

# 1.引言

## 1.1 目的及范围

该架构文档对 BBS 论坛整体结构进行描述，本论坛旨在建成一个自由开放的论坛，用于计算机行业专业人士技术交流，用户注册即可登陆使用。论坛界面友好，简单易用，易于维护且具有良好的可扩展性，非常适于中、小型论坛系统。论坛预计运用 Struts2 和 Hibernate 两大框架技术的整合进行系统的开发，采用 B/S 体系结构，采用免费、小巧、易用的 MySQL 作为数据库服务器，Tomcat 作为服务器。

## 1.2 文档结构

文档的组织结构如下：

### 第一部分 引言

本部分主要概述了文档内容组织结构，使读者能够对文档内容进行整体了解，并快速找到自己感兴趣的内容。同时，也向读者提供了架构交流所采用的视图信息。

### 第二部分 架构背景

本部分主要介绍了软件架构的背景，向读者提供系统概览，建立开发的相关上下文和目标。分析架构所考虑的约束和影响，并介绍了架构中所使用的主要设计方法，包括架构评估和验证等。

### 第三、四部分 视图及其之间的关系

视图描述了架构元素及其之间的关系，表达了视图的关注点、一种或多种结构。

### 第五部分 需求与架构之间的映射

描述系统功能和质量属性需求与架构之间的映射关系。

### 第六部分 附录

提供了架构元素的索引，同时包括了术语表、缩略语表。

## 1.3 视图编档说明

所有的架构视图都按照标准视图模板中的同一种结构进行编档。

# 2.架构背景

## 2.1 系统概述

B/S 结构(Browser/Server，浏览器/服务器模式)是 WEB 兴起后的一种网络结构模式，Browser 指的是 Web 浏览器，极少数事务逻辑在前端实现，但主要事务逻辑在服务器端实现。B/S 架构的系统无须特别安装，只有 Web 浏览器即可，因此十分适合当前的 Web 开发，在这种结构下，用户工作界面是通过 WWW 浏览器来实现，极少部分事务逻辑在前端（Browser）实现，但是主要事务逻辑在服务器端（Server）实现，形成所谓三层结构。

分层模式用于构建可分解为多组子任务的程序，每个子任务都在某个抽象层，每个层对上一个更高层提供服务。分层风格有效支持了软件设计的逐级抽象，具有较好的可扩展性，支持软件复用，但也要注意降低层次之间的耦合程度。

模型-视图-控制器模式，该模式也叫 MVC 模式，划分交互程序为 3 个部分：模型——包含核心功能和数据，视图——显示信息给用户（多个视图可被定义），控制器——处理用户输入。它通过分割用户信息的内部陈述和呈现、接受方式来实现，解耦组件并允许高效的代码复用。

## 2.2 架构需求

### 2.2.1 技术环境需求

客户端与数据库之间加入了一个中间层，也叫组件层。B/S 三层体系是应用程序将业务规则、数据访问、合法性校验等工作放到了中间层进行处理。通常情况下，客户端不直接与数据库进行交互。

基于 B/S 架构的工程需要我们掌握编写 Web Server 服务器的能力，并且具有一台可以当做 Web Server 服务器的设备，服务器也可以设立在云上，方便维护与更新。系统在开发过程中有如下设计约束：开发语言为 Java，采用关系型数据库存放数据，采用基于 UML 的面向对象分析与设计方法进行开发，采用 B/S 架构。

### 2.2.2 功能需求

对于 BBS 论坛系统来说，最主要的功能就是用户浏览帖子、发布帖子以及回复帖子。同时考虑用户所发布帖子的内容需要健康且合法，这就需要管理员加强对帖子的审核。如果用户想要在论坛发布帖子，前提是需要注册且处于登录状态。此外，为了便于从内容上对帖子加以区分，方便管理员对帖子的管理，还需要将论坛分为多个版块。还需要将用户划分为不同级别，根据用户级别的不同在论坛中为用户分配不同的权限，以进一步加强对论坛的管理。

综上，论坛既要实现对于帖子的管理，又要实现对于用户的管理，还要兼顾到各个功能模块之间的协调合作，共同完成 BBS 论坛系统的业务需求。下面采用 UML 用例模型对系统功能需求进行分析。

构建 UML 用例模型的第一步是确定模型中的使用者有哪些，确定使用者的原则有：谁是系统的维护者、谁是系统的参与者等。一般维护者处于系统内部，对系统有绝对的控制权；而参与者一般都位于系统的外部，处于系统的控制之外。

经过对 BBS 论坛系统的分析，可以确定本系统用例模型有二种，用户（用户分为游客和注册用户）和系统管理员。

### 2.2.3 质量属性需求

- ◆ 系统应支持 100 人以上同时访问服务器并支持 500 人以上同时访问数据库，服务器的响应时间不应该超过 2 秒。
- ◆ 所有用户在保证网络连接的情况下可同时通过局域网和互联网访问系统。
- ◆ 系统必须保证数据的安全访问，用户需要通过用户名和密码进行身份认证，同时对数据的访问要进行授权认证。

## 2.3 主要设计决策及原理

本系统采用 SSH 框架开发，系统架构主要分为四层：表示层、业务逻辑层、数据持

久层和域模块层(实体层)。

在表示层中，首先通过 JSP 页面实现交互界面，负责传送请求(Request)和接收响应(Response)，然后 Struts 根据配置文件(struts-config.xml)将 ActionServlet 接收到的 Request 委派给相应的 Action 处理。

在业务逻辑层中，管理服务组件的 Spring IoC 容器负责向 Action 提供业务模型(Model)组件和该组件的协作对象数据处理(DAO)组件完成业务逻辑，并提供事务处理、缓冲池等容器组件以提升系统性能和保证数据的完整性。

在持久层中，则依赖于 Hibernate 的对象化映射和数据库交互，处理 DAO 组件请求的数据，并返回处理结果。

采用上述开发模型，不仅实现了视图、控制器与模型的彻底分离，而且还实现了业务逻辑层与持久层的分离。这样无论前端如何变化，模型层只需很少的改动，并且数据库的变化也不会对前端有所影响，大大提高了系统的可复用性。下图为系统架构图：

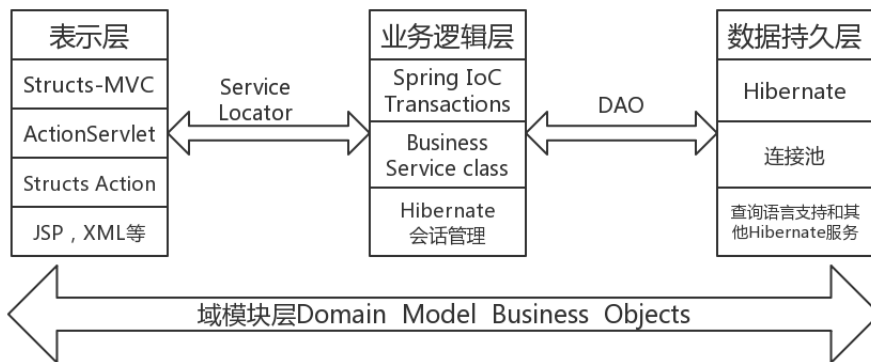


图 1 SSH 框架图

### 3.视图

#### 3.1 逻辑视图

##### 3.1.1 顶层逻辑视图

##### 3.1.1.1 主表示

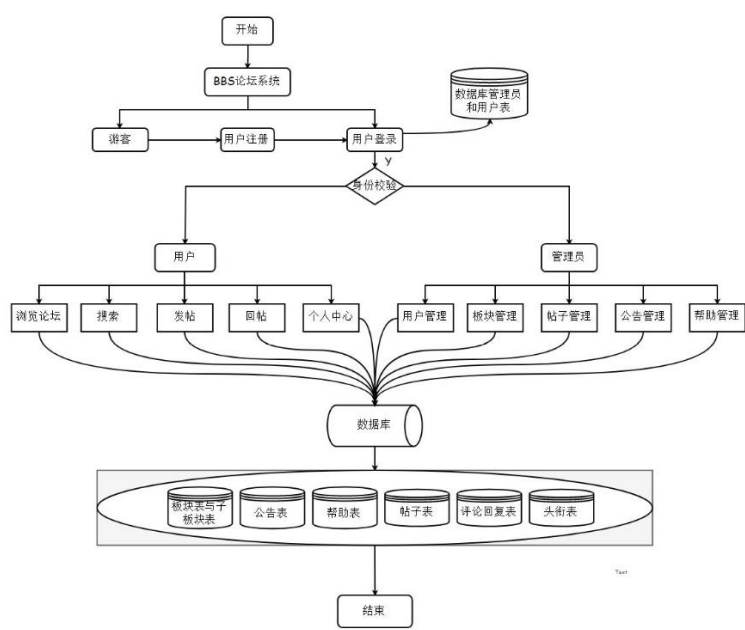


图 2 模型主表示图

3.1.1.2 构件目录

A.构件及其特性

构件	描述
用户	需要包含 ID、用户名、密码、邮箱等信息注册成为论坛用户；可以使用论坛系统中自己拥有权限的那部分功能如发帖、回帖等。
管理员	需要包含 ID、用户名、密码，可以进行用户管理、版块管理、帖子管理、公告管理和帮助管理。
搜索	用户可以输入关键词进行搜索帖子。
帖子	需要包含帖子 ID、主题和一定的描述信息，需要进行增删改查四种操作
公告	包含主题和公告信息，需要进行增删改查四种操作
帮助	包含主题和帮助信息，需要进行增删改查四种操作
板块	包含板块主题并存储这一主题的帖子，需要进行增删改查四种操作
用户管理	管理员对用户进行相应管理。
板块管理	对板块进行增删改查
帖子管理	对帖子进行增删改查

公告管理	对公告进行增删改查
帮助管理	对帮助进行增删改查

3.1.1.3 上下文图

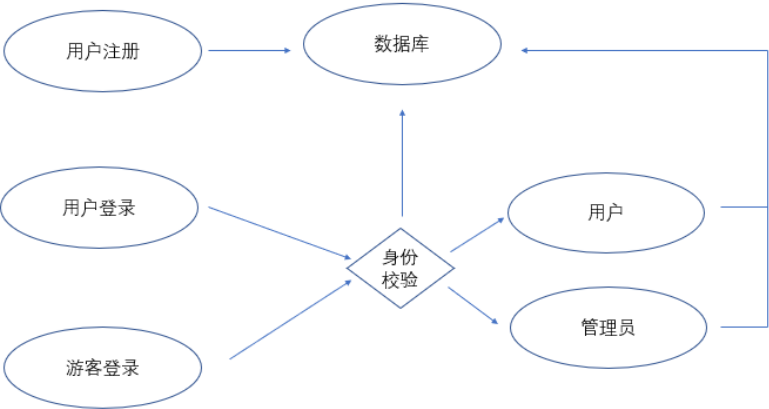


图 3 上下文关系图

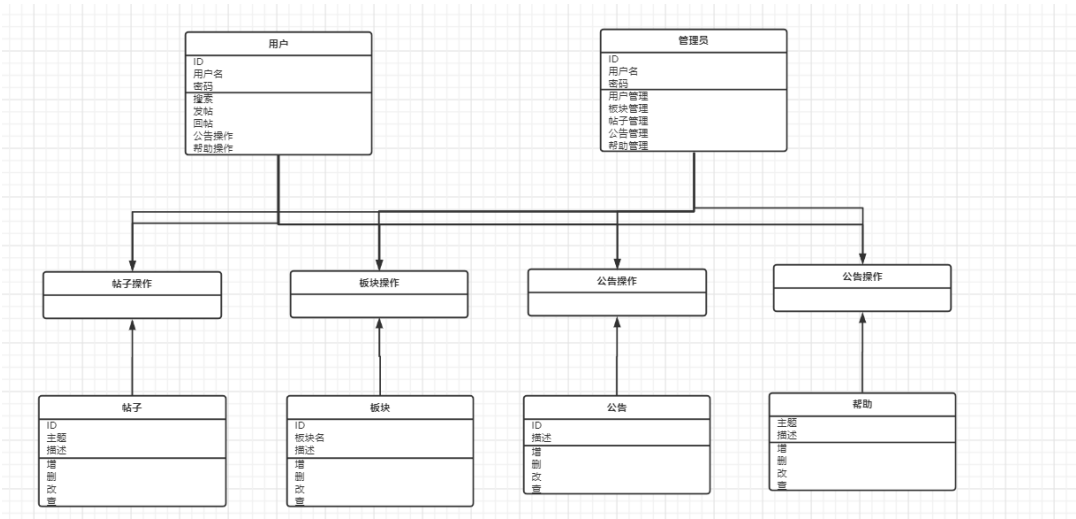


图 4 上下文 ER 图

3.1.1.4 可变性

无我们采用面向对象的编程方式，代码复用性高，可变性强，后期可通过新增类来进行功能的增加。

3.1.1.5 原理

根据系统分析，系统工作流程主要涉及游客工作流程、用户工作流程和管理员工作流程。

游客工作流程:游客是未注册论坛账号的使用者，游客进入论坛后，只能浏览论坛中公开的信息，比如浏览论坛公告、论坛帮助和论坛帖子。游客可以注册成为论坛用户。

用户工作流程:用户是注册论坛账号的使用者，用户进入论坛后，输入用户名和密码，经过系统校验后成功登录论坛。用户不仅拥有游客的所有权限，还可以搜索帖子、发布帖子、回复帖子和查看个人中心。

管理员工作流程:管理员进入论坛后，输入管理员账号和密码，经过系统校验后成功登录论坛。管理员可以进行用户管理、版块管理、帖子管理、公告管理和帮助管理。

3.1.1.6 相关视图

见附录 6.1

3.2 开发视图

3.2.1 顶层开发视图

3.2.1.1 主表示

针对开发的系统情况，确定了如下图所示的开发技术架构，下文详细说明了前后端的交互和服务关系：

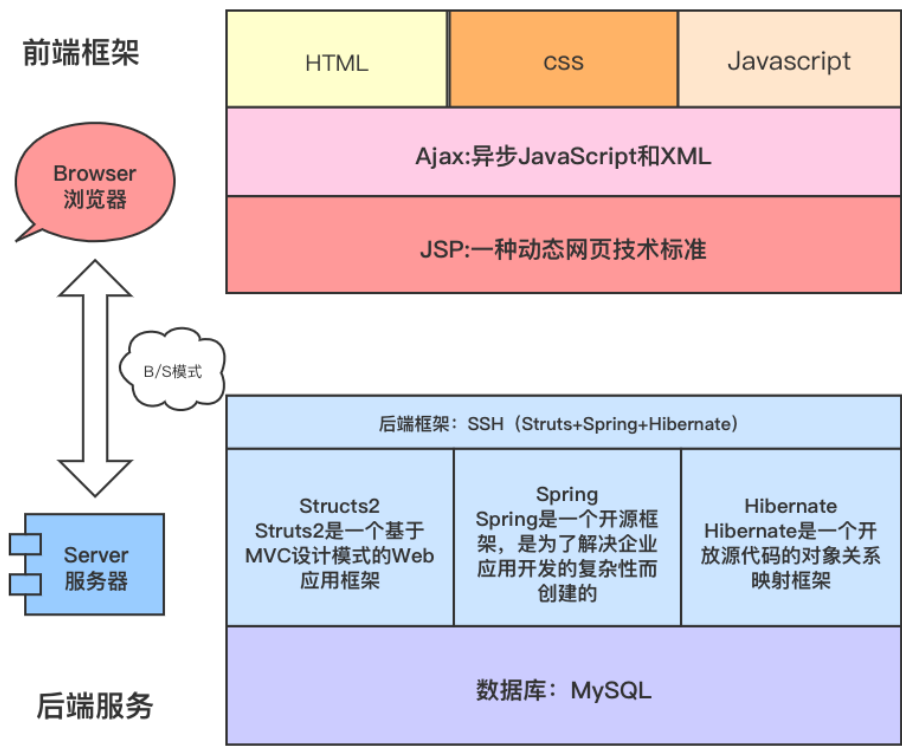


图 5 开发视图

3.2.1.2 构件目录

A. 前端构件

前端构件	描述
------	----

CSS	<p>叠样式表(Cascading Style Sheets)是一种用来表现HTML（标准通用标记语言的一个应用）或XML（标准通用标记语言的一个子集）等文件样式的计算机语言。CSS 能够对网页中元素位置的排版进行像素级精确控制，支持几乎所有的字体字号样式，拥有对网页对象和模型样式编辑的能力。</p> <p>CSS 是一种定义样式结构如字体、颜色、位置等的语言，被用于描述网页上的信息格式化和现实的方式。CSS 样式可以直接存储于 HTML 网页或者单独的样式单文件</p>
Javascript	<p>JavaScript 是一种基于对象和事件驱动并具有相对安全性的客户端脚本语言。同时也是一种广泛用于客户端 Web 开发的脚本语言，常用来给 HTML 网页添加动态功能，比如响应用户的各种操作。</p>
Ajax	<p>Ajax 全称为“ Asynchronous JavaScript and XML ”（异步 JavaScript 和 XML），是指一种创建交互式网页应用的网页开发技术。Ajax 技术可以提供高度交互的 Web 应用，给予用户更丰富的页面浏览体验，Ajax 功能的实现主要依靠 XMLHttpRequest 对象及其属性和方法、发送请求和处理响应。</p>
JSP	<p>JSP 本是一个简化的Servlet设计，它是在传统的网页HTML（标准通用标记语言的子集）文件(*.htm,*.html)中插入 Java程序段 (Scriptlet)和 JSP 标记(tag)，从而形成 JSP 文件，后缀名为 (*.jsp)。用 JSP 开发的 Web 应用是跨平台的，既能在 Linux 下运行，也能在其他操作系统上运行。JSP 技术使用 Java 编程语言编写类 XML 的 tags 和 scriptlets，来封装产生动态网页的处理逻辑。网页还能通过 tags 和 scriptlets 访问存在于服务端的资源的应用逻辑。</p>

## B. 后端构件

控件类型	后端控件	介绍
------	------	----

数据库	MySQL	MySQL是一个开放源码的小型关联式数据库管理系统，开发者为瑞典MySQL AB公司。MySQL被广泛地应用在Internet上的中小型网站中。由于其体积小、速度快、总体拥有成本低，尤其是开放源码这一特点，许多中小型网站为了降低网站总体拥有成本而选择了MySQL作为网站数据库。
SSH后端 服务框 架	Struts2	Struts2 是一个基于 MVC 设计模式的 Web 应用框架，它本质上相当于一个 servlet，在 MVC 设计模式中，Struts2 作为控制器 (Controller) 来建立模型与视图的数据交互。
	Spring	Spring是一个开源框架，是为了解决企业应用开发的复杂性而创建的。Spring使用基本的JavaBean来完成以前只可能由EJB完成的事情。Spring的用途不仅限于服务器端的开发。从简单性、可测试性和松耦合的角度而言，任何Java应用都可以从Spring中受益。
	Hibernate	Hibernate 是一个开放源代码的对象关系映射框架，它对 JDBC 进行了非常轻量级的对象封装，使得 Java 程序员可以随心所欲的使用对象编程思维来操纵数据库。Hibernate 的核心接口一共有 5 个，分别为 :Session、SessionFactory、Transaction、Query 和 Configuration。通过这些接口，不仅可以对持久化对象进行存取，还能够进行事务控制。

### 3.2.1.3 软件开发架构视图

软件在完成部分开发，需要为用户提供服务时，需要进行服务设计，该部分主要分成开发平台、服务器、客户端三部分，其设计如下图：



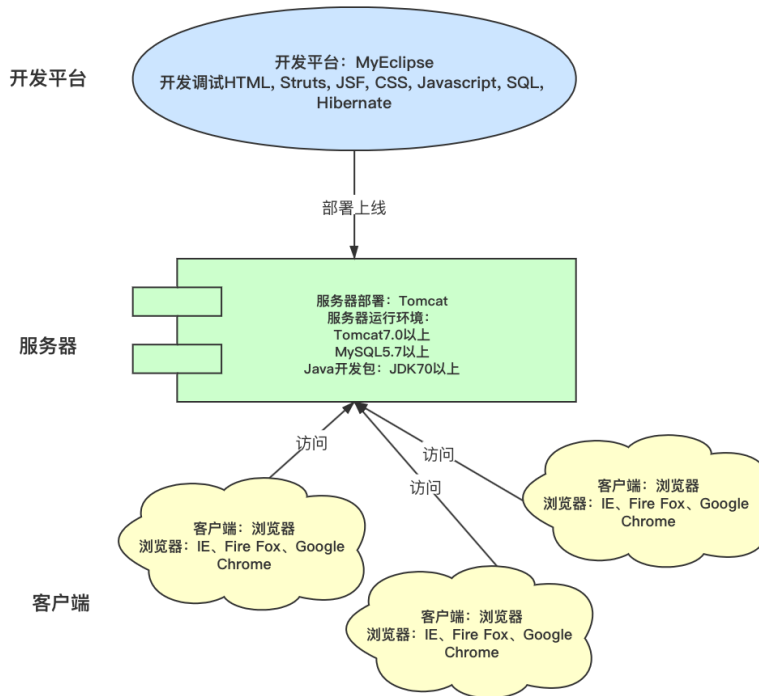


图6 开发视图

#### 3.2.1.4 构件连接方法

**B/S（Browser/Server）模式：**即浏览器和服务器结构。它是随着Internet技术的兴起，对C/S结构的一种变化或者改进的结构。在这种结构下，用户工作界面是通过WWW浏览器来实现，极少部分事务逻辑在前端（Browser）实现，但是主要事务逻辑在服务器端（Server）实现，形成所谓三层结构。

B/S 模式最大的优点就是可以在任何地方进行操作而不需客户端安装任何专门的软件。只要有一台能上网的电脑就能使用，客户端零维护，系统的扩展非常容易。

#### 3.2.1.5 集成开发架构

##### 集成框架架构：

集成 SSH 框架的系统从职责上分为四层：表示层、业务逻辑层、数据持久层和域模块层，以帮助开发人员在短期内搭建结构清晰、可复用性好、维护方便的 Web 应用程序。其中使用 Struts 作为系统的整体基础架构，负责 MVC 的分离，在 Struts 框架的模型部分，控制业务跳转，利用 Hibernate 框架对持久层提供支持，Spring 做管理，管理

struts 和 hibernate。

### 框架连接方法:

用面向对象的分析方法根据需求提出一些模型，将这些模型实现为基本的 Java 对象，然后编写基本的 DAO(Data Access Objects)接口，并给出 Hibernate 的 DAO 实现，采用 Hibernate 架构实现的 DAO 类来实现 Java 类与数据库之间的转换和访问，最后由 Spring 做管理，管理 struts 和 hibernate。

## 3.3 运行视图

### 3.3.1 顶层运行视图

#### 3.3.1.1 主表示

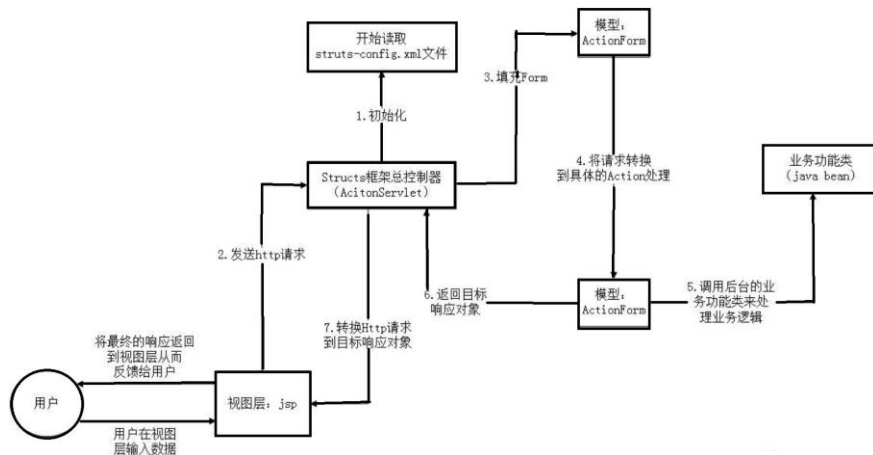


图 7 论坛运行视图

#### 3.3.1.2 构件目录

##### A. 构件及其特性

构件	描述
Struts2	Struts2 是一个基于 MVC 设计模式的 Web 应用框架，它本质上相当于一个 servlet，Struts2 作为控制器来建立模型与视图的数据交互。
Spring	Spring 是一个轻量级控制反转(IoC)和面向切面(AOP)的容器框架。
Hibernate	Hibernate 是一个开放源代码的对象关系映射框架，对 JDBC 进行了轻量级的对象封装，它将 POJO 与数据库表建立映射关系，是一个全自动的 orm 框架，hibernate 可以自动生成 SQL 语句，自动执行。

## B. 关系及其特性

### Struts

- 1、使用者通过浏览器客户端发送 HTTP 协议的数据请求。
- 2、HTTP 请求到达服务器后，经过控制器，使用框架结构填充 FormBean。
- 3、经过填充的数据请求通过框架中控制器的 Action 组件进行数据业务处理。
- 4、经过控制器处理过的业务数据调用 struts 的基础部件实现业务逻辑。
- 5、将业务逻辑经过转换形成 JSP 业务进行处理，并将处理结果返回
- 6、用户在客户端或者浏览器上得到 HTTP 请求的响应。

### Hibernate

Hibernate，负责跟数据库的交接。通过持久化数据对象，进行对象关系的映射，并以对象的角度来访问数据库。Hibernate 有自己的 HQL 语句，HQL 语句在面向对象编程的角度上，通过 Session 的 createQuery 方法创建一个 query 对象，由这个对象来完成对数据库的增删改查等操作。通过 Struts2 中 Action 的返回值，来调用 Dao 层中的业务处理。

### Spring

Spring，使用基本的实体 JavaBean 来完成以前只能用 EJB 完成的事情。核心是控制反转（IOC）和面向切面编程（AOP）。Spring 通过采用依赖注入的方式，通过属性的 Setter 和 Getter 方法来注入这个对象的属性，不完全依赖于容器的 API，且查询依赖与代码实现了解耦。

## C. 构件接口

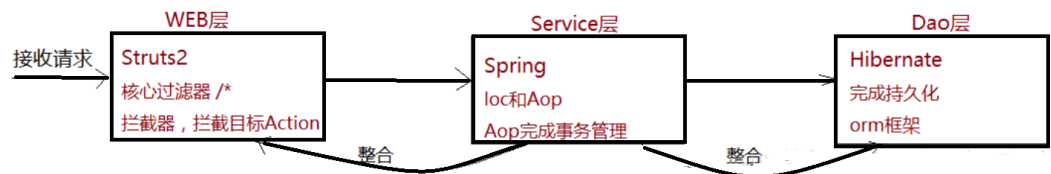


图 8 构件接口

## D. 构件行为

### struts2 行为

1. 客户端初始化一个指向 Servlet 容器（例如 Tomcat）的请求
2. 这个请求经过一系列的过滤器（Filter）。
3. 接着 FilterDispatcher 被调用，FilterDispatcher 询问 ActionMapper 来决定这个请求是否需要调用某个 Action。
4. 如果 ActionMapper 决定需要调用某个 Action，FilterDispatcher 把请求的处理交给 ActionProxy。
5. ActionProxy 通过 Configuration Manager 询问框架的配置文件，找到需要调用的 Action 类。
6. ActionProxy 创建一个 ActionInvocation 的实例。

7. ActionInvocation 实例使用命名模式来调用，在调用 Action 的过程前后，涉及到相关拦截器（Interceptor）的调用。

8. 一旦 Action 执行完毕，ActionInvocation 负责根据 struts.xml 中的配置找到对应的返回结果。返回结果通常是（但不总是，也可能是另外的一个 Action 链）一个需要被表示的 JSP 或者 FreeMarker 的模版。在表示的过程中可以使用 Struts2 框架中继承的标签。在这个过程中需要涉及到 ActionMapper。

### Spring 行为

1. springmvc 请所有的请求都提交给 DispatcherServlet, 它会委托应用系统的其他模块负责对请求进行真正的处理工作。

2. DispatcherServlet 查询一个或多个 HandlerMapping, 找到处理请求的 Controller.

3. DispatcherServlet 请请求提交到目标 Controller

4. Controller 进行业务逻辑处理后，会返回一个 ModelAndView

5. Dispatcher 查询一个或多个 ViewResolver 视图解析器, 找到 ModelAndView 对象指定的视图对象

6. 视图对象负责渲染返回给客户端。

### hibernate 行为

1. 通过 Configuration().configure(); 读取并解析 hibernate.cfg.xml 配置文件。

2. 由 hibernate.cfg.xml 中的<mappingresource="com/xx/User.hbm.xml"/>读取解析映射信息。

3. 通过 config.buildSessionFactory() 得到 sessionFactory。

4. sessionFactory.openSession() 得到 session。

5. session.beginTransaction() 开启事务。

6. persistent operate;

7. session.getTransaction().commit() 提交事务

8. 关闭 session;

9. 关闭 sessionFactory;

#### 3.3.1.3 上下文图

无

#### 3.3.1.4 可变性

无

#### 3.3.1.5 原理

**Hibernate 的运行原理：** 读取并解析配置文件，读取并解析映射信息，创建 SessionFactory，打开 Session，创建事物 Transaction，持久化操作，提交事务，关闭 Session，关闭 SessionFactory。

#### spring 的运行原理：

1. springmvc 请所有的请求都提交给 DispatcherServlet, 它会委托应用系统的其他模块负责负责对请求进行真正的处理工作。

2. DispatcherServlet 查询一个或多个 HandlerMapping, 找到处理请求的 Controller.
3. DispatcherServlet 请求提交到目标 Controller
4. Controller 进行业务逻辑处理后, 会返回一个 ModelAndView
5. DispatcherServlet 查询一个或多个 ViewResolver 视图解析器, 找到 ModelAndView 对象指定的视图对象
6. 视图对象负责渲染返回给客户端。

### 3.3.1.6 相关视图

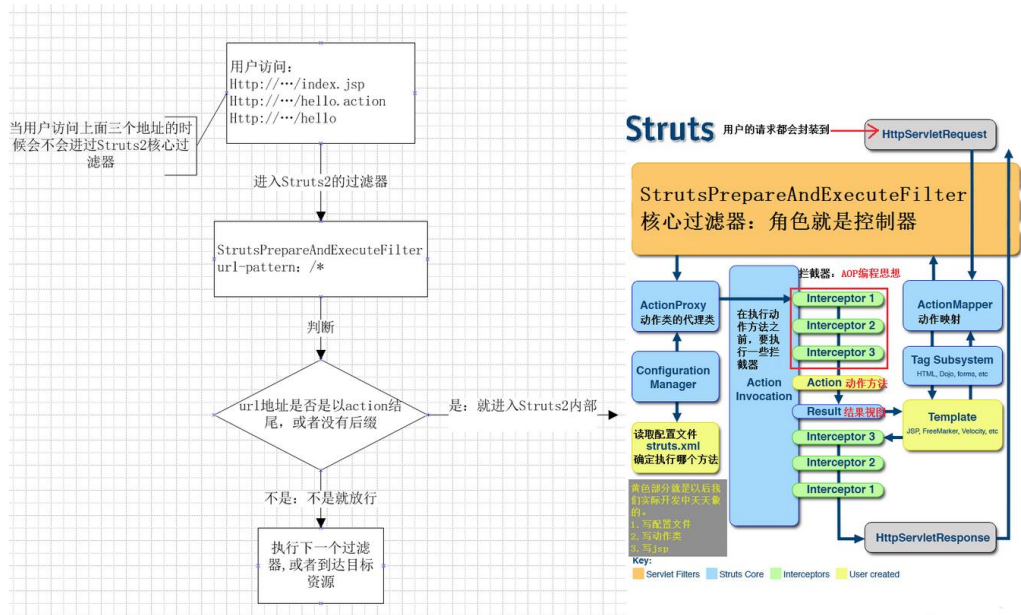


图 9 Struts 运行视图

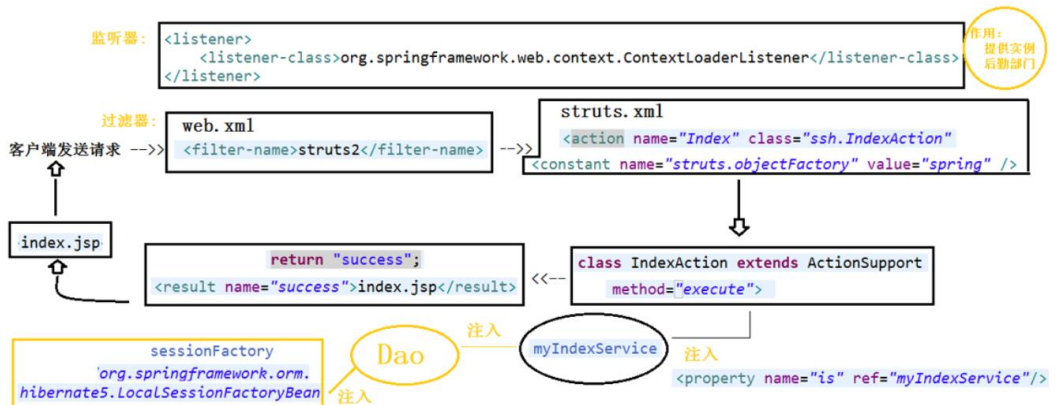


图 10 Spring 运行视图

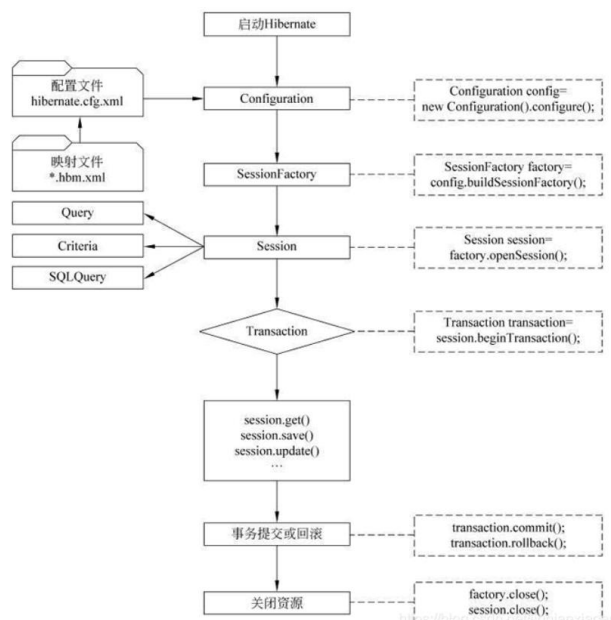


图 11 Hibernate 运行视图

### 3.4 部署视图

部署图描述的是系统运行时的结构，展示了硬件的配置及其软件如何部署到网络结构中。

#### 3.4.1 主表示

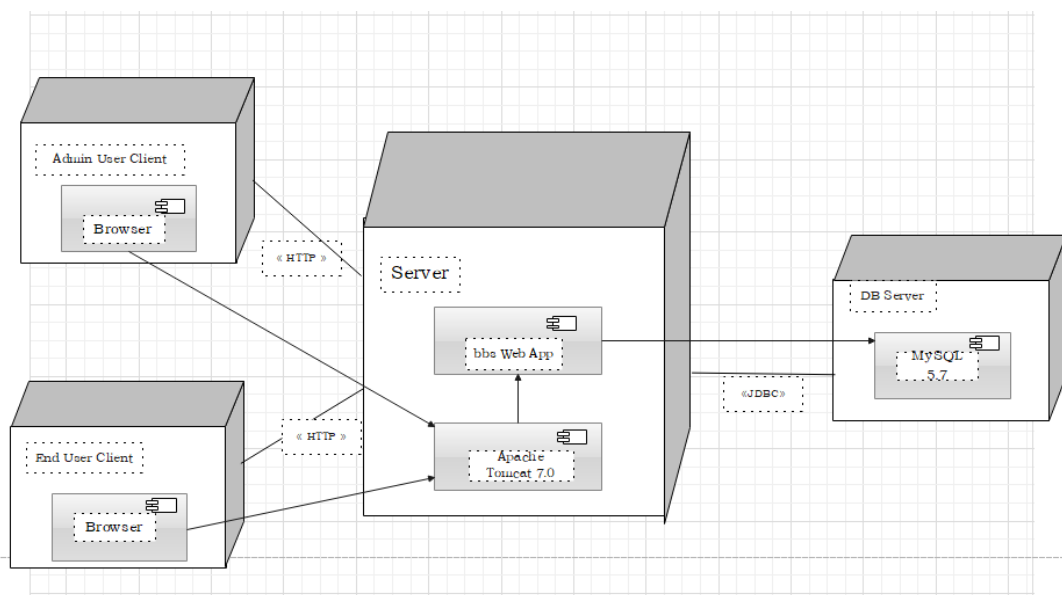


图 12 论坛系统部署图

### 3.4.1.2 构件目录

#### A. 构件及其特性

构件	描述
End User Client	普通用户：通过浏览器访问 bbs，拥有普通用户权限，拥有浏览论坛，发帖、回帖等权限，支持的浏览器包括 IE、Fire Fox、Google Chrome 等。
Admin User Client	管理员用户：通过浏览器访问 bbs，拥有管理员权限，主要负责系统的后台管理工作，支持的浏览器包括 IE、Fire Fox、Google Chrome 等。
Server	应用服务器运行 bbs 论坛系统，系统部署在 Tomcat7.0 容器上，它与数据服务器通过 jdbc 连接。
DB Server	数据服务器运行 MySQL5.7
bbs Web App	论坛系统
Apache Tomcat 7.0	Tomcat 服务器是一个免费的开放源代码的 Web 应用服务器，最新的 Servlet 和 JSP 规范总是能在 Tomcat 中得到体现。
MySQL5.7	一个开放源码的小型关联式数据库管理系统

#### B. 关系及其特性

用户通过浏览器访问，所以只要连通网络，用户端便可进入论坛。服务器端还需要提供 MySQL 的数据库等。

HTTP 超文本传输协议，一个网际协议。

系统部署在 Tomcat7.0 容器上，它与数据服务器通过 JDBC 连接。JDBC JAVA 数据库连接，一套为数据库存取编写的 Java API。

#### C. 构件接口

客户端与服务端的连接可以是局域网或互联网。

#### D. 构件行为

End User Client：该组件表示普通用户。普通用户分为游客和注册用户，其中游客只能浏览论坛公共部分的内容，还可以注册成为论坛用户；注册用户则可以登录论坛、搜索帖子、发帖、回帖和查看个人中心。

Admin User Client：该组件表示系统管理员。系统管理员主要负责系统的后台管理工作，如用户管理、帖子管理、版块管理、公告管理、帮助管理。

**Browser:** 该组件表示在 Web 浏览器上运行的应用程序的用户界面。使用 Web 浏览器, bbs 的用户可以打开网站, 获得自己的权限, 在论坛上进行相应的操作。

**Server:** 该组件表示应用服务器。

**bbs Web App:** 对于 BBS 论坛系统来说, 最主要的功能就是用户浏览帖子、发布帖子以及回复帖子。同时考虑用户所发布帖子的内容需要健康且合法, 这就需要管理员加强对帖子的审核。如果用户想要在论坛发布帖子, 前提是需要注册且处于登录状态。此外, 为了便于从内容上对帖子加以区分, 方便管理员对帖子的管理, 还需要将论坛分为多个版块。还需要将用户划分为不同级别, 根据用户级别的不同在论坛中为用户分配不同的权限, 以进一步加强对论坛的管理。因此, 论坛既要实现对于帖子的管理, 又要实现对于用户的管理, 还要兼顾到各个功能模块之间的协调合作, 共同完成 BBS 论坛系统的业务需求。下面采用 UML 用例模型对系统功能需求进行分析。

**Apache Tomcat 7.0:** 该组件表示 Web 应用服务器。系统部署在 Tomcat7.0 容器上, Tomcat 不仅仅是一个 Servlet 容器, 它也具有传统的 Web 服务器的功能: 处理 html 页面。

**MySQL5.7:** 该组件表示数据库管理系统。一个开放源码的小型关联式数据库管理系统。论坛系统数据库用来存储管理员和用户信息, 用于验证信息保证安全性; 存储帖子信息; 存储评论回复信息; 存储公告信息; 存储版块信息等。

### 3.4.1.3 上下文图

无

### 3.4.1.4 可变性

(1)服务器操作系统: WIN 7 及以上 64 位系统

客户端 操作系统: WIN 7 及以上 64 位系统

(2)浏览器: IE、Fire Fox、Google Chrome。

(3)在 Windows 10 平台上开发, 后期考虑部署到 Linux 服务器上。

(4)相应速度方面: 在平台的服务量快速增长后, 为了提高用户的使用体验和反应速度, 可能需要对平台的语言框架、需求处理算法进行迭代, 便于带来更好的用户体验。

(5)服务质量方面: 在平台上线之初, 可能存在服务的的质量的问题(准确性、数据正确、数据风险等方面), 后期可能需要重新对系统的服务流程和数据处理流程进行重新设计和重构。

(6)考虑到后期的某些活动、技术交流等平台具有独立性的特点, 如果都集成在一个庞大的平台上, 在用户使用过程中加载的开销很大、速度也比较慢, 后续可以采用“微服务”框架, 将不同的服务对象对应的不同功能模式独立出来, 这种方案也需要对平台进行重构。

(7)平台灵活性方面: 需要能够较好地支持如上所述的改进需求, 保证技术团队在后



续的迭代过程中可以更加便捷地完成版本的兼容和技术的重构。

#### 3.4.1.5 原理

我们利用 UML 部署图来实现了论坛系统部署图。之所以该部分采用部署视图来反映论坛系统的系统架构，是因为部署图描述了部署和运行软件的物理网络配置。关注系统安装和部署在什么样的环境上。通过该部分我们可以很好的了解软件系统的可靠性，可伸缩性等性能。

论坛系统的用户分为普通用户（游客、注册用户）和系统管理用户，因此，这保证了用户在身份认证、授权控制和私密性等方面要求较高的安全性。

系统具有良好的可兼容性，可以在 Windows 的大部分版本的操作系统中正常的运行，另外，也支持不同浏览器版本，能够在微软的 IE 浏览器、谷歌浏览器等常用的浏览器上正常运行。

系统的易升级和管理也是系统开放性的表现。系统设计开发的功能要尽量简单、实用，用户在使用系统过程中能够感受到系统的快捷、方便，不存在很多累赘和多余的操作或者功能。

系统采用了 Tomcat 服务器。主要是因为 Tomcat 技术先进、性能稳定，而且免费，深受 Java 爱好者的喜爱并得到了部分软件开发商的认可，成为目前比较流行的 Web 应用服务器。

#### 3.4.1.6 相关视图

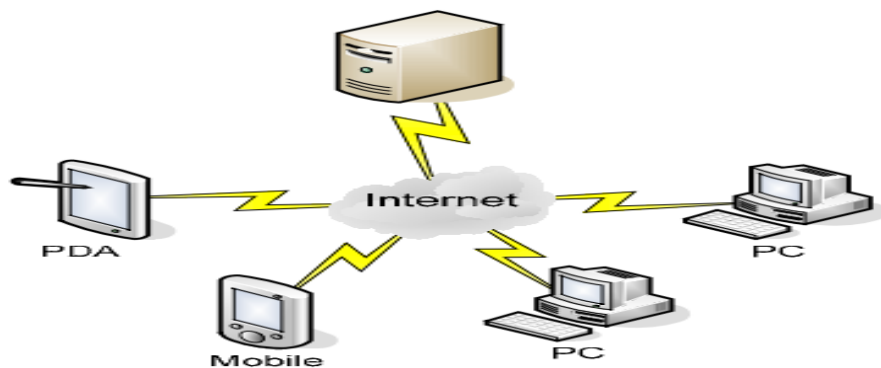


图 13 论坛系统网络结构图

### 3.5 用例视图

该部分是对软件架构的用例视图的描述，重点关注的是软件功能性需求。

#### 3.5.1 顶层用例视图

##### 3.5.1.1 主表示

见附录 6.1 用户视图，管理员视图

##### 3.5.1.2 构件目录

##### 1、构件及其特性

构件	描述	构件
用户注册、登录	用户注册	用户填写用户名、昵称、密码、邮箱等信息注册成为论坛用户
	用户登录	对用户输入的用户名，密码进行验证，验证通过后，该用户可以使用论坛系统中自己拥有权限的那部分功能，否则拒绝使用。
个人中心	查看信息	用户在个人中心，可以查看信息回复、查看自己发布的帖子、查看自己的评论回复。
	修改个人资料	用户在个人中心进行个人资料的维护。
	修改密码	用户在个人中心，可以修改密码。
	退出登录	用户在个人中心，可以退出登录。
搜索	搜索	用户可以输入关键词进行搜索帖子。
帖子相关操作	帖子列表	论坛首页中，帖子以列表形式展示给用户，需要展示热帖榜和新帖榜。
	发帖	用户可以发布帖子，需要在对应的版块分类下发帖，需要填写帖子标题和帖子内容等信息。
	回帖	用户可以回复其他用户发布的帖子。
公告展示	公告展示	论坛首页需要展示公告信息，按照时间顺序以列表形式展示。
帮助中心	帮助信息	论坛首页需要显示帮助信息，按照时间顺序以列表形式展示。
版块分类	版块分类	为了对帖子进行管理，需要将论坛分为多个版块，帖子隶属于版块。
用户管理	注册用户	对论坛注册的用户进行管理
	用户信息	查看用户详细信息
	删除用户	删除用户信息
	权限设置	给用户设置权限
版块管理	添加版块	添加版块

构件	描述	构件
	删除版块	删除版块
	修改版块	修改版块内容
	查看版块	查看版块内容
帖子管理	帖子维护	可以给用户发的帖子加精华和加置顶
	查看帖子	可以查看用户发的帖子
	删除帖子	可以删除用户发的帖子。
公告管理	添加公告	添加公告信息
	删除公告	删除公告信息
	修改公告	修改公告信息
	查看公告	查看公告信息
帮助管理	添加帮助	添加帮助信息
	删除帮助	删除帮助信息
	修改帮助	修改帮助信息
	查看帮助	查看帮助信息

## 2、关系及其特性

用例视图中存在关联关系、包含关系。

关联关系：用户与用例之间存在关联关系。

包含关系：

用户管理中包含了查看用户、删除用户、设置权限等用例的实现；

帖子管理中包含了删除帖子、查看帖子、置顶帖子等用例实现；

板块管理包含了添加板块、删除板块、修改板块、查看板块等用例实现

公告管理包含了添加公告、删除公告、修改公告、查看公告等用例实现；

帮助管理包含了添加帮助、删除帮助、修改帮助、查看帮助等用例实现。

## 3、构件接口

用户通过浏览器发送请求，服务端接受请求并进行数据的处理。应用程序与数据库进行交互（数据的查询修改删除等操作）等。

#### 4、构件行为

构建具体行为在构建及其特性中已经具体描述，在此不再赘述。

##### 3.5.1.3 上下文图

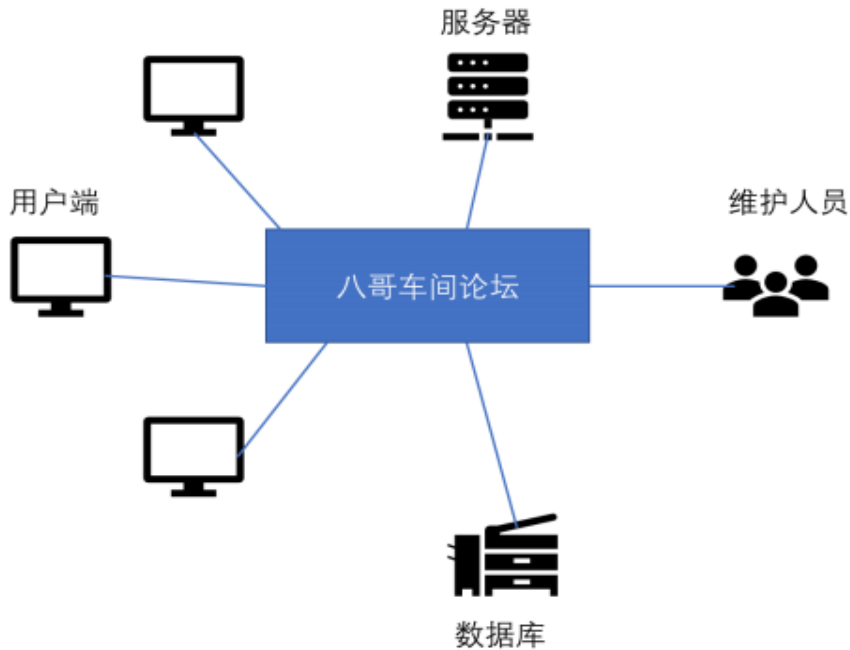


图 14 模型概念图

##### 3.5.1.4 可变性指南

对于功能需求方面的可变性主要表现在系统支持不同的用户拥有不同的权限，通过对用户进行身份认证保证授权控制和私密性。

另外，除了基本功能外，我们还增加了资源分享、资源接受、资源付费、公共资源、活动发布、活动参与、活动管理、活动结果发布、活动奖励发放等功能。这些增加功能我们团队根据时间合理进行完善。

##### 3.5.1.5 原理

我们利用 UML 用例图来实现了论坛系统用例图。之所以采用用例视图来反映我们的体系结构，是因为从系统用户的角度考虑问题，设计出的架构能够满足业务逻辑的需求。

##### 3.5.1.6 相关视图

见附录 6.1 用户视图的细化,管理员视图的细化

## 4.视图之间关系

### 4.1 视图之间关系说明

上述的 5 种视图，每一个视图只关心系统的一个侧面，5 个视图结合在一起才能反映系统的软件体系结构的全部内容。各个视图之间的关系如下图所示，其中 Physical View 就是物理视图(部署视图)，

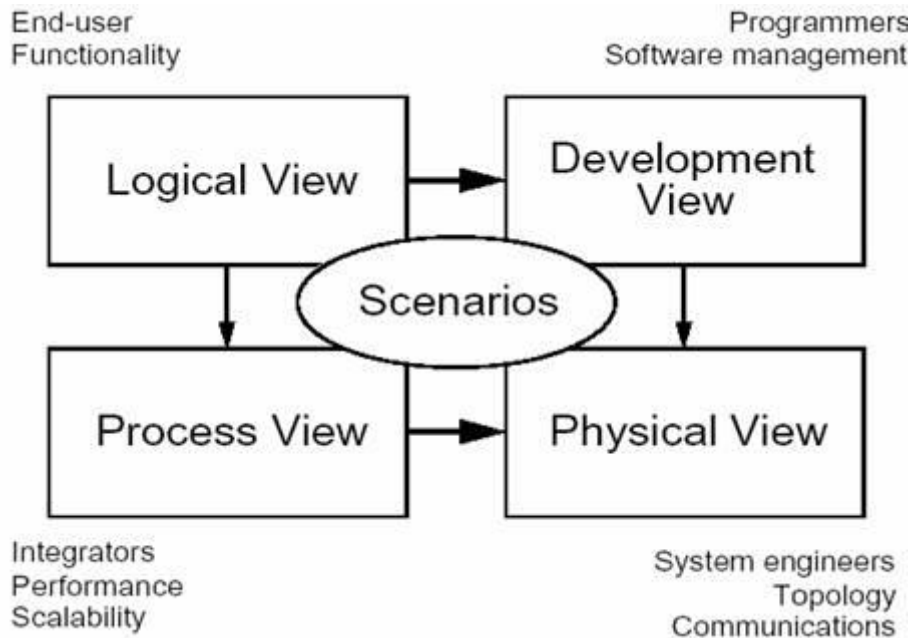


图 15 视图之间的关系

### 4.2 视图-视图关系

用例视图是五个视图的核心，是因为它确定了以下信息，而其它 4 个视图都是需要围绕着这些信息进行设计：

**系统边界：**有了边界，才能够确定系统的设计范围；同时，通过边界能够识别出系统需要与用户或其它系统进行交互；

**系统用户：**明确的用户定义是系统需求分析的先决条件；

**功能和场景：**通过识别出系统与用户或其它系统的交互，可以分析出系统需要提供哪些功能，以及这些功能存在哪些应用场景

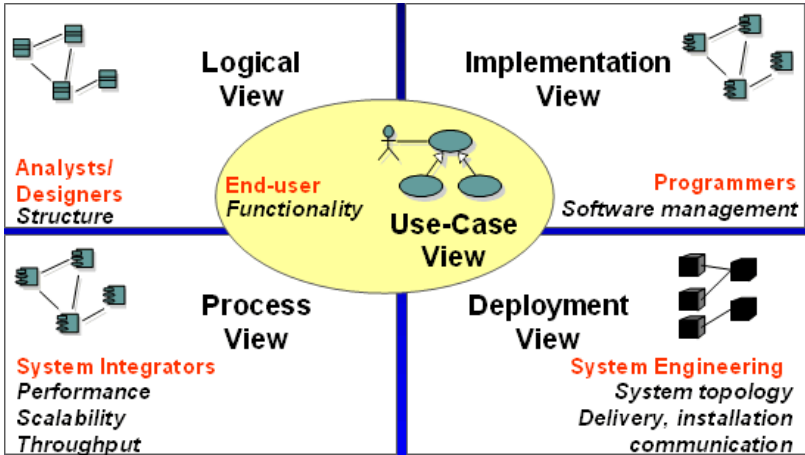


图 16 各视图的特点

(1) 用例视图和开发视图的关系

逻辑视图	开发视图
BBS 论坛系统	B/S 架构
BBS 论坛界面	CSS
用户界面交互	JavaScript + Ajax + JSP
管理员界面交互	
用户数据	MySQL 表 SSH 后端服务框架 开发视图
帖子数据	
BBS 系统服务	
管理员(用户管理, 帖子管理, 板块管理, 公告管理, 帮助管理)	
用户(浏览, 注册, 登录, 搜索, 发帖, 回帖, 个人中心)	
逻辑视图	

(2) 逻辑视图与运行视图关系

逻辑视图	运行视图
信息访问	使用 Hibernate, 通过持久化数据对象, 进行对象关系的映射, 并以对象的角度来访

	问数据库
界面与交互	属于客户端技术, 不在讨论范围
BBS 系统服务	使用 Struts,  1、HTIP 请求。 2、HITP 请求到达服务器, 经过控制器, 使用框架结构填充 FormBean。 3、经过填充的数据请求通过框架中控制器的 Action 组件进行数据业务处理。 4、经过控制器处理过的业务数据调用 struts 的基础部件实现业务逻辑。 5、将业务逻辑经过转换形成 JSP 业务进行处理, 并将处理结果返回
管理员(用户管理, 帖子管理, 板块管理, 公告管理, 帮助管理)	
用户(浏览, 注册, 登录, 搜索, 发帖, 回帖, 个人中心)	

### (3) 逻辑和开发视图与部署视图的关系

开发视图, 运行视图	部署视图
MySQL, Hibernate	MySQL5.7 on Database Server
SSH 服务框架, Struts 和 Spring	Apache Tomcat7.0 on Web Server
CSS	客户端浏览器, 要求支持运行 JavaScript 脚本
JS, AJAX, JSP	

## 5.需求与架构之间的映射

我们的架构选择分层模式、浏览器-服务器端模式、模型-视图-控制器三种混合模式。

### 需求:

论坛系统, 一个 Web 应用, 往往会随着用户需求而迭代版本, 这就要求对应用的易修改性有较高要求; 其次, 在开发新功能时, 能复用之前的逻辑或代码能减少开发人员的工作量, 并且不易出错, 易于复用也很重要; 安全性排第三位, 用户的隐私和网站的内部逻辑需要仔细保护, 防止泄露和被破解; 性能和数据表示往往是固定的, Web 应用的性能取决于服务器和代码逻辑, 前者和后者与其他应用相比, 往往差距不大, 数据表示就是键值对, 非常固定。

## 架构:

分层模式是 Web 应用的基本模式, 分层模式的安全性, 易修改性和易复用性使开发变得便捷, 并且各个层的数据表示独立, 底层数据表示更灵活, 但层数过多会导致系统冗余或性能下降。

我们的应用基于大众, 选择 B/S, HTTPS 的安全性非常可靠, 并且没有中间件, 系统轻便, 性能高。

模型-视图-控制器模式是最近几十年 Java Web 应用的宠儿, 该模式把表现层, 业务逻辑层, 数据层分开, 使得业务代码不用和前端代码混合在一起, 可维护性和灵活性都提高了很多. 三个模块之间接口固定, 模块独立易于修改, 安全性高, 完全使用模型访问数据库, 数据表示更灵活。

## 6.附录

### 6.1 架构元素索引

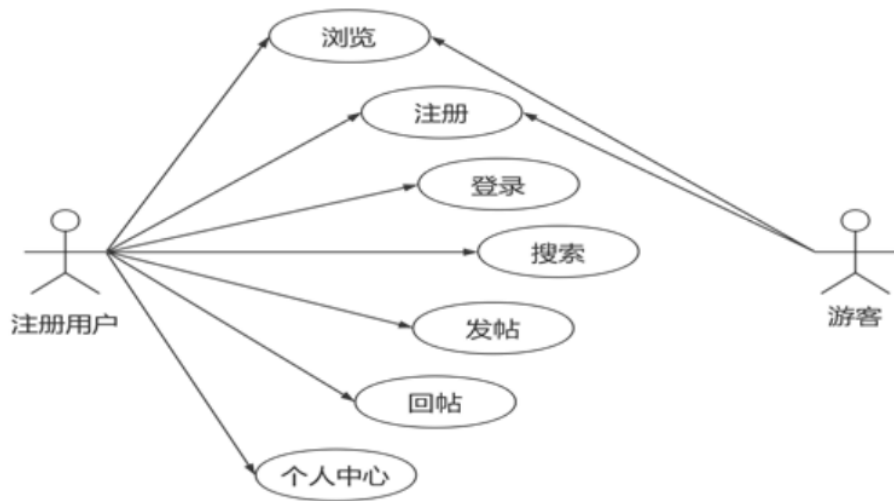


图 17 用户逻辑视图



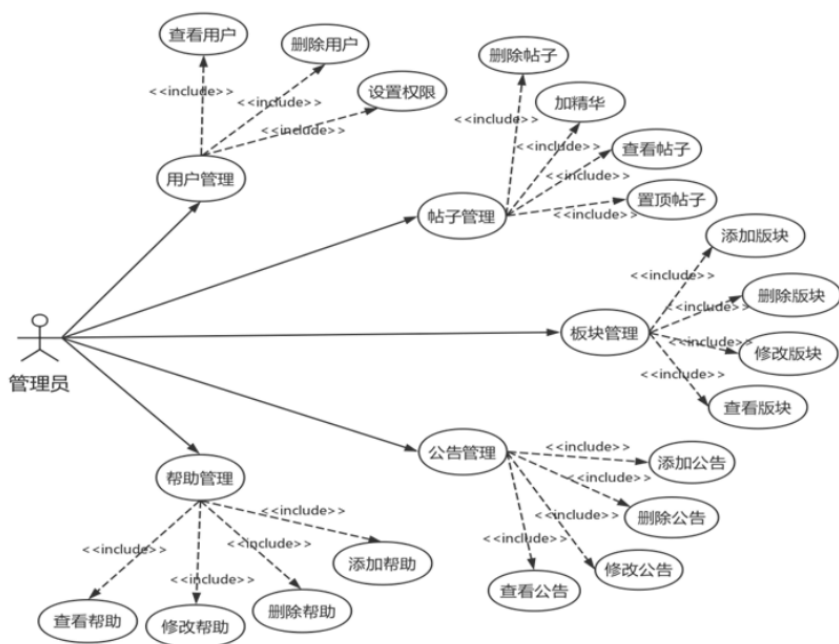
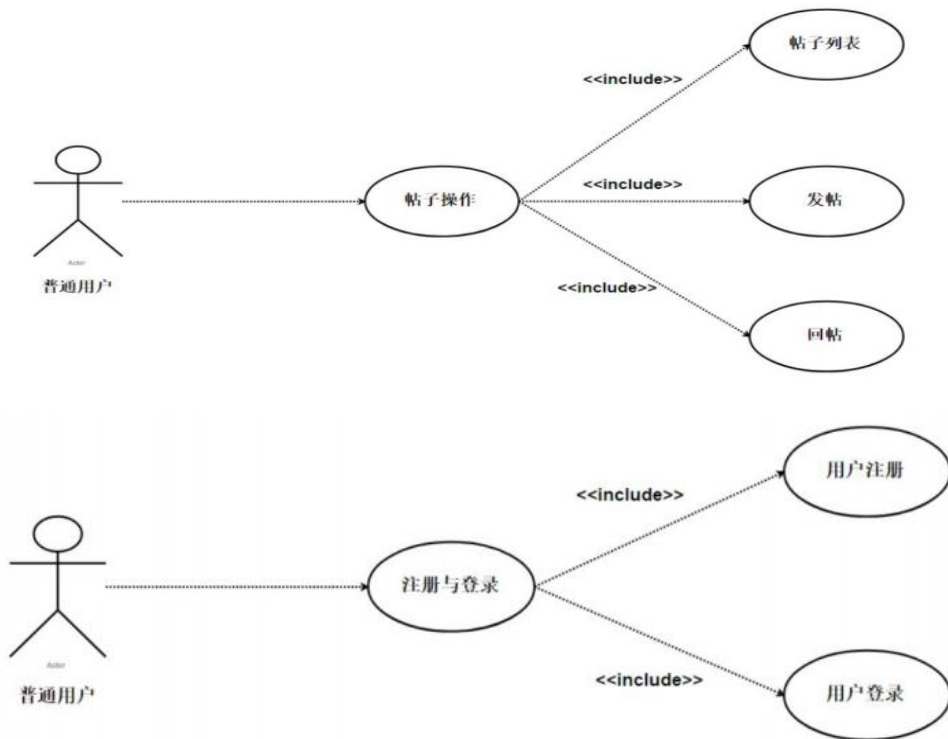


图 18 管理员逻辑视图



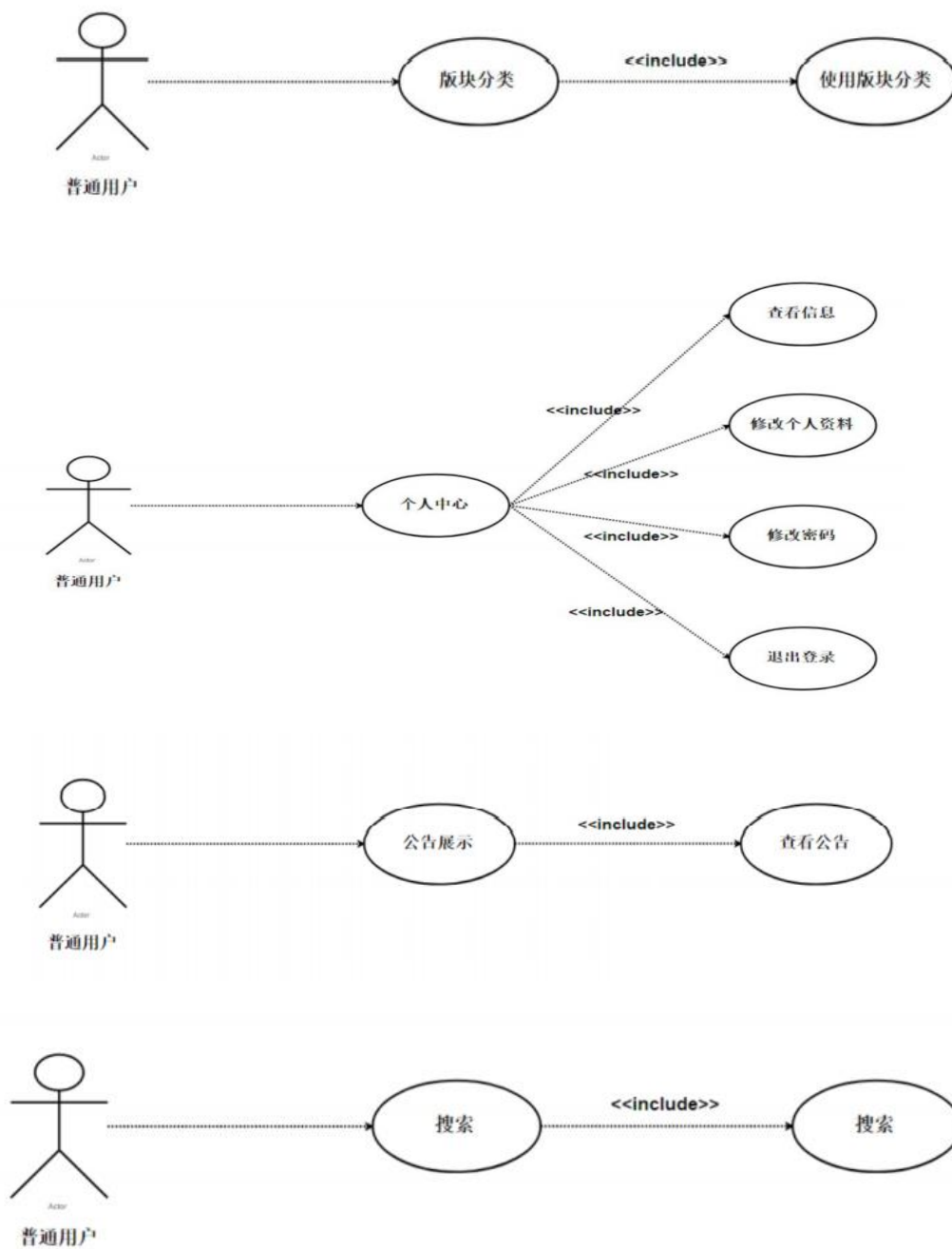
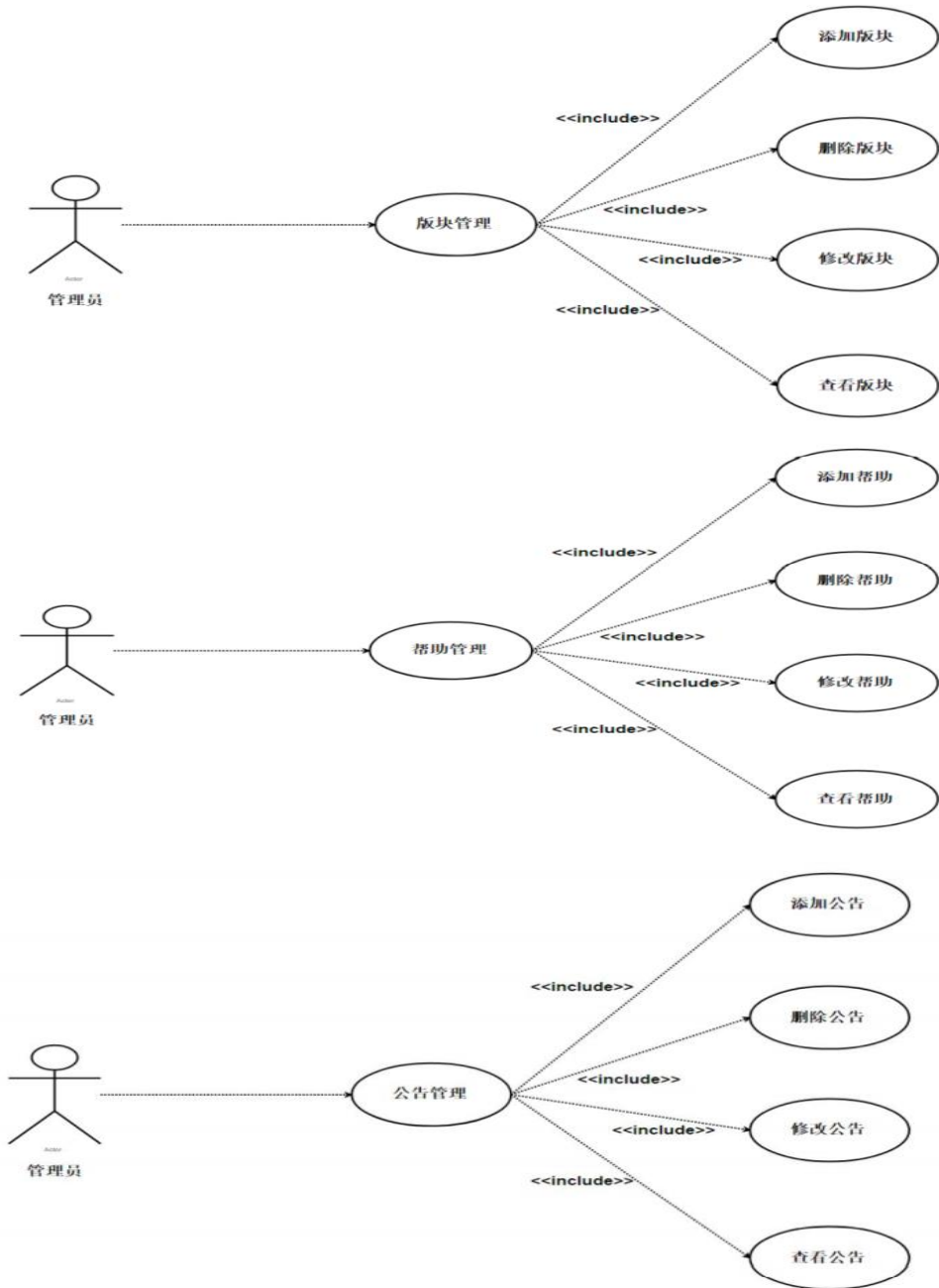


图 19 普通用户用例图的细化



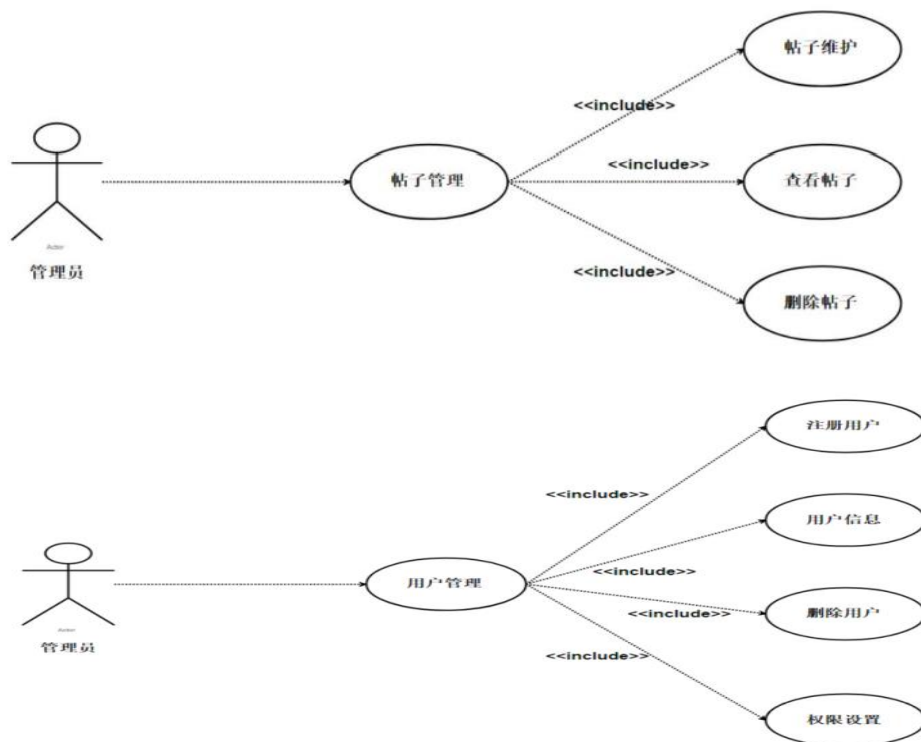


图 20 系统管理员用户用例图的细化

## 6.2 术语表

### B/S 结构:

B/S 结构 (Browser/Server, 浏览器/服务器模式), 是 WEB 兴起后的一种网络结构模式, WEB 浏览器是客户端最主要的应用软件。这种模式统一了客户端, 将系统功能实现的核心部分集中到服务器上, 简化了系统的开发、维护和使用。客户机上只要安装一个浏览器, 如 Netscape Navigator 或 Internet Explorer, 服务器安装 SQL Server、Oracle、MYSQL 等数据库。浏览器通过 Web Server 同数据库进行数据交互。

### JSP:

JSP 本是一个简化的 Servlet 设计, 它是由 Sun Microsystems 公司倡导、许多公司参与一起建立的一种动态网页技术标准。JSP 技术有点类似 ASP 技术, 它是在传统的网页 HTML (标准通用标记语言的子集) 文件(\*.htm,\*.html)中插入 Java 程序段(Scriptlet)和 JSP 标记(tag), 从而形成 JSP 文件, 后缀名为(\*.jsp)。用 JSP 开发的 Web 应用是跨平台的, 既能在 Linux 下运行, 也能在其他操作系统上运行。

JSP 技术使用 Java 编程语言编写类 XML 的 tags 和 scriptlets, 来封装产生动态网页的处理逻辑。网页还能通过 tags 和 scriptlets 访问存在于服务端的资源的应用逻辑。JSP 将网页逻辑与网页设计的显示分离, 支持可重用的基于组件的设计, 使基于 Web 的应用程序的开发变得迅速和容易。JSP(JavaServer Pages)是一种动态页面技术, 它的主要目

的是将表示逻辑从 Servlet 中分离出来。

### 6.3 缩略语

#### SSH:

SSH 是 Struts+Spring+Hibernate 的一个集成框架,是目前比较流行的一种 Web 应用程序开源框架。本系统使用 Struts2。Struts 对 Model, View 和 Controller 都提供了对应的组件。Spring 是一个轻量级的控制反转(IoC)和面向切面(AOP)的容器框架,它由 Rod Johnson 创建。它是为了解决企业应用开发的复杂性而创建的。Spring 使用基本的 JavaBean 来完成以前只可能由 EJB 完成的事情。Hibernate 是一个开放源代码的对象关系映射框架,它对 JDBC 进行了非常轻量级的对象封装,可以应用在任何使用 JDBC 的场合,可以在 Servlet/JSP 的 Web 应用中使用,也可以在应用 EJB 的 J2EE 架构中取代 CMP,完成数据持久化的重任。

#### DAO 组件:

DAO(Data Access Object)是一个数据访问接口,数据访问:顾名思义就是与数据库打交道。夹在业务逻辑与数据库资源中间。

### 6.4 保证项目的质量属性

#### 6.4.1 安全性

八哥车间 BBS 论坛采用 B/S 架构,客户端软件主要是 Web 页面。

安全性可分为:数据或功能访问控制安全和页面访问控制安全。

##### 6.4.1.1 数据或功能访问控制安全

数据或功能访问安全性可保证在预期的安全性情况下,不同授权的用户只能访问特定的功能,或者只能访问有限的数据库。例如,普通用户可进行输入数据、创建新账户等操作而不能随意删除数据或账户,只有管理员才能删除数据或账户。

##### 6.4.1.2 页面访问控制安全

(1)页面登录。B/S 架构软件必须测试登录用户名和密码的有效性、输入大小写的敏感性、用户登录是否有次数限制;而且要测试是否可以不登陆而直接浏览某个页面、IP 地址登录是否有限制等;

(2)超时限制。如果用户登陆系统后在一定时间内没有进行任何页面操作,应进行超时判断,强制用户重新登陆后才能正常使用系统。

(3)日志文件。日志文件是保证 Web 应用系统安全性的重要的工具。B/S 架构软件需要测试日志文件记录信息的完整性、各类操作的可追踪性。在服务器后台,还要检测服务器的日志记录是否正常进行。

(4)SSL。为保证信息在 Internet 上传输的安全性,B/S 架构软件会采用 SSL 技术。如果使用了 SSL,测试人员需要测试加密的正确性、检测信息的完整性,确定是否有相应的替代页面。当用户进入或离开安全站点的时候,是否有相应的提示信息。

(5)脚本安全性。服务器端的各类数据处理脚本常常成为安全漏洞而被黑客所利用。找出当前站点使用了哪些脚本语言,并研究该语言的缺陷,而且应该确定没有经过授权,相关人员就不能在服务器端保存和编辑相关脚本。

##### 6.4.1.3 安全性测试技术手段

为确保数据访问安全而进行的安全功能测试和为保证页面访问安全而进行的安全漏洞测试。安全功能测试基于软件的安全功能需求说明，测试软件的安全功能实现是否与安全需求一致；安全漏洞测试则站在攻击者的角度，以发现软件的安全漏洞为目的。

### 安全功能测试

Web 服务主要的安全功能需求包括身份认证、消息机密性、不可否认性、完整性、授权、可用性、访问控制、审计跟踪、安全管理、隐私保护等。安全功能测试主要针对上述安全内容的具体实现进行功能性测试验证。功能验证就是对软件需求中确定的有关安全模块的功能进行测试验证。开发者一般会在软件设计和开发过程中增加一些必要的安全防护措施，如权限管理模块、数据加密模块、传输加密模块等。安全功能验证可以采用与一般程序功能测试相似的方法，如黑盒测试、白盒测试或灰盒测试等方法来进行。

### 安全漏洞测试

安全漏洞测试就是识别软件的安全漏洞。漏洞指的是存在于一个系统内的弱点或缺陷，漏洞被利用可能造成软件受到攻击，使软件进入不安全的状态。

B/S 应用程序攻击包括：对程序本身的拒绝服务攻击、改变网页内容以及窃取用户信息等。常见的 B/S 应用安全漏洞有：已知弱点和错误配置、隐藏字段、后门和调试漏洞、跨站点脚本编写、参数篡改、更改 Cookies、缓冲区溢出、直接访问浏览等。

#### 参数篡改

WSDL 文件( Web 服务描述文件) 保存了 Web 服务提供的方法、调用这些方法需要的参数个数以及需要的参数类型等重要信息。参数篡改是指：首先对 WSDL 文件进行扫描，寻找 Web 服务调用可接受的参数类型，故意发送 Web 服务不期望的数据类型，对 Web 服务进行攻击。

#### XML 解析器攻击

XML 消息可以对递归实体进行扩展。基于这一特征，攻击者通过恶意构造包含大量递归嵌套元素的消息，例如构造嵌套 100000 层的消息，用来耗尽服务器资源或者使 XML 解析器崩溃，达到拒绝服务攻击的目的。

#### 注入式攻击

SOAP(简单对象访问协议) 是一种轻量的、简单的、基于 XML 的协议，它被设计成在 Web 上交换结构化的和固化的信息。SOAP 消息携带了 Web 服务调用需要的参数，而这些参数极有可能是 SQL 查询语句或者 XPATH 查询语句的一部分。注入式攻击者构造相关的查询语句或者认证语句用来绕过数据库认证，从而执行非法查询操作、恶意篡改数据或者非法执行系统命令等。例如下面的 XPATH 语句用来在数据库中查询操作者的用户名和密码：`user [name = 'nare'and pass = 'misback']` 攻击者通过插入字段 `or 1 = 1`，构造新语句，导致系统返回所有的用户信息：`user [name = 'nare' or 1 = 1or'and pass = 'mis-back']`

#### 跨站脚本攻击

跨站脚本攻击(也称为 XSS) 指利用网站漏洞从用户那里恶意盗取信息。用户在浏览网站、使用即时通讯软件或者在阅读电子邮件时，通常会点击其中的链接。攻击者通过在链接中插入恶意代码，就能够盗取用户信息。对于跨站脚本攻击进行防范主要有两方面：

验证所有输入数据，有效检测攻击；对所有输出数据进行适当的编码，以防止任何已成功注入的脚本在浏览器端运行。

#### 6.4.2 可复用性

可复用性（Reuseability），顾名思义，是指可以重复使用的程度。对于我们的论坛项目，主要考虑后端代码复用，前端代码复用和线程复用。

##### 6.4.2.1 后端代码复用

###### 使用接口和抽象类

先定义接口或者抽象类，而不是根据现实对象直接定义类。用抽象类的实例去访问各种派生对象。

证据实例：对于整个论坛系统，有三种角色，分别是游客，注册用户，管理员。我们可以定义接口或抽象类 User，然后再定义 Guest, RegisteredUser, Administrator，这三个类都从 User 派生，有公共的方法签名。这样定义之后，使用基类 User 的指针使用同样的逻辑操作用户，管理员和游客。比如，现在有一个游客注册成了论坛用户，则只需改变对象引用，无需再写用户转换代码。



###### 类之间相互关联依赖时的方法划分

论坛系统的各个实体，不可能是相互独立的，比如发帖这个动作，必须要用到用户类和帖子类，在这种情况下，要合理划分方法，尽可能与事件独立是一个能让代码最大限度复用的好办法。

具体实例：用户发帖后，可以给用户增加积分/经验/人气等，该代码逻辑应该写在帖子类内部还是用户类内部？如果写在帖子类内部，则用户回复帖子时也有相关操作，则需要重新再写一遍，显然出现了冗余。所以该逻辑应该写在用户类内部。复用原则之一：公用方法与事件分离。

##### 6.4.2.2 前端代码复用

###### 使用 CSS

HTML 的样式如果直接写在 html 文件中，同样的样式会写很多遍，显然属于无用工，可以把样式分离到 CSS 文件中，使用时调用。

具体实例：论坛是 Web 应用，则论坛的整个主题是确定的，因而颜色基调是确定的，很多界面的风格相似，完全可以使用独立的 CSS 文件表示网页样式，再结合 Servlet 动态生成每个页面的特殊格式，可以达到最大程度的样式复用。

###### 线程池

任何一个优秀的服务器(容器)必然要实现线程池，尽可能最大限度地复用线程，因为创建和销毁一个线程的代价太高了，这样能省去大量服务器资源。

证据：任何高并发服务器都是这么做的，比如 Apache Tomcat 和 Nginx。

### 6.4.3 可修改性

当需求发生变更或者环境发生变化时，系统是否可以快速得到调整，就体现出系统的可修改性。好的系统要求可以方便快速修改。

系统的任何部分都是可修改的，需要关注的是修改的成本。由此可见可修改性战术的目标是控制实现、测试和部署变更的时间和成本。

#### 6.4.3.1 策略分类

可修改性可以分为三大类：

(1) **局部化修改**——目标是减少由某个变更直接影响的模块的数量

1、维持语义的一致性。2、预期期望的变更。

(2) **防止连锁反应**——目标是限制对局部化的模块的修改，以防止对某个模块的修改间接地影响到其他模块

1、信息隐藏。2、维持现有的接口。添加接口。3、限制通信路径。4、使用仲裁者。

(3) **延迟绑定时间**——目标是控制部署时间并允许非开发人员进行修改

1、运行时注册。2、配置文件。3、多态。4、组件更换。5、遵守已定义的协议

其中，无论是局部化修改还是防止连锁反应，都是基于“高内聚，低耦合”思想。那么根据软件设计模式的设计原则来分析论坛系统：

(1) **局部性变更应用**：主要表现在类、函数、方法和接口的时候，实现“高内聚，低耦合”。

#### 6.4.3.2 项目实施

我们采用了 Struts+Spring+Hibernate 集成框架来进行系统的开发，集成 SSH 框架的系统从职责上分为四层：表示层、业务逻辑层、数据持久层和域模块层，以帮助开发人员在短期内搭建结构清晰、可复用性好、维护方便的 Web 应用程序。其中使用 Struts 作为系统的整体基础架构，负责 MVC 的分离，在 Struts 框架的模型部分，控制业务跳转，利用 Hibernate 框架对持久层提供支持，Spring 做管理，管理 struts 和 hibernate。

论坛系统开发时设计类、函数、接口的过程中，秉持着“高内聚，低耦合”的思想。

(2) **连锁反应定义**：我们平时编程，无论是写函数还是写类，都需要被其他类还是函数调用，修改此函数或类就会影响到调用他的函数，这就是连锁反应。

防止连锁反应应用：

1、**信息隐藏**。信息隐藏就是把某个实体的责任分解为更小的部分，并选择使哪些信息成为公有的，哪些信息成为私有的。

在编程时，我们写类时，会开设私有成员，公有成员等。保证了信息隐藏。

证据实例：AnnounceServiceImpl 具有私有成员变量，公有的成员函数。



public: 可以被所有其他类所访问; private: 只能被自己访问和修改

```
public class AnnounceServiceImpl implements AnnounceService {

    private AnnounceDao announceDao;
    private PageDao pageDao;

    public AnnounceDao getAnnounceDao() {
        return announceDao;
    }

    public void setAnnounceDao(AnnounceDao announceDao) {
        this.announceDao = announceDao;
    }

    public PageDao getPageDao() {
        return pageDao;
    }

    public void setPageDao(PageDao pageDao) {
        this.pageDao = pageDao;
    }
}
```

2、维持现有的接口。该战术的模式包括添加接口、添加适配器、提供一个占位程序。

Java 是面向对象的程序设计语言，所以我们经常要用到类，在编写程序时，我们采用了关键字 implements 实现接口，进行接口的继承：

关于接口的作用：接口是软件编程的规范，可以降低耦合性，换句话说，可以让某个模块或功能能够重复利用，这样只要写这功能一次代码就 ok 了。其他地方要用到的，全部用接口调用来实现。接口的优势：a . “接口+实现” 最常见的优势就是实现类和接口分离，在更换实现类的时候，不用更换接口功能。b . 这对于做单元测试也非常有帮助

证据实例：

创建接口 UserService：

```
public interface UserService {

    /**
     * 添加用户
     *
     * @param user
     * @return
     */
    public boolean add(Users user);

    /**
     * 禁用用户
     */
}
```

类 UserServiceImpl 继承了接口 UserService：

```
public class UserServiceImpl implements UserService {  
    private UserDao userDao;  
    private PageDao pageDao;  
    private TopicDao topicDao;  
    private CommentDao commentDao;  
    private NewDao newDao;  
    private GradeDao gradeDao;
```

3、限制通信路径。限制与一个给定的模块共享的模块，减少联系，一旦变更影响会小很多。

4、使用仲裁者。插入仲裁者来管理依赖之间的关系。数据库的使用，通过数据库来管理不同的数据信息。

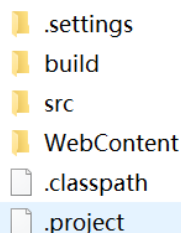
证据实例：论坛系统通过 mysql 来管理不同的数据信息

### (3) 推迟绑定时间

推迟绑定时间。将有可能的修改，尽量用配置文件，或者其他后期让非开发人员可调整的方式实现。

1、框架级可修改性：平台代码、应用代码、后台模块代码分离。

证据实例：我们采用了 Struts+Spring+Hibernate 集成框架来进行系统的开发，优点是论坛系统前后端的代码是分离的，框架可修改性提高。



- .settings
- build
- src
- WebContent
- .classpath
- .project

2、根据场景、调用后台模块出参入参可配原性修改，服务变更无需通过编码即可修改。

我们在做软件开发时，肯定会自觉的将需求抽象成软件的功能，这是毋庸置疑的，但是我们在开发前的架构设计阶段不单单要考虑软件的将要完成的功能，还要将软件的质量属性考虑进去，因为这将影响着一个软件的长期发展。

#### 6.4.4 开发难易

要开发一个论坛，需要设计如 html 的页面设计、java web 服务器的开发以及数据库的连接和操作等各方面的知识，而全面掌握开发一个论坛的所有知识对于我们当前阶段的知识储备来将太过困难，这无疑阻碍了我们工程的开发，因此如何将工程进行拆解、

分工，使组内成员可以各司其职，专注于补充自己负责模块的知识就成为了提高我们工程质量属性的一个关键点。

经过小组成员的调研和商讨，我们最终决定通过使用 SSH 框架来解决这个问题，以降低开发难度。集成 SSH 框架的系统从职责上分为四层：表示层、业务逻辑层、数据持久层和域模块层，以帮助开发人员在短期内搭建结构清晰、可复用性好、维护方便的 Web 应用程序。其中使用 Struts 作为系统的整体基础架构，负责 MVC 的分离，在 Struts 框架的模型部分，控制业务跳转，利用 Hibernate 框架对持久层提供支持，Spring 做管理，管理 struts 和 hibernate。

SSH 组合框架技术优势体现在四个方面：一、分离了 Java 代码和 HTML 代码，降低了对开发人员要求的复合度。二、是系统的层与层之间的工作相是对独立的，代码耦合度低。三即使脱离了 Spring 环境的 AOP 机制，也不会妨碍 AOP 实现业务的功能。四、与 SSH 配合使用的 Hibernate 等跨平台技术开源性极强促使了 SSH 框架的飞速发展。这无疑十分契合我们的工程，有利于缓解我们工程的燃眉之急。

而我们工程对于 SSH 框架的应用，体现在我们工程的架构和总体设计上，在此不再列出代码详解。

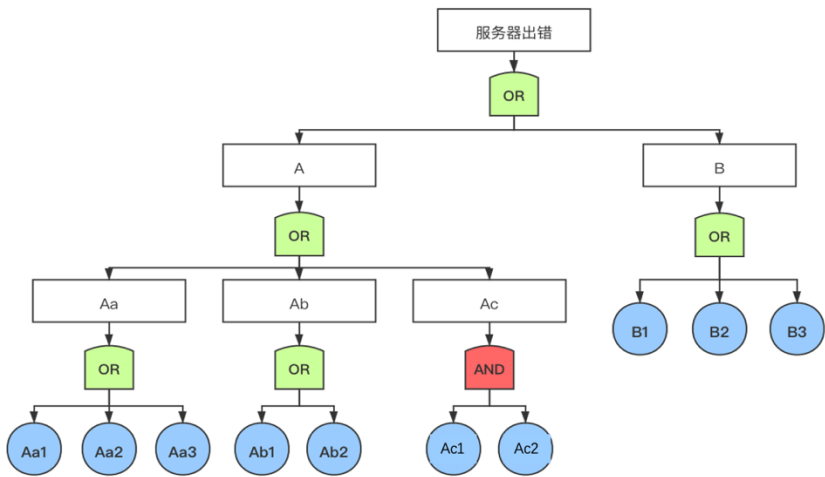
## 6.5 故障树分析

### 6.5.1 故障树 1:服务器出错

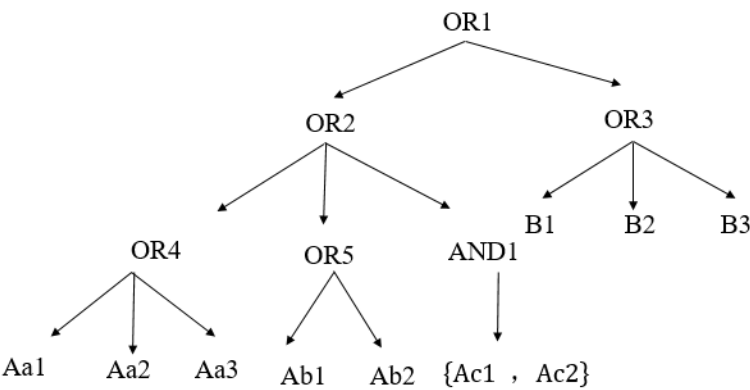
#### 6.5.1.1 符号说明

符号	含义
A	服务器无法返回正确的数据
Aa	服务器硬件宕机
Aa1	断电/意外关机
Aa2	服务器损坏
Aa3	电磁干扰
Ab	服务器系统出错
Ab1	服务器连接客户端过多，负载过大
Ab2	服务器返回错误数据
Ac	功能模块运行出错
Ac1	断言出错
Ac2	异常处理出错
B	服务器网络连接出错
B1	网络连接断开
B2	服务器网络连接端口被占用
B3	服务器网络服务超时

#### 6.5.1.2 图示



割集树



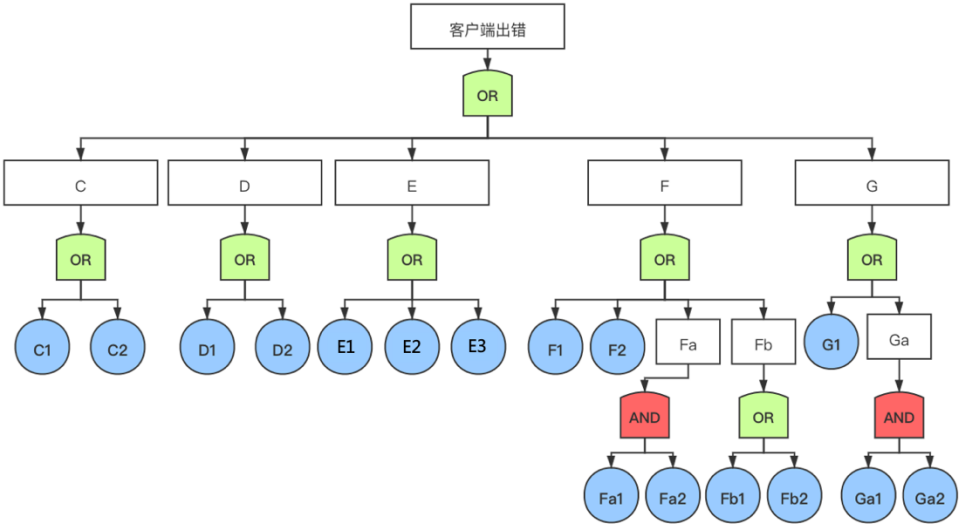
6.5.2 故障树 2：客户端出错

6.5.2.1 符号说明

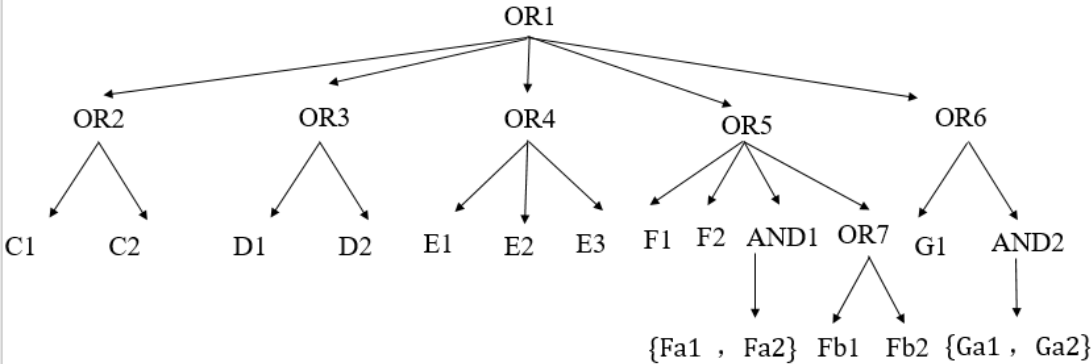
符号	含义
C	客户端未能成功发送数据到服务器
C1	客户端网络出错
C2	客户端突然退出致使数据传输中断
D	客户端无法接收服务器返回的数据
D1	客户端网络出错
D2	客户端负载多，内存不足
E	客户端无法正确显示服务器返回的数据
E1	客户端性能较差

E2	客户端浏览器内核版本过低，无法正常渲染
E3	客户端浏览器不支持论坛系统
F	客户端接收到访问错误
F1	填写表单中的数据存在问题
F2	服务器存在问题，无法提供服务
Fa	访问权限不足
Fa1	数据库存储的用户信息出错/丢失 and
Fa2	数据库检错失败
Fb	访问的数据出错
Fb1	访问的数据已经被删除
Fb2	访问的数据被相关政策屏蔽
G	客户端违反安全性的访问
G1	未识别以前的退出
Ga	不合法的访问
Ga1	密码泄露 and
Ga2	密码未改变

6. 5. 2. 2 图示



割集树

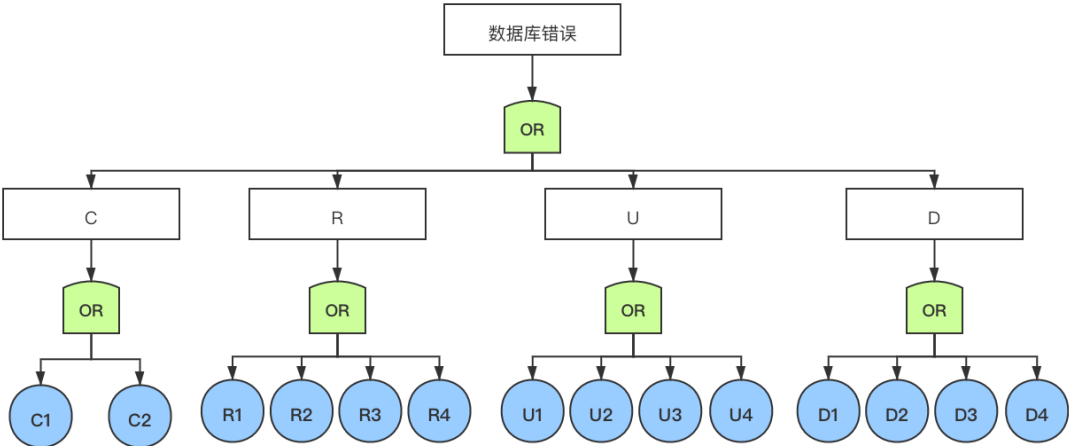


6.5.3 故障树 3 数据库错误

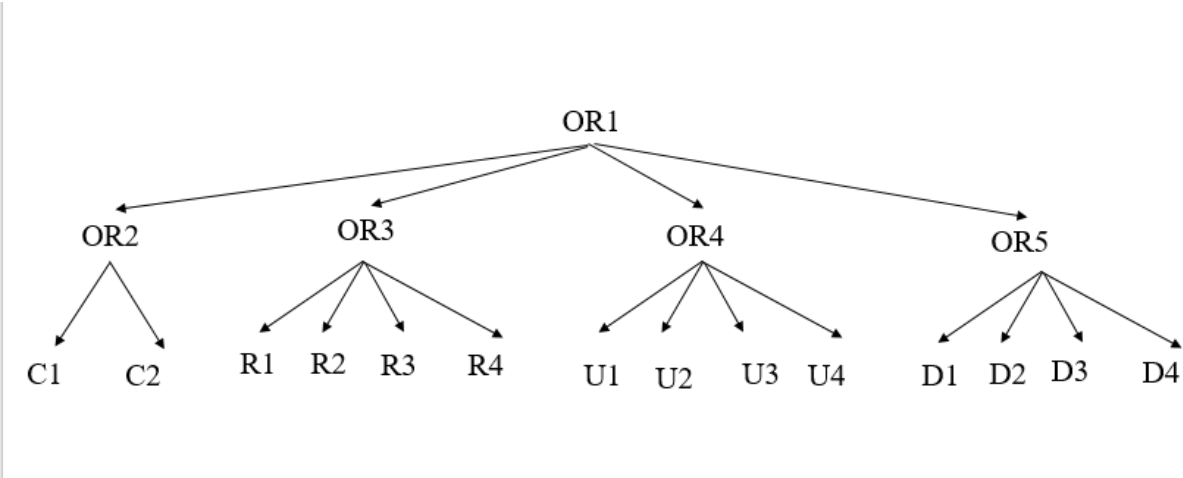
6.5.3.1 符号说明

符号	含义
C	“增”操作错误
C1	数据 id 重复
C2	数据格式/内容错误
R	“查”操作错误
R1	表不存在
R2	标识符不存在
R3	查操作指令存在中文编码字符
R4	缺失关键字及关键字条件错误
U	“改”操作错误
U1	数据权限不可修改
U2	找不到修改的数据
U3	修改数据格式不匹配
U4	修改触发事务死锁
D	“删”操作错误
D1	数据权限不可删除
D2	找不到修改的数据
D3	数据参与事务依赖，不可删除
D4	删除操作数据指代不明

6.5.3.2 图示



割集树



6.6 软件体系结构学习报告

6.6.1

软件体系结构：

软件架构设计：软件架构设计就是从宏观上说明一套软件系统的组成与特性，是一系列有层次地决策。

软件架构设计的方法：

- 1. 五视图法：（软件系统的不同的角色会站在不同的角度上提出的问题，我们就得从不同的视角来看待软件架构设计这项工作）
- 2. 五大视图：

视图	考虑的方面	工具	说明
逻辑架构	重点关注的是软件功能性需求。系统功能的划分、	结构图 UML 用例图 用例描述表	逻辑架构从系统用户的角度考虑问题，设计出的架构

	如何进行界面与用户的交互、设计接口等	鲁棒图 UML 类图等 (UML 已经成为对软件架构进行文档化的事实上的标准表示法。在视图文档的组织结构中, UML 主要用于表示元素或元素组的行为。)	能够满足业务逻辑的需求; 逻辑架构关注功能, 不仅包括用户可见的功能, 还包括为实现用户功能而必须提供的“辅助功能模块”
开发架构	重点关注的是开发编码实现方面的问题。 分层结构设计、开发技术选项、模块划分、开发规范、软件质量属性	绘图工具	从系统开发人员的角度来考虑, 设计的架构要易于理解、开发、单元测试等, 开发架构关注程序包, 不仅包括要编写的源程序, 还包括可以直接使用的第三方 SDK 和现场框架、类库, 以及开发的系统将运行于其上的系统软件或中间件。关注编译时刻的静态依赖关系。
数据结构	数据架构不仅仅要考虑开发中涉及到的数据库, 实体模型, 也要考虑物理架构中数据存储的设计。 数据如何存储、数据库表结构关系的设计、 如何设计实体、使用关系型数据库还是非关系型	Power Designer Visio	数据架构关注持久化数据的存储方案, 不仅包括实体及实体关系的存储格式、还包括数据传递, 数据复制, 数据同步等策略。
运行架构	运行架构关注的不再是全局而是局部, 着重关注那些		运行架构主要关注非功能需求;



	关键点与难点，常常需要技术攻关与预研。主要考虑控制流、通讯机制、资源争用、锁机制、同步异步、并发、串行，同时也要考虑质量属性。同步运行还是异步、并发还是串行、对象之间的交互、安全性、可靠性、性能等		运行架构关注进程、线程、对象等运行时概念，以及相关的并发，同步，通信等问题。运行架构关注运行期间各个单元的交互。
物理结构	物理架构主要考虑硬件选择和拓扑结构，软件到硬件的映射，软硬件的相互影响。网络方面、性能方面、部署方面		物理架构关注系统安装和部署在什么样的环境上。物理架构关注“目标程序及其依赖的运行库和系统软件”最终如何安装或部署到物理机器，以及如何部署机器和网络来配合软件系统的可靠性，可伸缩性等要求。

### 五视图法分析的意义：

- 1) 全面分析软件系统方方面面的问题
- 2) 尽早地发现和排除项目风险与不确定因素
- 3) 从不同角度去展现要设计的软件系统
- 4) 为项目进行中的干系人提供指导：
  - 逻辑架构描述系统功能，并指导系统测试
  - 开发架构规范软件的层次及代码风格
  - 数据架构指导数据库的设计
  - 运行架构定义了一些关键过程的设计
  - 物理架构明确软件如何部署与实施

### 经典的体系结构风格：

- 1) 数据流风格：批处理序列；管道/过滤器。

实例：编译器、

优点：

- 设计者可以将整个系统的输入、输出特性简单的理解为各个过滤器功能的合成。
- 基于管道-过滤器风格的系统具有较强的可维护性和可扩展性
- 支持一些特定的分析，如吞吐量计算和死锁检测等
- 管道-过滤器风格具有并发性

缺点：

- 交互式处理能力弱
- 具体实现比较复杂
  - 数据流同步问题
  - 数据加密与解析

## 2) 调用/返回风格：主程序/子程序；面向对象风格；层次结构。

实例：ISO、OSI 网络体系结构、

优点：

层次结构风格支持系统设计过程中的逐级抽象

- 基于层次结构风格的系统具有较好的可扩展性
- 层次结构风格支持软件复用

缺点：

- 并不是每个系统都可以很容易地划分为分层的模式
  - 很难找到一个合适的、正确的层次抽象方法。

## 3) 独立构件风格：进程通讯；事件系统。

## 4) 虚拟机风格：解释器；基于规则的系统。

## 5) 仓库风格：数据库系统；超文本系统；黑板系统。

## 6) 过程控制环路

## 7) C/S 风格

客户机/服务器（C/S）体系结构是基于资源不对等，且为实现共享而提出来的。

C/S 体系结构有三个主要组成部分：数据库服务器、客户应用程序和网络。

优点：

- 界面和操作可以很丰富
- 安全性高
- 响应速度快

缺点：

- 适用面窄
- 用户群固定

- 维护成本高

8) B/S 风格

9) B/S 体系结构主要利用不断成熟的 WWW 浏览器技术，特别是浏览器嵌入的多种脚本语言

优点：

- a) 维护和升级方式简单
- b) 交互性较强

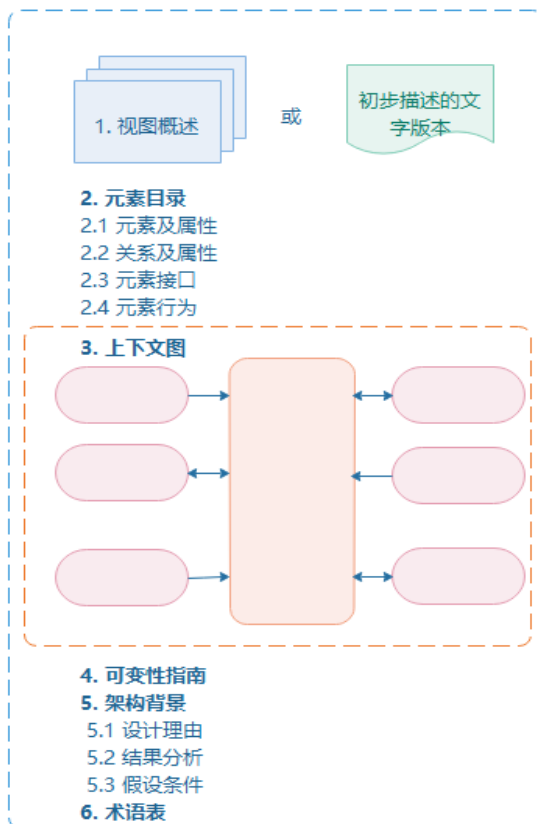
缺点：

- c) 在速度和安全性上需要花费巨大的设计成本
- d) 通常需要刷新页面（Ajax 等可以一定程度缓解该问题）
- e) 通信开销大

**软件架构文档化：**（阅读 Documenting Software Architectures ，2nd, Paul Clements  
<https://wiki.sei.cmu.edu/sad> 笔记）

目的：架构的主要用途是充当项目关系人之间进行交流的工具，文档则促进了这种交流——架构项目关系人希望从架构文档中获得自己所关心的架构信息。

视图编档：



## UML 建模

UML 通过图形化的表示机制从多个侧面对系统的分析和设计模型进行刻画。

10 种视图，四类：

1、用例图

2、静态图，包括 包图、类图、对象图。

类图的边表示类之间的联系，包括 继承、关联、依赖、聚合 等。

对象图描述在某种状态下或某一时间段，系统中 活跃的对象及其关系。

包由子包、类 组成。

3、行为图，包括 状态图、交互图、活动图，他们从不同的侧面刻画系统的动态行为。

交互图分为顺序图、合作图。顺序图强调对象之间 消息发送的时序。合作图更强调对象间的动态协作关系。

状态图描述对象的动态行为。

活动图描述操作序列，这些操作序列可以并发、同步，包含控制流、信息流。

4、实现图，包括 构件图、部署图。描述组成和分布情况。

部署图节点表示实际的计算机和设备，边表示节点之间的物理连接，也可以显示连接的类型及节点之间的依赖性。

## 系统架构的评估：

评估可以只针对一个体系结构，也可以针对一对一组体系结构。关注的是质量属性。

1、性能，是指系统的响应能力，多长时间对某个事件做出响应，或者某段时间内系统所能处理的事件的个数。

2、可靠性，是最重要的软件特性，平均失效等待时间 MTTF，平均失效间隔时间 MTBF

3、健壮性，不受错误使用和错误输入的影响。

4、可用性，正常运行的时间比例。经常用两次故障之间的时间长度或恢复正常的速度来表示。

5、安全性，阻止非授权用户。分为机密性、完整性、不可否认性、可控性 等特性。

6、可修改性，通过考察变更的代价衡量可修改性。

7.可扩展性，新特性来扩展软件系统，改进版本来替换构件并删除不需要的特性构件，需要松散耦合的构件。

8.可移植性。

### 主要评估方法：

1、SAAM 非功能质量属性的体系结构分析方法，是最早形式成文档并得到广泛使用的分析方法。最初它用于比较不同的软件体系结构，以分析 SA 的可修改性。

2、ATAM

#### 6.6.2

随着互联网和软件行业的发展，人们对软件的功能和性能需求越来越高，则软件的复杂性和开发难度也随之增加，则团队合作开发的大型软件成规模式增长。然而，数据表明，大系统软件开发中 70%的错误是由需求和软件设计阶段引入的;而且错误在系统中存在的时间越长则越难发现，解决这些错误的代价也越高。

伴随着这些问题的日益突出，整个软件系统结构的设计与规格说明便显得比在早期软件开发中占有重要地位的算法选择和计算问题的数据结构更为重要。由此便有了软件体系结构这一概念。

软件体系结构是软件需求和软件设计之间的一座桥梁，使得软件设计能够很真实地反映

并满足软件的需求，从而提高了软件需求和软件设计的质量。

我们往往比较熟悉的是二层 C/S 体系结构，即 Client/Server 结构，后台服务器负责管理数据，前台客户机负责与用户交互，但是该结构因为表现层和事务层都放在客户端，而数据逻辑层和数据存储层则放在服务端，这样的安排带来了很大限制，比如：①客户端很庞大，应用程序升级和维护时十分困难且耗资很大。对应用程序一个小小的改动，就必须通知或亲临每一个客户端去更新;新增或升级一台机器，都要把应用及其相关的文件安装在客户端上;②事务层不能与跨平台的客户端共享;③孤立了不同的逻辑组件;④没有统一的数据逻辑层来提供不同种类的数据存储层;⑤不支持 Internet 等，为了克服其中的一些缺陷，人们又提出了三层(多层)C/S 体系结构。

三层次客户机/服务器(C/S)结构是在常规客户机/服务器(C/S)结构上提出的，系统

在客户机和数据库服务器间添加一个应用服务器。三层 C/S 结构将应用功能分成表示层、功能层和数据层三个部分。表示层是应用的用户接口部分,它承担系统与用户间的对话功能,用于简单检查(不涉及有关业务本身的处理逻辑)用户输入的数据,显示应用的输出数据。在变更用户接口时,只需改写显示控制和数据检查程序,而不影响其他两层;功能层又称业务逻辑层,它将具体的业务处理逻辑编入程序中。表示层和功能层之间的数据传输要尽可能简洁。数据层就是数据库管理系统,负责管理对数据库数据的读写。

三层 C/S 的解决方案是:对这三层进行明确分割,并在逻辑上使其独立。原来的数据层作为数据库管理系统已经独立出来,所以,关键是要将表示层和功能层分离成各自独立的程序,并且还要使这两层间的接口简洁明了。

与传统的二层结构相比,三层 C/S 结构具有以下优点:①允许合理地划分三层结构的功能,使之在逻辑上保持相对独立性,从而使整个系统的逻辑结构更为清晰,能提高系统和软件的可维护性和可扩展性。②允许更灵活有效地选用相应的平台和硬件系统,使之在处理负荷能力上与处理特性上分别适应于结构清晰的三层。③应用的各层可以并行开发,各层也可以选择各自最适合的开发语言。④允许充分利用功能层有效地隔离开表示层与数据层,未授权的用户难以绕过功能层而利用数据库工具或黑客手段去非法地访问数据层,这就为严格的安全管理奠定了坚实的基础。

当然了,这和我们平常使用的各种编程框架很相似,写代码时,往往需要考虑自己在写哪一层的代码,从而确定自己的代码风格,比如 Web 前端往往写一些简单的业务逻辑,对数据库进行增删改查,不需要关心其他层次。从这样的角度看,知识往往都是相同的,特别是计算机领域。

### 6.6.3

软件体系结构是在软件系统的整体结构层次上关注软件系统的构建和组成,主要任务是研究软件系统是如何构建的、由哪些子系统或部件构成、这些子系统或部件之间的关系是什么。

当前对于软件体系结构没有统一明确的定义,但是从结构模式角度来说,软件体系结构是一个程序或系统的构件的组织结构,是它们之间的关联关系及其支配系统设计和演变的原则和方针。通俗地说,一个软件体系结构是由一组构件、连接件和它们之间的约束组成的,同时还包括系统需求和结构元素之间的对应关系。

在阅读了《Software Architecture in Practice》的第一部分后,我对软件的体系结构有了初步的认识。为什么在开发软件时,我们需要设计软件的体系结构?是为了每一个模块能够独立开发,然后有机且高效的结合在一起,而不是直接写成一个大的整体。例如我们要设计一个游戏,就要单独的引擎,单独的内容,单独美工,单独的音效等等。再细点就是一个引擎里要有单独的驱动模块、单独的描绘模块、单独的内容管理、单独的控制模块等等。而为了能够单独的开发各个子模块,就必须对整个系统进行分割、设计、然后才是一个个子模块的开发、测试。这些,都需要专精的人才进行独立的设计,然后再规划,统筹在一起,否则这会乱成一团,不利于开发、维护、调试等各种必要的操作。

我们通常通过视图(View)和结构(Structure)来讨论架构(Architecture),视图(View)是

一系列架构组件的描述,而结构(**Structure**)则是架构中的组件。架构通过各组件的性质,大体上可以分为以下三类:

**模块体系(Module Structures):** 内容为按功能分类后的架构组件。它负责描述组件的功能以及与其它组件的关系。模块体系包含: **分解(Decoposition):** 将模块进行分解,直至小到非常容易理解。子模块通常作为设计和开发的单位。**应用(Uses):** 能够将一个系统划分为不同的子集,以适应增量开发。**层(Layered):** 通常被设计用来隐藏低一层的实现细节,以带来可移植性。**类(Class or Generalization):** 类结构通过抽象和子类化,实现重用和功能的增强。

**运行体系(Component-and-Connector Structures):** 内容为运行时的架构组件以及与其它组件的交互组件。它负责描述运行时的主组件,公用的数据,处理中的数据流可并行的处理等等。执行体系包含: **处理与通信(Process or Communicating processes):** 描述处理或线程如何与通信处理整合。**并行(Concurrency):** 确定系统中的并行,定位可能出现的资源竞合。**数据共享(Shared data or Repository):** 适合用来描述针对共享数据的系统。显示在运行系统中数据如何被处理和消耗,能够被用来确保性能和数据完整性。**客户服务器(Client-server):** 适合用来描述一组项目交合的客户、服务器系统。

**部署体系(Allocation Structures):** 内容为开发或运行时的部署。部署体系包含: **部署(Deployment):** 执行(Implementatioin): 任务分配(Work assignment)。

除此之外,我还了解了对软件体系结构进行度量的一些指标,软件体系结构分析评价分为①定性分析:采用基于 **check list**, **questionnaire** 和场景的分析评价技术;②定量分析:采用基于度量指标、模拟、原型系统和数学模型等技术。在关注具体的质量特性时,在上述技术基础上,还需要特定的技术和方法支持。例如体系结构可靠性分析中的可靠性增长模型,安全性分析中的失效模式和效果分析(**failure modes and effect s analysis**, **FMEA**),以及性能分析中排队理论和调度理论的应用等。其中,**SAAM (Software Architecture Analysis Method)**方法 是第一个被广泛接受的体系结构分析评价方法,适用于可修改性、可拓展性以及功能覆盖等质量属性。**SAAM** 方法有着以下缺陷:没有提供体系结构质量属性的清晰的度量;评估过程依赖专家经验等,只适合对体系结构的粗糙评价。

计算机在各行各业的广泛使用使计算机软件的复杂性大幅度提高,随着复杂性的不断提高,很多问题接踵而至,软件体系结构的选择也就相当重要,同时对软件的分析评测就变得非常必要。所以,软件体系结构需要不断地优化、改进,对应的各种分析评测标准也要不断地完善。软件体系结构及其分析评测标准将会越来越受到广大工程人员的重视和研究,在

未来的软件开发和管理中会扮演极为重要的角色。

软件体系结构的作用犹如建筑工程中的“施工图纸”,或者称之为“蓝图”。没有图纸,不能开工。同样地,没有软件体系结构,就无法构造复杂的软件系统。目前,软件体系结构的设计在软件开发过程中扮演着越来越重要的地位,可以说选择一个适合自己的软件开发工程的体系结构是成功的一半,因此我们需要给予软件体系结构更多的关注。

#### 6.6.4

阅读章节：

《Software Architecture in Practice》--CHAPTER15：architecture in agile Projects

阅读内容目录：

15.1 How Much Architecture?

15.2 Agility and Architecture Methods

15.3 A Brief Example of Agile Architecting

15.4 Guidelines for the Agile Architect

15.5 Summary

15.6 For Further Reading

15.7 Discussion Questions

阅读此段原因：

由于在团队中主要负责敏捷开发的策略制定、时间规划和团队规划管理工作，需要输出敏捷开发相关的文档和安排图标，因此学习敏捷开发中的软件架构能够比较好的提升自己负责的工作质量，同时加深对敏捷开发的理解。

15.1 How Much Architecture?

重点一：敏捷开发的宣言

1. 我们的首要任务是通过尽早并持续交付有价值的软件来满足客户的需求。
2. 即使在开发后期，也欢迎不断变化的需求。敏捷流程利用变更来获得客户的竞争优势。
3. 频繁交付工作软件，从几周到几个月不等，而更倾向于较短的时间范围。
4. 在整个项目中，业务人员和开发人员必须每天一起工作。
5. 围绕有上进心的个人建立项目。给他们提供所需的环境和支持，并信任他们来完成工作。



6. 向开发团队内部传达信息的最有效方法是面对面的交谈。
7. 工作软件是进度的主要衡量标准。
8. 敏捷过程促进可持续发展。赞助者，开发者和用户应该能够无限期地保持恒定的步伐。
9. 持续关注技术卓越性和良好的设计可增强敏捷性。
10. 简洁性(最大化未完成工作量的艺术)是必不可少的。
11. 最好的体系结构，需求和设计来自自组织团队。
12. 团队会定期思考如何提高效率，然后相应地调整和调整其行为。

讨论：敏捷开发还是 BDUF (大前端设计) 的设计流程

在不同的工程项目中，应该采用截然不同的规划、策划、设计、实践策略。

- 系统性极强、对质量和系统的统一性、决策的安全性具有较高要求的工程项目，更加适合使用 BDUF (大前端设计) 的设计流程
- 质量要求平常、需要控制成本、需要快速启动的工程项目，更加适合于使用敏捷开发的流程

作者使用一个造摩天大楼和家里小院的例子形象地描述出了两者在选择上的差距点。

作者通过项目工程情况和返工率，得到了如下的结论：

成功的项目显然需要两种方法的成功融合。过多的前期计划和承诺会扼杀创造力和适应不断变化的需求的能力，敏捷度过高可能会造成混乱。没有人愿意在拥有不经过严格的计划和彻底的分析的飞行控制软件的飞机上飞行。同样，没人会花 18 个月的时间 为他们的最新手机型号，视频游戏或口红计划电子商务网站。

我们所有人都想要的就是最佳位置 - 乔治·费尔班克斯(George Fairbanks)所说的“足够的建筑”。这不仅是完成正确数量的架构工作的问题，而且还在于正确的时间完成它的问题。敏捷项目倾向于希望根据需要实时地发展体系结构，而大型软件项目传统上则倾向于进行大量的前期分析和规划。

## 15.2 Agility and Architecture Methods

作者提出：用于体系结构文档的方法称为 Views and Beyond

Views and Beyond 与 Agile 在以下方面着重达成共识:如果不需要信息,请不要花费资源来对其进行文档化。所有文档都应有预定用途和目标读者,并应以能够为两者提供服务的方式进行制作。

技术文档的基本原则之一是“为读者写信”。这意味着了解谁会阅读文档以及他们将如何使用它。如果没有听众,则无需制作文档。该原理在敏捷方法 中是如此重要,以至于它被赋予了自己的名字:YAGNI。YAGNI 的意思是“您将不需要它”,它指的是只有在确实有需要时才实施或记录某些东西的想法。不要花时间尝试预期所有可能的需求。

### 15.3 A Brief Example of Agile Architecting

作者以一个例子:WEB 会议系统对上述的原理进行了阐释

Web 会议系统是复杂而苛刻的系统。它们必须提供实时响应能力,竞争功能,易于安装和使用,轻巧的占地面积等。例如:

- 他们必须在各种硬件和软件平台上工作,其细节不受架构师的控制。
- 它们必须可靠并且提供低延迟的响应时间,尤其是对于实时功能,例如 IP 语音 (VoIP)和屏幕共享。
- 它们必须提供高安全性,但是必须在未知的网络拓扑以及一组未知的防火墙和防火墙策略中提供高安全性。
- 必须轻松修改它们,并轻松将它们集成到各种环境和应用程序中。
- 它们必须具有很高的可用性,并且必须由具有广泛 IT 技能的用户轻松安装和学习。

存在的问题:

#### 1、功能的矛盾和兼容性:

许多目标相互取舍。通常,安全性(以加密形式)是以实时性能(延迟)为代价的。可修改性是以缩短上市时间为代价的。可用性和性能通常以可修改性和成本为代价。

即使有可能收集,分析并确定所有相关数据,功能需求和质量属性需求的优先级,但在竞争激烈的市场(例如网络会议)中普遍存在的严格的上市时间限制将使我们无法做到这一点。试图支持所有可能的用途非常棘手,并且用户本身也不足以预见该系统的所有

可能用途。因此，仅以传统的需求启发方式询问用户他们想要什么，是不可能的。

## 2、对客户需求快速响应与产品的长期生存盈利

经典的“敏捷与承诺”问题：一方面，架构师希望快速提供新功能，并快速响应客户需求。另一方面，系统和公司的长期生存意味着必须针对扩展性，可修改性和可移植性进行设计。这可以通过基于少量常规应用的模式和策略为体系结构提供简单的概念模型来最好地实现。尚不清楚我们如何将我们的方式“演化”为这种架构。

一种比较好的架构方案，同时也是针对这种矛盾的解决方案：

使用两种方式进行问题的思考：

■ 自上而下 - 设计和分析建筑结构，以满足苛刻的质量属性要求和权衡

■ 自下而上 - 分析各种特定于实现的约束和特定于环境的约束，并为它们制定解决方案

对于给出的问题，作者提供了一种思考流程和规划思路：

- 1) 首先，WebArrow 团队快速创建并粗略地分析了初始软件和系统体系结构概念，然后从可以向客户展示的最关键功能开始，逐步实现并充实了它。
- 2) 每当出现新的要求或对问题域有了更好的了解时，他们都会调整体系结构并重构设计和代码。
- 3) 随着产品的发展，连续实验，经验评估和架构分析可帮助确定架构决策。

这里可以获得的经验是：

使体系结构流程敏捷并不需要彻底地重新发明敏捷实践或体系结构方法。WebArrow 团队对实验的重视是关键因素。这是我们实现敏捷的架构概念、实施和评估形式的方式。

### 15.4 Guidelines for the Agile Architect

Barry Boehm 及其同事开发了增量承诺模型，这是一个混合过程模型框架，旨在敏捷性和承诺之间找到平衡。该模型基于以下六个原理：

- A. 对成功至关重要的利益相关者的承诺和责任
- B. 利益相关者基于成功的谈判和权衡而“满意”（达到可接受的阈值）

- C. 系统定义和利益相关者承诺的增量和演进增长
- D. 迭代系统的开发与定义
- E. 交错的系统定义和开发，可以尽早部署核心功能，不断进行更改以适应变化，以及及时增长复杂系统，而不必等待定义每个需求和子系统的情况
- F. 风险管理 - 风险驱动的锚点里程碑，这是同步和稳定所有这些并发活动的关键

Booch 还指出：有效的敏捷过程将使体系结构随着系统的发展和成熟而逐步增长。成功的关键是要具有可分解性，关注点分离以及各部分几乎独立。

作者针对敏捷开发的系统架构给出的建议：

- A. 如果您要构建具有相对稳定且易于理解的要求的大型复杂系统，则最好预先进行大量架构工作（“大”的一些示例值请参见图 15.1）。
- B. 在需求含糊不清或不稳定的大型项目中，即使它只是“PowerPoint 架构”，即使它遗漏了许多细节，即使您只花了几天时间，也要快速设计一个完整的候选架构。根据情况，在执行峰值和实验时以及在功能和质量属性要求出现并巩固的情况下，更改和详细说明此体系结构。这种早期的体系结构将有助于指导开发，帮助早期的问题理解和分析，帮助确定需求，帮助团队进行协调，
- C. 在需求不确定的小型项目中，至少要设法就要采用的中心模式达成共识，但不要在前期的建设，文档编制或分析上花费过多的时间。

### 6.6.5

#### 基本概念

体系结构的设计就是设计组件与组件之间的联系。软件体系结构是具有一定形式的结构化元素，即构件的集合，包括处理构件、数据构件和连接构件。处理构件负责对数据进行加工，数据构件是被加工的信息，连接构件把体系结构的不同部分组合连接起来。对于软件体系结构没有统一明确的定义，但是从结构模式角度来说，软件体系结构是一个程序或，系统的构件的组织结构，是它们之间的关联关系及其支配系统设计和演变的原则和方针。一个软件体系结构是由一组构件、连接件和它们之间的约束组成的，同时还包括系统需求和结构元素之间的对应关系。连接件是构件的黏合剂，它也是一组对象。它把不同的构件连接起来，形成体系结构的一部分，一般表现为框架式对象或转换式对象(调用远程构件资源)。约束一般为对象连接时的规则，或指明构件连接的势态和条件。例如，上层构件可要求下层构件的服务，反之不行；两个对象不得递归

地发消息；代码复制迁移的一致性约束；以及在什么条件下此种连接 无效 等等。 可以将软件体系结构抽象地描述如下：软件体系结构 = 构件 + 连接件 + 约束。

### 几种常见的软件体系结构及特点

#### 管道和过滤器

把系统任务分成为几个独立的功能模块。这些功能模块采用通过系统的数据流连接。在管道-过滤器风格下，每个功能模块都有一组输入和输出。功能模块称作过滤器。功能模块间的连接可以看作输入、输出数据流之间的通路，所以称作管道。管道-过滤器风格的特性之一在于过滤器的相对独立性，即过滤器独立完成自身功能，相互之间无需进行状态交互。

优点：设计者可以将整个系统的输入、输出特性简单的理解为各个过滤器功能的合成。基于管道-过滤器风格的系统具有较强的可维护性和可扩展性。支持一些特定的分析，如吞吐量计算和死锁检测等。管道-过滤器风格具有并发性。

缺点：管道和过滤器常常导致批处理方式；在维护两个分离但相关的数据流时，有一定的困难；可能对数据传输标准有一定的要求，这就导致了为每一个过滤器进行数据语法分析和编码的额外工作，同时，在编写过滤器时增加了复杂性，降低了性能。

#### 层次结构：

分层结构采取层次化的组织方法，每一层向上一层提供服务，并利用下一层的服务。比较常见的有 TCP/ IP 协议，OSI 七层结构等。在 OSI 形成的过程中，把通信协议有关的不同任务交给不同的层来完成，使每一层的函数既得到良好的定义又与其他层分开。服务的定义是从底层向顶层形成的。

优点：层次结构风格支持系统设计过程中的逐级抽象。基于层次结构风格的系统具有较好的可扩展性。层次结构风格支持软件复用。

缺点：并不是每个系统都可以很容易地划分为分层的模式很难找到一个合适的、正确的层次抽象方法。

#### C/S 风格

客户机/服务器（C/S）体系结构是基于资源不对等，且为实现共享而提出来的。C/S 体系结构有三个主要组成部：数据库服务器、客户应用程序和网络。

优点：界面和操作可以很丰富，安全性高，响应速度快。

缺点：适用面窄，用户群固定，维护成本高。

#### B/S 风格

B/S 体系结构主要利用不断成熟的 WWW 浏览器技术，特别是浏览器嵌入的多种脚本语言。B/S 体系结构有三个主要组成部分：浏览器，Web 服务器，数据库服务器。

优点：维护和升级方式简单，交互性较强。

缺点：在速度和安全性上需要花费巨大的设计成本，通常需要刷新页面（Ajax 等可以一定程度缓解该问题），通信开销大。

面向对象 面向对象的程序设计其实是一种观念，主要是完成对象的封装、继承、多态等特性。在这种结构中，数据的表示和与之相连的属性的基本构造都被封装在一个抽象的数据类型或对象中。这种结构的部件就是对象，或者说是抽象数据类型的实例。

对象之间通过函数和过程调用发生相互作用。这里所说的对象就是 C++, JAVA 中的类,都是封装了特定功能的属性和事件。

优点: 对象的隐藏并不影响它的使用者,更改它的实现,不会对系统造成任何危害。将它们所操纵的数据与访问这些数据的操作绑在一起,是设计者易于将问题分解成相关对象之间的相互作用。

缺点: 为了实现某个对象与其他对象相互作用,必须知道对象的标志。这与管道形成了鲜明的对比;会产生连锁反应。比如 A 调 B, C 调 B,那么 A 对 B 的调用可能会对 C 造成影响

#### 6.6.6

从机器语言、汇编语言、过程式程序设计语言、面向对象程序设计语言、形式化(半形式化)规格说明语言(如体系结构描述语言)发展过程中,可以发现:计算机语言越来越适合于开发人员的思维活动模型,代码复用的级别也在不断地提升。体系结构技术的研究,使软件复用从代码复用发展到设计复用和过程复用。SA 是软件体系架构 (Software Architecture) 的缩写。作为软件需求与软件设计间的桥梁,D.E.Perry 将软件体系结构视为软件开发中第 1 类重要的设计对象。而 BarryBoehm 明确指出:“在没有设计出体系结构及其规则时,那么整个项目不能继续下去,而且体系结构应该看做是软件开发中可交付的中间产品。由此可见,体系结构在软件开发中为不同的人员提供了共同交流的语言,体现并尝试了系统早期的设计决策,并作为系统设计的抽象,为实现框架和构件的共享与复用、基于体系结构的软件开发提供了有力的支持

#### 软件体系结构的发展史:(软件体系结构综述 软件学报)

软件系统的规模在迅速增大的同时,软件开发方法也经历了一系列的变革。在此过程中,软件体系结构也由最初模糊的概念发展到一个渐趋成熟的技术。

20 世纪 70 年代以前,尤其是在以 ALGOL 68 为代表的高级语言出现以前,软件开发基本上都是汇编程序设计,此阶段系统规模较小,很少明确考虑系统结构,一般不存在系统建模工作。70 年代中后期,由于结构化开发方法的出现与广泛应用,软件开发中出现了概要设计与详细设计,而且主要任务是数据流设计与控制流设计。因此,此时软件结构已作为一个明确的概念出现在系统的开发中。

20 世纪 80 年代初到 90 年代中期,是面向对象开发方法兴起与成熟阶段。由于对象是数据与基于数据之上操作的封装,因而在面向对象开发方法下,数据流设计与控制流设计则统一为对象建模,同时,面向对象方法还提出了一些其他的结构视图。如在 OMT 方法中提出了功能视图、对象视图与动态视图(包括状态图和事件追踪图);而 BOOCH 方法中则提出了类视图、对象视图、状态迁移图、交互作用图、模块图、进程图;而 1997 年出现的统一建模语言 UML 则从功能模型(用例视图)、静态模型(包括类图、对象图、构件图、包图)、动态模型(协作图、顺序图、状态图和活动图)、配置模型(配置图)描述应用系统的结构。

90 年代以后则是基于构件的软件开发阶段,该阶段以过程为中心,强调软件开发采用构件化技术和体系结构技术,要求开发出的软件具备很强的自适应性、互操作性、可扩展性和可重用性。此阶段中,软件体系结构已经作为一个明确的文档和中间产品存在于软件

开发过程中,同时,软件体系结构作为一门学科逐渐得到人们的重视,并成为软件工程领域的研究热点,因而 Perry 和 Wolf 认为,“未来的年代将是研究软件体系结构的时代!”

纵观软件体系结构技术发展过程,从最初的“无结构”设计到现行的基于体系结构软件开发,可以认为经历了 4 个阶段:(1)“无体系结构”设计阶段:以汇编语言进行小规模应用程序开发为特征;(2)萌芽阶段:出现了程序结构设计主题,以控制流图和数据流图构成软件结构为特征;(3)初期阶段:出现了从不同侧面描述系统的结构模型,以 UMI 为典型代表;(4)高级阶段:以描述系统的高层抽象结构为中心,不关心具体的建模细节,划分了体系结构模型与传统的软件结构的界限,该阶段以 Kruchten 提出的“4+1”模型为标志。由于概念尚不统一、描述规范也不能达成一致认识、在软件开发实践中软件体系结构尚不能发挥重要作用。

### 体系结构的主要研究方向:

在基于构件和体系结构的软件开发方法下,程序开发模式也相应地发生了根本变化。软件开发不再是“算法+数据结构”,而是“构件开发+基于体系结构的构件组装”。那么,如何描述、表示、设计体系结构?如何验证一个体系结构是否符合期望的系统需求?如何基于体系结构开发实例化的软件系统?将是基于构件和体系结构的软件开发方法所必须研究和解决的核心问题。归纳现有体系结构的研究活动,主要包括如下几个方面:

- (1)体系结构理论模型的研究;
- (2)体系结构描述研究,主要研究体系结构描述语言及其支持环境、体系结构描述规范;
- (3)体系结构设计研究:包括体系结构设计方法、体系结构风格、体系结构设计空间等内容;
- (4)体系结构分析与验证:研究如何将软件的非功能特性转化为体系结构的需求,如何分析体系结构满足期望的需求的属性,对体系结构的语法、语义、类型失配等进行检查与验证的研究;
- (5)体系结构演化与复用研究:研究产品线中软件体系结构演进的理论与方法,从已有文档、系统设计和代码中逆向提取软件体系结构、体系结构复用等;
- (6)动态体系结构研究:研究软件系统由于特殊需要必须在连续运营情况下的体系结构变化与支撑平台;
- (7)基于体系结构的软件开发:研究引入体系结构后的软件开发过程、基于体系结构开发与中间技术集成、基于体系结构的程序框架自动生成技术等。目前,软件体系结构研究活动主要集中在软件体系结构描述语言 ADL,体系结构描述构造与表示,软件体系结构分析、设计与验证,软件体系结构发现、演化与复用等几个领域