**软件架构原型文档**

**团队名称：Anti-Gay-Fri 第26组**

**团队成员：PM:姜殊 成员：陈元杰 耿昱翔 金炜 莫其凡**

|  |  |  |
| --- | --- | --- |
| 变更时间 | 变更说明 | 变更人 |
| 2012/4/15 | **文档编写** | **耿昱翔** |
|  |  |  |
|  |  |  |

目录

[1 系统架构设计问题描述 2](#_Toc322297719)

[1.1 问题清单 2](#_Toc322297720)

[1.2 问题的具体描述 2](#_Toc322297721)

[2 解决方案的描述及相关约束条件 5](#_Toc322297722)

[2.1 解决方案描述 5](#_Toc322297723)

[2.2 相关约束条件 6](#_Toc322297724)

[3 各方案风险分析 9](#_Toc322297725)

[4 系统涉众描述 11](#_Toc322297726)

[5 系统架构描述 15](#_Toc322297727)

# 系统架构设计问题描述

## 问题清单

* S1: 代码设计采用哪些设计模式
* S2: 网络部分是分散在各个功能的包里，还是放在一个单独的包里
* S3: 如何布局文件的目录系统
* S4: 服务器采用何种设计结构
* S5: 如何登入和验证
* S6: 系统如何划分模块
* S7: 如何查看会议记录

## 问题的具体描述

* S1：代码设计采用哪些设计模式

|  |  |
| --- | --- |
| 要素 | 内容 |
| ID | S1 |
| 提出者 |  |
| 问题的描述 | 代码设计采用哪些设计模式 |
| 问题的影响 | 代码设计不具备良好的编程风格，层次不清，难以阅读和修改，代码之间的关系错综复杂，移植性比较差 |
| 解决问题目标 | 为编程选择合理的设计模式，在以后的代码开发中遵守这些设计模式，以设计出有层次，结构清晰的代码 |

* S2：网络部分是分散在各个功能的包里，还是放在一个单独的包里

|  |  |
| --- | --- |
| 要素 | 内容 |
| ID | S2 |
| 提出者 |  |
| 问题的描述 | 网络部分是分散在各个功能的包里，还是放在一个单独的包里 |
| 问题的影响 | 服务器端与客户端之间无法建立良好的连接，信息无法迅速转发，各个功能的网络部分造成相互干扰或是协作时，出现干涉等问题 |
| 解决问题目标 | 确立良好的代码结构，设计出良好的网络部分代码，确保服务器与客户端能够进行良好的通信，保证软件的正常运行 |

* S3：如何布局文件的目录系统

|  |  |
| --- | --- |
| 要素 | 内容 |
| ID | S3 |
| 提出者 |  |
| 问题的描述 | 如何布局文件的目录系统 |
| 问题的影响 | 没有良好的具有一定层次的文件目录系统，无法保证代码的层次结构，造成各个功能间的代码相互干涉，逻辑混乱，难以打动代码 |
| 解决问题目标 | 确立良好的文件结构，使代码层次分明。确保阅读代码时能够清晰明了的读懂，逻辑清晰 |

* S4：服务器采用何种设计结构

|  |  |
| --- | --- |
| 要素 | 内容 |
| ID | S4 |
| 提出者 |  |
| 问题的描述 | 服务器采用何种设计结构 |
| 问题的影响 | 会议时，服务器端直接和用户端通信，如果服务器缺乏良好的设计结构，可能直接导致服务器端和客户端连接或通信的异常，也可能导致数据交换的流畅程度较低 |
| 解决问题目标 | 确立良好的服务器端代码设计结构，使得服务器断和客户端之间通信良好，数据交流比较流畅，确保会议良好的进行下去 |

* S5：如何登入和验证

|  |  |
| --- | --- |
| 要素 | 内容 |
| ID | S5 |
| 提出者 |  |
| 问题的描述 | 如何登入和验证 |
| 问题的影响 | 用户无法正常通过信息验证登入，客户端持有太多的用户信息，导致了信息流失或是被窃的风险增大，使得代码的设计比较差 |
| 解决问题目标 | 保证使用该软件的用户及时登入系统，系统提供安全性保障，客户端在持有恰当的信息情况下保证用户能够正常登入 |

* S6：系统如何划分模块

|  |  |
| --- | --- |
| 要素 | 内容 |
| ID | S6 |
| 提出者 |  |
| 问题的描述 | 系统如何划分模块 |
| 问题的影响 | 用户在使用正常功能时，各个功能键可能相互干涉，模块的划分直接导致了代码编写的容易程度，并且影响各个模块间的耦合与内聚程度 |
| 解决问题目标 | 将系统划分为不同的模块，具有一定层次，结构，相互联系，统一，降低不同模块间代码的耦合程度，并且使得编码比较容易进行 |

* S7：如何查看会议记录

|  |  |
| --- | --- |
| 要素 | 内容 |
| ID | S7 |
| 提出者 |  |
| 问题的描述 | 如何查看会议记录 |
| 问题的影响 | 开完会议后由于记忆的有限性，无法后期对会议进行重新评审，无法借鉴往次会议的内容，并且可能导致一些重要的会议内容流失 |
| 解决问题目标 | 方便与会人员查看历史会议记录，以做出相应的设计调整，可以直接审视或查阅会议内容 |

# 解决方案的描述及相关约束条件

## 解决方案描述

* S1：代码设计采用哪些设计模式

方案1：采用MVC模式，大家运用不熟练。

方案2：采用分层结构。代码结构分为视图层、逻辑层和数据层，客户层持有视图层和逻辑层，服务器端持有数据层。

方案3：服务器与客户端建立通信时，编码采用观察者模式，将观察者和被观察的对象分离开，服务器端作为观察者查看客户端的连接请求，并且做出相应的响应

* S2: 网络部分是分散在各个功能的包里，还是放在一个单独的包里

方案1：将网络部分是分散在各个功能的包里

方案2：将网络部分放在一个单独的包里

* S3: 如何布局文件的目录系统

方案1：第一层文件以会议小组名字划分，第二层文件以不同小组的具体会议划分，第三层文件以功能划分

* S4：服务器采用何种设计结构

方案1：服务器端采用一个controller，用以监听控制与会人员发起会议的请求，并创建一个新的聊天room

方案2：分散控制风格。

* S5：如何登入和验证

方案1：客户端将与会人员的账号和密码信息传给服务器端，由服务器端识别和判定。

方案2：客户端本地验证。

* S6：系统如何划分模块

方案1：以功能划分，每个具体功能对应一个具体的代码块

方案2：以数据流划分，不同的数据处理流程成为一个模块。

* S7：如何查看会议记录

方案1：会议记录保存在服务器端，客户端下载查看。

## 相关约束条件

* S1：代码设计采用哪些设计模式
* 方案1：

约束1：客户端的所有的数据应该是从服务器获得的

约束2：服务器通过广播的方式将数据传送给各个客户端

约束3：它强制性的使应用程序的输入、处理和输出分开。应用程序被分成三个核心部件：模型、视图、控制器。它们各自处理自己的任务。

* 方案2：

约束1： 服务器端作为观察者，客户端作为被观察对象

约束2： 观察者将自己注册到被观察者的容器中时，被观察者不应该过问观察者的具体类型，而是应该使用观察者的接口

* S2：网络部分是分散在各个功能的包里，还是放在一个单独的包里
* 方案1：

约束1：每个功能具有独自的网络部分代码

约束2：连接服务器时，需要多线程

* 方案2：

约束1：不同功能的网络部分结合在一起

约束2：多个功能模块同时访问该包

* S3：如何布局文件的目录系统
* 方案1：

约束1：不同小组的每次会议都对应相应的会议名字

约束2：消息记录放在第三层文件中

* S4：服务器采用何种设计结构
* 方案1：

约束1： 会议发起人员发起会议需先向服务器端controller发送请求

* 方案2：

约束1：会议前需明确与会人员

* S5：如何登入和验证
* 方案1：

约束1：服务器端仅传送账号和密码信息，其他与用户成员相关的信息保留

* 方案2：

约束1：客户端持有和服务器端相同的对象，服务器端和客户端持有客户信息。

约束2：客户端需拥有专门文件保存用户信息

* S6：系统如何划分模块
* 方案1：

约束1 ：不同功能部分的代码之间相互独立

约束2：各个具体功能在使用时，可以共存

约束3：需降低模块之间的耦合度

* S7：如何查看会议记录
* 方案1：

约束1：同时访问，避免冲突。

# 各方案风险分析

* S1：代码设计采用哪些设计模式
* 方案1：

风险1： 需要花费相当可观的时间去考虑如何将MVC运用到应用程序，同时由于模型和视图要严格的分离，这样也给调试应用程序带来了一定的困难，每个构件在使用之前都需要经过彻底的测试

风险2：使用MVC同时也意味着你将要管理比以前更多的文件

* 方案2：

风险1：可能有些部件，鉴于两层之间，难以划分。

* 方案3：

风险1： 观察者和被观察对象之间的互动关系不能体现成类之间的直接调用，否则就将使观察者和被观察对象之间紧密的耦合起来

风险2：java的观察者模式固有的缺陷，被观察者不能继承其他子类。

* S2：网络部分是分散在各个功能的包里，还是放在一个单独的包里
* 方案1：

风险1： 网络部分之间相互独立，使得整体的耦合度很高

风险2：不方便不同功能间键代码的重用

风险3：代码整合时，可能会出现由于代码风格的不同而导致的诸多bug

* 方案2：

风险1：不利于分开编程，网络部分需一个人负责，否 则代码结构，风格不一

风险2：功能与代码没有实现一一对应，相互分离开来

* S3：如何布局文件的目录系统
* 方案1：

风险1：保存文件时，不同模块之间保存的文件可能相互干涉

* S4：服务器采用何种设计结构
* 方案1：

风险1：参与会议人员的自由性比较低，因为一上线就 会默认已经进入聊天模式

* 方案2：

风险1：发起议会人员的控制权降低

* S5：如何登入和验证
* 方案1：

风险1：客户端只保留用户的部分信息，不方便实现不同与会人员之间的沟通交流

* 方案2：

风险1：客户端持有与会人员对象，造成系统的安全性降低

风险2：增加模块之间的耦合度

* S6：系统如何划分模块
* 方案1：

风险1 ：不同功能之间可能产生些许影响

* S7：如何查看会议记录
* 方案1：

风险1：不同功能之间可能产生些许影响

# 系统涉众描述

* 项目经理

|  |  |
| --- | --- |
| 代表 | 项目经理 |
| 说明 | 项目的重要管理人员之一 |
| 职责 | 负责整个团队的开发管理工作，监督团队进度 |
| 成功标准 | 项目在可接受的时间内，最终完成 |
| 参与方式 | 1. 管理项目资源。  2. 负责对外的联系事务。  3. 制定项目计划，保证计划实施。  4. 负责监督团队人员的工作，保证项目进度。  5. 协调团队队员之间的交流。  6. 管理文档资源，控制项目变更。 |
| 可交付工件 | 完整的可以成功运行的网络会议软件系统。 |
| 意见/问题 | 1. 系统要根据需求开发。  2. 获取必要的开发资源。  3. 项目要在可接受的时间内完成。 |

* 用户

|  |  |
| --- | --- |
| 代表 | 用户 |
| 说明 | 用户是软件开发人员 |
| 职责 | 网络会议系统的使用者，参与开发产品的需求开发和产品体验。 |
| 成功标准 | 1. 项目满足参会人员的需要。  2. 项目准时交付。  3. 项目达到预定目标。 |
| 参与方式 | 参与开发产品的需求开发和产品体验。 |
| 可交付工件 | 无 |
| 意见/问题 | 1. 系统可以提高开会效率。  2. 系统可以使会议人员一对一的私下交流。  3. 系统可以方便会议人员实现资料共享。  4. 系统的用户界面要看上去比较舒适。  5. 系统最好可以记录会议内容。 |

* 需求工程师

|  |  |
| --- | --- |
| 代表 | 需求工程师 |
| 说明 | 项目需求开发的主要参与者。 |
| 职责 | 与用户交流，开发项目需求，进行需求管理，形成需求文档。 |
| 成功标准 | 1. 获取稳定的需求。  2. 至少开发出并且要明确用户基本需求。 |
| 参与方式 | 需求开发 |
| 可交付工件 | 需求文档 |
| 意见/问题 | 1. 需求可能会变化，因此需求文档也可能会变化。  2. 第一次迭代周期的需求并不十分明确。 |

* 体系结构架构师

|  |  |
| --- | --- |
| 代表 | 体系结构架构师 |
| 说明 | 系统中上层建筑的规划者。 |
| 职责 | 负责给出系统的中高层抽象结构 |
| 成功标准 | 1. 系统结构具有良好的弹性。  2. 系统结构要便于维护。  3. 系统具有需求文档所要求的功能。  4. 系统的架构要优美。 |
| 参与方式 | 设计系统的中高层建筑 |
| 可交付工件 | 软件系统的体系机构文档和详细设计文档。 |
| 意见/问题 | 1. 由于需求的变化可能会导致系统总体结构的变化。  2. 随着迭代的进行体系结构会不断优化。  3. 用户的有些需求会导致体系结构设计的困难。 |

* 软件工程师

|  |  |
| --- | --- |
| 代表 | 软件工程师 |
| 说明 | 代码的编写者 |
| 职责 | 根据详细设计文档，编写代码 |
| 成功标准 | 1. 代码简洁。  2. 注释清楚。  3. 算法高效。 |
| 参与方式 | 编写底层代码 |
| 可交付工件 | 具体的程序代码 |
| 意见/问题 | 1. 系统结构的不合理设计会导致软件构造的困难。  2. 可能因为赶进度，而放弃优化代码，或者必要的注释缺失。 |

* 质量保障人员

|  |  |
| --- | --- |
| 代表 | 软件质量保障员 |
| 说明 | 对软件进行测试，保证软件质量 |
| 职责 | 根据需求文档，体系结构文档和详细设计文档，进行测试 |
| 成功标准 | 发现程序的大部分Bug |
| 参与方式 | 集成测试 |
| 可交付工件 | 测试平台和测试代码 |
| 意见/问题 | 项目可能因为进度的缘故而缩短测试时间 |