软件架构文档

面向智慧工厂的准实时监管系统

项目组组号：NO.1

项目组负责人：周汉辰

联系电话：18621966896

电子邮箱：1341634255@qq.com

2016 年 10 月

文档信息

标题	面向智慧工厂的准实时监管系统软件架构文档
作者	周汉辰
创建日期	2016/10/25
上次更新日期	2017/01/03
版本	V3.1
组号	NO.1

修改历史

日期	版本	说明	作者
2016/10/25	V1.0	初稿	周汉辰
2016/10/27	V2.0	增加用例视图、数据视图、进程视图	周汉辰、曹雨婷、张鹤腾
2016/11/08	V3.0	修改用例视图	曹雨婷、张鹤腾
2017/01/03	V3.1	增加 rule 说明	曹雨婷

目录

一、 简介

1. 目的

本文档将从构架方面对“面向智慧工厂的准实时监管系统”进行综合概述，其中会使用不同的构架视图来描述系统的各个方面。为系统的整体实现提供指导和依据。

2. 范围

本文档适用于“面向智慧工厂的准实时监管系统”项目开发的整个生命周期，包括项目的每个阶段，覆盖项目每一项工作任务。适用于所有参与本项目的相关成员以及相关组。但由于需求的改变以及对性能要求的提高，软件架构会随之进行调整，软件架构文档也会随之改变，因此不能保证软件架构文档的不变性。

3. 定义、首字母缩写词和缩略语

定义及缩略语	含义
可视化	利用计算机图形学和图像处理技术，将数据转换成图形或图像在屏幕上显示出来
数据推送前置	负责前端和后台的数据进行通信的模块
数据组装	项目核心模块，通过读取配置数据将源数据组装成目标数据的模块
配置管理	与用户进行交互，负责生成源数据与目标数据的配置信息
数据获取	与其他业务数据库进行交互，通过配置信息获取源数据
数据池	数据中心，存放配置数据、源数据和目标数据
目标数据	用户需要监控的数据，可能是源数据本身，也可能是多个源数据的组合
源数据	可以通过其他业务系统交换数据库直接获取的数据

4. 概述

本文档将明确“面向智慧工厂的准实时监管系统”的构架表示方式、构架的目标和约束、性能和质量等，并通过一系列视图来表示“面向智慧工厂的准实时监管系统”的软件构架，视图包括：用例视图、逻辑视图、部署视图、数据视图和进程视图。使用 visio 或 Enterrise Architecture 进行 UML 建模。

二、 构架目标和约束

1. 性能

Performance1：请求的响应时间要在 2 秒以内；
Performance2：系统允许多线程并发；
Performance3：系统的数据量为 MB 级。

2. 安全性

Safety1：配置管理系统应该只允许经过验证和授权的用户访问。

3. 可维护性

Modifiability1：在系统的基本源数据格式发生变化时，系统应该不受影响；
Modifiability2：可增加新的源数据和目标数据。

4. 可用性

Availability1：系统的可用性要达到 95%。

5. 可靠性

Reliability1：客户端与服务器通信时，如果网络故障，系统提醒网络故障。

6. 易用性

Usability1：系统应尽可能多地让用户以点击鼠标方式完成任务。
Usability2：配置管理员经过简单培训就可以对目标数据与源数据进行配置。

7. 业务规则

BR1：目标数据的产生规则支持源数据的加、减、乘、除、均值等运算；
BR2：数据的刷新频率为：秒、分、时等；
BR3：数据类型为：单值或列表。

8. 约束

CON1：客户端系统将运行在 Windows 7，Windows 8，Windows 8.1、Windows XP，MacOS，Linux 操作系统上；

CON2：客户端系统使用宽显示屏显示最终界面；
CON3：系统要主要使用 Java 语言进行开发；
CON4：用户因在多地点工作，监管系统要在网络上分布为多个服务器和多个客户端。

三、 用例视图

1. 订单数据显示功能

该功能包含 3 个子功能：查看今日新增订单数据、查看销售数据排名和查看客户区域分布饼图。

查看今日新增订单数据包含：今日新增订单数量，今日新增订单金额、今日新增订单的分布地区、今日排产订单、订单完成及时率（包括：加急订单、变更订单和异常订单）、追踪订单状态（包括：处理中、未处理和紧急）、订单内容分类饼图。

查看销售数据排名包含：品牌订单量 TOP3 柱状图，面料订单量 TOP3 柱状图。

查看客户区域分布：显示不同区域的用户分布饼图。

用例图如图 4-1 所示。

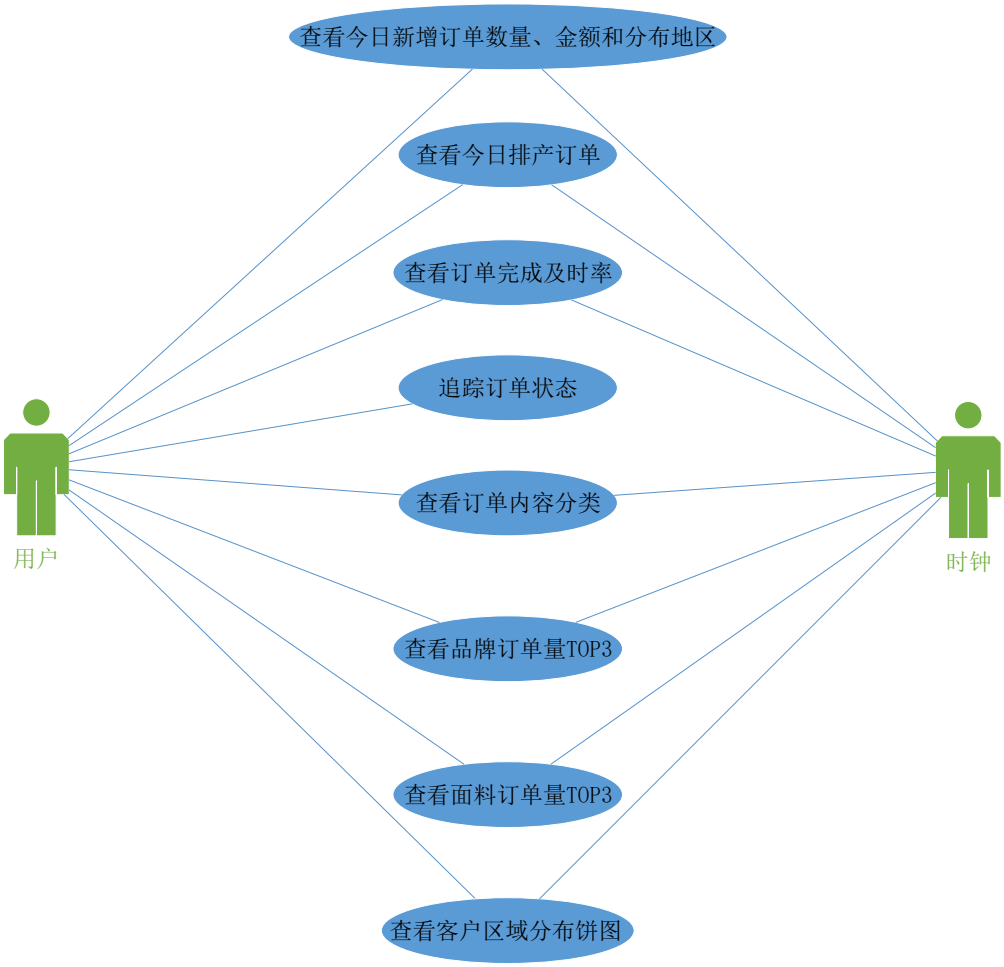

图 4-1 订单数据显示功能用例图

2. 客服数据显示功能

该功能用于查看客服数据，包含：查看客服工单数量和查看客服在线状态（包括：在线人数和总人数）。

用例图如图 4-2 所示。

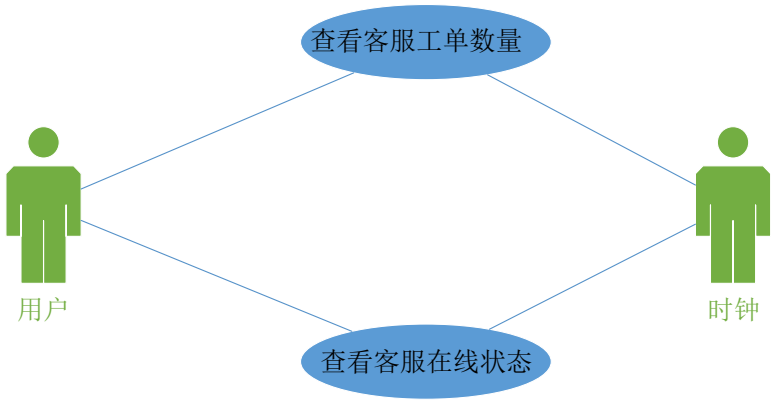

图 4-2 客服数据显示功能用例图

3. 智能吊挂系统监管功能

该功能包含：查看不同产线的吊挂系统状态（包括：已完成数和总数）、查看工作站暂压、查看配对工作站时间效率、查看暂压的裁片数量和查看服装配对事件。

用例图如图 4-3 所示。

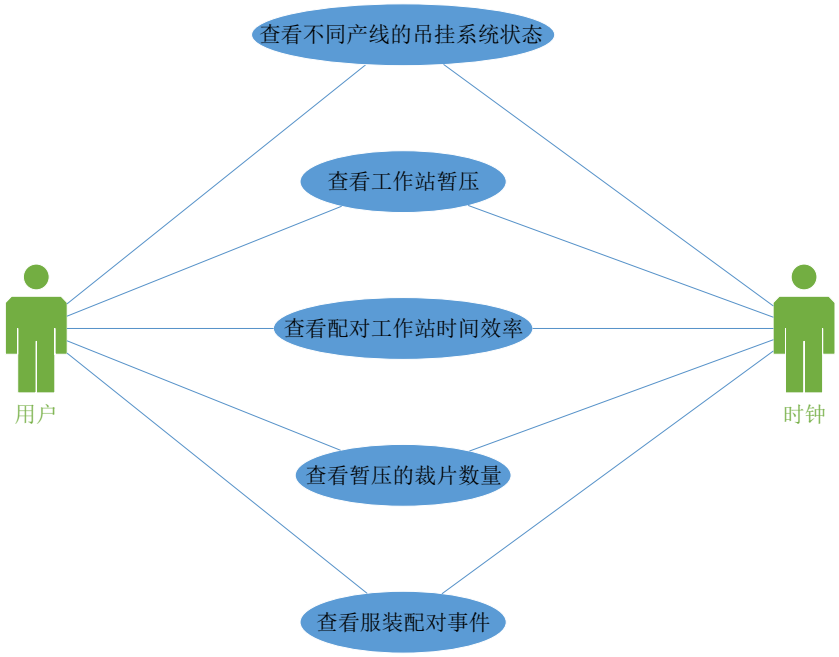

图 4-3 智能吊挂系统监管功能用例图

4. 自动裁床系统监管功能

该功能包含：查看不同品牌的裁床状态（包括：正在使用数和总数）、查看有效工作时间、查看刀头的移动时间和查看平均裁剪数量。

用例图如图 4-4 所示。

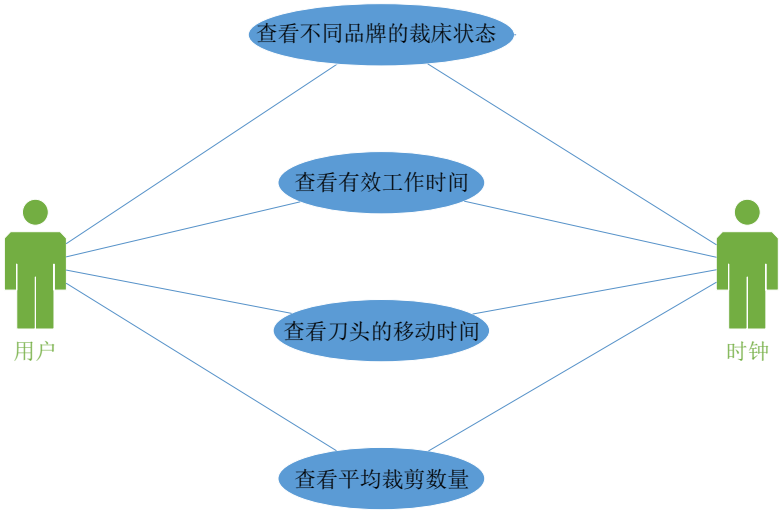

图 4-4 自动裁床系统监管功能用例图

5. 智能排程系统监管功能

该功能包含 2 个子功能：查看正在打版、裁剪、缝制、整烫、包装的产品数量和查看排产信息。

查看排产信息包含：查看排产成功率、查看排产冲突率和查看排产失败数量。

用例图如图 4-5 所示。

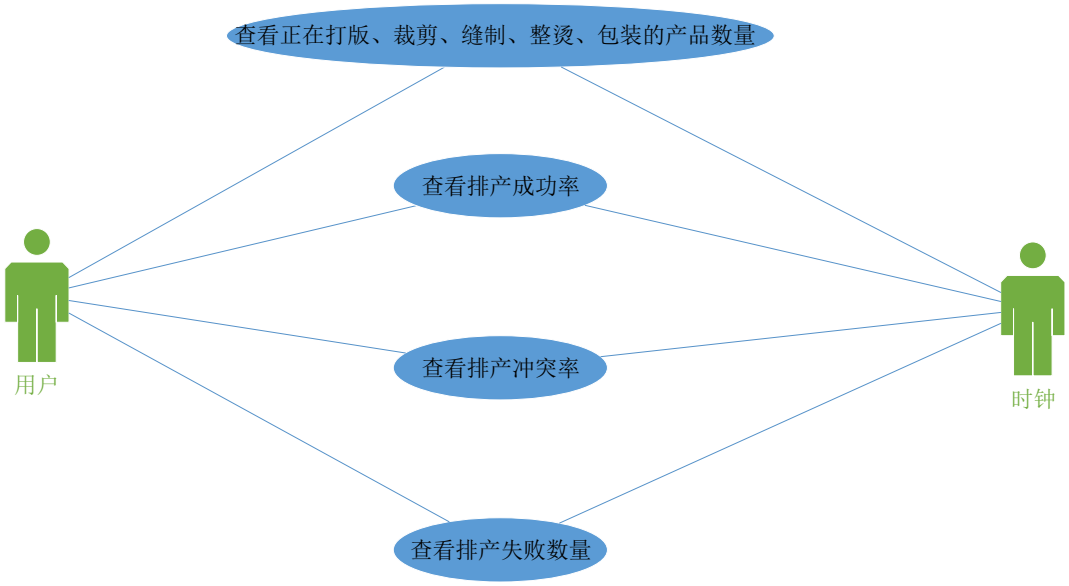

图 4-5 智能排程系统监管功能用例图

6. 监管设备功能

该功能包含 4 个子功能：查看各个 AVG 小车的当前站、停留时间和运送面料，查看其他设备的开机率（包括：已开机设备数和总设备数）和设备运行状态，查看机房环境的温度、湿度和电压（仪表盘显示）以及查看关键资源可用性折线图。

查看关键资源可用性包含：查看主机可用性、查看数据库可用性和查看中间件可用性。

用例图如图 4-6 所示。

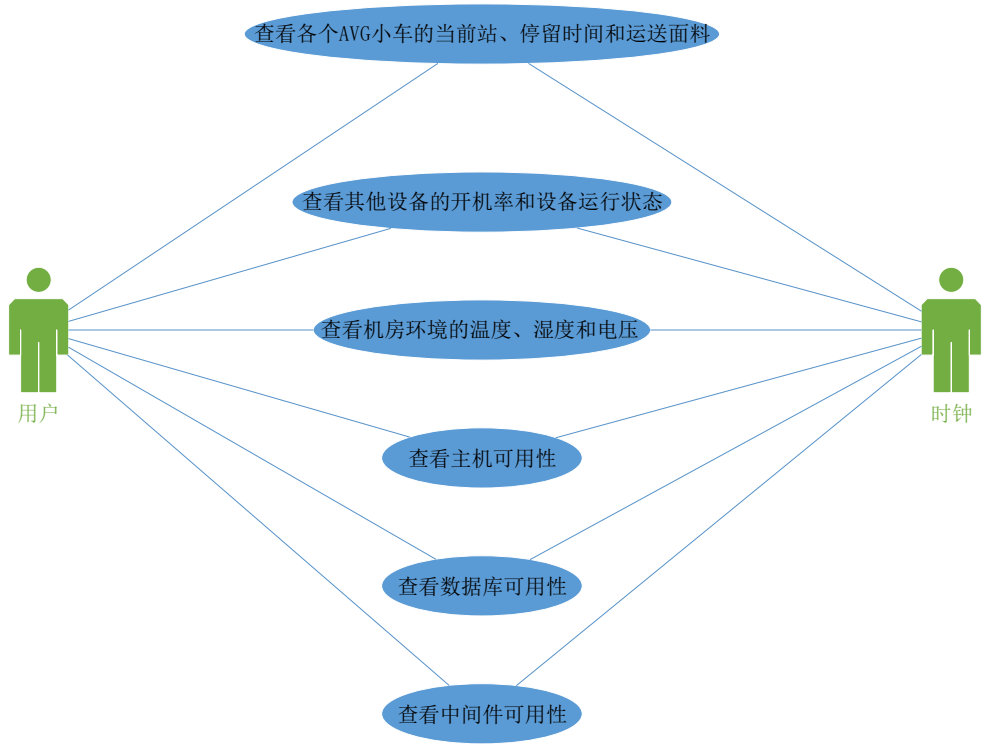

图 4-6 监管设备功能用例图

7. 智能生产控制功能

该功能包含：查看各个车间的出勤（包括：出勤人数和总人数），查看各个车间的日产量和人均产量，查看各个车间的平均工时和返工率以及查看产品质量（包括：一次检验合格率和返修合格率）、不合格原因分析饼图及问题栈点 TOP3。

用例图如图 4-7 所示。

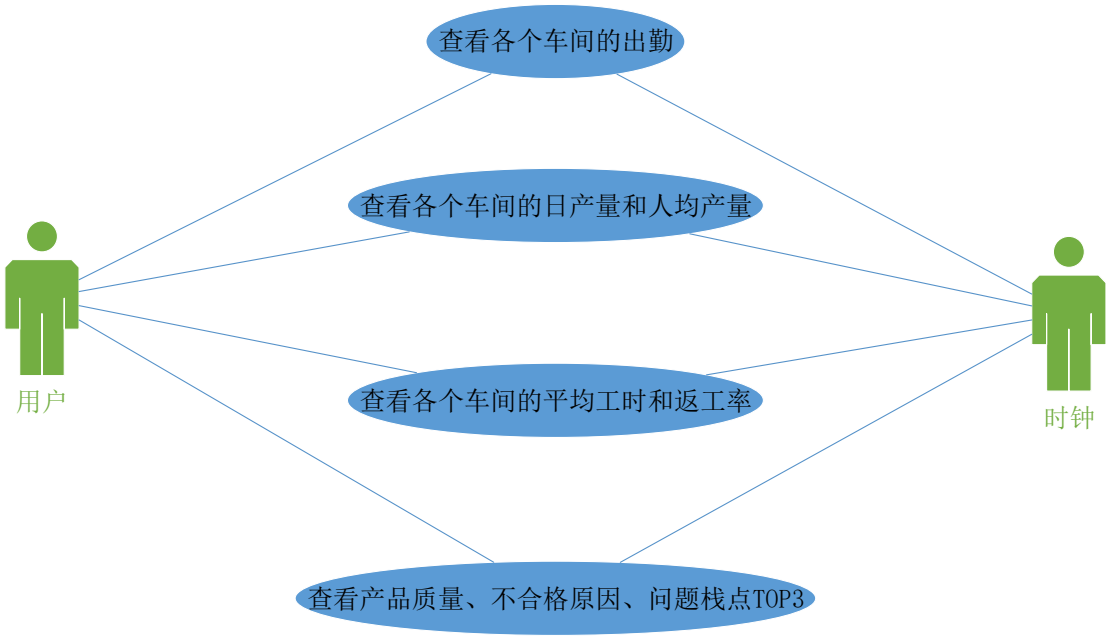

图 4-7 智能生产控制功能用例图

8. 供应链监管功能

该功能包含 2 个子功能：查看今日订单的面料、辅料、配件、供货进度和供货及时率以及查看计划达成率。

查看计划达成率（条形图显示）包含：查看采购计划达成率、查看外协发货计划达成率和查看批发出库计划达成率。

用例图如图 4-8 所示。

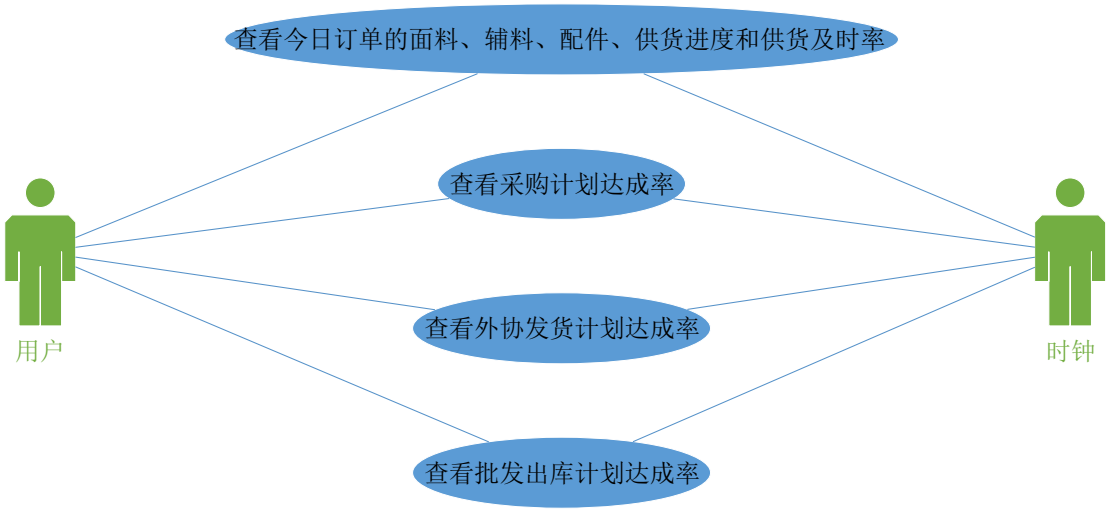

图 4-8 供应量监管功能用例图

9. 数据模型配置管理功能

该功能包含：配置源数据和配置目标数据。

用例图如图 4-9 所示。

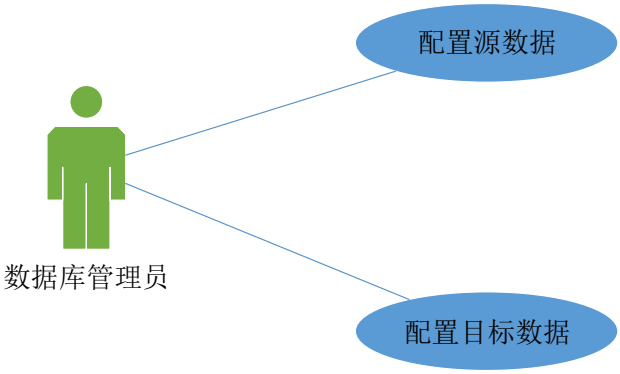

图 4-9 数据模型配置管理功能用例图

10.多数据库获取功能

系统能够使用多线程从不同业务系统的不同数据库（包括 MySQL、SQL Server）中同时进行多个源数据的获取，并使用多线程同时进行多个目标数据的组装。服务器端能够监听客户端请求，返回客户端请求的目标数据。

四、 逻辑视图

1. 概述

本项目的中心是数据，所以采用仓库风格的架构。项目一共分为五个模块，分别是：数据可视化模块、数据推送前置模块、数据组装模块、数据获取模块和配置管理模块，所有模块围绕数据池工作。

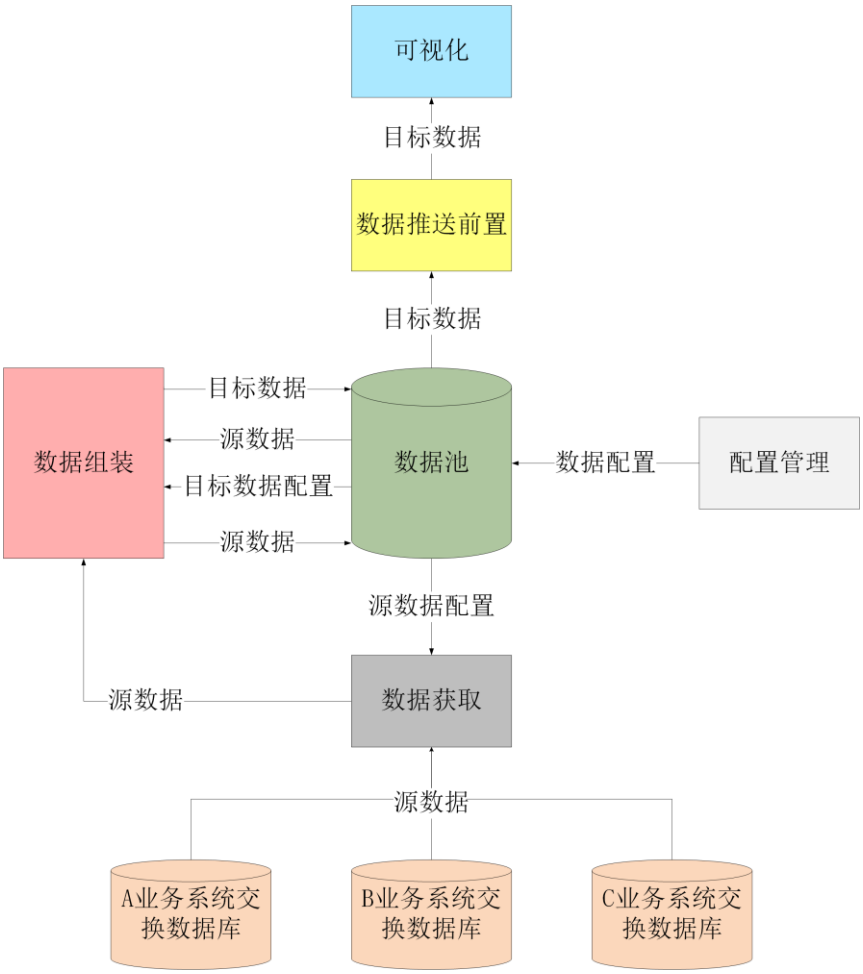

图 5-1 架构逻辑视图

当需要展示某一目标数据时，流程如下：

1）配置管理员将目标数据与目标数据所对应的源数据在配置管理模块进行配置；

2）配置模块将配置数据放入数据池中；

3）数据组装模块读取目标数据配置；

4）数据组装模块向数据获取模块发送获取源数据请求；

5）数据获取模块从数据池中读取源数据配置；

6）数据获取模块通过配置从其他业务系统交换数据库中读取源数据；

7）数据获取模块将源数据返回给数据组装模块；

8）数据组装模块将源数据放入数据池中；

9）数据组装模块以固定频率组装目标数据并放入数据池中；

10）可视化模块以固定频率向数据推送前置模块请求目标数据；

11）数据推送前置模块从数据池中获取对应目标数据并返回给可视化模块；

12）可视化模块对目标数据进行可视化。

2. 可视化模块

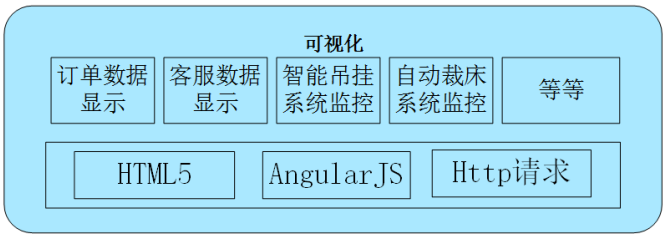

图 5-2 可视化模块

可视化模块负责数据的可视化展现，使用 HTML5、AngularJS 技术与用户交互，使用 Http 请求与数据推送前置交互。

3. 数据推送前置模块

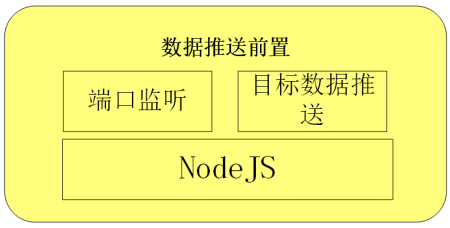

图 5-3 数据推送前置模块

数据推送前置模块接受客户端请求，将目标数据从内存数据集返回到客户端，内存数据集使用 MemCached 高速缓存系统，减少数据库压力，提升效率。网络通信连接使用 NodeJS，服务器端监听客户端请求，建立连接。

4. 数据组装模块

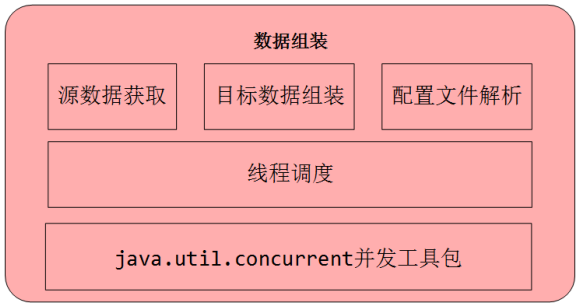

图 5-4 数据组装模块

数据组装模块中的数据读取线程将数据获取模块取得的数据存入内存 DB，数据组装线程从内存 DB 中取源数据并按照配置管理文件中的数据计算规则计算目标数据。

5. 数据获取模块

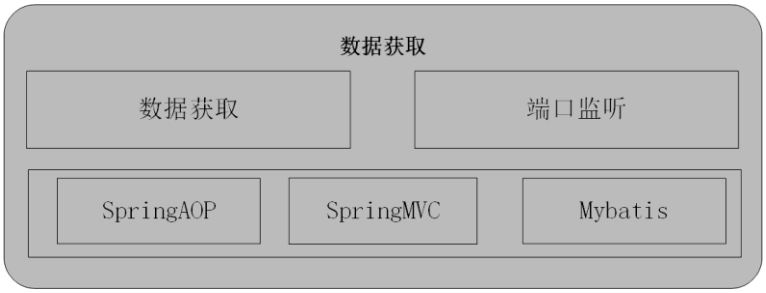

图 5-5 数据获取模块

数据获取模块使用 Spring aop+SpringMVC+MyBatis 从多个不同的数据库中获取数据，提供给数据组装模块。

6. 配置管理模块

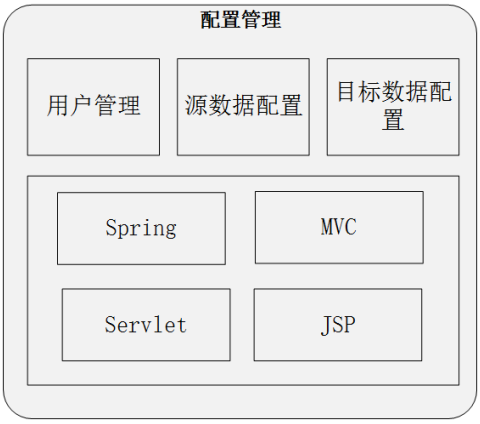

图 5-6 配置管理模块

使用 MVC 架构，采用 Spring 注解，通过 JSP 向服务器发送 Servlet 请求。配置管理模块根据配置管理员的输入，生成特定的 json 配置文件，储存源数据和目标数据的类型、刷新频率、来源等信息。

7. 数据池

图 5-7 数据池

数据池采用 MemCached 存放源数据与目标数据，MongoDB 作为配置数据的持久化平台，MySQL 作为目标数据的备份数据库。

五、 部署视图

1. 部署视图及描述

配置管理、业务系统数据获取、业务分析、管理控制、Memcached 在不同服务器上进行部署，方便今后进行分布式拓展。拓展方案如下：

1）业务系统数据获取：多服务器，为每一个不同的业务系统单独部署。

2）业务分析：多台服务器处理不同业务，一台主服务器制定业务分解策略，多个从服务器之间采用硬件负载均衡，直接在外部网络和服务器之间安装负载均衡器，设备专业性能优越。

3）Memcached：自身设计方便较好的分布式拓展，可多点部署。

4）管理控制：针对不同业务开启多端口监听，不同端口可使用不同协议。

因此，部署视图如图 6-1 所示。

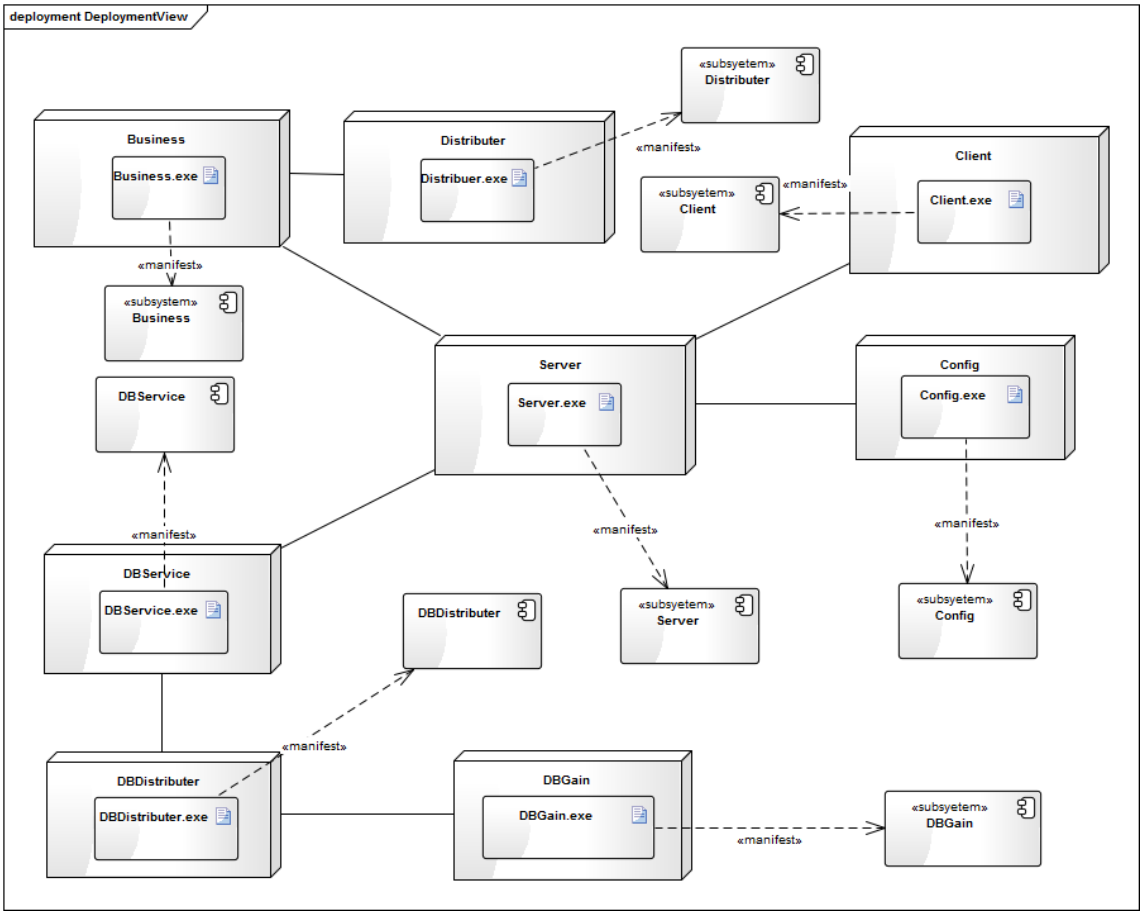

图 6-1 部署视图

可视化模块：Client：

数据推送前置模块：Server（监管服务器）

数据组装模块：Business（业务分析主从服务器）、Distribuer（业务分析分解策略制定）

数据获取模块：DBService（业务系统数据获取服务）、DBGain（业务系统数据获取多服务器）、DBDistributer（数据获取多服务器分解策略制定，目前：为每一个不同的业务系统单独部署）

配置管理模块：Config（配置管理服务器）

2. 硬件设施分布

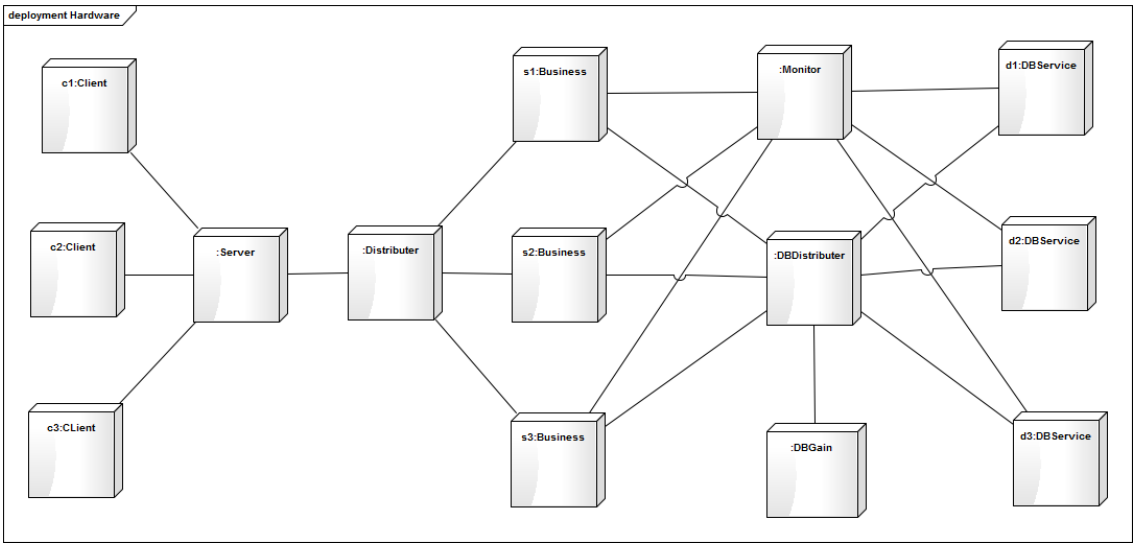

图 6-2 硬件设施分布描述

Client：客户端可视化

Server：数据推送前置

Distributer：数据组装多服务器分解策略制定

Business：数据组装

DBDistributer：数据库多服务器分解策略制定，目前：为每一个不同的业务系统单独部署

DBGain、DBService：数据获取

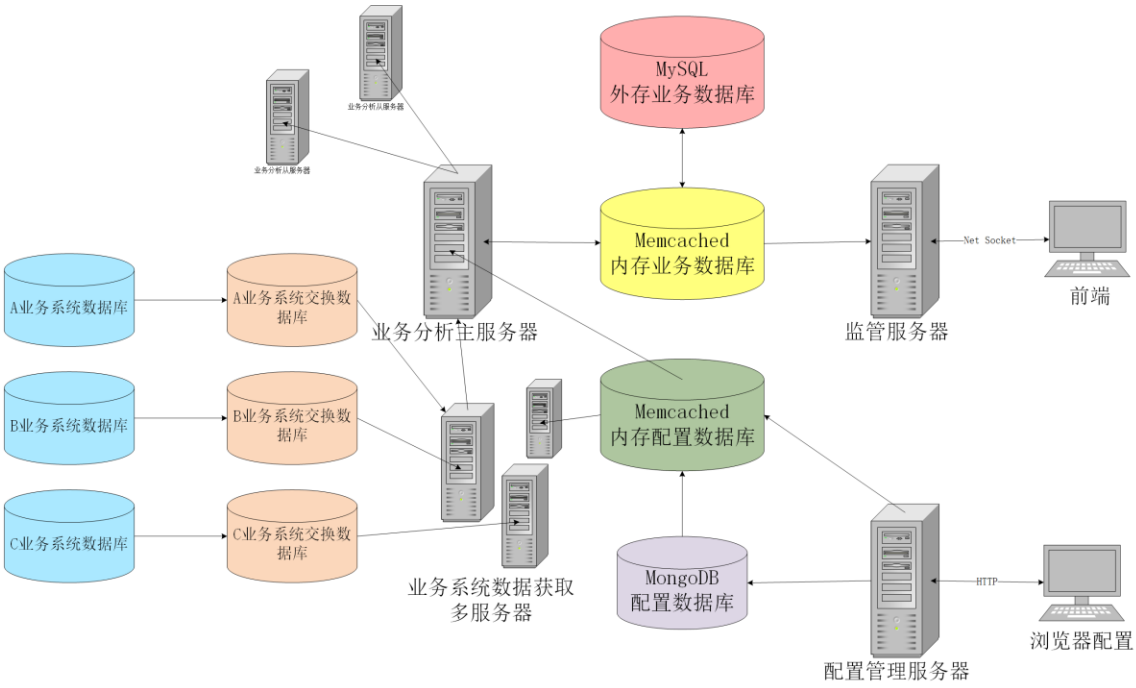

图 6-3 部署实际示意图

六、 数据视图

本项目使用非关系型数据库，数据格式将通过 json 形式展示，存储在 MongoDB 数据库中。

1. 源数据配置

表名：sourceData

```
{
  "_id": ObjectId(""),
  "name": "",
  "type": "single || list",
  "tupleNum": "5",
  "frequency": "1m",
  "source": "string",
  "SQL": "string"
}
```

字段说明：

id：主键，唯一
name：源数据名
type：源数据类型，单值或列表
tupleNum：源数据单项的元素个数
frequency：源数据刷新频率
source：数据库配置信息来源
SQL：数据获取 SQL

2. 目标数据配置

表名：goalData

```
{
  "_id": ObjectId(""),
  "name": "",
  "type": "single || list",
  "tupleNum": "5",
  "frequency": "10",
  "rule": {
    “ruleName”: “sort”,
    “key”: “value”,
    “order”: “desc”
  },
  "dataSourceList": [
        {
            "name": "s_sourcedata",
            "frequency": 10
        },
```

```
        {
            "name": "s_sourcedata",
            "frequency": 10
        }
    ]
}
```

字段说明：

id：主键，唯一

name：目标数据名

type：目标数据类型，单值或列表

tupleNum：目标数据单项的元素个数

frequency：目标数据刷新频率，以秒为单位

rule：目标数据的产生规则

允许为{}；

rule 不为{}时：

ruleName：值为"sort"或"exp"，分别表示排序和按表达式计算

key：规则应用的字段

order：ruleName 值为 sort 时，表示排序方式，值为"desc"或"asc"

expression：ruleName 值为 exp 时，表示计算目标数据的表达式内容，操作数和操作符之间以空格隔开

dataSourceList：目标数据所需要的源数据名列表,name 和 frequency 成对出现

rule 的三种情况示例如下：

```
"rule": {}
"rule": {
    “ruleName”:“sort”,
    “key”:“value”,
    “order”:“desc”
}
"rule": {
    “ruleName”:“exp”,
    “key”:“value”,
    “expression”:“s_1+s_2”
}
```

3. MemCached 配置

MemCached 中不需要表名

存储用 set（key，value，lifetime）

```
{
  "name": "",
  "type": "list || single",
  "sort": "",    //当 type=list 才有，否则为空
  "sortKey": "",  //当 type=list 才有，否则为空
  "frequency": "",
```

```
    "tupleNum": "",
    "values": ""    //计算结果
}
```

七、 进程视图

图 5-1 描述了系统的主要进程关系。用户通过配置管理模块向 MemCached 中写入源数据以及目标数据的配置，数据获取服务拿到源数据的配置，数据组装线程与数据获取线程拿到目标数据的配置。数据获取线程会根据相应的配置以一定的周期向数据获取服务请求数据，并将数据写入 MemCached；而数据组装线程则根据配置，从 MemCached 读入源数据，进行组装后将目标数据写回 MemCached。用户配置好后，可以请求配置好的目标数据，系统会从 MemCached 读出。

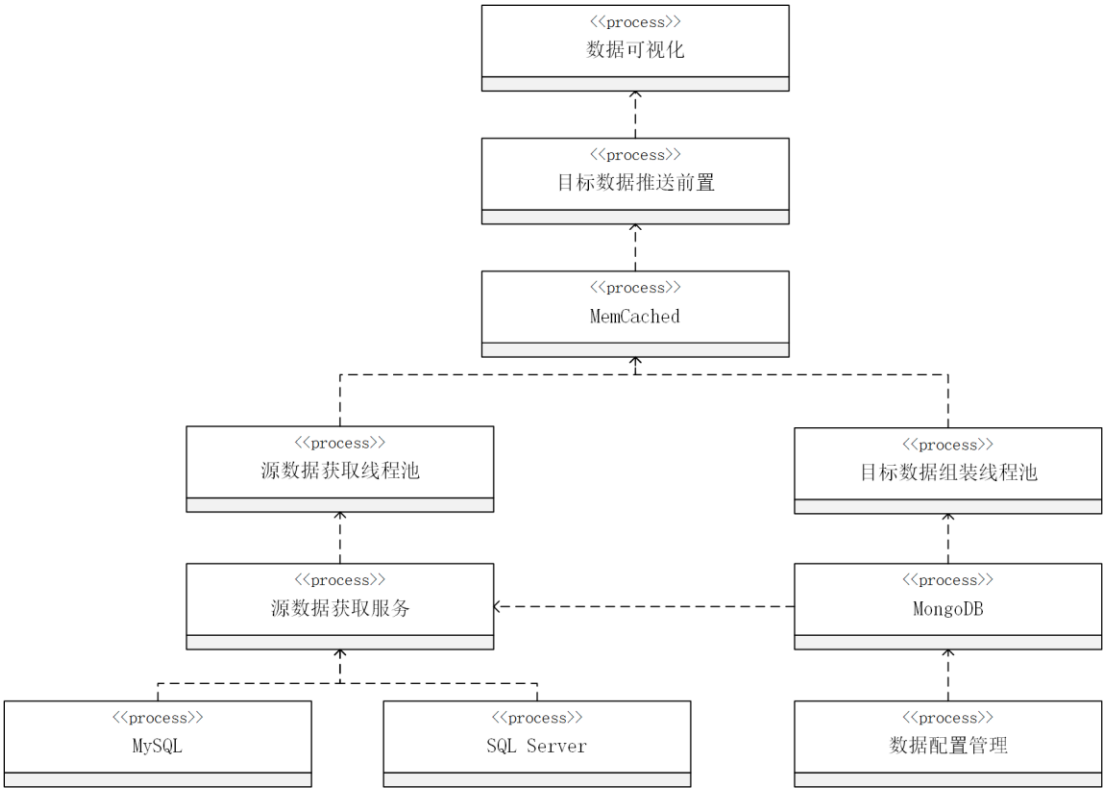

图 5-1 进程视图