义对象的。具体的继承思想是：在基类中定义纯虚函数，这些函数代表着各类应用可能要用到的功能。然后开发者通过实现这些纯虚函数就可以设计出需要的派生，然后通过派生对象的创建就可以实现不同的应用功能。

（3）代理的思想。北京大学的方娟教授提出在传统的架构之上增加移动代理的思想，代理有很多种，每个移动代理都能执行一个具体的任务，整个系统就通过代理层的运转实现各种各样的功能。手机上的中间架构层不宜设计的如此复杂，但是受此启发，我们可以为每种服务设置一个代理，后面称之为业务代理对象。这样做的目的是为了解决多个应用调用同一种服务的问题，后文将有详细描述。

2.2 本章小结

本章对应用程序架构理论进行了详细的综述，首先应用程序架构的定义，应用程序设计的模式和软件架构的目标，并且较详细的分析了架构的国内外研究现状。通过这些理论的理解和分析，具体给出了软件架构的必备组件，分析了它们的功能，并给出了模型图。最后，谈到了本文的中间架构层设计所受到的理论启发。

火龙果•整理
uml.org.cn

3 应用程序架构设计

3.1 分层架构模式

3.1.1 分层模式介绍

分层是人们认识世界、解决事物复杂性和多样性的一种方法[28]。分层模式是最常见的一种架构模式，它描述的是这样一种架构设计过程：从最低级别的抽象开始，称为第 1 层。这是系统的基础。通过将第 J 层放置在第 J-1 层的上面逐步向上完成抽象阶梯，直到到达功能的最高级别，称为第 N 层。各层之间要能相互移植通用，除了接口功能一致以外，数据的通讯格式也必须是一致的，相应的还应有一组数据格式描述。

因而分层模式就可以定义为：将解决方案的组件分割到不同的层中。每一层中的组件应保持内聚性，并且应大致在同一抽象级别。每一层都与它下面的各层保持松散耦合。

分层模式的关键点在于确定依赖：即通过分层，可以限制子系统间的依赖关系，使系统以更松散的方式耦合，从而更易于维护。

典型的分层方式是[29]应用程序专用功能位于上层，跨越应用程序领域的功能位于中层，而配置环境专用功能位于低层。层的数量与组成取决于问题领域和解决方案的复杂程度。通常而言只有一个应用程序专用层。应当把子系统组织成分层结构，架构的上层是应用程序专用子系统，架构的低层是硬件和操作专用子系统，中间件层是通用服务。

3.1.2 分层原则

分层模式虽然可以很好的实现功能的划分，提高底层的复用率，但是它却把系统复杂化化了，如果分层过多会直接影响到构件软件的可靠性，而保证软件的高可靠性是分层的一个重要原则。

根据参考文献[30]的模型原理：在基于构件的软件开发中，软件系统由构件与连接件组成。基于构件的软件可以看作一个 Markov 过程。通过估算构件与连接件在整个系统中的使用次数，估算整个系统的可靠性。

例如，当某个系统由 N 个构件组成，L_i 的可靠性为 R_{L_i}，使用次数为 U_i，层连接

件 $L_{j,k}$ 的可靠性为 $R_{L_{j,k}}$，使用次数为 $V_{j,k}$，系统的可靠性为 R_s：

$$R_s = \prod_{i=1}^{N} pow(R_{L_i}, U_i) \times \prod_{j=1}^{N} \prod_{k=1}^{N} pow(R_{L_{j,k}}, V_{j,k}) \qquad (1)$$

其中 L_i (i=1,2,…N)是软件系统中 N 个不同构件，$L_{j,k}$ 表示第 L_j 个构件与第 L_k 个构件之间的连接件。公式前部分的因此表示构件的可靠性，后部分的因子表示连接件的可靠性。它把整个系统看作一个串联系统，系统的可靠性就是所有串联因子的可靠性的乘积。其模型如下：

在我们所设计的架构分层体系中，情况稍有不同，不同之处在于：分层模式中有一条可见度原则，它规定信息的传递是不管至上而下还是至下而上，都要逐层传递。各子系统只能与同一层及其相邻层的子系统存在依赖关系，而不会出现跨层访问的情况。因此，考虑到架构分层体系的这种特殊性，上述公式应稍作修改如下：

$$R_s = \prod_{i=2}^{N-1} pow(R_{L_i}, U_i) \times \prod_{i=2}^{N-1} pow(R_{L_{i,i+1}}, V_{i,i+1}) \times$$
$$\prod_{i=2}^{N-1} pow(R_{L_{i,i-1}}, V_{i,i-1}) \times pow(R_{L_1}, U_1) \times pow(R_{L_{1,2}}, V_{1,2}) \qquad (2)$$
$$\times pow(R_{L_N}, U_N) \times pow(R_{L_{N,N-1}}, V_{N,N-1})$$

其中 $V_{i,i+1}$ 与 $V_{i,i-1}$ 分别表示第 L_i 层与 L_{i+1} 和 L_{i-1} 层的交互次数。公式的前三个部分因子表示出了第一层和第二层之外的系统可靠性，因为第一层没有下层，第 N 层没有上层，所以单独计算，后面四部分的因子表示第一层和最后一层及其中间件的总体可靠性。

可靠性参数的值域是 0~1，它们作为乘数会使结果值变小。而对于一个架构而言，分层越多，不仅层的可靠性参数增多，而且连接件也越多，从而连接件的可靠性参数也越多，最终导致可靠性值变小。因此根据计算公式，整体可靠性的增强要求分层越少越好。

结论：在给架构分层的时候，如果分得过少，容易造成模块的臃肿和模块内部的

火龙果•整理
uml.org.cn

混乱；如果分的过多，又会降低整个系统的可靠性。因此，分层原则必须充分考虑这两方面的因素，在保证可靠性的前提下，对系统分而治之，从而进行有效的组织。

3.2 架构层设计结构

根据 3.1 的分层原则：条理化原则和可靠性原则，作这样的分层考虑：最上层是应用层，它对应手机的应用程序，主要负责手机应用的启动、初始化、关闭以及相关控件的调用和显示，并通过图形界面直接与用户进行交流；其次，将界面调用和业务功能管理都放在应用层的结果是界面代码与逻辑功能代码混在一起，结构不清，不易维护，因此，可以在应用层中划分出一个业务层，专门负责包装各类手机应用服务，并向上提供相关的服务给应用层调用，前面提到的业务代理对象将在这一层产生；再次，本文的应用软件是用 java 语言开发的，事实上大多数手机应用程序都是基于 java 技术的，而 java 存在的一些安全缺陷很可能被病毒制造者利用[33]，从而对手机进行攻击，因此在应用层与业务层之间设计一个安全层；最后，考虑到应用与底层协议栈的交互，可以分出一个适配层[19]，负责存储数据，与平台进行数据双向传输。

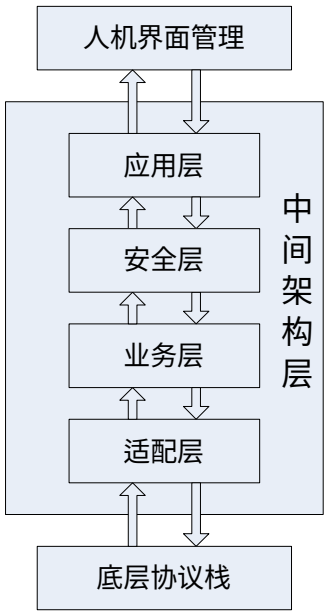

图 3-1 中间架构层模型

综上所述，将架构分为四个层次，从上到下依次为：应用层(Application_Layer)，安全层(Security_Layer)，业务层（Business_Layer）和适配层(Adapter_Layer)，这 4 个层次相对独立，这种松耦合的方式增强了架构的可维护性[29]。如图 3-1 所示。

3.2.1 应用层设计

新应用的创建思想为：首先为应用层定义一个基类——Android_Application，开发者通过继承机制可以设计出需要的派生类，然后通过派生类对象的创建就可以实现不同的应用功能。

（1）应用的注册

在 Android 系统中，除了本身自带的应用外，用户可以随意向系统注册新的应用：

```
#define  Activity_Camera   " mCam "
#define  Activity_GTalk    " mGtalk "
```

火龙果•整理
uml.org.cn


```
#define  Activity_GEarth   " mGEar "
#define  Activity_MMS      " mMMS "     /*Multi-Message Service*/
…
```

接下来只需为应用编写一个类对象，然后重新对系统编译即可。

（2）消息传递机制

在一般的操作系统中，诸如 windows，linux，系统都有不同的应用程序来处理不同用户的每个不同的操作，系统的监视程序捕捉到用户的操作，然后利用消息机制将操作以一定的形式发送给应用程序；而对应的应用程序再接受到消息后，就按照一定的机制来处理消息，如果消息队列为空，则直接处理用户操作，否则就插入等待队列，这种应用程序处理用户操作的响应过程，就称为消息驱动。

类似于一般的操作系统，手机对应的操作系统也采用消息驱动的形式[32]，不过对于一般的手机而言，其操作系统过于简单，无法完全实现消息机制，无法进行处理机的调度以及相应的时间片轮转。也就是所谓的第一代计算机系统，单系统单任务单处理机，无法实现应用程序的并行处理。既然大多数手机并不能进行 CPU 的调度和时间片的轮转，那么如果让每个应用程序都对应一个消息队列，那么在同一时间只能有一个程序能运行。这显然是不行的。

那么如何实现多个任务的同时运行呢？我们的思想是：将手机程序从“勤劳型”变成“懒惰型”，即各个程序并不时刻监视自己的消息队列，相反的，它们呈现完全的“被动”状态——等着被调用。为此，程序员需要预先为它们创造得以“偷懒”的条件——创建消息映射器，每个消息映射器都对应一个应用程序，这样，一旦发生当前消息的映射，消息映射器便可以直接触发对应的应用程序。其模型如下：

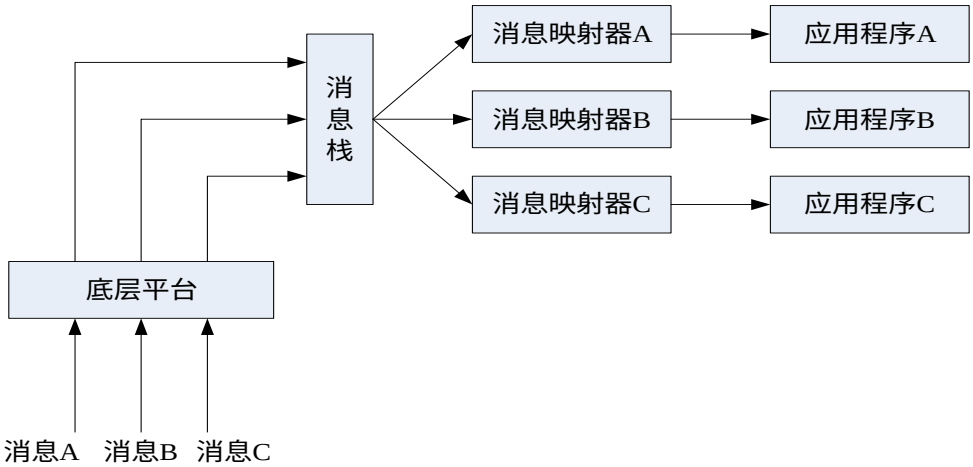

图 3-2　消息传递机制设计图

火龙果•整理
uml.org.cn

在平台之上放置一个消息队列，当消息队列接受到消息时，将消息发送到一个消息栈中，栈中按应用的注册顺序排列了多了消息映射器，消息从底层平台传到消息栈中后，就开始去找对应的消息映射器，这里采用责任链的形式，将所有的消息映射器排列成链状，到来的消息依次流过这条链时，每个消息映射器都对该消息进行识别，如果是与自己对应的消息就将其截获并进行处理。当消息映射器找到对应的消息时，就会调用相应的回调函数通知应用程序采取相应的动作。如果应用程序被删除，其对应的消息映射器也会同时被销毁。流程如下：

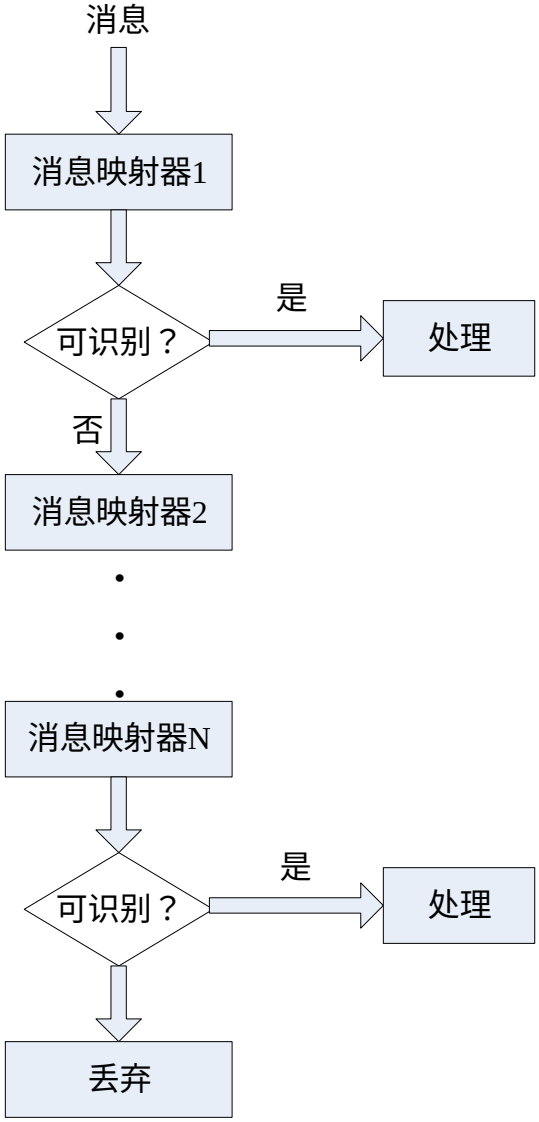

图 3-3　消息处理过程

消息链的创建方法是：采用双链表的创建思想，即将每个消息映射器作为链表的一个节点，每个节点分为三个域：数据域，指向下一个节点的指针域和指向对应应用程序的指针域。容易看出这里的消息链与双链表的区别在于将每个节点的回溯指针变成了指向应用程序的指针，因此，寻找对应的消息映射器的过程就相当于一个单链表的查找过程，可以运用相应的算法。插入和删除消息管理器只需调用单链表的插入和删除节点的算法即可，而这两种操作在单链表中都是很容易实现的。

3.2.2 安全层设计

安全层的设计思想也是先定义一个安全基类：Android_Security，然后通过继承机制设计出多个派生类，通过派生类对象的创建实现不同的安全级别。

（1）安全机制

对于手机软件来说，一般可以选择实现的安全措施有[34]：

① 创建安全管理器

安全管理器是一个 Java.lang 包中的抽象类，可以使用 GetSecurityManager() 方法返回一个应用程序当前的安全管理器。尽管不能根除病毒，但这个方法能在一个潜在的破坏性操作完成之前对其进行控制。

② 加密技术

若采用明文直接保存的方式，一旦攻击者进入系统获得用户密码，就将对系统数据安全造成无法挽回的损失。使用加密技术，用户重要的通讯数据的安全性可以得到较大保障。

③自动数据恢复技术

该技术提供了一种可靠的数据恢复方式，当主机上某数据区损坏时，采用备用数据块恢复；而当主机彻底瘫痪时，则直接用全数据块的备份对系统进行还原，与 Windows 的一键还原系统有异曲同工之妙。该技术可以解决现有技术中不能满足不中断系统业务的情况下进行数据恢复的技术难点。

④字节码校验[35]

字节码校验是在字节码层面保障应用下载的安全性。其目的是加强被下载的字节码的静态约束。这些限制保证了字节码可以被虚拟机安全的执行而且不会绕过更高安全等级的安全机制。创建字节码验证器可以确保很多异常情况的发生，具体有①不匹配的对象，②非法转换的数据，③伪造的指针，④超越限制的访问，⑤操作数的栈溢出等等，能防止 Java 运行装入有害类文件。

⑤创建类注册机制

火龙果•整理
uml.org.cn

从宏观上看，任何程序都是类的集合，将所有的类进行注册和装入后，系统就可以确定整个可执行程序的内存布局。当一个类有运行请求时，系统会首先查看“花名册”上该类的相关信息，若审核通过才予以放行，否则视为病毒类。

⑥沙箱技术

沙箱是一款专业的虚拟类软件，它通过重定向技术，把程序生成和修改的文件，定向到自身文件夹中。通过加载自身的驱动来保护底层数据，属于驱动级别的保护，在分层模型中，这个技术是保证应用层安全的一种安全机制。对于手机病毒，它并不查杀和拦截，而是在内部构造一个虚拟空间，把病毒和木马限制在这个空间内，使它们不能对真实的手机环境进行攻击。通过这样的方式使主机资源得到保障。

当然还有防范手机病毒的安全措施数不胜数，在此不一一列举。

（2）安全机制的注册

我们为每个安全机制定义一个 ID，用一个整型常量表示。

```
Security_Manager = TheFirstId；
Security_ByteCodeVeri = TheSecondId；
Security_ Encryption = TheThirdId；
Security_ClassPlugIn = TheFourthId；
Security_SandBox = TheFifthId；
…
```

3.2.3 业务层设计

和应用层一样，业务层的设计思想也是先创建一个业务基类：Android_Business，然后通过继承机制设计多个子对象来实现不同的业务。

在文献中的传统的三层模型中，将与业务层功能对应的服务层细分为服务代理层，服务注册层和服务执行层，但是在手机系统里，这样的细分优越性并不大，相反会造成资源的浪费。但是以此为参考，可以对业务层进行如下设计。

（1）业务的注册

```
#define Business_Message    “mMessage”
#define Business_Alerm      “mAlerm”
#define Business_Clock      “mClock”
…
```

手机启动时，会对所有已注册的业务进行相关的创建工作。一项业务被创建之后，

火龙果•整理
uml.org.cn

一般在初始化时会为其绑定一个适配器对象。

（2）业务代理对象

手机系统起来时，只能创建一个业务类实例，但是实际上，可能同时有多个应用程序需要使用这项业务。为解决这个问题，参考文献 20 中移动代理层的思想，可以为每个业务类实例创建多个对象作为代理，对应于不同的应用，我们称之为“业务代理对象(Business-Broker Object)”，它相当于业务的一个“线程”。如下图所示：

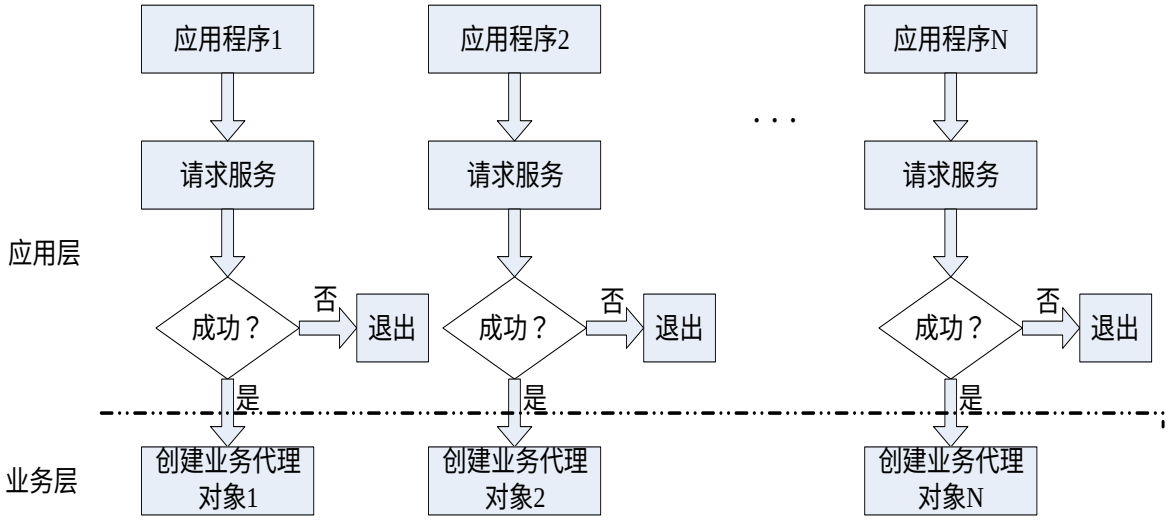

图 3-4 业务代理对象工作机理

3.2.4 适配层设计

适配层的创建原则也是继承机制：先定义适配器的基类 Android_Adapter，然后以此为父类创建多个派生类，不同的子对象对应于不同的适配器。

作为业务层的下层，适配层是直接为业务层提供服务的，这里我们将适配器类与业务类一一对应并进行绑定，即每个适配器的工作是：向上提供对应业务所需要的数据，向下将上层的数据传送给底层平台。

适配层的绑定过程是，分别获得服务和适配器的 ID，然后将它们绑定在一起。其函数形式如下：

```
Service_ID  =  getServiceID(Service_Name)；
Adapter_ID  =  getAdapterID(Adapter_Name)；
adapterBind(Adapter_ID,Service_ID)；
```

当手机启动时，适配器被创建并绑定到相应的业务上。

火龙果•整理
uml.org.cn

3.3 本章小结

本章对软件架构的模型进行了具体的结构设计。首先，介绍了分层模式的定义和一般规则，然后，给出了软件架构的可靠性估计模型。在对各种原则进行了充分权衡之后，本章给出了基于 Android 平台的软件设计的一个合理架构，从上到下依次是应用层，安全层，业务层和适配层，并具体给出了各层的实现方法。在对应用层的设计当中，本章提出了一种适合手机软件开发的消息传递机制。

火龙果•整理
uml.org.cn

4 应用程序架构实现

我们依照这个架构，进行了基于 Android 平台环境的即时聊天软件的开发工作。具体实现的是 Google Talk 手机客户端。应用程序用 Java 语言编写，在一个名叫 Dalvik 的设备上运行，Dalvik 是一个为运行在 linux 内核顶层的嵌入式应用所设计的虚拟机。

Gtalk 全名是 Google Talk，是 Google 公司于开发的一款即时通讯工具，类似于国内的 QQ，虽然功能远远不及 QQ，但也可以支持文字信息和语音聊天。Gtalk 的登录有两种方式，第一种类似于 QQ，但是这种方式需要下载 Gtalk 客户端软件；第二种方式是使用 Gmail 帐号登录 Google 邮箱，这个邮箱的功能之一可邀请朋友聊天，与客户端的聊天功能是一样的。Gmail 联系人可导入 GoogleTalk，并支持检查 Gmail 信件。GoogleTalk 使用开源的 Jabber 和 XMPP 协议提供实时扩展信息和交流，它可以通过跨平台的集成 IM 客户端来登录，实现跨平台的登录聊天，并且聊天记录和 Gmail 绑定，使用方便。

4.1 即时通讯工具及其功能

4.1.1 即时通讯工具

即时通讯工具[44]是通过即时通讯技术来实现在线聊天，交流的软件，已经有超过 20 年的历史。在这些年间，即时通讯软件从一个作为公告作用的工具逐渐成为年轻人所广泛使用的通讯工具，而到现在，它早已不是年轻或时髦的代名词，而是男女老少都能够普遍接受的事物。

在现实生活中，即时通讯工具越来越多，在中国流行的有 QQ、ICQ、网易 POPO、MSN、Jabber、百度 Hi 等。多年前 QQ 的异军突起，让国人疯狂的迷恋上了网上聊天。很多公司都开发了自己的即时聊天工具，供员工之间交流。很多商务人士在自己的名片联系方式上写上自己的 QQ 号码。还有一些员工为了避免面对而与客户交谈时有时候不成功所造成的尴尬局面等用即时通讯与他们进行交谈。

即时通讯的一个主要发展趋势就是缩小移动设备和电脑的距离[45]，让ＰＣ机上能安装的即时通讯软件更多的移植到手机上来。而从目前的态势上看，手机的功能必将越来越强大，手上电脑的普及指日可待。这个发展趋势的依据在于：桌面即时通信目

火龙果•整理
uml.org.cn

前在世界范围内已拥有好几亿用户，移动设备的持有者更是普及，若能将即时通讯系统与移动通信系统结合，必将获得更多的市场，因为用户甚至不需要有一个接受新事物的过程。

4.1.2 即时通讯工具的功能

即时聊天工具种类繁多，一般来说，它们具备以下功能：

（1）用户登录。即时聊天软件一般是个人的工具，它需要用户登录认证后方可使用。

（2）用户状态。用户可以跟据需要，设置自己的状态，如在线、忙碌或隐身等等。

（3）好友管理。好友管理最主要的功能就是添加好友和删除好友，并设置陌生人对自己的访问权限。另外，为了便于区分和查找，用户可以将与自己聊天的人群归为不同的种类，如大学同学、足球俱乐部会员等。

（4）即时聊天。这是即时通讯工具的最主要的功能，通过文字、语音甚至视频可以实现个人之间的实时通讯。

（5）聊天记录管理。一般包括查看聊天记录、删除聊天记录和查询某条聊天记录。

（6）文件传输。用户之间通过即时聊天工具发送文件或接受对方传过来的文件。

有一些即时聊天工具还进行了业务拓展，增加了许多附加功能，在此不一一列举。

本文开发的 Gtalk 聊天软件实现了即时聊天工具的一些基本功能，包括用户登录、好友管理、文字交互和聊天记录管理。

4.2 实验平台——Android 平台

Android[32]是 Google 开发的基于 Linux 平台的开源手机操作系统。它包括操作系统、用户界面和应用程序 —— 移动电话工作所需的全部软件，而且不存在任何以往阻碍移动产业创新的专有权障碍。Google 与开放手机联盟合作开发了 Android，这个联盟由包括中国移动、摩托罗拉、高通、宏达电和 T-Mobile 在内的 30 多家技术和无线应用的领军企业组成。Google 通过与运营商、设备制造商、开发商和其他有关各方结成深层次的合作伙伴关系，希望借助建立标准化、开放式的移动电话软件平台，在移动产业内形成一个开放式的生态系统。

4.2.1 Android 的特性

作为一款最新的智能手机平台，Android 有如下特性[41]：

1. 应用程序框架可重用及可复写组件组成。
2. 针对移动设备优化过的 Dalvik 虚拟机。

3. 整合浏览器，该浏览器基于开源的 WebKit 引擎开发。

4. 提供了优化过得图形系统，该系统由一个自定义的 2D 图形库；一个遵循 OpenGL ES 1.0 标准(硬件加速)的 3D 图形库组成。

5. 使用 SQLite 来实现结构化数据的存储 。

6. 媒体方面对一些通用的 audio，video，和图片格式提供支持(MPEG4，H.264，MP3，AAC，AMR，JPG，PNG，GIF) 。

7. GSM 技术(依赖硬件) 。

8. 蓝牙，EDGE，3G 和 WiFi(依赖硬件) 。

9. Camera，GPS，指南针，和加速计 (依赖硬件)。

10. 非常丰富的开发环境，包括一个设备模拟器，调适工具，内存和效率调优工具和一个 Eclipse 的插件。

4.2.2 Android 的结构

下图展示了 Android 操作系统的主要组件结构[41]。

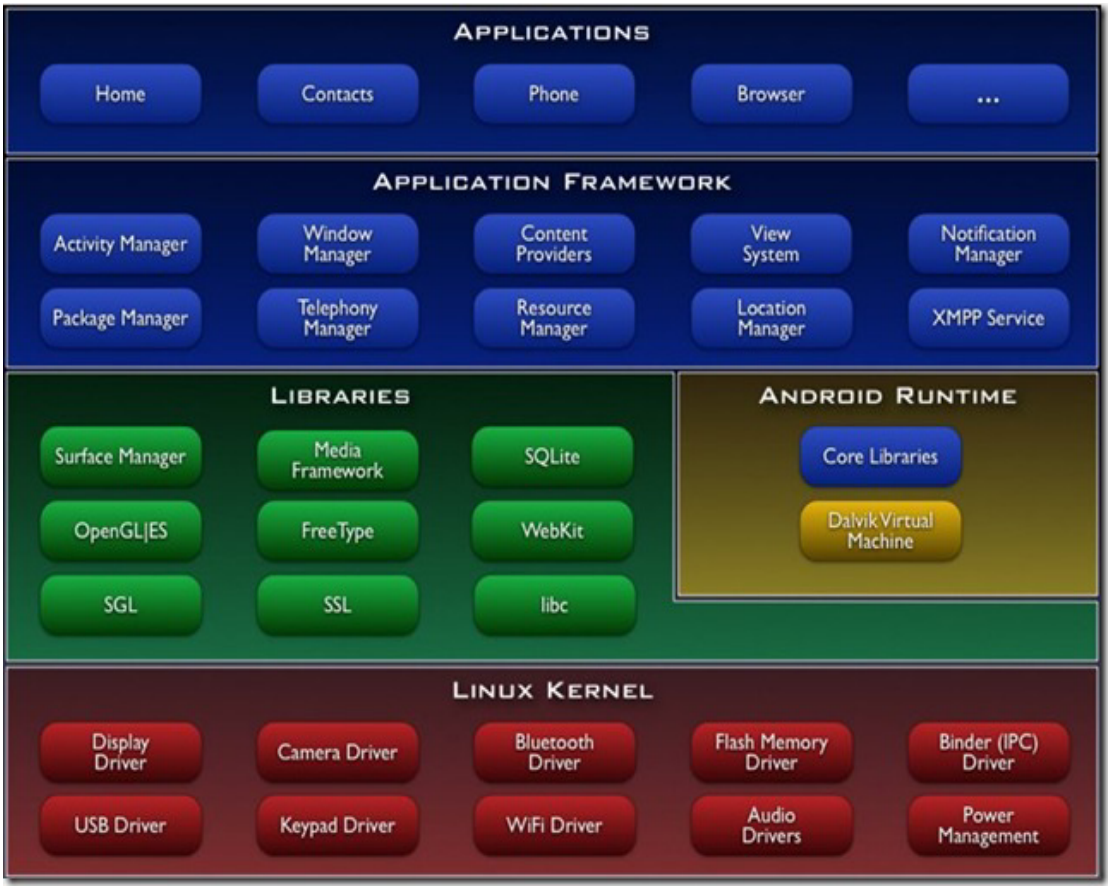

图 4-1 Android 平台的结构

火龙果•整理
uml.org.cn

（1）应用程序

Android 会附带一些核心的应用程序，包括一个 email 客户端，短信程序，日历，地图，浏览器，通讯录等等。所有的应用程序都是由 Java 语言完成的。

（2）应用程序框架

开发者对主要应用程序的相同框架接口有充分的利用权。设计应用程序架构是为了简化组件的重用；任何应用程序可以提供它自己的功能，而其它任意的应用程序都可以将这些功能引为己用（前提是在框架规定的安全范围以内）。这个相同的机制允许用户进行组件的替换。

开发人员可以跟那些核心应用一样，拥有访问框架 APIs 的全部权限．应用的系结构设计简化了各组件之间的重用；任何应用都可以分发自己的组件，任何应用也可以使用这些分发的组件（应用的使用方法需遵循框架的安全性约束）。这个机制对用户来说同样适用，他们也可以替换组件。

隐藏在每个应用后面的是一套服务和系统，其中包括：

1.一套表现丰富,扩展性很强的View，通过这些View您可以创建包括list，grid，文本框，按钮，甚至一个嵌入浏览器的应用；

2.内容提供者，通过它可以在某个应用内访问其他应用的数据(例如通讯录)，或者共享自己的数据；

3.资源管理者，提供对静态资源的访问能力，例如国际化文本和 layout 文件；

4.消息管理者，通过该机制，所有的应用都可以在状态栏显示自己的提醒信息；

5.事件管理者，用于管理应用的生命周期，并且提供了通用的向后导航功能。

（3）库

Android 包含一套 C/C++库，被用于 Android 系统中的各种组件中。这些功能通过 Android 应用框架展现给开发人员。下面列出一些核心库：

1.系统 C 库：实现了 BSD-derived 的标准 C 系统库(libc)，用于嵌入式的基于 Linux 的设备。

2.媒体库：基于 PacketVideo 的 OpenCORE；该库提供了对许多流行的音频和视频格式的播放回放和录制功能，同样包括静态图片文件，包括 MPEG4，H.264，MP3，AAC，AMR，JPG，和 PNG 等。

3.界面管理： 用于管理不同应用对图形显示子系统和 2D 和 3D 图形层的访问。

4.核心库网：一个时尚的 web 浏览器引擎，为 Android 浏览器和内嵌的 web view 提供实现。

5.SGL - 隐藏的 2D 图形引擎。

6.3D 库： 基于 OpenGL ES 1.0 APIs 实现的库；该库用于 3D 图形加速或提供高优化的 3D 软件光栅器。

7.自由类型： 位图和向量模式的字体绘制。

8.SQLite：一个强大的，轻量的关系型数据库引擎，用于所有的应用。

（4）Android 运行状态

Android 包含一组核心库，提供了 Java 语言核心库内的大部分功能。 每一个 Android 应用运行在自己的进程里，用该应用自己的 Dalvik 虚拟机实例. Dalvik 可以让一个设备高效的运行很多个 VM. Dalvik 虚拟机执行 Dalvik Executable (.dex) 格式的文件，该格式的文件经过优化 占用很小的内存. 该虚拟机是基于存储器的，运行经过 Java 语言编译器的类，这些类通过"dx"工具被转换成.dex 格式。Dalvik 虚拟机依赖 Linux kernel 来实现一些潜在功能，例如线程和底层内存管理.。

（5）Linux 内核

Android 基于 Linux version 2.6 内核来提供系统的核心服务，例如安全机制，内存管理，进程管理，网络堆栈和驱动模块. 该内核还扮演着一个虚拟的中间层，该层介于硬件和其余的软件堆栈.

4.3 应用程序开发流程

应用程序的基本架构已经搭建好，在这个架构体系下，我们为各个部分添加具体的功能就可以开发出所需要的应用。

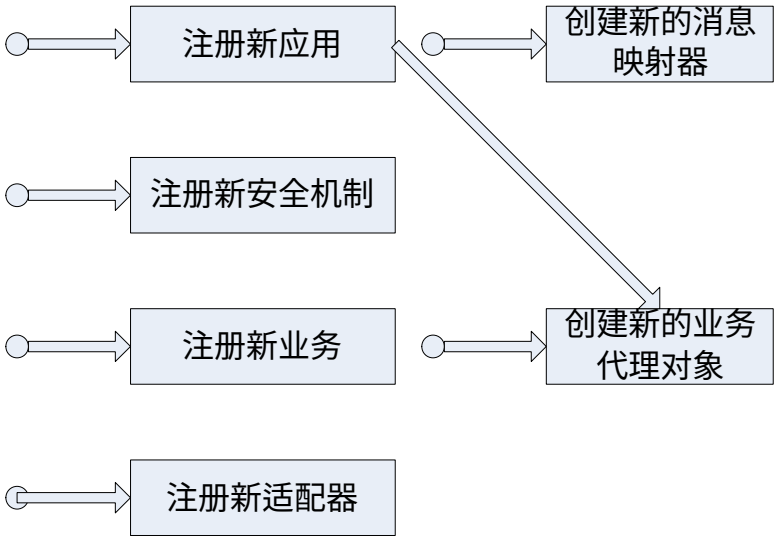

图 4-2　应用程序开发流程

4.4 应用程序功能实现

我们将 Gtalk 聊天工具划分为四个基本的模块：登录模块、好友管理模块、文字交互模块和聊天记录管理模块。

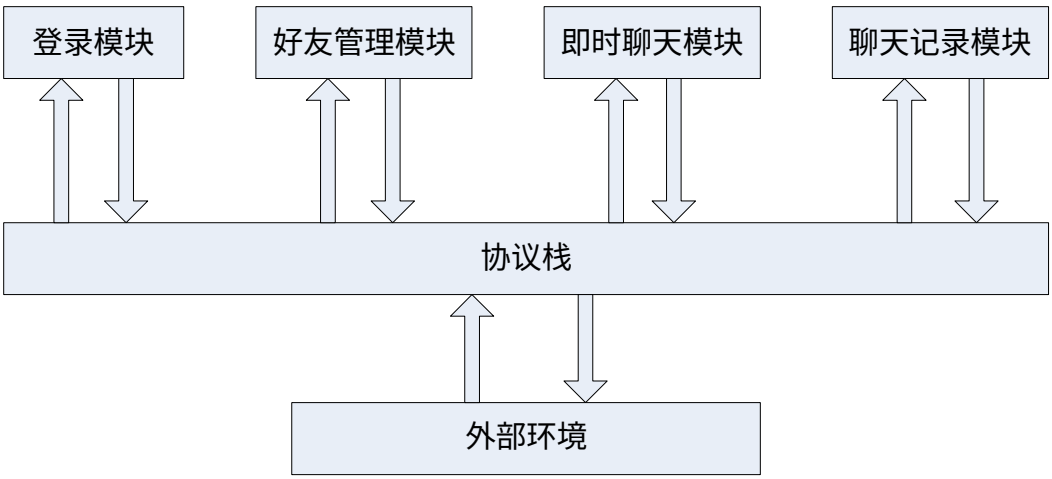

图 4-3 Gtalk 的模块结构图

基于本文设计的架构，我们在 Android 平台环境下开发出了一款即时聊天软件：Google Talk。该应用程序的功能模块实现过程和结果显示如下。

首先，根据模块功能的划分，我们将整个软件分为五个主页面，它们的功能分别是：登录界面，好友列表界面，删除和添加好友界面，即时聊天界面和聊天记录管理界面。在Android中，在Android中，每个类都是一个activity，对应一个手机界面，我们必须提前将它们都注册到一个用于显示的.XML文件中。这些类的注册如下：

```
<activity class=".FriendsList" android:label="GTalk Client">
activity class=".SubClass" android:label="GTalk Client">
 </activity>
 <activity class=".Delete" android:label="GTalk Client">
 </activity>
 <activity class=".GTalkClient" android:label="GTalk Client">
 </activity>
 <activity class=".CRecord" android:label="GTalk Client">
```

当应用程序启动时，FriendList这个类对应的应用首先被激活。

4.4.1 登录模块的实现

（1）应用层的实现

应用层的注册如下：

#define Activity_Login “Login”

应用层调用登录服务的方法如下：

bindService((new Intent()).setComponent(com.google.android.xmppService.XmppConstants.XMPP_SERVICE_COMPONENT), null, mConnection, 0)；

（2）业务层的实现

这个模块的功能就是让用户登录，因此，它需要调用系统登录服务，我们为这个服务注册如下：

#defined Business_Login “DL”

（3）适配层的实现

由于需要通过外部服务器的认证，所以这个应用必须与协议栈进行交互，适配层的绑定必不可少，绑定方法是：

Service_ID = getServiceID(XMPP_SERVICE_COMPONENT)；

Adapter_ID = getAdapterID(XMPP_ADAPTER_COMPONENT)；

adapterBind(Adapter_ID，Service_ID)；

这样，当手机启动时，负责与协议栈进行交互的适配器就被创建和绑定到登录业务上了。

Gtalk 的登录模块实例界面如下图：

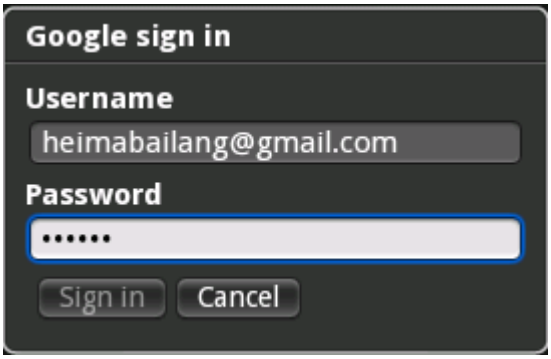

图 4-4 登录界面

用户在输入用户名（Gmail 账号）和密码，通过验证后，就可以进入聊天系统了。

4.4.2 好友管理模块的实现

（1）应用层的实现

应用层的注册为：

```
#define Activity_FriendManage “FManage”
```

实现框架为：

```
public class FriendsList extends ListActivity {
    public void onCreate(Bundle icicle) {
            super.onCreate(icicle);
        ……
        }
    ……
}
```

（2）业务层的实现

从功能上看，它需要实现添加好友和删除好友两种功能，因此，我们需要在业务层注册删除好友和添加好友两个服务：

```
#define Business_FriendAdd      “FA”
#define Business_FriendDelete “FD”
```

除此之外，这个应用还需要一个能与用户进行图像交流的界面，因此，需要注册一个界面显示服务：

```
#defined Business_FriendInterface “FI”
```

这个服务的实现方法是在一个.XML 文件中为应用程序界面中的各个控件精确定位，在此不赘述。我们将这个文件命名为 friend.xml.应用层调用该服务的方法是 setContentView 函数。

```
setContentView(R.layout.friendslist)；
```

另外，在控件方面，应用程序需要一个好友 Gmail ID 的输入框，一个添加好友的确认按钮和一个删除好友的确认方法。这些服务都需要我们在业务层进行注册。

输入框的注册方法是：

```
#define Business_Input “FriendName”
```

应用层对它的调用形式是：

```
EditText edt = (EditText)findViewById(R.id.FriendName)；
```

即先通过 findViewById 函数找到这个服务，然后将这个服务添加到应用程序的编

辑框变量中。

添加好友确认按钮在业务层的注册方法是：

#define Business_AddFriend “AddFriend”

应用层对它的调用形式是：

Button btn = (Button)findViewById(R.id.AddressFriends)；

即先通过 findViewById 函数找到这个服务，然后将这个服务添加到应用程序的按钮变量中。

删除好友服务比较复杂，方法是：应用程序界面的菜单项里调用一个服务，该服务用来确定被选中的好友名单，并将其删除。这项服务在业务层的注册方法是：

#define Business_DeleteFriend “DeleteFriend”

应用层对它的调用方法是：

```
public boolean onCreateOptionsMenu(菜单变量名){
                menu.add(删除好友项)；
                menu.add(退出项)；
        }
public boolean onOptionsItemSelected(菜单项){
                int id = item.getId()；
```

即先通过 onCreateOptionMenu 函数创建一个菜单，该菜单有两个两个项目，删除好友项和退出项。然后通过 onOptionItemSelected 函数确认需要删除的好友。

（3）适配层的实现

添加和删除好友都需要与外部的服务器进行数据交互，所以同样需要在适配层绑定一个适配器。方法是：

```
Service_ID   =   getServiceID(服务名称)；
Adapter_ID   =   getAdapterID(适配器名称)；
adapterBind(Adapter_ID，Service_ID)；
```

具体实现方法是：

```
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
                        android.R.layout.simple_list_item_1,items)；
                this.setListAdapter(adapter)；
```

即先建立一个适配器数组类型的适配器变量，它通过协议栈与外部网络进行交互，能够以列表的形式将好友显示出来，然后，通过 setListAdapter 函数，我们将这个适配器变量设置到好友管理业务的服务上。

Gtalk 的好友管理模块实例界面如下图：

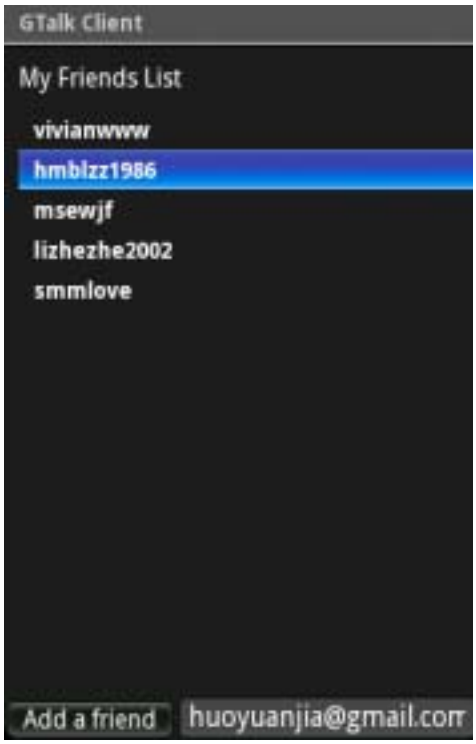

图 4-5 添加好友界面

图 4-6 删除好友界面

在右下角的编辑框里输入要添加的好友账号，点击左下角的确认按钮，就可以将好友添加到自己的好友列表里了。移动蓝色滚动条到需要删除好友的名称上，点击菜单上的确认项，就可以将好友删除了。

4.4.3 文字交互模块的实现

（1）应用层的实现

应用层的注册为：

```
#define Activity_Chatting “Chatting”
```

实现框架为：

```
public class GTalkClient extends Activity implements View.OnClickListener {
public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
      ……
        }
    ……
```

火龙果•整理
uml.org.cn

}

（2）业务层的实现

从功能上看，该模块需要实现接受和发送信息的功能，因此在业务层上设计至少四个服务项目：聊天内容输入框，聊天记录查看框，此外还有确认发送和查看聊天记录的按钮控件。

聊天内容输入框服务的注册为：

#define Business_ChatInput “ChatInput”

应用层对它的调用方法为：

EditText mSendText = (EditText) findViewById(R.id.输入框名称)；

同理，聊天记录查看框服务的注册为：

#define Business_ChatView “ChatView”

应用层对它的调用方法为：

TextView mRecipient =（TextView) findViewById(R.id.查看框名称)；

确认发送按钮控件服务的注册为：

#define Business_SendConfirm “Confirm”

应用层对它的调用方法为：

Button mSend = (Button) findViewById(R.id.按钮名称)；

查看聊天记录按钮控件服务的注册为：

#define Business_ChatView “View”

应用层对它的调用方法为：

Button mSend = (Button) findViewById(R.id.按钮名称)；

可以看出，应用层对业务层的以上四种服务的调用方法都是：通过 findViewById 方法找到需要的服务，再将该服务添加到应用程序的变量中，从而实现与业务层的交互。

此外，业务层还需要向应用层提供一个显示界面的服务，这个服务注册如下：

#define Business_ChattingInterface “CI”

具体的实现方法是在一个.XML 文件中为应用程序界面中的各个控件精确定位，在此不赘述。我们将这个文件命名为 chat.xml.

应用层调用该服务的方法是 setContentView 函数。

setContentView(R.layout.chat)；

这里出现了一个问题：跳转到这个应用后，业务层提供的界面显示服务不能总是一样的，这是因为我们打算在界面的最上面显示正在与哪个好友聊天。例如，“ 与张

火龙果•整理
uml.org.cn

三丰聊天中”或“与大傻聊天中”等。这取决于用户在好友管理界面上点击的到底是哪个好友。

解决这个问题的办法是：在上个应用的业务层中添加一项传递参数的服务，让系统记住用户点击的好友名。该项服务的注册如下：

```
#define Business_DeliverName “Dname”
```

具体实现方法为：

```
protected void onListItemClick(ListView l, View v, int position, long id){
        String str=l.getSelectedItem().toString()；
        try{
            Intent intent = new Intent()；
              intent.setClass(FriendsList.this, GTalkClient.class)；
              intent.setData(new ContentURI(str))；
              startSubActivity(intent,REQUEST_TYPE_A)；
          catch(Exception e){
        }
        }
```

该服务利用 onListItemClick()函数实现了对点击好友事件的响应。其中函数的第一个参数是 ListView 类型，它对应了系统的好友列表。首先，通过 getSelectedItem()函数从好友列表中获取当前选中的好友名字，将它的值赋给 String 类型的变量 str。然后，定义一个 Intent 变量，它在程序中的意义可以理解为意图。那么，这个意图具体是什么呢：

(a)用 setClass()函数实现界面的跳转（从好友管理界面跳转到即时聊天界面）；

(b)用 setData()函数实现好友名的传递。

注意，这里我们用到了 try{}…catch()的形式，是为了排除可能出现的异常情况，体现了安全的思想。

在即时聊天应用中，如何接收上个应用传递过来的好友名称这个数据呢？采用本文设计的架构层的思想，我们仍然是在业务层为应用层添加一个服务，这个服务的功能是接收上个应用传递过来的数据。该服务的注册为：

```
#define Business_AcceptName “AName”
```

具体实现方法是：

```
private Intent i    = getIntent()；
```

即用 getIntent()函数获取在系统中存储的上个应用传递过来的数据，将这个数

火龙果•整理
uml.org.cn

据的值赋给 Intent 型变量 i,这样,变量 i 就可以代表正在与用户聊天的好友名称了。

（3）适配层的实现

考虑到即时聊天模块需要与外界进行交互，因此需要在适配层上进行适配器的绑定，用于与底层协议栈进行数据的双向传输。适配器的绑定方法是：

Service_ID = getServiceID(服务名称)；

Adapter_ID = getAdapterID(适配器名称)；

adapterBind(Adapter_ID，Service_ID)；

这样，当手机启动时，负责与协议栈进行交互的适配器就被创建和绑定到登录业务上了。

Gtalk 的文字交互模块实例界面如下图：

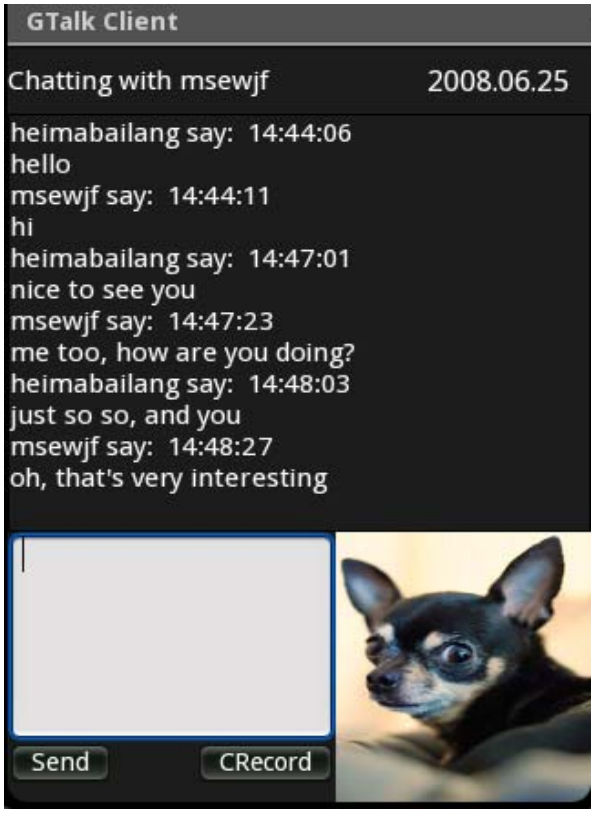

图 4-7 即时聊天模块

在上个应用的好友管理界面里，点击好友名称就可以进入这个即时聊天界面。最上面显示了与自己聊天的好友名称和聊天日期。中间的黑色框框里显示了本次的聊天内容，包括精确到秒的时间。下面的白色输入框是让用户进行输入的，当输入完毕后，点击左下角的 send 按钮，就可以将聊天内容发送出去，并马上显示到中间的黑色框

中。点击 Crecord 按钮可以查看与该好所有的聊天记录，界面会跳转到下一个应用中。

4.4.4 聊天记录模块

（1）应用层的实现

应用层的注册为：

#define Activity_ChatRecord　“CRecord”

实现框架为：

```
public class CRecord extends ListActivity{
public void onCreate(Bundle icicle) {
                super.onCreate(icicle);
              ……
                }
            ……
}
```

（2）业务层的实现

该模块有两个功能，显示聊天记录和删除聊天记录，因此，我们在业务层为它注册两个服务项目：聊天记录界面显示服务和删除聊天记录服务。

聊天记录界面显示服务的注册如下：

#define Business_RecordInterface　“RI”

具体的实现方法是在一个.XML 文件中对应用程序界面中的各个控件的位置进行描述，在此不赘述。我们将这个文件命名为 record.xml.

应用层对它的调用方法为：

setContentView(R.layout.crecord)；

与即时聊天应用类似，聊天记录应用界面不可能总是一样的，因为不同的好友就有不同的聊天记录，它同样需要区分不同的好友。如何知道应该显示与哪个好友的聊天记录呢？我们采取与前面相同的方法。即首先在即时聊天应用的业务层上添加一个服务，这个服务的作用就是记住正在与用户聊天的好友名称。该服务项的注册为：

#define Business_FriendName “FN”

具体实现方法是：

```
try{
    Intent intent=new Intent()；
```

火龙果•整理
uml.org.cn


```
                intent.setClass(GTalkClient.this, CRecord.class)；
                intent.setData(new ContentURI(i.getData().toString()))；
                startSubActivity(intent,REQUEST_TYPE_A)；
            }
            catch(Exception ex){}；
```

与前面类似，定义一个 Intent 变量，用这个变量实现我们所需要的两个功能：

(a)用 setClass()函数实现界面的跳转（从好友管理界面跳转到即时聊天界面）；

(b)用 setData()函数实现好友名的传递。注意函数里面的参数 i，这个参数的值正是从上个应用（也就是好友管理应用）中继承过来的，所以它就代表着正在与用户进行即时聊天的好友。

这里我们同样用到了 try{}…catch()的形式，是为了排除可能出现的异常情况，体现了安全的思想。

在聊天记录管理应用中，如何接收即时聊天应用传递过来的好友名称这个数据呢？我们仍然是在业务层为应用层添加一个服务，这个服务的功能是接收上个应用传递过来的数据。该服务的注册为：

```
#define Business_KnownName “KName”
```

具体实现方法是：

```
private Intent i    = getIntent()；
```

即用 getIntent()函数获取在系统中存储的上个应用传递过来的数据，将这个数据的值赋给 Intent 型变量 i，这样，变量 i 就可以代表需要显示的用户好友名称了。

删除聊天记录的算法与删除好友的算法类似。方法是：应用程序界面的菜单项里调用这个服务，该服务用来确定删除聊天记录，并将其全部删除。删除聊天记录服务的注册为：

```
#define Business_DeleteRecord “DeleteRecord”
```

应用层对它的调用方法是：

```
public boolean onCreateOptionsMenu(菜单变量名){
            menu.add(删除记录项)；
            menu.add(退出项)；
        }
```

即先通过 onCreateOptionMenu 函数创建一个菜单，该菜单有两个两个项目，删除记录项和退出项。可以看出，这项服务与删除好友服务的不同之处在于：删除聊天记录服务执行的结果是删除所有记录，并没有设置选择删除某一条记录的功能。与数字

火龙果•整理
uml.org.cn

签名功能类似，这在一定程度上能起到防止抵赖的作用。

（3）适配层的实现

显示聊天记录和删除聊天记录的都服务需要与外部的服务器进行数据交互，因此需要在适配层绑定一个适配器。方法是：

Service_ID = getServiceID(服务名称)；

Adapter_ID = getAdapterID(适配器名称)；

adapterBind(Adapter_ID，Service_ID)；

具体实现方法是：

```
ArrayList<String> results = new ArrayList<String>()；
this.setListAdapter(new ArrayAdapter<String>(this,
                                    android.R.layout.simple_list_item_1_small,
results))；
```

即先建立一个数组列表类型的适配器变量，它通过协议栈与外部网络进行交互，能够以列表的形式将与好友的所有聊天记录详细的显示出来，然后，通过 setListAdapter 函数，我们将这个适配器变量绑定到聊天记录管理的业务上。

Gtalk 的聊天记录模块实例界面如下图：

图 4-8 聊天记录模块

在上个即时聊天模块里，点击查看聊天记录按钮就可以进入到这个界面。最上

面显示了与哪个好友的聊天记录。中间的黑色框里显示了具体的聊天记录（包括聊天用户，时间和具体时间）。下面的菜单项里可以选择删除聊天记录和退出界面两种功能。

4.4.5 各层工作总结

本文在各层所做的工作总结如下：

（1）应用层

我们注册了四个应用：

登录模块：　　#define Activity_Login　“Login”

好友管理模块：　#define Activity_FriendManage “FManage”

即时聊天模块：　#define Activity_Chatting “Chatting”

聊天记录模块：　#define Activity_ChatRecord　“CRecord”

其中登录模块的实现是直接利用了系统自带的 XMPP 协议。

好友管理模块的实现框架是：

```
public class FriendsList extends ListActivity {
    public void onCreate(Bundle icicle) {
            super.onCreate(icicle);
        ……
        }
      ……
}
```

即时聊天模块的实现框架是：

```
public class GTalkClient extends Activity implements View.OnClickListener {
public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
      ……
        }
      ……
}
```

聊天记录模块的实现框架是：

```
public class CRecord extends ListActivity{
```

火龙果•整理
uml.org.cn


```
public void onCreate(Bundle icicle) {
            super.onCreate(icicle);
          ……
            }
        ……
}
```

（2）业务层

在这一层所注册的业务列举如下：

登录业务： #defined Business_Login “DL”

添加好友业务： #define Business_FriendAdd “FA”

删除好友业务： #define Business_FriendDelete “FD”

好友界面显示业务： #defined Business_FriendInterface “FI”

添加好友的输入框业务： #define Business_Input “FriendName”

添加好友的按钮业务： #define Business_AddFriend “AddFriend”

删除好友的菜单业务： #define Business_DeleteFriend “DeleteFriend”

聊天内容输入框业务： #define Business_ChatInput “ChatInput”

聊天记录显示框业务： #define Business_ChatView “ChatView”

确认发送按钮业务： #define Business_SendConfirm “Confirm”

查看聊天记录按钮业务： #define Business_ChatView “View”

聊天界面显示业务： #define Business_ChattingInterface “CI”

传递好友名字业务一： #define Business_DeliverName “Dname”

接收好友名字业务一： #define Business_AcceptName “AName”

聊天记录界面显示业务： #define Business_RecordInterface “RI”

传递好友名字业务二： #define Business_KnownName “KName”

接收好友名字业务二： #define Business_KnownName “KName”

删除聊天记录业务： #define Business_DeleteRecord “DeleteRecord”

（3）适配层

登录模块：

Service_ID = getServiceID(XMPP_SERVICE_COMPONENT)；

Adapter_ID = getAdapterID(XMPP_ADAPTER_COMPONENT)；

adapterBind(Adapter_ID，Service_ID)；

好友管理模块：

火龙果•整理
uml.org.cn

Service_ID = getServiceID(AddFriend or DeleteFriend)；
Adapter_ID = getAdapterID(适配器名称)；
adapterBind(Adapter_ID，Service_ID)；
具体实现方法是：
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
android.R.layout.simple_list_item_1,items)；
this.setListAdapter(adapter)；
即时聊天模块：
Service_ID = getServiceID(Confirm)；
Adapter_ID = getAdapterID(适配器名称)；
adapterBind(Adapter_ID，Service_ID)；
聊天记录模块：
Service_ID = getServiceID(DeleteRecord)；
Adapter_ID = getAdapterID(适配器名称)；
adapterBind(Adapter_ID，Service_ID)；

4.5 本章小结

本章是架构设计的具体实现，实验平台是 Android，编程语言是 Java。首先介绍了即时聊天的现状和基本功能。接着，根据前面章节对架构层的研究和分析，设计出了软件的具体开发流程和模块划分，最后基于本文设计的架构，在 Android 平台成功的开发出了 Gtalk 这款即时聊天软件。

Android 是基于 Linux 内核的一个手机操作系统。它是针对移动设备优化过的 Dalvik 虚拟机,并且提供了优化过得图形系统，该系统由一个自定义的 2D 图形库；一个遵循 OpenGL ES 1.0 标准(硬件加速)的 3D 图形库组成。 另外，它为开发者提供了非常丰富的开发环境，包括一个设备模拟器，调适工具，内存和效率调优工具和一个 Eclipse 的插件。Android 的结构除了底层的 Linux 内核外，还包括一套 C/C++库，Android 运行时，应用程序框架和上层应用程序。

Gtalk 这款即时聊天软件分为四个主要功能模块：登录模块，好友管理模块，即时聊天模块和聊天记录管理模块。在这些应用的开发过程中，本文充分体现了分层架构的思想，即将每个应用都分为应用层，安全层，服务层和适配层，各层各司其职。架构搭好后，我们只需要往各层添加需要的内容即可。应用层主要负责界面显示和与

火龙果•整理
uml.org.cn

用户的交流，它的功能主要是靠调用业务层的服务实现的；安全层负责系统的安全，保证应用程序的正确运行；业务层则根据应用层的功能需要，生成各种各样的服务项供应用层调用；适配层则专门负责与协议栈的数据交换，从而实现用户与外部网络的联系。

另外，在后三个模块的应用开发过程中，根据后来应用的需求，我们两次对前面已经完成的应用架构进行修改（添加业务层的服务）。这并不代表我们的架构设计理念是错误的，相反的，这正是体现了架构的迭代设计思想。在迭代设计中，每一次对架构的修改，就使架构更趋于成熟和稳定。

火龙果•整理
uml.org.cn

5 结论和展望

5.1 主要结论

随着 3G 时代的到来，手机软件的复杂度越来越高，传统的面向对象的软件开发的思想已经难以满足产品更新换代的速度需求。针对这个问题，本文提出了一种可以缩短软件开发周期，提高产品质量的方法，那就是在平台之上建立一个中间架构层，让开发者能够在这个架构之下实现各个软件的功能，从而让开发者抛开对应用结构设计的考虑，专注于各应用具体功能的设计与实现，上从而大大减少开发项目的时间，缩短应用软件的开发效率。这个中间架构层分为四个层次，从上到下依次是应用层、安全层、业务层和适配层。应用这个架构，我们在 Android 平台上，成功的开发出 GTalk 这款即时聊天软件，证明了该架构在 Android 平台上的成功性。

在中间架构层的设计过程中，本文给出了一种估计架构可靠性的方法，并给出了计算公式。在应用层的设计中，本文给出了一种适合手机软件开发的消息传递机制：将应用程序的主动形式改为被动形式，可以大大节省系统开销。

5.2 展望

本文设计的中间架构层虽然在应用上获得了成功，但是，由于自身局限和外部环境的原因，仍然有待改进的地方，主要表现在以下几个方面：

（1）安全机制并没有得到彻底贯彻。在架构的应用层下面是安全层，作这样设计的初衷是计划在这个层面上设计种种安全机制以预防和消除未来不可预知的手机病毒或木马，然而事实上，本文在这个层面上所做的安全机制不尽完善。之所以这样，有以下几个原因：

a.Android 平台已经自带完善的安全机制，它自身就能处理各种各样的安全问题；

b.手机系统资源有限，不可能像 PC 机上一样，编写复杂的算法以确保系统安全。这也是本文忽略安全机制设计的最主要的原因；

c.安全问题的处理并不是本文设计的重点。尽管如此，在应用的开发代码中，已多次体现了安全的思想。

（2）架构层设计的目标之一就是减少系统之间的耦合性，保持松耦合状态的系统

火龙果•整理
uml.org.cn

更容易维护。本文设计的架构层基本上达到了这样的目标，各个模块之间基本上可以相互独立的进行开发，但是，在第五章的开发过程描述中，已两次出现了这样的情况：在下一个应用开发过程中，需要依赖上一个应用的服务，从而不得不重新为上一个应用进行添加服务。虽然基于架构层，这样的修改并不复杂（只需要在业务层添加一个服务项即可，并不会对原有的代码块产生影响），但是这仍然表明，如果能减少或者消除这样的依赖关系，那么本文的架构设计可以变得更完美。

（3）现在手机平台种类繁多，不同的平台可能拥有不同的协议规范，在这样的情况下，对平台具有严重依赖性的适配层就必须重新改写。 因此，本文设计的架构层虽然在 Android 平台下获得了成功，但是，它不能适用于所有的手机平台。但是，不管怎样，这样的设计理念是可以为所有的手机平台软件开发作为参考的。

（4）消息机制未能在架构设计中表现。在应用层的设计中，本文设计了一个适合手机消息传递的机制，它的设计初衷是为了解决某些手机操作系统不能进行 CPU 的调度和时间片轮转的问题。但是本文采用的 Android 手机操作系统本身能够支持多线程的轮转，已经具备良好的消息传递机制，所以本文直接利用平台本身的消息机制就可以。尽管如此，本文设计的消息传递机制在其它的手机平台上可能有借鉴的价值。

针对以上几点，今后待改进的地方总结如下：

（1）根据不同应用的重要性级别，在各个安全层上设计出不同的安全机制，然而，它们的算法不宜过于复杂。

（2）深入理解系统结构，划分耦合性更为松散的模块结构，让架构层的应用效率更为显著。

（3）建立统一的平台协议规范，在这个基础上，建立适用于所有手机平台的架构层，软件设计师更容易进行相互交流与合作。

火龙果•整理
uml.org.cn

致 谢

在攻读硕士学位和论文的写作期间，我得到了导师黄金国副教授的亲切关怀，教诲和鼓励。两年期间，黄金国老师对我的工作和研究情况甚是关心。在项目方面，黄老师时常鼓励我参加公司的项目开发，培养自己的动手能力和创新能力，并多次给我这样的参与机会，让我受益匪浅，并最终成功完成了 Android 平台上的 Gtalk 开发。在论文的撰写方面，黄老师为我提供了广阔的思路，让我收获良多。在生活方面，黄老师是我们的朋友，经常与我们亲切聊天，教我们为人处事的道理。黄老师严谨的治学态度，渊博的学识和孜孜不倦的教诲，让我学会了如何做学问，也学会了如何为人。我要衷心的感谢黄老师在学习和生活上给我的帮助和支持！

我还要感谢华中科技大学信息与系统技术研究所的张新访教授、向文教授、王同阳副教授、朱建新副教授、刘怀兰副教授、张翔立副教授、周培源副教授、艾南平老师等。感谢这些老师对我在学习上的指导和科研上的支持。

感谢王俊峰、覃明彬、万旭阳、周方圆、郭建平等实验室的同学在学习和生活上给我的支持和帮助，我们大家有缘聚在一起，互相帮助，共同进步。

最后，我要感谢我的父母，是你们给了我生命，是你们永远无条件的支持我，感谢你们的养育和关怀。

火龙果•整理
uml.org.cn

参考文献

[1] 杨剑.分部解构智能手机发展趋势[J].电子与电脑，2005.10: 21-23
[2] 朱清峰，葛文海.手机技术现状及发展概述[J].邮电设计技术，2005.09: 37-41
[3] 杨菠.智能手机开发平台与操作系统简介[J].通信世界网，2003.11: 24-27
[4] 陈榕.关于 3G 手机移动软件运行平台的思考[J].和欣技术白皮书，2005.01: 59-62
[5] 段涛.基于 ARM9 和嵌入式和 Linux 智能手机开发平台的研究[D].武汉：武汉理工大学自动化学院，2007.05
[6] 胡玉同.基于构件技术的智能手机软件开发模型的研究[D].上海：上海师范大学数理信息学院，2005.04.30
[7] 陈翌.嵌入式软件开发技术[M].北京:国防工业出版社，2003
[8] 刘乾发.基于 Ajar 平台的手机软件架构的研究与应用[D].浙江：浙江大学软件学院，2007.5
[9] 胡倩，杨子华，曹燚.智能手机软件平台的研究与开发[J].通信技术. 2007.11: 30-36
[10] vietor shterm.软件工程方法[M].北京:机械工业出版社，2002
[11] 董佩嘉.基于 Symbian 0S 的智能手机应用软件开发[D].四川：成都理工大学计算机学院，2007.05
[12] 张磊，廖建新，陈俊亮.移动智能网多媒体消息业务的体系结构及其协议[N].通信学报，Vol.25.No.2004
[13] 奚焜,袁军,李树广.基于 Symbian 平台应用的开发模式[J].计算机工程，2003.9,34-36
[14] 郭学里，张健，等.网络程序设计[M」.武汉:武汉大学出版社，2004
[15] 马智强.移动平台上多媒体信息服务的设计与实现[N].北京理工大学校报，2003.02.01, 35-38
[16] 左爱群，黄水松.软件体系结构的发展与研究[J].计算机工程与应用.1999，3: 25-29
[17] NICK JM, FOSTER I. Grid services for distributed system integration[J]. IEEE Computer, 2002,35(6):37-46
[18] 姜滨.深入理解软件架构[J].信息技术.2007，36(4): 7-8

火龙果•整理
uml.org.cn

[19] Sabdra Haseloff. Designing Adaptive Mobile Applications[J]. Fraunhofer Institute for Software and Systems Engineering ISST.2001:131-138

[20] Fang Juan.Multi-layer Grid Protal Architecture Based on Mobile Agent[D]. College of Computer Science, Beijing University of Technology. 2007

[21] Guoyou Zhang,Yinzhang Guo.Research on the Architecture-based Adaptive Grid Application[J]. D-CABES 2004 PROCEEDINGS,2004，48-50

[22] Roy Thomas Fielding.架构风格与基于网络的软件架构设计[D]. 李锟、廖志刚、刘丹、杨光，译.加州大学欧文分校博士论文，2000

[23] ZHOU Yu,MA Xiaoxing,TAO Xianping,LU Jian. Towards a Component Framework for Architecture-Based Self-Adaptive Applications[J]. Wuhan University Journal of Natural Sciences. Vol.11,No.5,2006,1227-1232

[24] 李洋.手机软件技术剖析[N].中国计算机报，2003.06(2)

[25] Tatsuro Masamura. Towards 4th Generation Mobile Communication[R]. The International Forum on Future Mobile Telecommunication & China-EU Post Conference on Beyond 3G Beijing，2002，101-110

[26] 王学龙.嵌入式 Linux 系统设计与应用.清华大学出版社，2001，24-30

[27] 陈榕.中间件技术在嵌入式操作系统中的应用[J].计算机应用，2001，19(2):41-47

[28] 莫东松，彭晓东，夏克俭.面向对象的分层应用程序的三种结构模式[J].计算机应用研究，2003(2):34-35

[29] 章远阳，杨芙清，邵维忠.面向对象系统软件的分层构造模型[N].软件学报，1993，4（5）：1-5

[30] 周娜琴，张友生.构件软件的可靠性估算模型[J].计算机应用，2008，28（6）：1630-1634

[31] Chen Chong. Design and Implementation of Mobile Software Agent System[D]. Computer Seience Department, Beijing University of Aeronautics and Astronautics，1999

[32] 林建民.嵌入式操作系统技术发展趋势[J].计算机工程，2001.10，27(10)

[33] 黎筱彦，曲保章.Java 的安全缺陷[J].现代计算机，1999,（5）：79-80

[34] 郭前进.手机病毒分析及智能手机杀毒软件设计[J].河北：河北工业大学计算机学院，2007.02

[35] Xavier Leroy. Bytecode verification on Java Smart card[J]. Software

火龙果•整理
uml.org.cn

Practice and Experience, 2002, Vol.32, No.4: 319-340

[36] 郭春雷.基于 Jabber 协议的手机即时通讯系统的研究与实现[D].贵州：贵州大学计算机软件学院，2007.05

[37] 宁勇.嵌入式操作系统综述[N].邵阳学院学报。2004.06, 1(2)

[38] 张严岐.即时消息传递软件服务器模块的设计[J].计算机与应用，2004.02: 36-40

[39] D.Gannon, J.Alameda. Building Grid Portal Application from a Web Service Component Architecture[EB/OL]. http://www.extreme.indiana.edu/~aslom/papers/ PortalsInProceeding IEEE 2005.pdf.2005

[40] 过晓华.基于 Symbian 平台的 BlueTooth 网络聊天系统的研究[D].四川：重庆大学软件工程学院，2006.10

[41] Van Schyndel R.Tirkel A，Osborne c.A digital watemrarking IEEE Proeeeding[R].International conference on Image Processing，Austin，Tex. IEEE Press，1994:86-90

[42] Sang ho Kim, Choon Seong Leem. Security Threats and Their Countermeasures of Mobile Portable Computing Devices in Ubiquitous Computing Environments[J]. ICCSA 2005, LNCS 3483, 2005:79-85

[43] Janne Sinkkonen, Sanuel Kaski Clustering Based on Conditional Distributions in an Auxiliary Space[J]. Neural Computation, 2001(14):217-239

[44] 周骏，徐林，李征.基于 Agent 的 Jabber 即时消息系统研究及其实现[J].计算机工程与设计，2005.06: 31-37

[45] 2008 年即时通讯领域呈现 6 大发展趋势[N].Market Trends，2008，Page 34

[46] Android document. Released by google company

[47] Balzer,R. Living in the Next-Generation Operating System[J]. IEEE Software, 1987.11:77-85

[48] Steve R.White. Open Problems in Computer Virus Research[J]. Virus Bulletin Conference, 2004(22):101-105

Android平台环境下应用程序架构的设计与实现

作者：罗震
学位授予单位：华中科技大学

火龙果•整理
uml.org.cn

本文链接：http://d.g.wanfangdata.com.cn/Thesis_D084981.aspx