JiaoGoBang

软件架构文档

版本 <1.1>

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| <27/06/2016> | <1.0> | <初始版本> | <陈俊> |
| <08/07/2016> | <1.1> | <修改版本1> | <陈俊> |
|  |  |  |  |
|  |  |  |  |

目录

1. 简介 4

1.1 目的 4

1.2 参考资料 4

2. 用例视图 4

2.1 联网双人对战 4

2.2 人机对战 4

2.3 复盘 5

3. 逻辑视图 5

3.1 概述 5

3.1.1 概述 5

3.1.2 界面层（客户端） 5

3.1.3 业务逻辑层（客户端） 6

3.1.4 实体层（客户端）/数据访问层（服务器端） 7

3.1.5 业务逻辑层（服务器端） 8

3.1.6 对于一般请求的类图 8

4. 进程视图 9

5. 部署视图 9

6. 实现视图 10

7. 数据视图（可选） 10

8. 核心算法设计（可选） 10

软件架构文档 （简化版）

# 简介

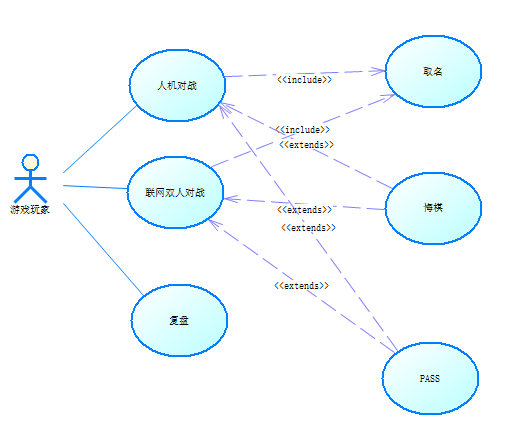
## 目的

本文档将从构架方面对系统进行综合概述，其中会使用多种不同的构架视图来描述系统的各个方面。它用于记录并表述已对系统的构架方面作出的重要决策。

## 参考资料

《软件工程原理》，2013-02，高等教育出版社

# 用例视图

本节是对软件架构的用例视图的描述，在此选取了最为核心的三个用例，其余用例及用例规约详见SRS软件需求规约。

## 联网双人对战

简要说明：该用例需要在取名（取名即给自己取个昵称，并发给服务器端记录）后才能进行。该用例允许玩家通过WLAN与其他在线玩家进行五子棋双人对战。

## 人机对战

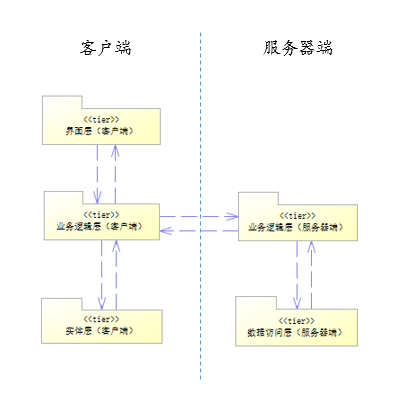
简要说明：改用例需要在取名后才能进行。该用例允许玩家通过WLAN与服务器端的人机AI进行五子棋对战。在第一次迭代中AI采用普通的人机AI，在第二次迭代中AI将采用深度学习实现。

## 复盘

简要说明：允许用户选择保存过的棋盘，查看某场过往比赛的流程。保存的棋盘保存在客户端。

# 逻辑视图

## 概述

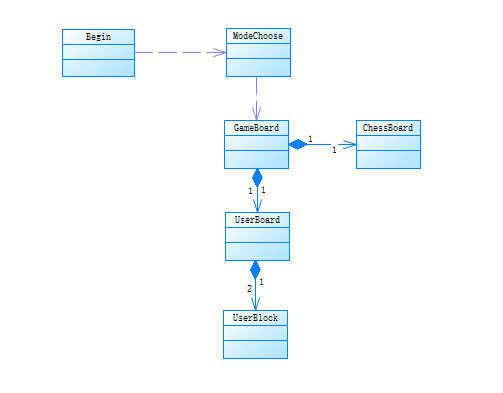


由于使用C/S架构，将程序分为客户端和服务器端两部分，界面层主要放置一些边界类，用于与用户进行交互，用户的输入或点击传递到客户端的业务逻辑层，逻辑层通过对实体层的一些调用，将信息包装后通过socket传递给服务器端的业务逻辑层，服务器实现业务逻辑后，若需要保存，则通过数据访问层保存数据，然后将数据发送给客户端。在构架方面具有重要意义的设计包

### 概述

本五子棋软件共分为三个大包，分别为Client（客户端包），Server（服务器包），以及Common（客户端和服务器共享包，主要放置实体类）。客户端主要分为两个包，即UI包（界面层），Action包（业务逻辑层），Action包分为send包和receive包，分别用于发送请求和接收回复。Server包主要包括一个Service包，用来处理客户端发来的各种请求，并将回复发回给客户端。Common包放置客户端和服务器端共享的实体类，即可看作客户端的实体层和服务器端的数据访问层。在客户端包中，放有ClientThread类和ActionThread类，用于处理客户端的线程；在服务器端有Server类，ServerThread类和ServiceThread类，用于并行处理服务器接收到的请求。同时，服务器端还有AI类和Score类，用于进行AI的落子操作。

### 界面层（客户端）

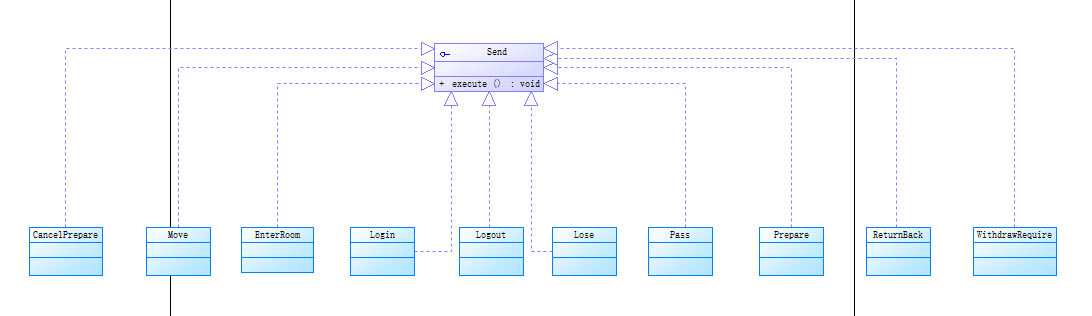


界面层主要包括以上类。Begin为初始界面，用户输入昵称取名后，通过点击Begin上的登录按钮，服务器端认证通过后，将会new一个ModeChoose界面，通过该界面可以选择人机和人人对战，人机和人人对战主要由GameBoard类来展示，GameBoard类包括一个Chessboard（用来展示棋盘和棋子）以及一个UserBoard（展示对战双方信息及执子情况），UserBoard包括两个UserBlock，分别包括每个玩家的信息和执子情况。当这些界面被初始化后，将其作为UI类的static对象保存起来，方便Action类调用。

### 业务逻辑层（客户端）

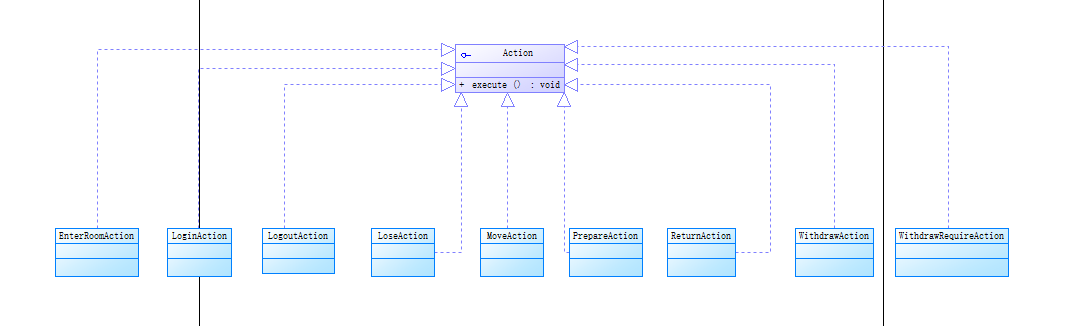
业务逻辑层主要包括两个包，包括一个Send包，用于向服务器端转发数据；另一个为Receive包，用于接收服务器端发来的数据。

Send包：



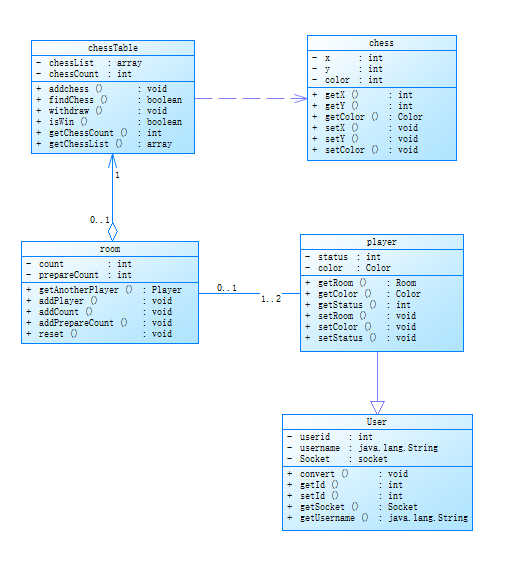
Send包用于在界面层的listener中被调用，用于向服务器端发送请求，让服务器端进行相应的功能操作，再将返回信息发送给客户端。Send包内的所有类都实现了Send的execute()接口，用于执行各自的操作。其包含的操作主要有取消准备、落子、进入房间、取名（Login，表示用户给自己取昵称后，交与服务器端记录保存）、登出（从服务器端删除用户记录）、认输、PASS、准备、返回、发送悔棋请求。

Receive包：



Receive包由ActionThread线程调用，ClientThread接收到服务器端发送的response后，通过返回的方法，调用ActionThread，由ActionThread来根据返回的方法调用相应的Action，达到处理的目的。Receive包中的类都实现了Action的execute()接口，用于接收服务器端对于各类请求的response。其包含的操作有接收进入房间回复、接收取名回复、接收登出回复、接收认输回复、接收准备回复、接收落子回复、接收返回回复、接收悔棋请求以及接收悔棋请求结果的回复。

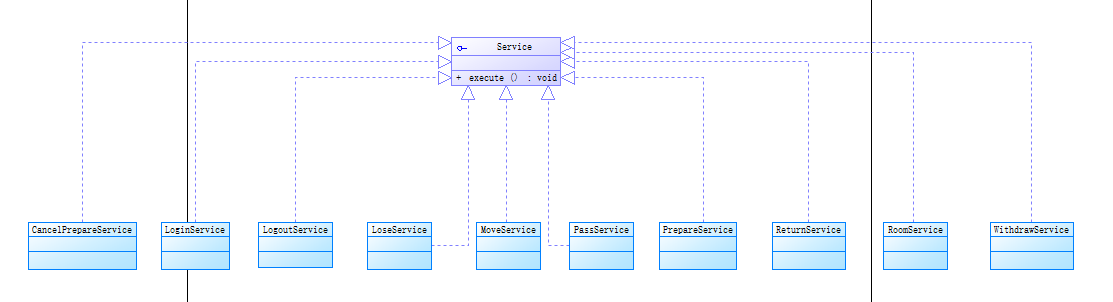
### 实体层（客户端）/数据访问层（服务器端）



客户端和服务器端共用相同的实体类，放在Common包中。包括一个User类，用于实现客户端与服务器端的身份确认与交互；一个继承自User的Player类，用于保存一局五子棋游戏中玩家的状态。Room类，包括一到两个玩家，以及一个chessTable棋盘；chessTable为棋盘类，用于记录各个棋子的位置，可以通过调用addChess（chess）添加棋子到棋盘。Chess类为棋子类，包括它的坐标及棋的颜色。

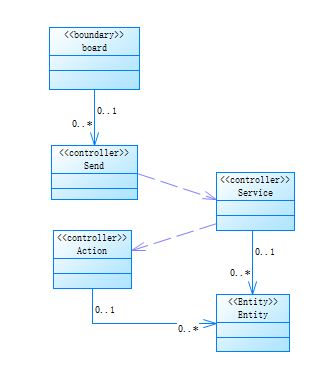
同时，在服务器端有一个AI类（图中未画出），用于进行ai的下棋操作。其有setChess函数，输入为chessTable类棋盘，输出为Chess棋子。

### 业务逻辑层（服务器端）

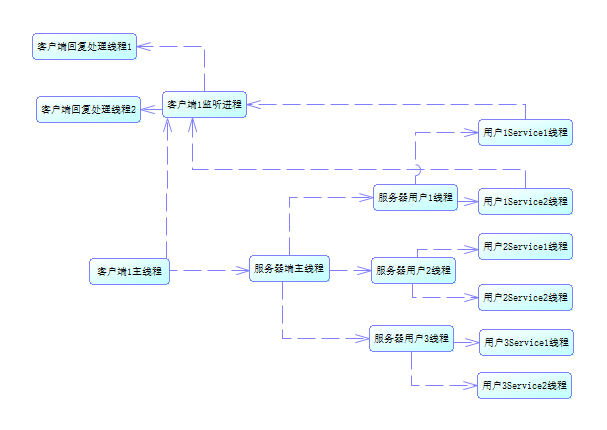


服务器端的业务逻辑层主要是将客户端发送过来的数据进行处理，并执行相应操作，并转回给客户端。Service包中的类都实现了Service的execute()接口，用于处理相应的事件。具体实现的事件如上图所示，其实现的事件基本与用户发出的请求类型雷同。接收到信息后，Service对相应事件进行处理，如需转发，则将response转发给对应客户端。

### 对于一般请求的类图

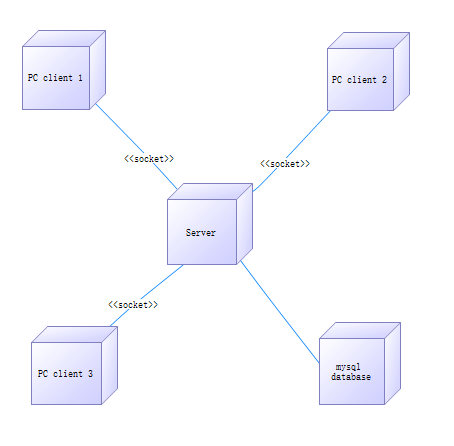


# 进程视图



每个客户端有一个主线程，主线程第一次发信息给服务器端时，会生成一个监听线程，这个线程会一直获取服务器端发送的信息，直至Socket连接断开。客户端第一次连接服务器端时，服务器端为该Socket单独分配一个线程，之后客户端发送request时，直接发送给该线程进行处理。对于用户的不同请求，该线程会分配不同的Service线程，来对用户的request进行服务，并发送给相应客户端，以保证其并行性。客户端接受response后，根据response的类型，为response分配不同的回复处理线程，以保证并行处理服务器的回复。

# 部署视图

本软件采用C/S架构，客户端通过socket套接字与服务器端相连，服务器端有数据库，用于储存一些必要的数据（比如深度学习需要的棋谱信息）。用户在客户端进行一些操作，客户端处理后将数据发送到服务端，服务器处理后，将处理后的返回数据发送到客户端，以此实现C/S 的交互。

# 实现视图

本视图描述了实现模型的整体结构、软件分解为实现模型中层和子系统的情况。本软件主要分为服务器端和客户端两大部分。每一个部分按照三层架构来分，分为界面层（服务器端不需要）、业务逻辑层、数据访问层（实体层）。客户端主要包括边界类（界面显示），一部分的业务逻辑以及需要的实体层；服务器端不需要界面层，其实现大部分的业务逻辑，并包含相关的事体以及与数据库的访问。

# 数据视图（可选）

数据库中只存储有用户的用户名和密码等信息，内容较少，故不需要数据视图。

# 核心算法设计（可选）

本软件的最终实现需要实现深度学习算法，并实现合理的落子选择器和棋局评估器。在初始版本中，需要实现简单的五子棋AI，应该会使用极大极小值搜索中应用alpha-beta剪枝。