<My PC Logo>

软件架构文档

版本 <1.2>

**13组**

[要定制 Microsoft Word 中的自动字段（选中时显示灰色背景），请选择 File>Properties，然后将 Title、Subject 和 Company 等字段替换为此文档的相应信息。关闭该对话框后，通过选择 Edit>Select All（或 Ctrl-A）并按 F9，或只是在字段上单击并按 F9，可以在整个文档中更新自动字段。对于页眉和页脚，这一操作必须单独进行。按 Alt-F9，将在显示字段名称和字段内容之间切换。有关字段处理的详细信息，请参见 Word 帮助。]

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| 2020/10/30 | <1.0> | <初代架构文档>：用例视图、物理视图、逻辑视图 | 窦嘉伟、敖宇晨 |
| 2020/11/06 | <1.1> | <架构文档初步完成 & 修改用例视图> | 窦嘉伟、敖宇晨 |
| 2021/01/06 | <1.2> | <完善架构文档> | 付玉晗 |

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 概述 4

2.2 架构上的重要用例 4

2.2.1 <Use case1 用户系统 规约> 5

2.2.2 <Use case2 单机模式 规约> 5

2.2.3 <Use case3 匹配机制 规约> 5

2.2.4 <Use case4 协作模式 规约> 5

2.2.5 <Use case5 对战模式 规约> 5

3. 逻辑视图 5

3.1 概述 5

3.2 逻辑视图的结构层次 5

3.3 在架构方面具有重要意义的设计包 6

3.3.1 Application层 6

3.3.2 Business Service层 6

3.3.3 Middleware层 7

4. 进程视图 7

5. 部署视图 8

6. 实现视图 8

6.1 Desktop PC 9

6.2 Server 9

6.3 Login/Register System 9

6.4 Turtle Draw 9

6.5 Communicate System 9

7. 数据视图 9

8. 核心算法设计 9

8.1 客户端与服务器端通信机制算法 9

8.1.1 轮询 9

8.1.2 长轮询 10

8.2 命令语句解析算法 10

软件架构文档

# 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

1. My\_PCLogo项目迭代计划1.0 2020.09 本项目组

2. 项目软件需求规约 2020.09 本项目组

3. 项目数据库设计规范 2020.10 本项目组

4. 项目词汇表 2020.09 本项目组

5. 谷歌编程规范（试用于本项目） 2020.09 本项目组 （摘自《谷歌编程规范》）

6. 项目要求 2020.xx <软件工程原理与实践>课程要求

7. 课程教材《软件工程原理》 2013.xx 沈备军 陈昊鹏 陈雨亭编著

8. 用例模型 2020.10 本项目组

9. Vision文档2.0 2020.10 本项目组

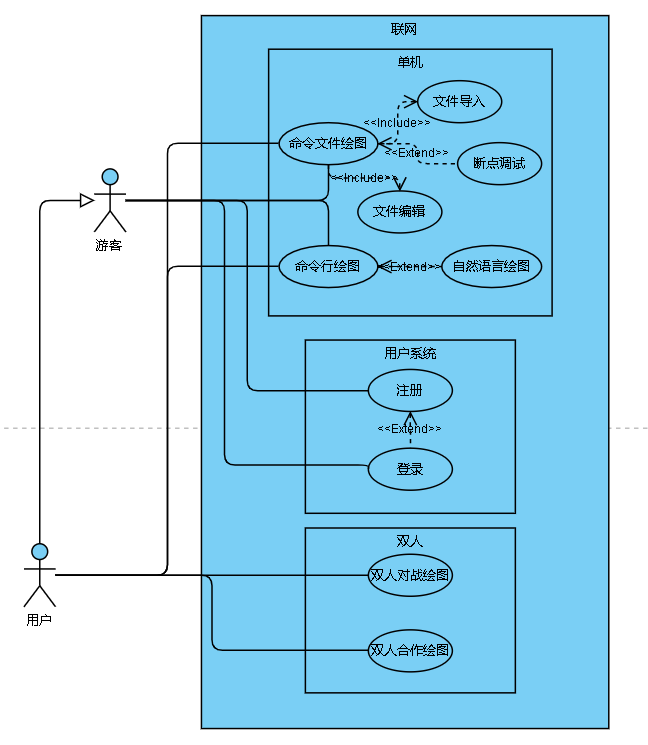
# 用例视图

## 概述

本部分描述了架构设计中的用例视图，它描述了一组场景或用例，这些场景或用例强调或说明了架构设计中的一些重要的特点。

## 架构上的重要用例

整体上包括三个模块：单机模式、用户系统、双人模式。其中用户系统和双人模式需要在联网下才能使用，单机模式则在联网与不联网下均能使用，所以将单机模式包含在联网下绘制出。



图表 1用例图

### <Use case1 用户系统 规约>

若想联网使用，需要登录，用户拥有一个账户。游客可以选择填写注册信息，发送到后台注册接口，注册新账户。注册表单包含用户名，邮箱/电话号码，密码等信息。游客用户若已有账户，通过填写账号及密码，发送到后台登录接口，校验成功即可登录。若登录时提示没有账户则令游客前往注册账户。

### <Use case2 单机模式 规约>

在单机模式下，用户可以有两种使用方式：

1. 命令行模式

每一条指令输入后，经过解析器解析，确认合乎语法规范后，交付给执行器将结果绘制在画布上。输入指令可以是直接键入命令行，或者使用自然语言通过语音方式输入命令行（进阶需求）

1. 命令文件模式

文件可以自行在编辑器上编写或者直接将文件导入编辑器。通过丰富的指令组合，解释器确认在合乎语法规范后，绘制出丰富多彩的图案。还可设置断点进行调试，分步运行

### <Use case3 匹配机制 规约>

用户发送匹配请求，由后台自动匹配。匹配到其它用户后，双方需要在有限时间内（如30秒）确认。双方在时间限制内均确认后建立连接，即可进行联网游戏。若在规定时间内尚有用户未确认，则本次匹配失效。

### <Use case4 协作模式 规约>

1. 网上双人单海龟协同绘图

已经匹配的两名用户，通过轮流发送指令控制单个海龟绘图，每人需要在固定的操作时间限制内（如30秒）发送合乎语法的指令（在本地校验合法后才能发出），否则海龟的控制转移到另一玩家手中。

1. 网上双人双海龟协同绘图

已经匹配的两名用户，各自发送指令控制自己的海龟进行绘图，双方可以自由协调沟通，共同完成绘图。

### <Use case5 对战模式 规约>

网上双人双海龟绘图对战。已经匹配的两名玩家用户，各自发送指令控制自己的海龟绘图，在规定时间结束后，经由评分模块通过一定的评分机制决出胜者。

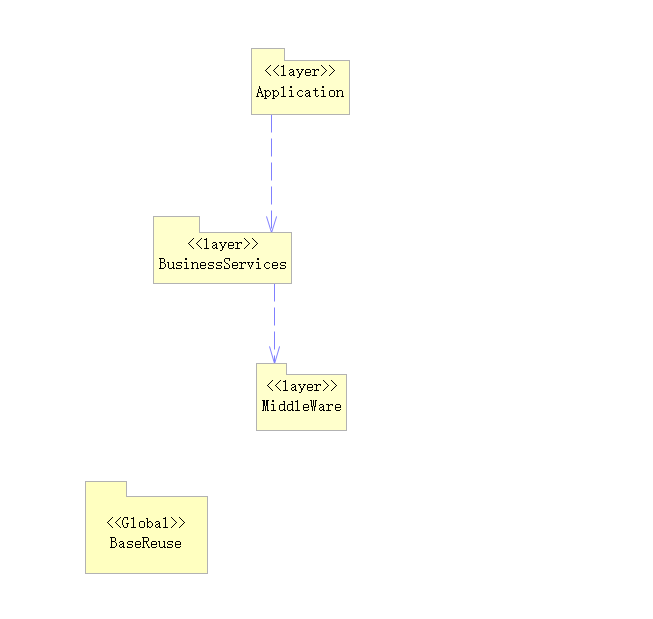
# 逻辑视图

## 概述

本部分描述了架构的逻辑视图，它描述了服务和功能在项目架构中的层次结构，此外，还描述了层次与包之间的关系。

## 逻辑视图的结构层次

本项目的逻辑视图由三个主要层次组成：应用层、业务服务层、中间层。

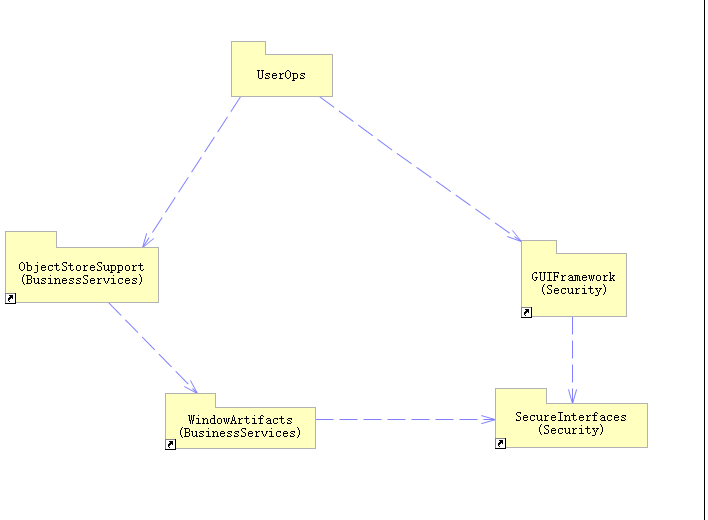


图表 2逻辑视图的三个层次及其关系

## 在架构方面具有重要意义的设计包

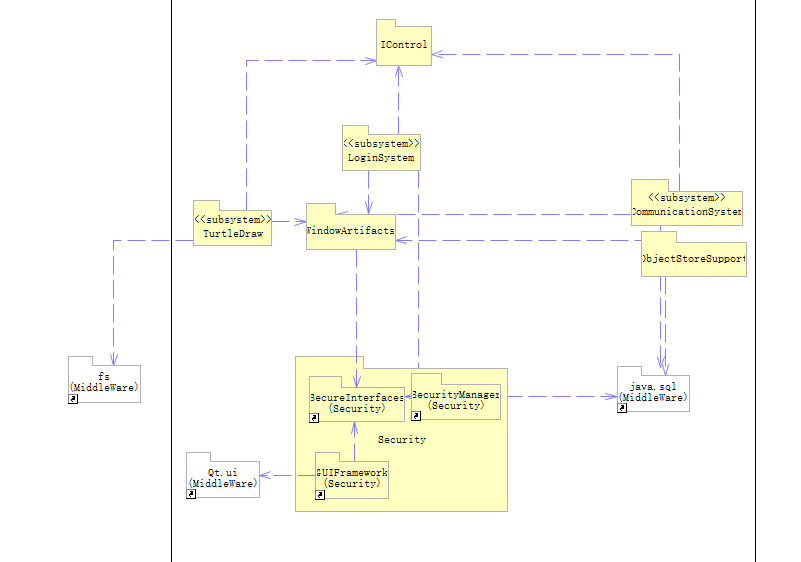
### Application层

Application层依赖于业务服务层。



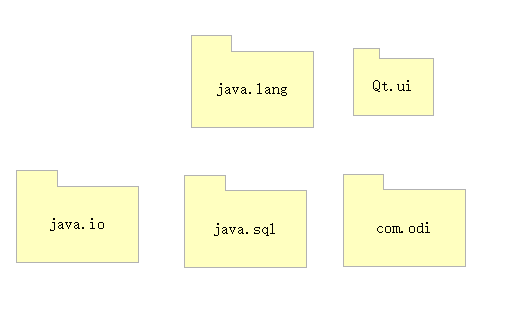
图表 3应用层

### Business Service层



图表 4业务服务层

### Middleware层

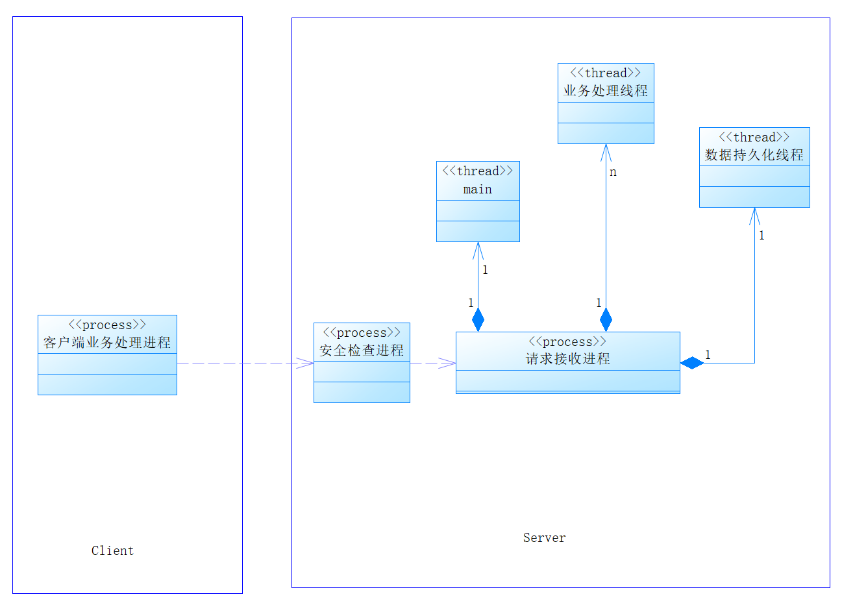


图表 5中间层

# 进程视图

本节说明是对软件架构的进程视图的描述，描述了在系统执行中涉及的任务(进程和线程)及它们的通信和交互。进程视图说明了软件客户端与服务端进程之间的通信与交互。

首先，客户端业务处理进程发起请求，经过服务端安全检查进程检查确认后发往服务端的请求接收进程。请求接受进程创建一个线程组，针对每一个发送请求的客户端加入一组新的业务处理线程和数据持久化线程到线程池中。业务处理线程用以处理业务，数据持久化线程则通过与数据库的交互实现数据的持久化。

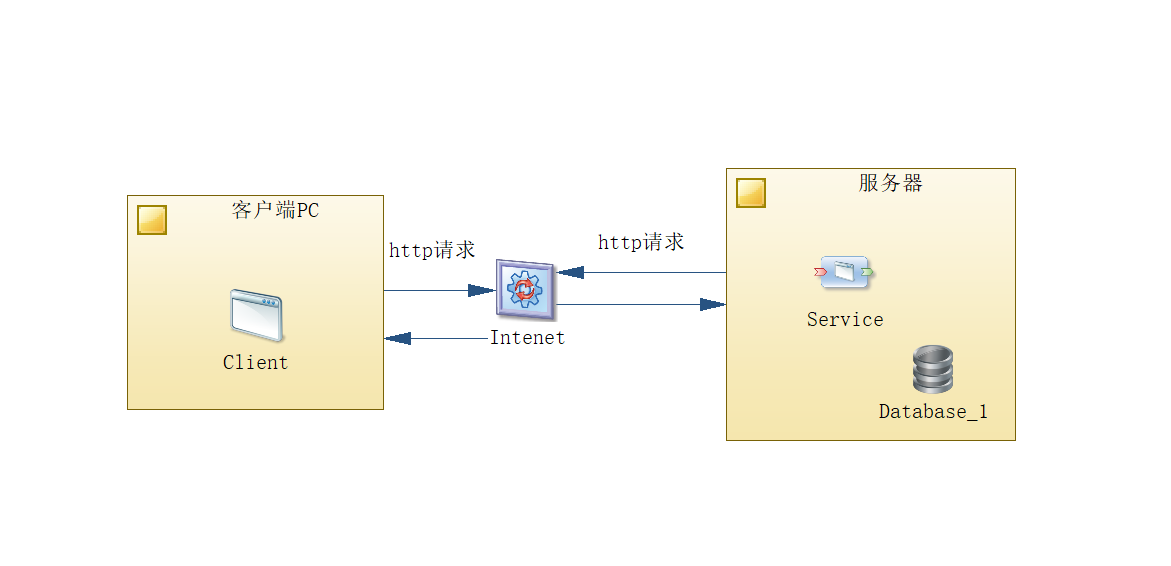


图表 6进程视图

# 部署视图

本节是软件架构的部署视图即物理视图，描述用来部署和运行本软件的一种物理网络（硬件）配置，并描述了该配置中执行该软件的物理节点及其互联情况。另外，还描述了进程视图中的各进程到物理节点的映射。

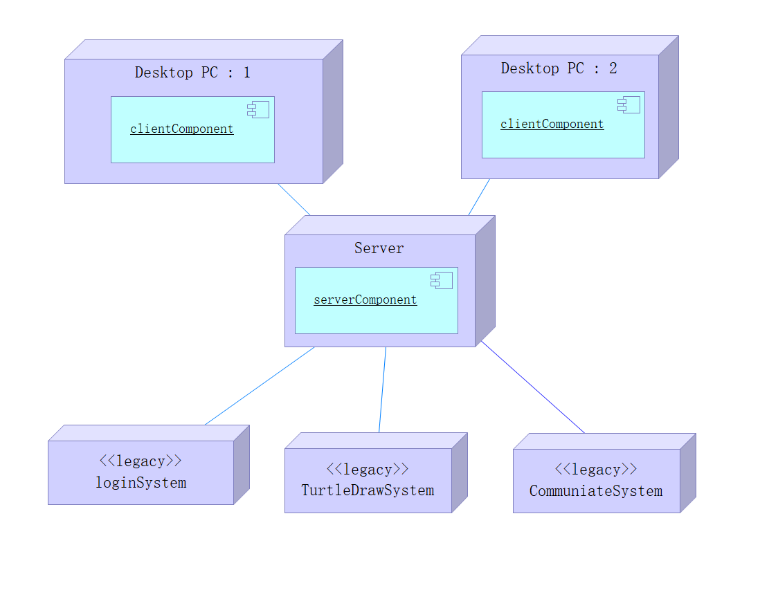
Client通过调用QT中的网络库，使用HTTP协议连接网络，并发送请求，服务器使用HTTP协议连接网络并接收来自client的请求，经过server的处理将结果返回给client，部分需持久化存储的数据储存在服务器的MYSQL和MongoDB的数据库中。



图表 7物理视图

# 实现视图

架构的部署视图描述了用于最典型平台配置的各种物理节点。还描述了任务(从Process视图)到物理节点的分配。本节按物理网络配置组织;每个这样的配置都由部署图来说明，然后是到每个处理器的进程映射。



图表 8 实现视图

## Desktop PC

用户通过desktop pc控制turtle，与用户交互并与server连接.

## Server

Server是一个unix服务器，用户可以通过internet和http请求访问server.

## Login/Register System

包含了完整的用户登录可以通过服务器使用.

## Turtle Draw

包含了用户控制turtle的过程.

## Communicate System

包含了用户交互过程，好友邀请以及对战交互等等.

# 数据视图

本项目不选择做数据视图。本项目对数据存储的需求只有用户账号数据的存储以及对命令文件、游戏数据的存储，在复杂程度上远不及其它传统信息系统项目。

# 核心算法设计

## 客户端与服务器端通信机制算法

客户端与服务器的通信使用长轮询算法和socket通信。

### 长轮询

长轮询算法应用在用户对战交互、对局匹配队列、用户交流、好友列表。

客户端向server端发送长轮询请求，server端接受请求后将其hold住，只有当server端需要向客户端发送信息时才返回。

用户登录后向server端发送长轮询，当有好友登录或发送消息或者匹配成功时长轮询返回。用户端在对局中，若不是自己回合则和server端保持长轮询连接。

### Socket通信

使用TCP/IP协议建立socket连接，实现数据传递。

使用springboot集成socket实现用户间通信。

## 命令语句解析算法

对于一行语句（一个字符串），首先解析字符串头部是否是合法的指令，如果不是则报错提示。然后根据不同指令从字符串的后部解析出响应数量的参数，若不足则报错提示，然后执行该指令，若后续还有未解析完的语句，则进入下一个指令的解析过程中。