**南京大学 软件学院**

12

**软件架构设计文档**

**软件设计交流系统 第三循环**

**萌军赶死队：刘羽佳、苗羿、卿荣山、王恬、完颜章伟**

**软件架构设计文档**

**更新历史**

|  |  |  |  |
| --- | --- | --- | --- |
| 版本号 | 作者 | 时间 | 内容 |
| 01 | 刘羽佳 | 2012-3-26 | 文档框架化 |
| 02 | 刘羽佳 | 2012-3-27 | 添加概述与文件共享交互图 |
| 03 | 刘羽佳 | 2012-3-28 | 添加文字聊天交互图 |
| 04 | 刘羽佳 | 2012-4-2 | 加入物理、逻辑视图 |
| 05 | 完颜章伟 | 2012-4-10 | 加入系统架构设计部分 |
| 06 | 完颜章伟 | 2012-5-1 | 更新系统模块划分 |

目录

[1 概述 3](#_Toc323594205)

[1.1 编写目的 3](#_Toc323594206)

[1.2 对象与范围 3](#_Toc323594207)

[1.3 名词与术语 3](#_Toc323594208)

[1.4 文档组织结构 3](#_Toc323594209)

[2 系统架构设计 4](#_Toc323594210)

[2.1 架构分析 4](#_Toc323594211)

[2.2 架构设计思想 4](#_Toc323594212)

[2.3 架构体系描述 5](#_Toc323594213)

[2.3.1 服务端 5](#_Toc323594214)

[2.3.2 客户端 5](#_Toc323594215)

[2.4 系统模块划分和描述 6](#_Toc323594216)

[2.4.1 模块划分 6](#_Toc323594217)

[2.4.2 模块描述 7](#_Toc323594218)

[2.5 模块接口设计 11](#_Toc323594219)

[2.5.1 Controller层 11](#_Toc323594220)

[2.5.2 Module层 13](#_Toc323594221)

[3 系统视图和系统建模 15](#_Toc323594222)

[3.1 场景视图 15](#_Toc323594223)

[3.2 逻辑视图 16](#_Toc323594224)

[3.3 进程视图 17](#_Toc323594225)

[3.3.1 白板功能顺序图 18](#_Toc323594226)

[3.3.2 音频功能顺序图 19](#_Toc323594227)

[3.3.3 文件上传/下载功能顺序图 19](#_Toc323594228)

[3.3.4 账户管理功能顺序图 22](#_Toc323594229)

[3.3.5 文字聊天功能顺序图 25](#_Toc323594230)

[3.4 开发视图 26](#_Toc323594231)

[3.5 物理视图 26](#_Toc323594232)

# 概述

## 编写目的

本文档的撰写目的在于，基于第二循环中《需求规格说明文档》中所定义的需求，对软件设计交流系统进行模块划分，并对各个划分好的模块进行实现层面上的设计与说明，为后续的代码实现阶段的基础，即实现系统需求到系统实现的过渡。

## 对象与范围

* 预期阅读对象：萌军赶死队项目组成员、审阅指导老师。
* 文档范围：软件设计交流系统的模块划分以及模块内部的设计。

## 名词与术语

无。

## 文档组织结构

**系统架构设计**

**场景视图**

**逻辑视图**

**系统视图化及建模（4+1视图）**

**进程视图**

**开发视图**

**物理视图**

# 系统架构设计

## 架构分析

根据上一阶段的需求分析和本阶段的体系架构原型，本项目需要提供一个具有电子白板、音频聊天、文件共享、文字聊天功能的软件。另外，为了方便管理和保证安全性，该软件也需要提供账号登录系统和房间系统。于是，可将软件按照功能点分为电子白板、音频聊天、文件共享、文字聊天、账号系统和房间系统6个子系统。

本项目的界面、逻辑和数据结构可以进行分离，是典型的可以应用MVC架构的实例。故本项目可以采用MVC架构，将软件分为View、Control和Module三个模块，分别负责界面的显示、流程逻辑的控制和数据集合的操作。

由于本项目需要满足同一公司多个异地设计师同时交流的问题，所以我们决定在整体架构上采用C/S（客户端/服务器）架构，即多个用户使用客户端访问位于服务器的服务端程序。

## 架构设计思想

在账号管理、房间操作、文件传输、音频聊天的功能中，服务端在接收客户端的命令后在本机进行处理，然后通知其他客户端进行刷新。

由于白板功能和文字聊天功能中，服务端需要以较快的速度处理多个客户端的请求，所以我们决定让服务端不负责命令的处理，只把客户端发出的命令转发给其他客户端。而客户端负责向服务器发送操作的信息，并接收从服务器转发来的其他客户端的操作信息。最后，客户端根据这些操作信息来显示操作结果。

## 架构体系描述

### 服务端

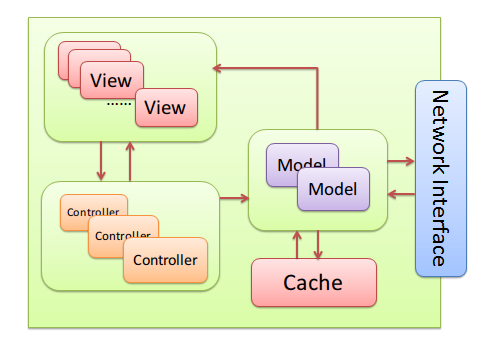
服务端负责处理客户端的登录、出入房间、文件传输、音频传输请求，并转发客户端的白板操作、文字聊天信息。用户只要打开服务端程序就能打开服务端的服务器，来允许客户端进行登录。

**View层**由Java完成，提供一个关闭按钮来关闭服务器；

**Controller层**负责服务器的建立和关闭

**Module层**中包含服务器的实现，并为客户端提供一个服务器的远程接口。该层实现了服务器的主要职能。

### 客户端



客户端负责发送用户请求和接收处理其他用户的请求。和传统的MVC类似，View主要用于呈现信息，Model用于本地业务逻辑并和数据层交互。Controller则是前两者间的桥梁。下面就各个模块作必要的补充说明：

**View层**基于Java，负责呈现系统要表达的信息，并提供用户输入的接口；

**Controller层**负责事件派发和调度，一个Controller可能会对应数个Model或View，但是对应每个View都只提供一个小的接口，以保证View的调用权限。

**Module层**中提供各个功能所需的数据实体及其为Controller层提供的接口，并且包含白板和文字部分的一些业务逻辑（其他逻辑在服务器完成）。

除此之外，客户端还将涵盖必要的资源加载和程序设置工具，这里不作表示。他们可以通过“单件模式Singleton Pattern”来初始化实现，或只提供静态方法。

## 系统模块划分和描述

### 模块划分

#### 客户端

如上文描述，客户端采用MVC架构。

按照功能划分，可以将客户端分为以下子系统：

* 电子白板：负责管理白板指令的发送、接收和白板的显示。
* 音频聊天：负责用户的音频输入输出和网络传输、解码。
* 文件共享：负责文件的远程传输。
* 文字聊天：负责文字信息的发送和接收。
* 账号系统：负责发送用户的登录请求和管理员账户的用户管理请求。
* 房间系统：负责房间信息的获取、显示，和用户加入房间请求的发送。

#### 服务端

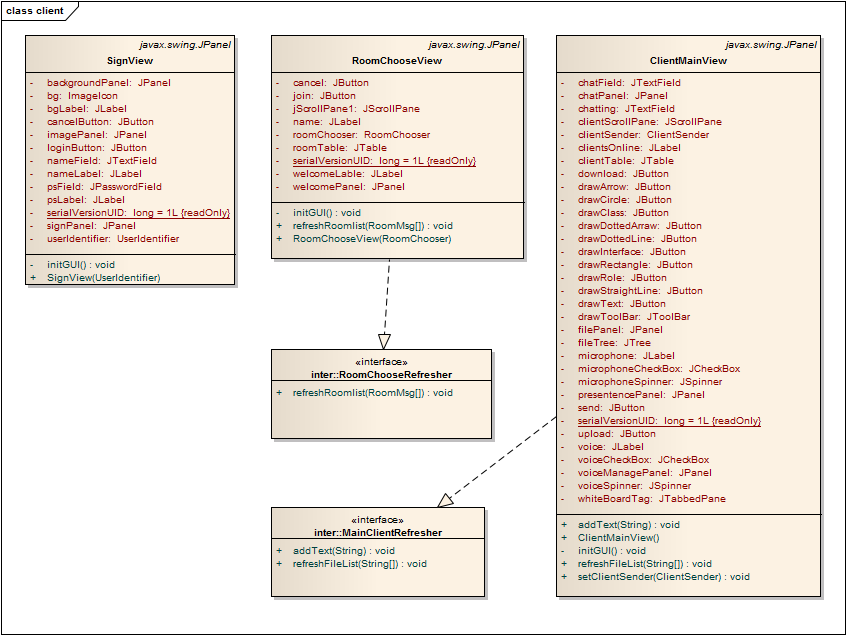
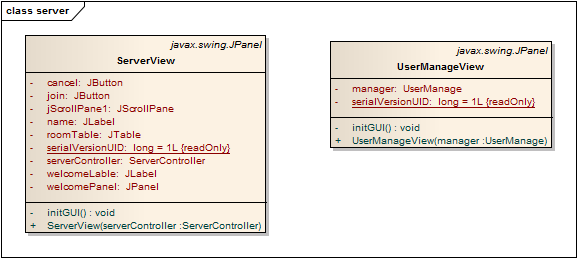
如上文描述，服务端采用简化的MVC架构，MV之间不需要连接。

按照功能划分，可以将服务端分为以下子系统：

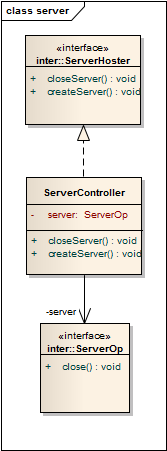
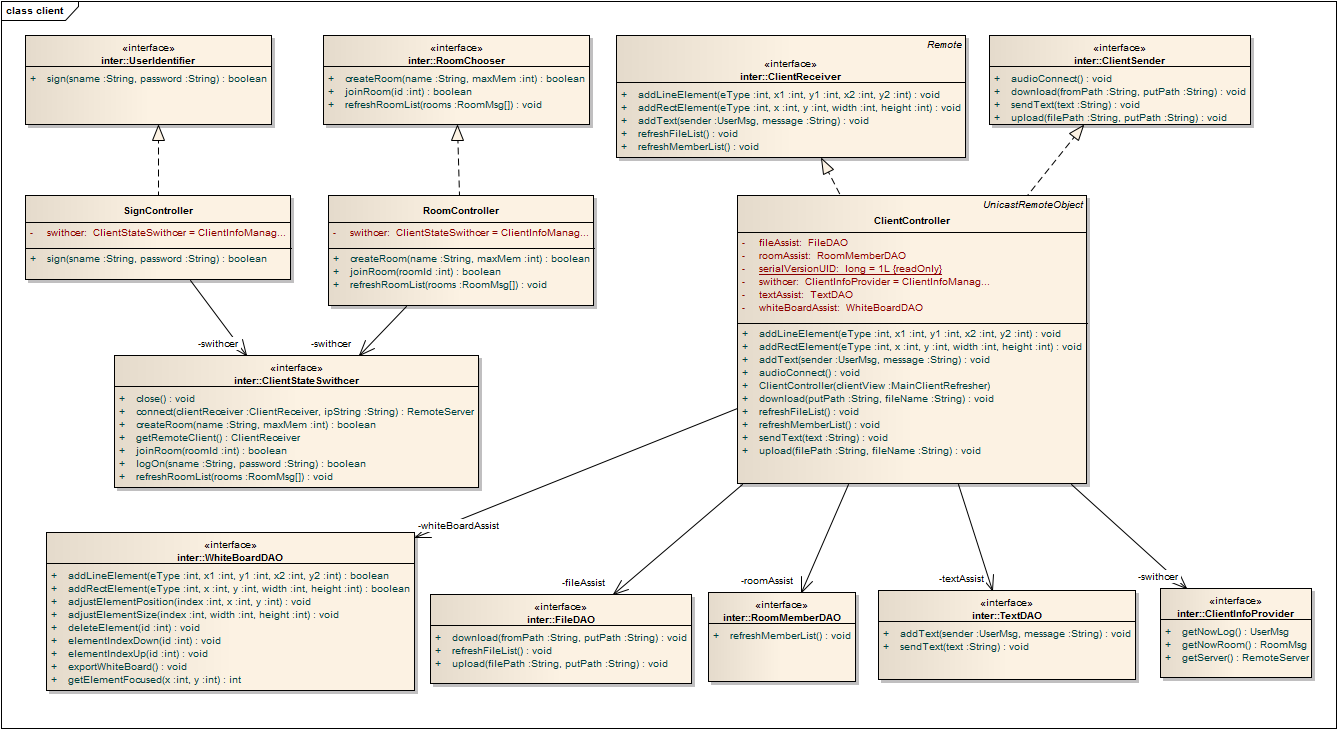
* 电子白板：负责管理白板指令的转发。
* 音频聊天：负责传输用户发送的音频流信息。
* 文件共享：负责服务器文件的远程传输。
* 文字聊天：负责文字信息的转发。
* 账号系统：负责处理用户的登录请求和用户管理请求。
* 房间系统：负责房间信息的管理，和用户加入房间信息的广播。

### 模块描述

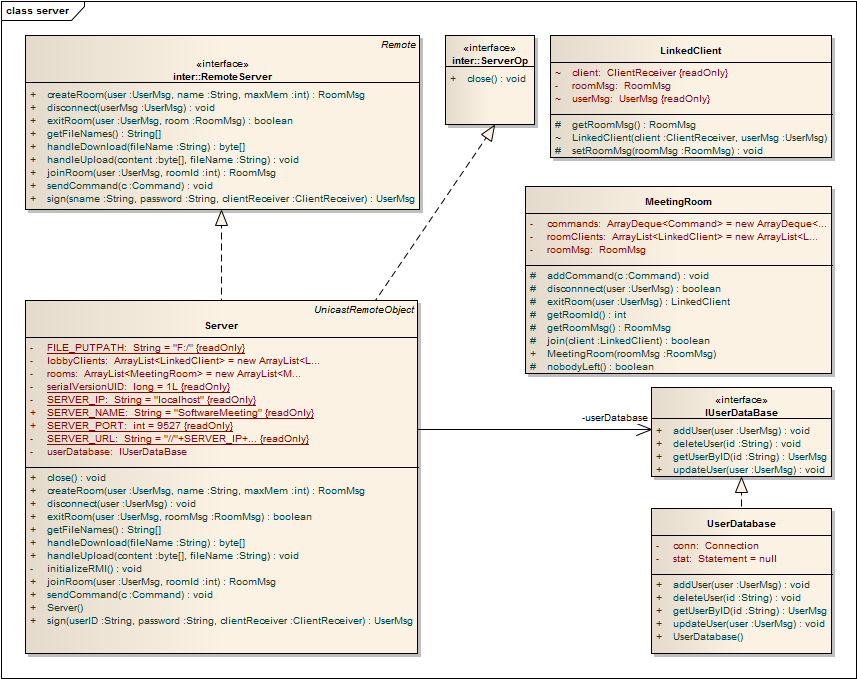
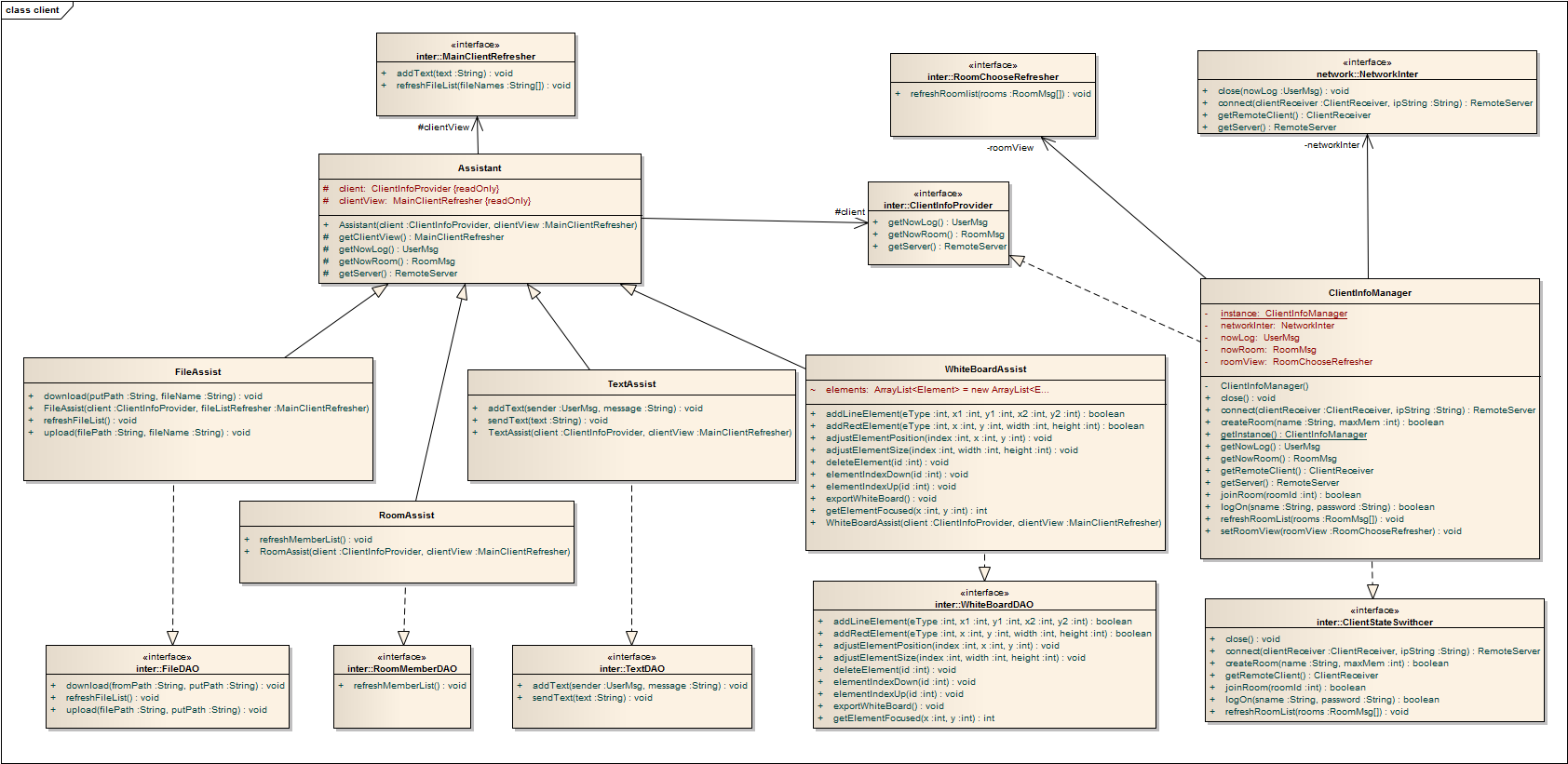
#### View



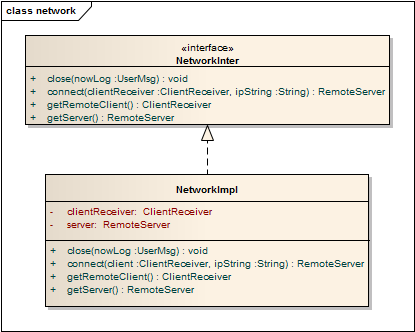
#### Controller



#### Module



#### Network



## 模块接口设计

### Controller层

|  |  |  |
| --- | --- | --- |
| 接口名称 | 接口说明 | 接口内容 |
| UserManage | 供账号管理界面使用，提供对账户的增删改操作。 | /\*\*按照给出的信息添加一个用户。\*/  **public** **void** addUser(UserMsg usr);  /\*\*按照给出的信息，把对应id的用户信息以usr信息覆盖。\*/  **public** **void** editUser(UserMsg usr);  /\*\*删掉对应id的用户。\*/  **public** **void** deleteUser(String id); |
| UserIdentifier | 供登录界面使用，提供验证用户名密码的方法。 | /\*\*验证输入的用户名和密码是否正确。  \* **@return** 是否登录成功 \*/  **public** **boolean** sign(String sname,String password); |
| RoomChooser | 用户在房间选择界面里的操作接口。提供用户加入房间的方法。 | /\*\*登录于此的用户加入对应id的房间。  \* **@return** 加入是否成功 \*/  **public** **boolean** joinRoom(**int** id);    /\*\*登录的用户创建一个新的房间  \* **@return** 创建是否成功 \*/  **public** **boolean** createRoom(String name,**int** maxMem); |
| ClientInfoProvider | 让Assist类查看客户端信息用的接口。 | /\*\* **@return** 远程的服务器引用 \*/  **public** RemoteServer getServer();  /\*\* **@return** 现在登录着的用户的信息 \*/  **public** UserMsg getNowLog();    /\*\* **@return** 现在所在的房间的信息 \*/  **public** RoomMsg getNowRoom(); |
| ClientSender | 客户端用于发送信息的接口。此接口负责与服务器接口进行连接。 | /\*\*将本客户端连接到服务器。  \* **@return** 连接是否成功 \*/  **public** **boolean** connect(String ipString) **;**  /\*\*关闭此客户端和服务器的连接，并在关闭前告知服务器自己的编号。\*/  **public** **void** close();  /\*\*上传一个文件到服务器。\*/  **public** **void** upload(String filePath,String putPath);  /\*\*下载一个文件到本机。\*/  **public** **void** download(String fromPath,String putPath);  /\*\*发送一条文字信息。\*/  **public** **void** sendText(String text);  /\*\* 开始接受和发出声音，开始时调用一次\*/  **public** **void** audioConnect();  /\*\*添加一个白板命令。\*/  **public** **void** sendWhiteBoardCommand (CommandOnWhiteBoard wbc); |
| ClientReceiver | 服务器端回调客户端所用的接口。客户端必须要实现这个接口。 | /\*\*刷新房间列表。 \*/  **public** **void** refreshRoomList();    /\*\*刷新文件列表。 \*/  **public** **void** refreshFileList();    /\*\*刷新房间成员列表。 \*/  **public** **void** refreshMemberList();    /\*\*在文字信息窗口中添加一条信息。 \*/  **public** **void** addText(UserMsg sender,String message);    /\*\*添加一条白板命令。\*/  **public** **void** addWhiteCommand(CommandOnWhiteBoard c);    /\*\*撤消上一步的白板命令。 \*/  **public** **void** undoWhiteCommand(); |
| ServerHoster | 供服务端的界面使用，提供开关服务器的接口。 | /\*\*创建可用的服务器\*/  **public** **void** createServer();  /\*\*关闭服务器 \*/  **public** **void** closeServer(); |

### Module层

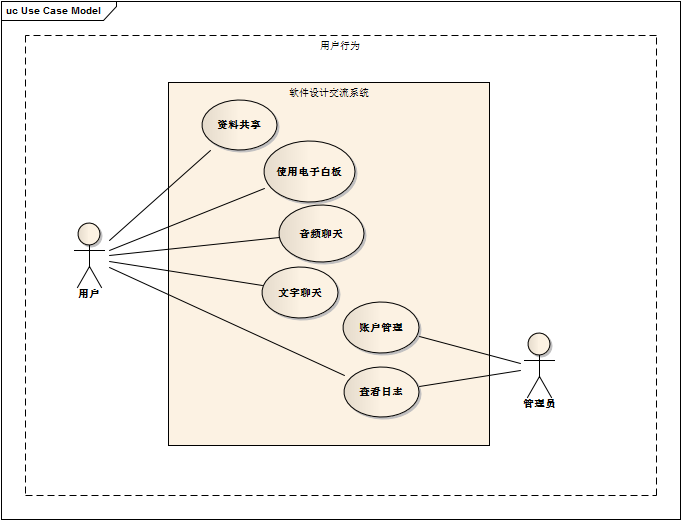
|  |  |  |
| --- | --- | --- |
| 接口名称 | 接口说明 | 接口信息 |
| RemoteServer | 服务器端供客户端远程调用的接口。 | /\*\*用户登录系统，获得自身的信息。  \* **@return** 用户的信息\*/  **public** UserMsg sign(String sname,String password,ClientReceiver clientReceiver);    /\*\*用户加入一个房间  \* **@return** 加入的房间信息 \*/  **public** RoomMsg joinRoom(UserMsg user, **int** roomId);  /\*\*用户新建一个房间  \* **@return** 创建的房间信息 \*/  **public** RoomMsg createRoom(UserMsg user, String name, **int** maxMem);    /\*\*退出房间，返回大厅\*/  **public** **boolean** exitRoom(UserMsg user,RoomMsg room);    /\*\*切断与某个客户端的连接\*/  **public** **void** disconnect(UserMsg userMsg);  /\*\*将一个命令对象传输到服务器去。\*/  **public** **void** sendCommand(Command c); |
| FileDAO | 控制层访问文件共享子系统所用接口。 | /\*\*上传一个文件到服务器。\*/  **public** **void** upload(String filePath,String putPath);    /\*\*下载一个文件到本机。\*/  **public** **void** download(String fromPath,String putPath);    /\*将底层的文件列表刷新后更新到界面里 \*/  **public** **void** refreshFileList(); |
| NetDAO | 客户端的网络连接辅助接口。其实现类需封装网络连接的实现细节。 | /\*\*将本客户端连接到服务器。  \* **@return** 连接到的服务器对象\*/  **public** RemoteServer connect(String ipString);  /\*\* **@return** 客户端的远程引用\*/  **public** ClientReceiver getRemoteClient();    /\*\*关闭此客户端和服务器的连接，并在关闭前告知服务器自己的编号。\*/  **public** **void** close(); |
| RoomDAO | 控制层访问文件共享子系统所用接口。 | /\*\*对应用户加入对应id的房间。  \* **@return** 加入的房间信息 \*/  **public** RoomMsg joinRoom(UserMsg user,**int** roomId);    /\*\*对应用户创建房间。  \* **@return** 创建的房间信息 \*/  **public** RoomMsg createRoom(UserMsg user,String name,**int** maxMem);    /\*\*刷新房间列表。 \*/  **public** **void** refreshRoomList();    /\*\*刷新房间成员列表。 \*/  **public** **void** refreshMemberList(); |
| TextDAO | 控制层访问文字聊天子系统所用接口。 | /\*\*发送一条文字信息。\*/  **public** **void** sendText(String text);    /\*\*添加一条文字消息到窗口。\*/  **public** **void** addText(UserAssist sender, String message) ; |
| UserDAO | 控制层访问用户操作子系统所用接口。 | /\*\*按照给出的信息添加一个用户。\*/  **public** **void** addUser(UserMsg usr);    /\*\*按照给出的信息，把对应id的用户信息以usr信息覆盖。\*/  **public** **void** editUser(UserMsg usr);    /\*\*删掉对应id的用户。\*/  **public** **void** deleteUser(String id);    /\*\*验证输入的用户名和密码是否正确。  \* **@return** 登录到的用户 \*/  **public** UserMsg logOn(String sname,String password); |
| WhiteBoardDAO | 控制层访问电子白板子系统所用接口。 | /\*\*添加一个元素 \*/  **public** **boolean** addElement(**int** eType);  /\*\*返回被选择的元素 \*/  **public** **int** getElementFocused(**int** x, **int** y);  /\*\*调整元素大小 \*/  **public** **void** adjustElementSize(**int** index, **int** width, **int** height);  /\*\*调整元素位置 \*/  **public** **void** adjustElementPosition(**int** index, **int** x, **int** y);  /\*\*输出白板内容到图片 \*/  **public** **void** exportWhiteBoard(String path); |

# 系统视图和系统建模

## 场景视图

此处用《需求规格说明文档》中定义明确的用例图作为场景视图，其中用例的大小代表需求的关键程度。

场景视图如下：

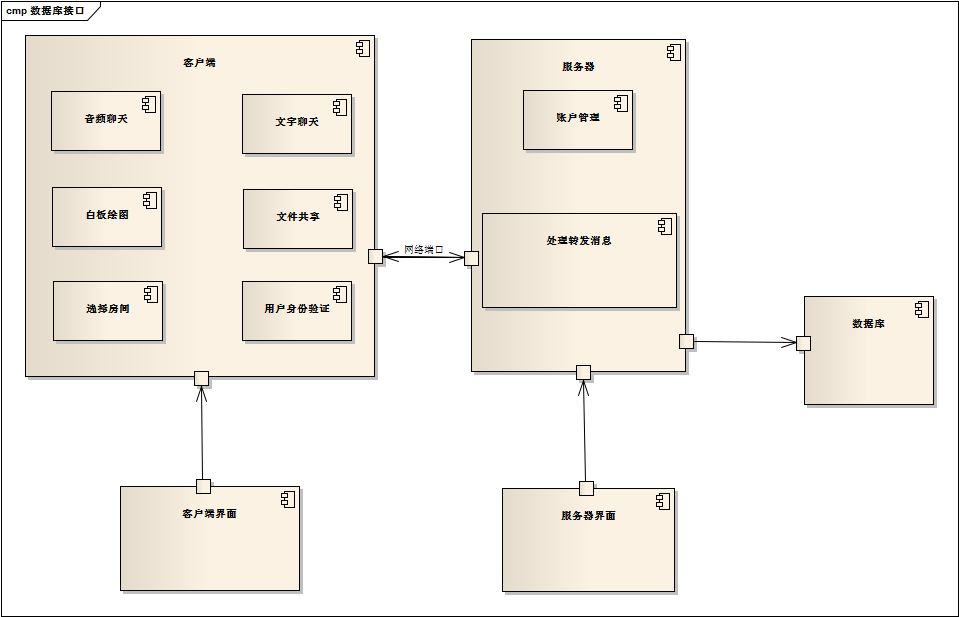


从图中可以总结出以下需求及其相应优先级：

|  |  |
| --- | --- |
| 优先级 | 模块 |
| 01 | 白板功能 |
| 02 | 音频功能 |
| 03 | 资料共享 |
| 04 | 账户管理 |
| 05 | 文字聊天 |
| 06 | 查看日志 |

## 逻辑视图

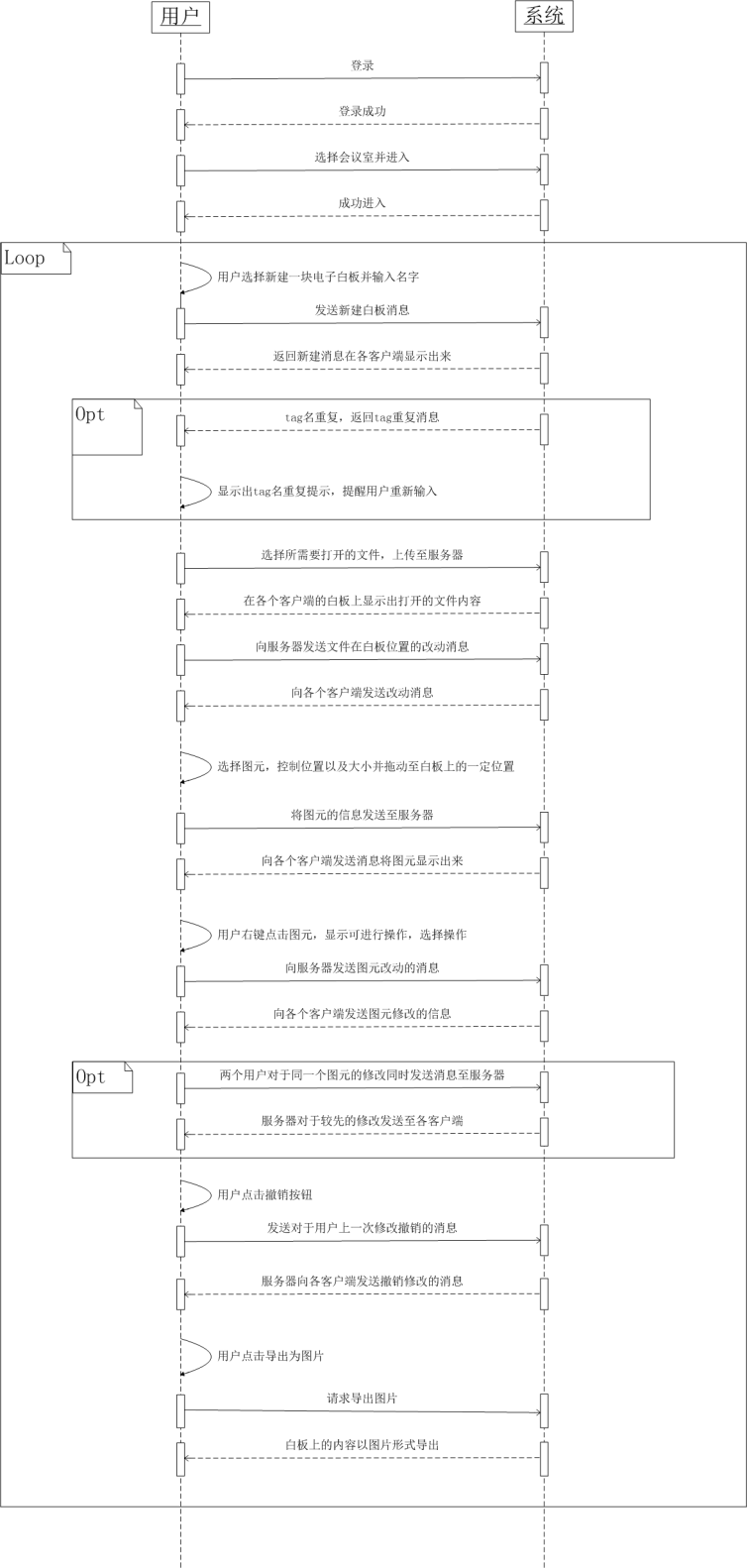
逻辑视图关注系统的高层逻辑结构，针对终端用户，给出系统功能的大致描述。



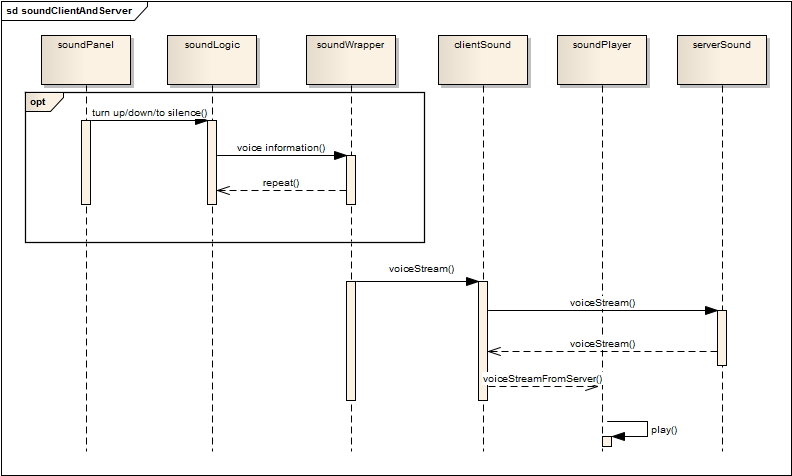
## 进程视图

进程视图针对系统集成人员，主要描述系统并发和系统同步机制的相关进程和线程。此处利用UML交互图（系统顺序图）来表示。各个进程的优先级同2.1场景视图所示。

### 白板功能顺序图



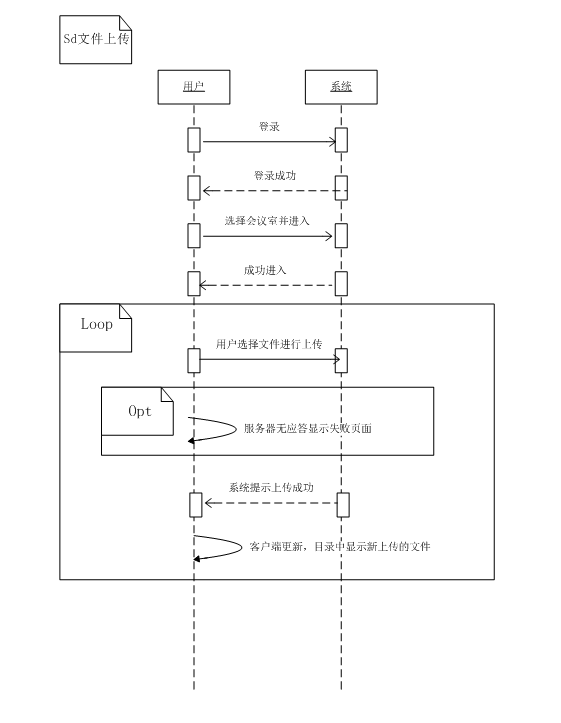
### 音频功能顺序图



### 文件上传/下载功能顺序图

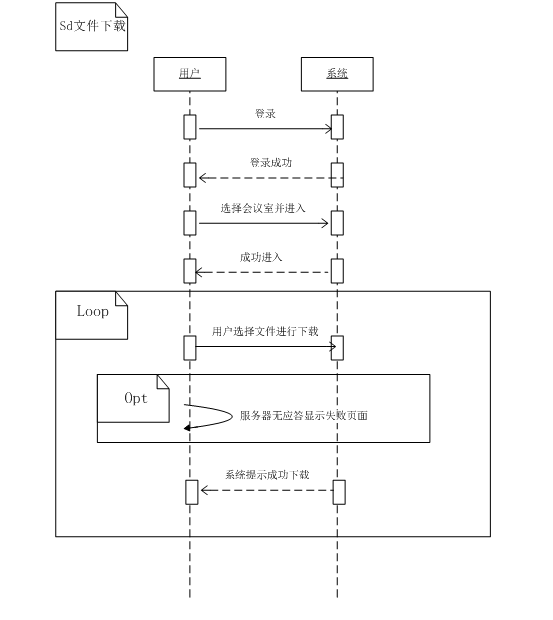
#### 文件上传

文件上传顺序图如下：



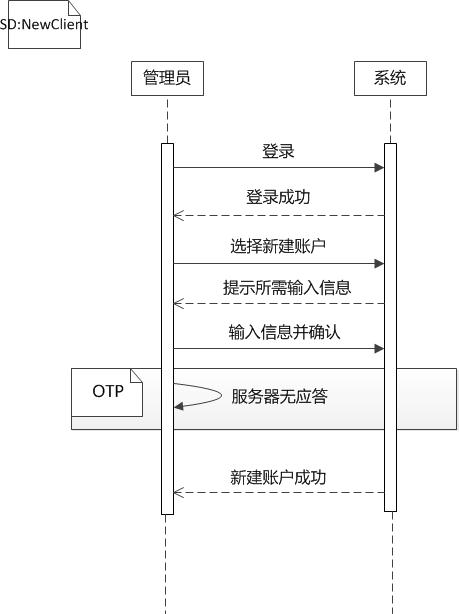
#### 文件下载

文件下载顺序图如下：

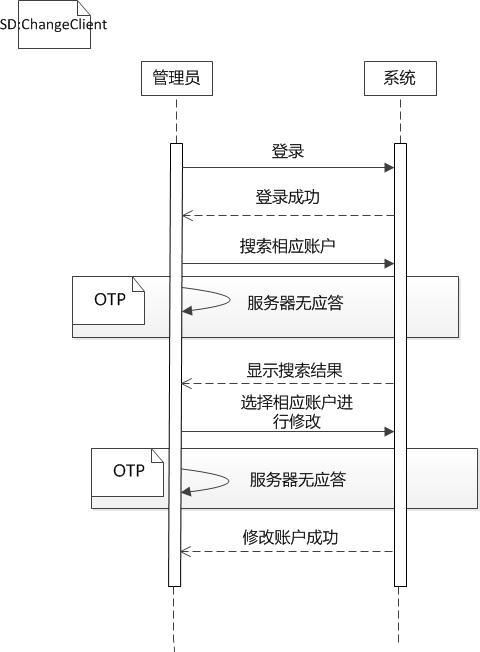


### 账户管理功能顺序图

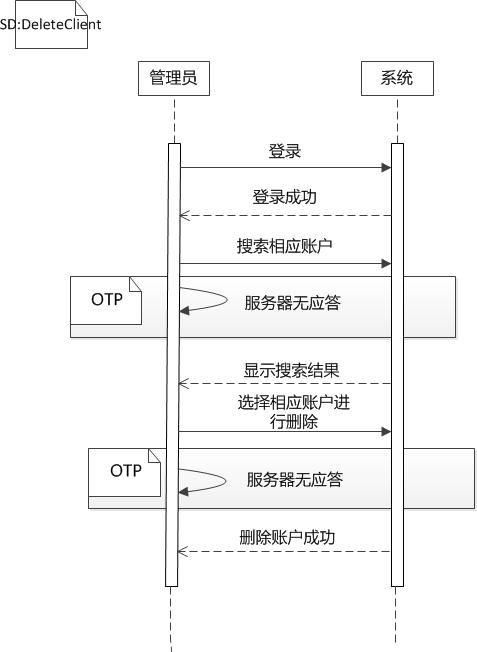
#### 新建账户



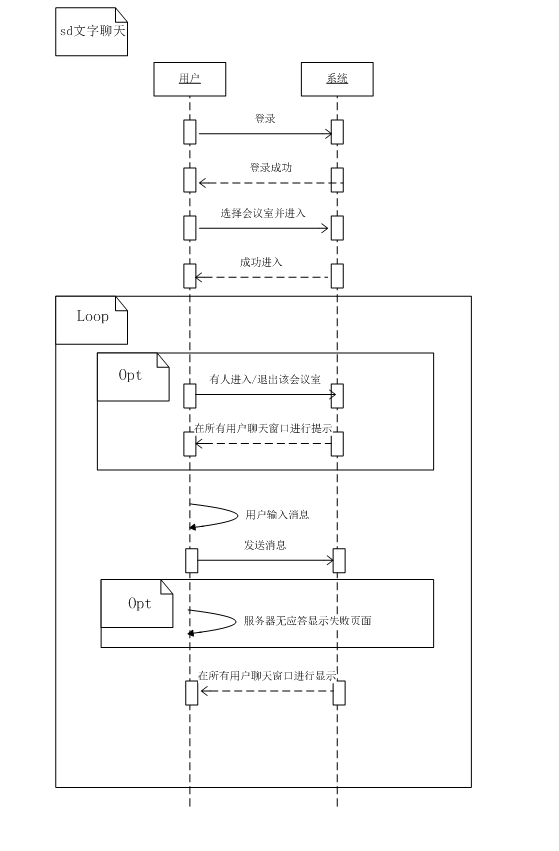
#### 账户查看/修改



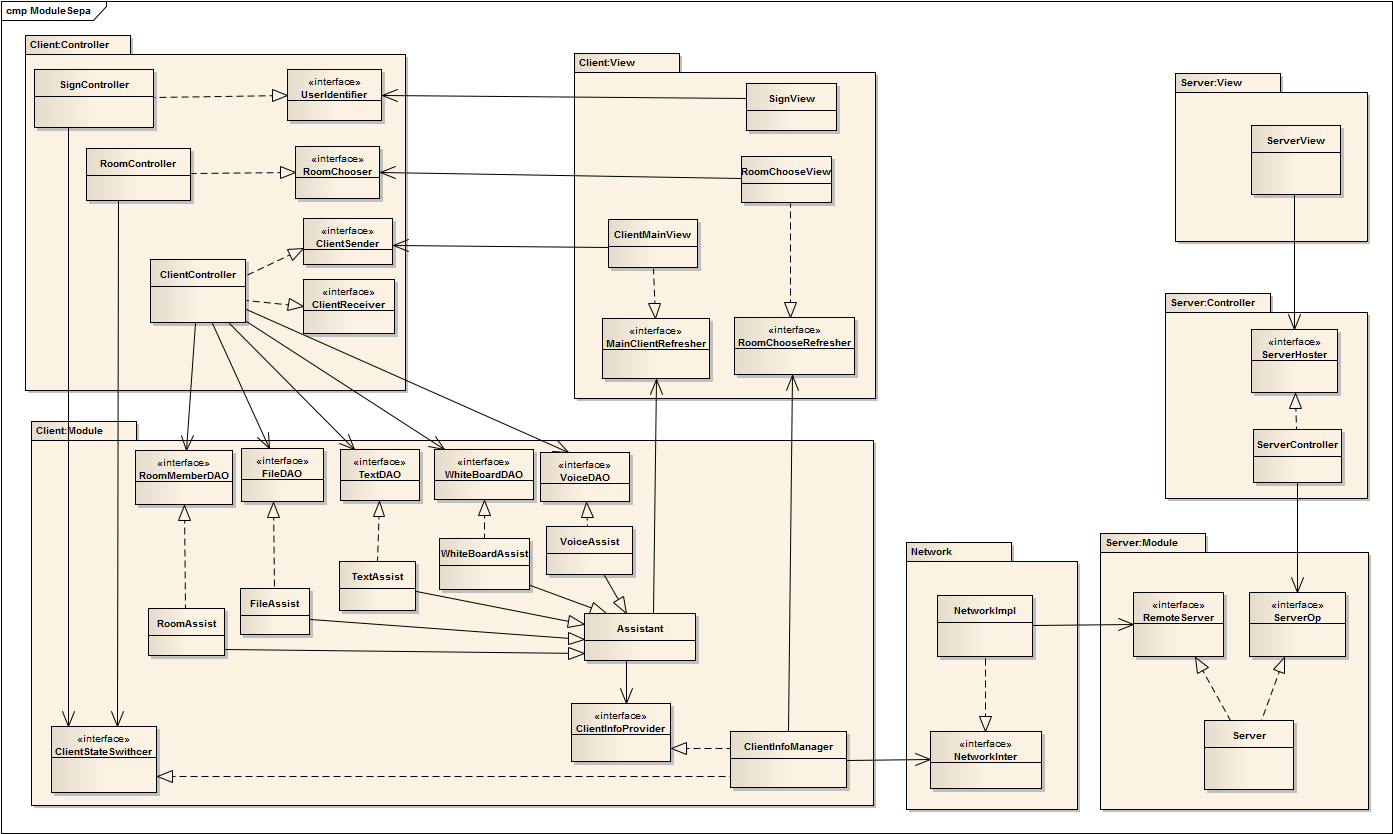
#### 账户删除



### 文字聊天功能顺序图



## 开发视图



## 物理视图

物理视图针对系统工程师，重点描述系统的硬件部署情况（包括部署状况和拓扑逻辑）。

