PC Logo

软件架构文档

版本 <1.0>

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| <2020/11/8> | <1.0> | 初次撰写完成软件架构文档 | 苏浩然 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目录

[1. 简介 4](#_Toc55766582)

[1.1 目的 4](#_Toc55766583)

[1.2 参考资料 4](#_Toc55766584)

[2. 用例视图 4](#_Toc55766585)

[2.1 命令行绘图 5](#_Toc55766586)

[2.2 命令文件操作 5](#_Toc55766587)

[2.3 调试 5](#_Toc55766588)

[2.4 网上双人绘图 5](#_Toc55766589)

[2.5 调用子过程 5](#_Toc55766590)

[2.6 语音绘图 5](#_Toc55766591)

[2.7 管理个人信息 5](#_Toc55766592)

[2.8 作品集 5](#_Toc55766593)

[3. 逻辑视图 6](#_Toc55766594)

[3.1 概述 6](#_Toc55766595)

[3.2 桌面应用程序(DesktopClient) 6](#_Toc55766596)

[3.2.1 View Tier 7](#_Toc55766597)

[3.2.2 Controller Tier 7](#_Toc55766598)

[3.2.3 Model Tier 7](#_Toc55766599)

[3.3 Server 7](#_Toc55766600)

[3.3.1 Service Tier 8](#_Toc55766601)

[4. 进程视图 8](#_Toc55766602)

[5. 部署视图 9](#_Toc55766603)

[6. 实现视图 9](#_Toc55766604)

[6.1 DesktopClient构件 10](#_Toc55766605)

[6.1.1 GUI构件 10](#_Toc55766606)

[6.1.2 UserCommand构件 11](#_Toc55766607)

[6.1.3 FileProcess构件 11](#_Toc55766608)

[6.1.4 Communication构件 11](#_Toc55766609)

[6.1.5 Interpreter构件 11](#_Toc55766610)

[6.2 Server构件 11](#_Toc55766611)

[6.2.1 Controller构件 11](#_Toc55766612)

[6.2.2 Service构件 11](#_Toc55766613)

[6.2.3 Dao构件 12](#_Toc55766614)

[7. 数据视图 12](#_Toc55766615)

[8. 核心算法设计 12](#_Toc55766616)

[9. 性能 13](#_Toc55766617)

[10. 质量 13](#_Toc55766618)

软件架构文档 （简化版）

# 简介

## 目的

本文档将从架构方面对系统进行综合概述，其中会使用多种不同的架构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做出的重要决策。

本文档对PC Logo软件的架构视图进行了概述，该软件开发者为软件工程原理第14组。

本文档依次对PC Logo的用例视图、逻辑视图、部署视图、实现视图、数据视图进行了简要概述，之后描述了软件中的核心算法，最后阐述了架构设计中为应对关键非功能性需求的设计战术。并没有为该软件设计进程视图，具体原因在进程视图一节会做出说明。

## 参考资料

[1]沈备军,陈昊鹏,陈雨亭.软件工程原理[M].北京:高等教育出版社,2013:104-107.

[2]软件架构文档示例,https://oc.sjtu.edu.cn/courses/23844/files/1609978

[3]PC Logo 系统术语表.v1.0.上海:上海交通大学,2020-09-26.

[3]PC Logo 软件需求规约.v1.1.上海:上海交通大学,2020-10-08.

# 用例视图

本部分包括对软件架构的用例视图的描述。用例视图描述了一些重要核心功能的场景和用例集，这些场景和用例能较全面地覆盖软件架构或者体现了架构中的特殊之处。

PC Logo软件的用例如下：

* 命令行绘图
* 命令文件操作
* 调试
* 网上双人绘图
* 调用子过程
* 语音绘图
* 管理个人信息
* 作品集

这些用例均由使用软件的用户发起。

用例图如下。

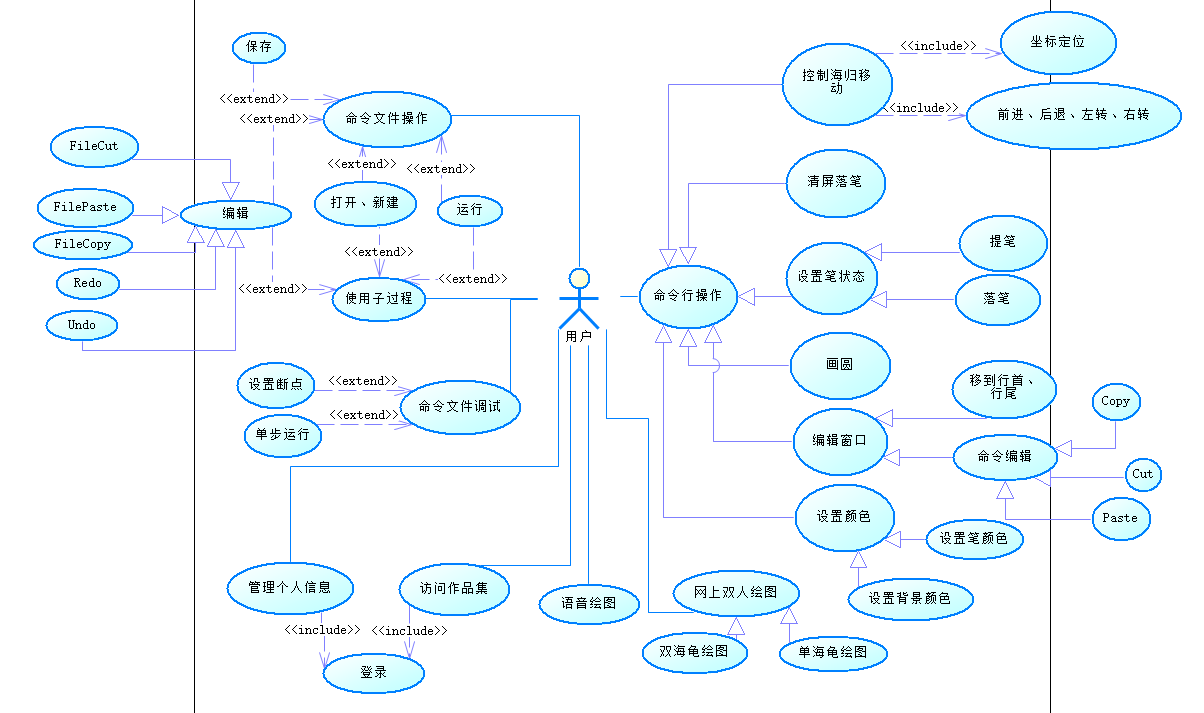


图 1用例图

与架构相关的核心用例如下。

## 命令行绘图

简介：用户在命令行输入指令，小海龟根据指令移动实现绘图。

## 命令文件操作

简介：用户可以对命令文件进行操作，包括新建、打开文件，编辑文件，运行文件中的指令，保存文件等。

## 调试

简介：用户可以在命令文件中设置断点，在运行时可以运行到断点处停止并单步运行。

## 网上双人绘图

简介：用户可以在网络上搜索其他用户并邀请一个其他用户共同绘图，可以双人双海龟绘图或双人单海龟绘图。

## 调用子过程

简介：用户可以创建并调用子过程。

## 语音绘图

简介：用户选择语音绘图后可以发出语音指令输入命令，点击运行后可以进行绘图。

## 管理个人信息

简介：用户登录后，可以查看、修改个人信息，包括昵称、邮箱等

## 作品集

简介：用户可以向自己的作品集中上传作品，也可以查看其他用户已经上传的作品。

# 逻辑视图

## 概述

PC Logo软件总体上采用了C/S架构，桌面应用程序和网页端作为客户端，与服务器端进行通信以实现登录、联机等功能。总体架构如下图。

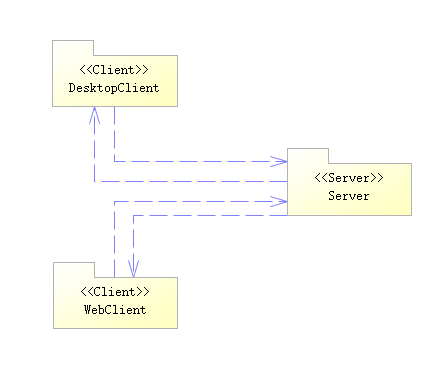


图 2 逻辑架构-总体

WebClient部分仅由网页构成，比较简单，不做赘述。下面说明桌面应用程序部分和Server的架构。

## 桌面应用程序(DesktopClient)

桌面应用程序部分架构如下图。

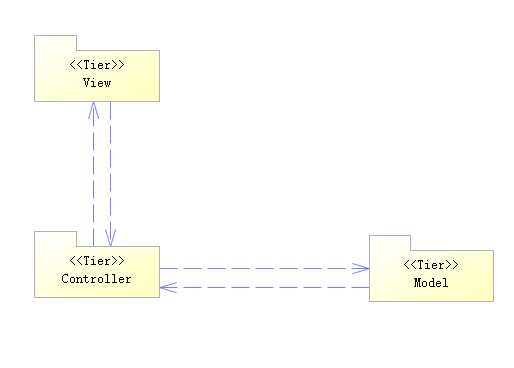


图 3逻辑架构-桌面应用程序

桌面应用程序的架构采取了MVC的风格，并且将功能实现（Model）与界面展示（View）解耦，通过Controller进行控制调度。View发生变化时，通过Qt中信号槽的机制告知Controller，再由Controller发送信号到Model进行处理，Model处理完后，发送信号给Controller，再由Controller发送信号告知View进行重新渲染。

### View Tier

主要负责渲染、展示用户界面。当用户进行操作时，将用户进行的操作以信号形式发送给Controller；当界面需要重新渲时，Controller会将信号发送给View，View收到信号后会进行渲染。

### Controller Tier

主要负责View层和Model层之间的信号转发以及调度。

### Model Tier

主要负责桌面应用程序具体服务和计算的实现，包括解释运行用户输入的Logo语言程序，进行文件操作等。其具体设计如下图。

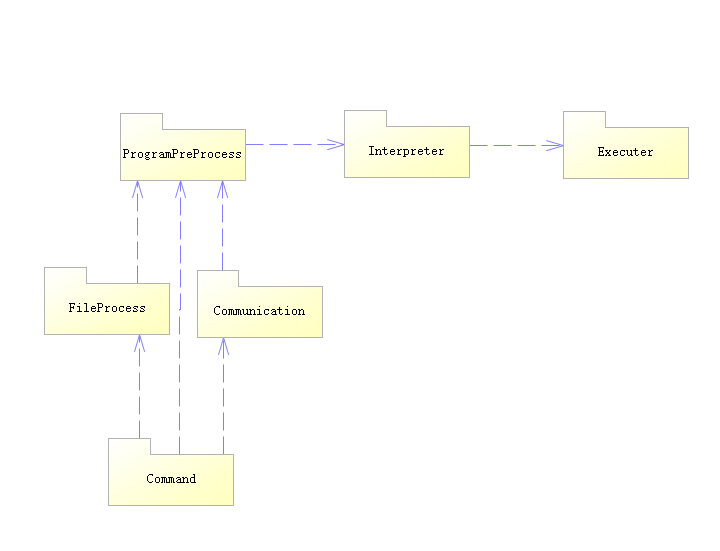


图 4 逻辑架构-Model层

Model层整体上采用了“管道-过滤器”风格的架构，Controller层转发过来的信号先由Command进行处理，判断需要执行的命令，如果涉及到文件操作，会调用FileProcess处理，如果涉及联机，会调用Communication进行处理。用户想要运行的程序最终会传递给ProgramPreProcess进行预处理，通过后会调用Interpreter进行解释，生成抽象语法树，然后调用Executer遍历语法树执行程序（发送信号给Controller进行渲染）。

## Server

Server端架构如下图。

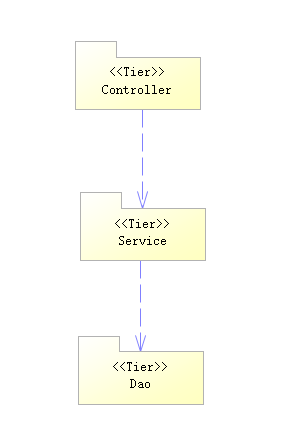


图 5 逻辑架构-Server

服务器端采用了分层架构，其中Controller层负责接收网络请求并调用相关服务，Service层负责具体的业务流程，Dao层负责与数据库进行通信，进行数据访问。由于本系统中服务器需要实现的功能比较简单，故不再对Controller层和Dao层做进一步阐述，下面描述Service层包含的服务。

### Service Tier

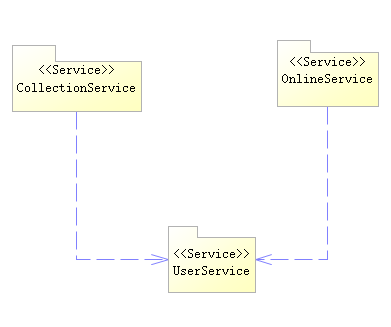


图 6 逻辑架构-Service层

Service层实现的服务比较简单，包括用户服务，作品集服务和联机服务。其中用户服务主要负责管理用户基本信息，作品集服务主要负责管理用户上传到自己作品集中的Logo程序，联机服务主要负责双人绘图时的搜索、联机等功能。其中作品集服务和联机服务都对用户服务有依赖。

# 进程视图

由于PC Logo软件的桌面应用程序部分是单进程单线程的桌面应用程序，且服务端使用Springboot框架，多线程并发由框架托管，不需要开发者实现，故本软件没有设计进程视图。

# 部署视图

PC Logo软件的部署视图如下图。

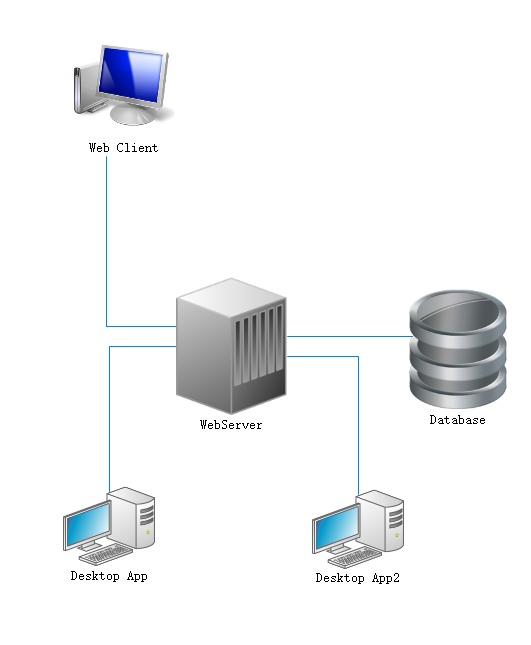


图 7 部署试图

本系统的客户端有两种，一种是Desktop App，即桌面应用程序，这是本系统中最主要的部分，软件的大部分功能由桌面应用程序独立实现；另一种客户端是Web端，部分功能（如作品集、帮助文档等）需要用户通过网页访问，故会有网页端。两种客户端均可以通过网络访问Web Server以实现登录等功能，两个不同的桌面客户端可以通过互联网访问Web Server进行联机，Web Server可以通过局域网访问数据库。

# 实现视图

PC Logo的实现视图如下图。

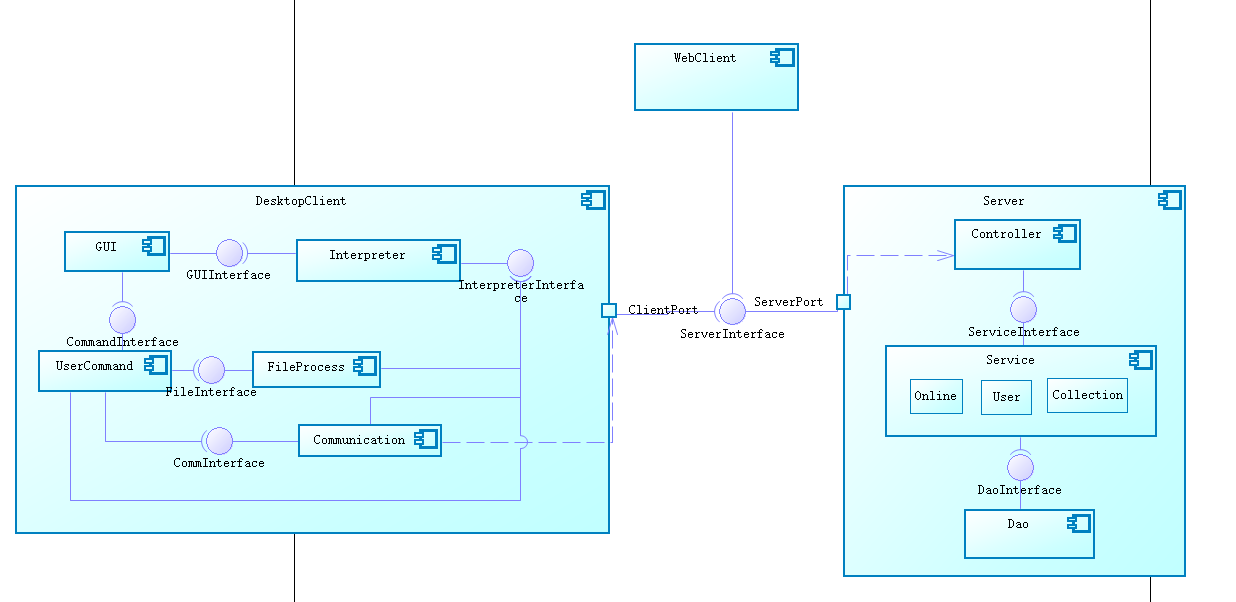


图 8 实现视图

实现视图基本与逻辑视图相符。软件总体上分为三个构件，即DesktopClient构件、WebClient构件和Server构件。下面主要说明DesktopClient构件和Server构件。

## DesktopClient构件

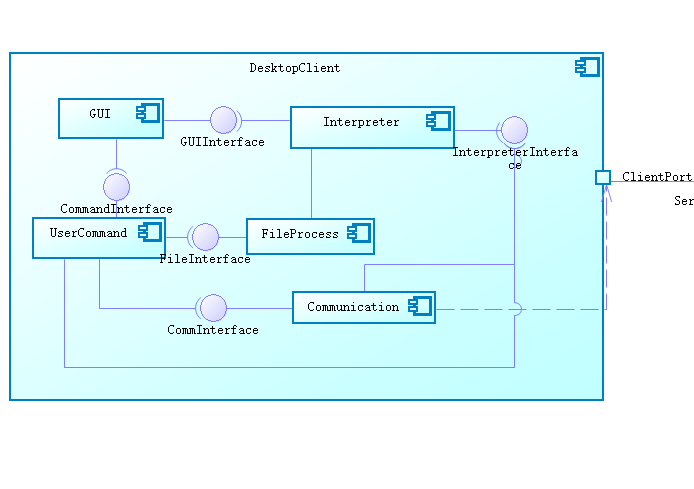


图 9 实现视图-桌面应用程序

本构件主要负责该软件绝大部分功能的实现。控制器构件负责信号的转发调度，简洁起见图中省略没有画出。其余子构件如下。

### GUI构件

负责用户界面的渲染展示，调用UserCommand构件提供的接口（通过控制器转发）以传递用户的操作或指令。提供重新渲染的接口给解释器以展示程序执行效果。

### UserCommand构件

为GUI构件提供接口接收指令，之后按照指令需求调用FileProcess或Communication或Interpreter构件的接口以完成操作。

### FileProcess构件

完成各种文件操作。提供接口给UserCommand，处理完文件后可能会调用Interpreter构件的接口以执行文件中的Logo程序。

### Communication构件

完成各种网络操作。提供接口给UserCommand，如果收到联机用户的程序可能会调用Interpreter接口完成联机绘图。该构件还可能通过DesktopClient的端口通过互联网调用服务构件提供的接口。

### Interpreter构件

负责对程序进行检查、解释和执行，提供接口给上述三个构件。程序执行后，通过调用GUI提供的接口（通过控制器转发）以将执行结果绘图展示。

## Server构件

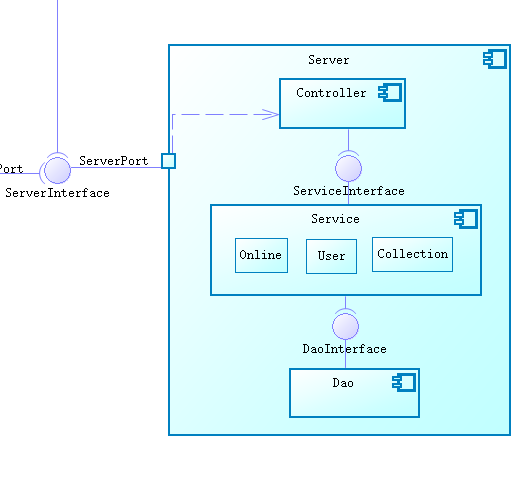


图 10 实现视图-Server

Server构件通过服务器端口给WebClient和DesktopClient提供接口。接口由Controller实现。

### Controller构件

主要负责接收网络请求并调用Service构件提供的接口完成服务。

### Service构件

负责实现具体的服务，包括用户、作品集、联机三部分。可能会调用Dao构件提供的接口以访问数据。

### Dao构件

负责对数据进行访问。

# 数据视图

本PC Logo软件需要在数据库中永久性存储的数据很少，主要为服务端存储用户个人信息及其作品集信息。数据间的关系也很简单，如下图。

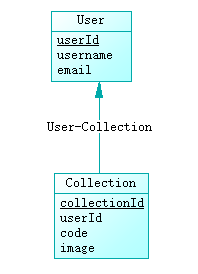


图 11 数据视图

上图说明了需要在服务端数据库中存储的数据，十分简单直观，不做赘述。

# 核心算法设计

下面给出递归解析Logo语言指令的核心算法。

- 词法分析

- 输入：string

- 输出：list[word]

- 解释：根据空格将输入串分割为一串word。

- word类型

- int: 十进制数字

- keyword: FD BK RT LT PU PD SETXY SETPC SETBG STAMPOVAL REPEAT PROCEDURE BEGIN END CALL

- symbol：[ ]

- 命令行语句解析

- 核心函数：parseLine(list[word], left, right)

即解析列表中下标从left到right 的单词串

- 输出：commandList

- commandList: {command, commandList}

- command: {type, args}

| type | args |

| ------------ | ----------------- |

| SETPC | color |

| SETBG | color |

| SETXY | x y |

| SETANGLE | angle |

| STRAIGHTMOVE | distance |

| OVALMOVE | x y |

| REPEAT | times commandList |

- 方法：递归解析

Eg. REPEAT 5 [ FD 100 REPEAT [ 4 RT 20 ] ] FD 20

通过词法解析得到

list[word] = {"REPEAT", 5, "[", "FD", 100, "REPEAT", "[", 4, "RT", 20, "]", "]", "FD", "20"}

调用parseLine(list[word], 0, 13)

返回commandList(command(REPEAT, 5, parseLine(list[word], 3, 10)), parseLine(list[word], 11, 12))

其中parseLine(list[word], 3, 10) 得到REPEAT command的第二个参数

parseLine(list[word], 11, 12) 得到返回的commandList的第二个参数

- 命令文件解析

- 输出：commandFile(commandList, procedureList)

- 对于非precedure部分正常解析为commandList 并进行连接

- procedureList中的procedure: name + commandList

识别到PROCEDURE关键字时，向下找到BEGIN和END，并将中间的语句解析为commandList

# 性能

本PC Logo软件对性能的主要需求如下。

* 吞吐量：每秒处理事务数为100
* 响应时间：在低于100并发的情况下，系统响应时间不超过2s

为了实现以上需求，架构中采取了将软件的主要功能集成于桌面应用程序的方式，服务端只实现登录等简单功能，降低了服务端的压力，从而可以提高性能；此外服务端使用了Springboot框架，由Springboot托管并发线程，也有助于实现较好的并发性能。

# 质量

本PC Logo软件对性能和质量的主要需求如下。

* 可用时间百分比：99.8%以上
* 使用小时数：1000小时
* 平均故障间隔时间：PC Logo的平均故障间隔时间为一个月
* 平均修复时间：PC Logo的平均修复时间为6小时

为了实现以上需求，架构中将主要功能分为不同模块实现，并将View层和Model层解耦，从而在发现某个功能或模块出现问题时可以及时修复或替换并且不影响其他模块正常工作，从而有助于降低修复时间，提高可用时间。