****



**研 究 生 毕 业 论 文**

**（申请工程硕士学位）**

|  |  |
| --- | --- |
| **论文题目** | 基于Play的生产线数据管理系统的  设计与实现 |
| **作者姓名** | 陈一玮 |
| **学科、专业名称** | 工程硕士(软件工程) |
| **研究方向** | 软件工程 |
| **指导教师** | 李传艺 |

**20**18**年** 5 **月** 16 **日**

**学 号： MF1632009**

**论文答辩日期： 年 月 日**

**指 导 教 师： （签字）**

**基于Play的生产线数据管理系统的**

**设计与实现**

|  |  |
| --- | --- |
| **作 者:** | **陈一玮** |
| 指导教师: | **李传艺** |

|  |
| --- |
| **南京大学研究生毕业论文** |
| **(申请工程硕士学位)** |

|  |
| --- |
| **南京大学软件学院** |
| **20**18**年**5**月** |

**The Design and Implementation of Product Line Management System Based on Play Framework**

**Chen, Yiwei**

**Submitted in partial fulfillment of the requirements for the degree of Master of Engineering**

Supervised by

Lecturer **Li,Chuanyi**

Software Institute

**NANJING UNIVERSITY**

Nanjing, China

May, 2018

# 摘 要

随着新一代的信息技术的高速发展，以及劳动力、原材料与能源成本迅速增长，各个企业开始认识到利用自动和信息化的手段改进制造过程、寻求产业升级的重要性，世界各国都在加速新技术和新的工业革命。当前企业拥有工厂大量的信息数据，如何管理好这些数据并能够充分利用这些数据为生产创造价值，为工厂提高效率，以及帮助产品制造商降低生产成本是公司目前首先要考虑的问题。

面对以上这些问题，生产线机器数据管理系统就显得极其重要，该系统通过使用系统实现传统工厂设备和生产的状态高度透明化，提升产能和生产效率。Play框架是一个快速的全栈Web开发框架，因此系统选择它来实现系统部分功能，Play将应用分为表现层和模型层，遵循了MVC模式，表现层又分为视图和控制层。由于Play的CRUD模块不能实现较复杂的页面，因此部分页面采用AngularJS，数据库使用Oracle。

本文按照功能模块对系统进行总体设计，在文中结合了实际情况从多个角度分析系统需求，并根据详细设计来对各个模块进行实现展示。基于本平台的一个重要的应用是生产线调度，能够帮助生产计划部门在选择生产线时不必做很多工作就得到一个合理的且节省成本的生产方案。多个生产线调度问题是服务组合的一个特殊的应用场景，本篇论文也会给出生产线调度的具体算法。项目整体目标是优化生产线利用率，最佳循环时间，减少变更和增大工厂间的透明度。

**关键词**：工业、数据管理、Play框架、生产线调度、服务组合

# Abstract

With the development of information technology and rising cost of labor, raw material, and energy sources, it is generally realized by enterprise managers that manufacture improving and industry upgrading can be achieved with automated and information-based methods. World-wide countries are promoting new technologies and industrial revolution. So far, companies have a large amount of industrial data, how to manage and make full use of these data for production, improve efficiency for plants, and reduce production costs for product manufacturers are the most important issues that companies must consider.

Product line management system plays an important role in plants. It can achieve a high degree of transparency in the state of plant equipment and production, improve productivity and production efficiency in traditional plants. Play framework is a full-stack framework, it follows the MVC pattern and divides the application into presentation and model layers, where the presentation layer can be divided into the view and control layers. As Play CRUD cannot implement more complex pages, some pages are developed using AngularJS in this system, and database is oracle.

In this thesis, the system is designed in accordance with the function module, and the system requirements are analyzed in several aspects, this thesis will explain how to implement the system. Product line scheduling is an important function based on this platform, which helps to get a reasonable solution to choose a line. Product line scheduling is also a special application of service composition, this thesis will give the algorithm for the product line scheduling. The project goal is to optimize utilization of the line, reduce changes and increase transparency among plants.

**Keywords**: Industry, data management, Play framework, product line scheduling, service composition

# 

**目 录**

[摘 要 I](#_Toc514886057)

[Abstract II](#_Toc514886058)

[图 目录 V](#_Toc514886059)

[表目录 VII](#_Toc514886060)

[第一章 引言 1](#_Toc514886061)

[1.1 项目背景 1](#_Toc514886062)

[1.2 现状分析及必要性 2](#_Toc514886063)

[1.3 解决的问题及本文主要研究的工作 4](#_Toc514886064)

[1.4 本文的组织结构 6](#_Toc514886065)

[第二章 技术综述 7](#_Toc514886066)

[2.1 Play框架 7](#_Toc514886067)

[2.2 前端使用技术 9](#_Toc514886068)

[2.2.1 AngularJS 9](#_Toc514886069)

[2.2.2 Play CRUD 10](#_Toc514886070)

[2.3 Json Web Token 11](#_Toc514886071)

[2.4 Java Persistence API 12](#_Toc514886072)

[2.5 基于粒子群算法的服务组合 13](#_Toc514886073)

[2.6 本章小结 14](#_Toc514886074)

[第三章 工厂生产线数据管理系统的分析与设计 15](#_Toc514886075)

[3.1 系统需求分析 15](#_Toc514886076)

[3.1.1 项目总体规划 15](#_Toc514886077)

[3.1.2 用户特点 15](#_Toc514886078)

[3.1.3 功能要求 16](#_Toc514886079)

[3.1.4 非功能要求 17](#_Toc514886080)

[3.1.5 系统用例 17](#_Toc514886081)

[3.2 系统总体设计 24](#_Toc514886082)

[3.2.1 系统模块设计 24](#_Toc514886083)

[3.2.2 系统类结构设计 24](#_Toc514886084)

[3.3 系统详细设计 26](#_Toc514886085)

[3.3.1 工厂数据总览模块详细设计 26](#_Toc514886086)

[3.3.2 生产线搜索模块详细设计 29](#_Toc514886087)

[3.3.3 参数维护模块详细设计 30](#_Toc514886088)

[3.3.4 数据统计导出模块 31](#_Toc514886089)

[3.4 系统数据库设计 31](#_Toc514886090)

[3.5 本章小结 34](#_Toc514886091)

[第四章 工厂生产线数据管理系统的实现 35](#_Toc514886092)

[4.1 数据连接及处理的实现 35](#_Toc514886093)

[4.2 工厂数据总览模块 36](#_Toc514886094)

[4.3 生产线搜索模块 38](#_Toc514886095)

[4.4 参数维护模块 40](#_Toc514886096)

[4.5 数据统计导出模块 41](#_Toc514886097)

[4.6 生产线组合的实现 44](#_Toc514886098)

[4.6.1 问题描述 44](#_Toc514886099)

[4.6.2 问题建模 45](#_Toc514886100)

[4.6.3 解决方案 48](#_Toc514886101)

[4.6.4 仿真实现和结果 54](#_Toc514886102)

[4.7 本章小结 56](#_Toc514886103)

[第五章 总结与展望 57](#_Toc514886104)

[5.1 总结 57](#_Toc514886105)

[5.2 进一步工作展望 57](#_Toc514886106)

[参 考 文 献 58](#_Toc514886107)

[致 谢 61](#_Toc514886108)

[版权及论文原创性说明 62](#_Toc514886109)

# 

# 图 目录

[图1.1数据治理维度 3](#_Toc514886284)

[图2.1 Play框架结构 7](#_Toc514886285)

[图2.2 Play在路由中绑定方法 8](#_Toc514886286)

[图2.3事件模型服务器工作流程 8](#_Toc514886287)

[图2.4 AngularJS中MVVM模式的运用 10](#_Toc514886288)

[图2.5启动CRUD模块 10](#_Toc514886289)

[图2.6 CRUD模块配置路由 11](#_Toc514886290)

[图3.1系统用例图 18](#_Toc514886291)

[图3.2系统总体结构 24](#_Toc514886292)

[图3.3系统类图 25](#_Toc514886293)

[图3.4生产线查询时序图 26](#_Toc514886294)

[图3.5生产线调整时序图 27](#_Toc514886295)

[图3.6添加新机器时序图 27](#_Toc514886296)

[图3.7修改机器信息时序图 28](#_Toc514886297)

[图3.8删除机器时序图 29](#_Toc514886298)

[图3.9搜索生产线时序图 29](#_Toc514886299)

[图3.10添加可选参数时序图 30](#_Toc514886300)

[图3.11修改参数时序图 30](#_Toc514886301)

[图3.12数据导出时序图 31](#_Toc514886302)

[图3.13与用户相关联的表 32](#_Toc514886303)

[图3.14与机器关联的表 33](#_Toc514886304)

[图3.15数据流图 34](#_Toc514886305)

[图4.1 Play框架数据库连接配置 35](#_Toc514886306)

[图4.2数据模型实体类 35](#_Toc514886307)

[图4.3数据模型实体类操作 36](#_Toc514886308)

[图4.4配置操作方法路由 36](#_Toc514886309)

[图4.5工厂数据总览模块的实现流程 37](#_Toc514886310)

[图4.6具体工厂详情页 37](#_Toc514886311)

[图4.7具体机器信息查看页面 38](#_Toc514886312)

[图4.8生产线搜索实现流程图 39](#_Toc514886313)

[图4.9生产线查询模块 39](#_Toc514886314)

[图4.10继承CRUD的相关类 40](#_Toc514886315)

[图4.11参数维护页面 41](#_Toc514886316)

[图4.12数据导出实现流程图 42](#_Toc514886317)

[图4.13数据统计导出模块页面 42](#_Toc514886318)

[图4.14导出参数部分代码展示 43](#_Toc514886319)

[图4.15电子助力转向系统结构图 44](#_Toc514886320)

[图4.16生产任务模型 45](#_Toc514886321)

[图4.17算法1 50](#_Toc514886322)

[图4.18生产线调度方案 51](#_Toc514886323)

[图4.19生产线复用时间示例图 52](#_Toc514886324)

[图4.20算法2 53](#_Toc514886325)

[图4.21实验的生产任务工作流模型 54](#_Toc514886326)

[图4.22实验结果比较 56](#_Toc514886327)

# 表目录

[表4.1符号含义表 41](#_Toc514585527)

[表4.2产品部分参数表 43](#_Toc514585528)

[表4.3生产线的效率与单价表 51](#_Toc514585529)

[表4.4任务执行结果 52](#_Toc514585530)

# 第一章 引言

## 1.1 项目背景

博世（BOSCH）公司是一家全球汽车零部件、汽车技术解决方案配套供应商，硬件基本上做到极致，接下来就要在保持硬件优势的基础上，着眼于更多的软件和服务业务。公司把硬件上的优势作为一个撬板，来撬动软件和服务方面的工作，同时用公司的服务和软件来促进硬件销售，相辅相成。在向工业4.0转型的过程中，公司供应链内所有的业务合作伙伴必须都紧密连接在一起，除此之外，制定开放标准来确保安全，随时提供快速，灵活的信息系统都是必不可少的。工业4.0是一个物联网与互联网组合的工业应用场景，最大限度地提高了生产效率。基于相关数据，生产商可以通过预先清楚当前季度该生产多少的量，对应收货方的出货又是多少来保证工厂的有效运行，减少本地库存。这些都需要以信息化为基础，即工业3.0+各种信息采集[黄阳华, 2015]。目前的网络流程仅限于一家工厂，但在工业4.0的应用场景中，工厂之间的界限将被解除，以连接多个工厂及地理区域[缪学勤, 2014]。

在工业领域，数据一直都是一个很大的问题，当今的企业信息系统大多不能连接商业应用程序，而且这些新系统通常分布在多个组织中无法统一。[夏虹，2017]中提出了一种面向工业数据开放的服务平台体系结构与工业数据系统在企业数据中心的工作流程。[B Fabian, 2012]提出了一个信息系统体系结构，有效地将面向服务的体系结构的概念与语义技术相结合，该体系结构提供了提取、链接和提炼综合的手段。在本系统的应用场景中数据是以分层的方式进行存取，把基于参数设备的与设备相关的参数和生产数据分离，生产数据是一种时效性数据，分为两种，在一个生产批次中生产中的数据即实时数据，生产过的是历史数据。实时性的数据用来做系统预警，实时检测机器是否出现故障等，历史数据用于做一些趋势分析，统计等工作。本文中的系统处理的是静态数据，把基于参数设备部分的数据，作为一种标准提供给工厂使用，而如果有其他部门开发其他与设备相关的系统时，都需要从标准数据系统中获取标准数据，这就解决了不同的系统分布在不同组织中导致信息不统一的问题。

本系统需要做的是对数据的管理，所要管理的主要是工厂中与机器和参数相关的静态数据。所谓的静态数据就是指工厂生产线上的机器设备的相关信息，这些数据可以用来提供给工人生产使用，也可以提供给决策人员使用等，而动态数据是指生产时数据，是一种时效性数据。当工厂最终目标达成，即将静态数据管理与动态数据处理连接起来时，就可以通过动态数据的预警、分析等功能及时地发现机器出现问题，并能够通知提醒该系统下的提供生产数据的机器进行自动升级，这样就使得组件和系统能够获得自我意识和自我预测性，也使管理层更加了解工厂现状。此外，来自不同部件的信息的点对点比较和融合提供了部件和系统水平的精确的预测，并迫使工厂管理部门在尽可能最好的时间触发所需维护，以实现及时维护，零停机。

目前本系统的基础部分是提供给工厂来使用，包括工人来查取数据，工厂专家来补充标准化的数据等等，这些都只是对机器设备的一个管理操作。企业有所有工厂的数据，想要作为服务提供给需要的用户，那么必须要制定出一个可以为用户解决问题的方案。因此本篇论文提出一个基于粒子群优化（PSO）算法得到的能够帮助客户获取产品在规定时间内生产成本最小的生产线组合方案。这里的生产线组合其实就是服务组合的一个特殊应用场景。

服务组合就是将一些简单的服务，组合成一个具有更强大、更完整功能的服务。在本篇论文中，也将讨论这样一个特殊的场景，将每个生产线生产产品看作一种服务，即不同的生产线将会提供不同的服务，最后得到一组有序的服务以满足用户的特定产品需求。

## 1.2 现状分析及必要性

发展信息化服务的前提是落实基础设施，这里以SMT领域为例，Surface Mounting Technology表面贴装技术（SMT）已成为内容多元、跨学科的综合性技术。在近些年，表面贴装技术又到了新的发展时期，并成为电子组装的主流技术[杜江淮, 2016]。可以看出，目前在工业生产方面已达到较高的水平，但想要有长足的发展，必须要开拓新的天地。

数据对于SMT领域来说就是新石油，当前用户希望从SMT生产线上的不同机器收集尽可能多的数据，以便了解工厂的生产流程并提高工厂的质量。而数据治理是企业当前所要面临的挑战（见图1.2）。AE（Automative Electronics）工厂的许多方面缺少或违反了产品和流程标准，在过去10至15年，SMT领域已经对机械和工艺的标准化已经形成了强烈的思维。对于数据部分，现在必须开始一个新的旅程。

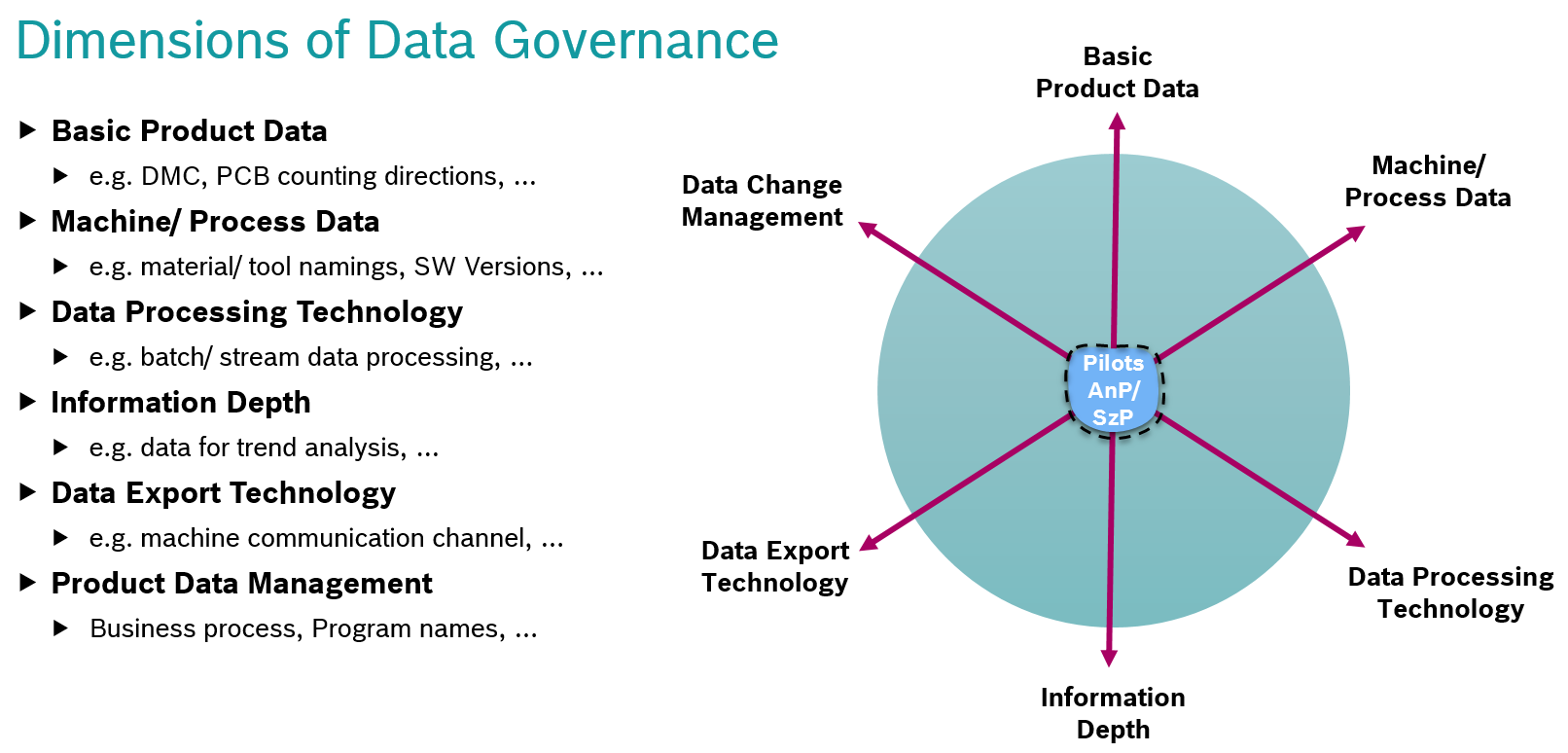


图1.1数据治理维度

信息数据是企业成功的最重要的因素之一，越早获取到相关信息对企业越有利，它可以来自产品或是平台开发的任何阶段。在企业的AE技术路线图之外，有许多新技术（例如0201元件，更薄的PCB，0.3mm间距元件）。这些新技术目前在产品团队和工厂之间进行讨论和实施，但可能会因为工厂之间的不透明导致下一个生产相关产品的工厂错过这些新技术，并且没有人知道这些新技术是否可以引入AE路线图中，以及是否已经有其他工厂实施的成功案例等等。

维护在SMT的生产效率中也很重要。各个工厂需要评估当前维护活动以便将最佳实践定义为标准。为创建一个标准化维护清单，需遵循以下步骤：通过比较当前维护活动并考虑相应指标来找出最佳实践；支持在试点工厂实施最佳实践；确定最佳实践标准；支持为所有AE工厂推出新的维护标准。通过上述步骤得到的最佳实践标准数据就是系统所要维护的机器参数，使用系统还有助于提高SMT效率和降低成本，促进工厂之间的合作使项目运行起来更加高效。

作为一个工业企业的目标是工业智能化发展，即打造一个智能工厂[王华，2013]。智能工厂就是可以利用其优势，制造出成本低廉、又有针对性的产品。其次是它高度自动化的流程，因此企业无需要重新设计内部流程，比较灵活。服务间的相互关联可以使用IT系统纵向和横向的整合来实现。下面以汽车产品为例，来阐述创建人工智能的好处：每个消费者的需求喜好都不相同，有些顾客可能希望将轮胎的材质或色彩改变一点，让产品更符合自己的心意。生产商有必要为客户生产具有个性和定制化和产品和设计，为达到这样的需求就需要一个高效、灵活、且可以扩展的生产，生产各种产品的数量减少，将这一切合并起来就有了比传统工厂更人性化的智能工厂。将来的智能工厂会通过集成系统的方式来体现可扩展性、敏捷性和灵活性等特性[王喜文，2013]。本系统帮助企业了解智能制造领域相关数据标准规范，以此基础标准数据为推手，可以促进工业生产中各类数据的有效利用，提高各个工业部门的生产效率，帮助制造业实现智能化转型。

在较为灵活的生产过程中，产品与产品之间应该是相互关联的，系统根据产品带有的生产相关的信息提示，来进行对应的生产工作，可以改变生产线来将生产的单元连接到一起，从而就改变生产。IT系统在智能生产中显得尤为重要，最终需要把各个系统有机的整合到一起，让他们能够相互协调同时工作。在整个供应链中，垂直整合这个产品线，纵向整合工厂内部，然后可以通过其它技术配合IT系统，使系统更好的工作。

在目前的行业环境下，以最低的成本提供高质量的服务或产品是成功的关键，工业企业为了增加利润和声誉，尽可能地获得尽可能多的业绩。通过这种方式，各种数据源可以提供有关工厂不同方面的有价值的信息。在这个阶段，可以利用数据来了解当前的运行情况，检测故障。例如在生产中，可以通过IT系统向工厂管理层提供整体设备有效性信息以突出显示问题的根本原因和系统中可能的错误。还可以借助系统中的数据，快速找到哪一条生产线可以满足工厂对部件的生产需求，这个可以看做是系统为用户提供的一项服务，但一般可用的原子服务往往只是单一或者简单的功能开发，但是在面临复杂的用户需求和复杂的业务流程时就会发挥不了作用。创造增值服务，通过将多个现有的生产任务安排到工作流中来应对复杂的需求，是推动这一领域迈出的重要一步。找出最佳的服务组合方案可以帮助用户做出决策给出成本最低的生产产品方案。

## 1.3 解决的问题及本文主要研究的工作

在上一小节中提到IT系统在实现工业4.0的过程中是不可缺少的，以下介绍目前系统所要解决的问题：

首先是对机器数据的管理和维护、其次可以提高各个工厂之间的透明度，促进新技术的发展、提供数据标准化，上传的机器信息为系统提供的经过标准化后的数据，同时解决了数据查询效率低，人工管理效率低等问题。

降低工厂内的误报率是一个需要持续改进的过程。工厂专家们确定了一系列可持续优化的措施。在过去几年从常规机器校准的机器基础到复杂的布局特定优化措施，以及使用特定的检查策略使得虚报率得到了优化。下一步的工作是要通过一个远程连接获取各个工厂数据提供给决策者，即决策者可以直接通过使用系统就能够浏览所有工厂数据，而不是通过别的部门获取电子报表。

另外一个问题，每个AE工厂的正常情况是20条左右SMT生产线，1000多种不同产品。但是应如何决定产品的生产线，按周期时间还是按客户要求，受技术限制还是要按经验，甚至所有在一起，这是一个要考虑的问题。因此企业需要一个方案，来帮助工厂的生产计划部门在做出这些决定时只需点击几下就可以进行决策。主要目标是优化生产线利用率，减少变更和得到工厂整体更大的透明度，以及在保证质量和时间的前提下达到成本最优。并且可以实现有成效的计划方式。

本文所生产线数据管理系统，是一个管理各工厂生产线下设备参数信息，使传统工厂设备和生产的状态高度透明化，通过线上查询工厂机器数据，提升产能和生产效率、提供统计导出等功能统计各工厂相关信息的同时为工厂的生产计划部门做决策提供依据。

首先本文对工业方面相关背景做了相关调查，并简单介绍了公司的业务背景，以及系统要解决的问题。针对工业数据存取问题，本文提出了将参数设备的设备静态数据与生产产品时的动态数据进行分离管理的方案。然后详细地介绍了开发的生产线数据管理系统的设计与实现。本文把生产线数据管理系统的基础实现部分分为四个模块来详细介绍。

基于系统实现部分本文提出了基于PSO算法的生产线组合调度算法。该算法可以得到生产产品调度生产线时能够使成本最优的调度方案，同时帮助用户进行决策选择出在规定时间内是成本最低的生产线组合，来提高服务组合的效率。

## 1.4 本文的组织结构

第一章 引言部分。首先介绍了项目的开发背景和实际意义，对工业方面相关背景做了简单的调查和介绍，系统要解决的问题。针对工业数据存取问题，本文提出了将参数设备的设备静态数据与生产产品时的动态数据进行分离管理的方案。

第二章 技术综述。主要对项目开发过程中使用到的相关技术理论与技术研究做介绍。首先讨论了项目开发的整体过程中用到的Play框架，了解到该框架一些比较好的特性和开发方法；然后分别对可视化，持久化等部分用到的相关技术也做了相关介绍；最后是本篇论文为该项目提出的一个拓展内容涉及到的算法技术介绍。

第三章 工厂生产线数据管理系统的分析与设计。在这一章节中本文提出了工厂生产线数据管理系统的相关需求，同时对项目整体的设计做描述，以及对项目各个模块的划分、对系统涉及的数据库表等进行了设计。

第四章 工厂生产线数据管理系统的实现。重点阐述了项目各个模块的实现细节，并讲解了基于基础系统新增部分的实现细节。

第五章 总结与展望。总结已完成的工作、还有哪些不足，以及未来还需要完成的任务。

# 第二章 技术综述

本章是对项目开发中所涉及到的部分相关技术做介绍。主要有系统开发过程中用到的Play框架、AngularJS等，还有本文用来找出最佳生产线组合方案的基于PSO算法的服务组合研究方法。

## 2.1 Play框架

Play框架是一个开源的Web框架，是由Zenexity公司的开发团队利用业余时间开发，2009年10月发布了Play的1.0完整版并正式开放了源代码，此后不断更新版本。Play框架潜心于开发效率，并以Restful为目标，是Java敏捷开发的最佳参考 [范凯, 2009]。Play既不失Java的稳定性，又同ROR、Django一样轻巧，遵循于MVC模式，与大多数基于JVM的Web应用程序开发框架不同，Play不是基于Servlet标准构建的，而是具有HTTP和WebSocket协议的轻量级抽象。作为一个快速的全栈式JavaWeb应用开发框架，Play包括了Web开发的各个方面，如图2.1所示[Manuel, 2016]：

图2.1 Play框架结构

此外Play框架还具有以下特性：

1. 无状态的模型：Play遵循了REST风格，是一个扩展性非常强的无共享框架。Play所有请求遵循这样的执行过程：1）发送HTTP请求，2）Router对请求进行匹配、控制器中的Action开始执行，3）应用代码将获取的数据发给Action，4）Action将数据传到view绘制视图并展示给用户，5）HTTP响应返回[Play, 2017]。
2. 迅速定位错误：当程序出问题时，Play会在页面中提示问题代码块并定位该问题发生的具体位置。
3. Java与Scala：Play2.x用Scala和Java开发，因此具备对Java和Scala[Odersky, 2013]编程语言的原生支持的优势。
4. RESTFul: 在传统Web开发框架中，都是使用Servlet将Http协议隐藏，程序员无法清楚某个请求与操作的对应关系。Play框架的设计却大不相同，例如可以在route文件中写如图2.2来获取用户列表。则/users/list这个URL对应到UsersController中list方法。

|  |
| --- |
| GET /users/list controllers.UsersController.list |

图2.2 Play在路由中绑定方法

1. 高效方便：Play框架有热重载的功能，使用者不需要手工地去编译部署重启服务器等操作，当编辑相关文件并保存后，只需对页面刷新就可以查看更新后的界面，提高了程序员的开发效率。

一般的JavaWeb开发框架采用的HTTP Server大多是基于线程模型实现，比如ApacheTomcat。而Play的 HTTP Server是基于事件模型且具有异步非阻塞的特点。图2.3展示了Play框架的事件模型服务器的运行模式：

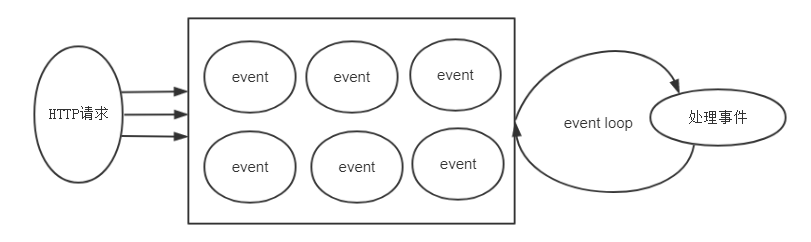


图2.3事件模型服务器工作流程

当使用者发送一个请求时，一般情况下该请求会有多个操作，而Play可以把请求分为单个的事件，再异步去处理这些它们。比如当某事件正在被处理，若当前线程一直等这个event执行完成再去执行下一个event，就会造成资源的浪费。因此这段时间消息线程（event loop）可以去执行已经到达事件列表中的别的event。在事件完成后，就会发一个中断（也算一个事件），再继续加入到事件列表中等被执行。这种异步非阻塞的模式使Play框架可以用较少的资源去应付较大流量的访问，因此Play框架可以迅速地响应用户操作。

总的来说，Play是在Java社区中成长并且与其他社区已存在的快速Web框架（如Django、RubyonRails等）相媲美的开发框架。

## 2.2 前端使用技术

本系统的可视化分为两个部分：一个是对各个工厂生产线的管理，以及对各生产线机器的管理等操作，这一部分对所有用户都可见；另一个是对所有参数的维护，这一部分只有工厂专家可以参与，只用到一些基本的操作及校验，且不需要复杂的界面。对于生产线机器管理等可视化部分使用AngularJS。参数维护部分系统使用Play框架的 CRUD模块来实现，因为这一部分主要是对参数的一些基本操作无需复杂的界面，简单整洁即可。

### 2.2.1 AngularJS

AngularJS是由Google公司开发，Google作为互联网行业的巨头拥有强大的社区支持。其次它的声明式的模板功能非常丰富，包含了强大的Angular指令[Bacon et al, 2013]，是一个相对较为完整的前端MVVM框架，包含了可视化部分所有的功能。可以很灵活地自定义Directive。ng的模块化的思想引入了Java的部分内容，比如依赖注入，开发者可以方便快捷地写出复用度高的代码，对于实施敏捷开发团队来说也有很大的帮助。但最大的好处还是能够简化开发。

HTML作为一种静态的、声明式的语言，当用它来声明包含动态的Web应用时就不好处理了，而AngularJS是为了解决其不足而设计，可以通过指令等让用户的浏览器页面识别新语义，使应用更加生动，可读性也更高[Barrow, 2013]。AngularJS遵循MVVM模式（Model View ViewMode，模型—视图—视图模型）。MVVM模式为了实现View和Model的连接，引入专门的ViewModel（视图模型），借助框架内部双向绑定对MVP（Model View Presenter）模式转换[李嘉, 2018]，使得视图和模型之间产生了更深层次的分离与解耦。如图2.4所示，Angular中MVVM模式主要由四个部分构成。

View视图，负责页面的展示，View和Model通过ViewModel来协调与合作，ViewModel提供需要展示的数据给View，还提供了Command事件操作Model的方法，Angular中的$scope对象充任了ViewModel。Model模型，是业务领域的对象，是与业务相关的数据的封闭载体，Model不会有任何页面展示相关的内容。



图2.4 AngularJS中MVVM模式的运用

ViewModel对象的初始化工作是由Angular中的Controller负责的，它可以组合一个或多个service将获取到的获取业务领域Model放在ViewModel对象上，并且能够在启动和加载页面时可以被使用[何焕春, 2017]。

### 2.2.2 Play CRUD

Play CRUD模块用于快速生成一个基本的管理区域，能够提供易于管理的组件，该模块会检查模型类内部的结构，可以自动创建较为简单的管理表单等。比如开发人员可以非常方便地初始化一个登陆用户。CRUD为程序员提供一个可视化的对象管理界面，就好比数据库的客户端，Play在后台执行SQL语句，并对相关数据做处理。用户就可以在页面进行增删改查的操作，就不需要自己再去写操作数据的代码了，该模块主要负责系统中持久层的基本操作。Play为了操作JPA模型对象，其CRUD模块集成了非常多的Web接口。

只需要在application.conf中完成相关设置如图2.5，即可启动CRUD模块：

# Import the crud module

module.crud=${play.path}/modules/crud

图2.5启动CRUD模块

CRUD模块的路由还有一些规则，要在conf/routes文件中设置如图2.6：

# Import CRUD routes

 /admin            module:crud

图2.6 CRUD模块配置路由

当使用者打开/admin时，程序会加载CRUD路由的规则。每一个模型需要一个作为controllers.CRUD子类的controller，用来帮助CRUD管理域集成相关的模型对象。如在controllers目录中新建Users.java，并让该User类作为controllers.CRUD的子类，便实现了对Users模型CRUD的工作。

另外使用Play对字段进行校验，只需要导入play.data.validation.\*包下的类，并使用相关注解即可。

## 2.3 Json Web Token

Json Web Token (JWT)使用一种基于JSON的标准来传递声明，是一种URL安全方法，由于JSON的普用性，JWT可支持多种开发语言。由于这种token它的开销较小，并且可以轻巧地跨不同的域使用，尤其适用于分布式的单点登录应用场景[JWT, 2017]。它也常用在身份与服务的提供者之间，传送一些需要认证的标识信息，以便于从存储资源的服务器端得到这些资源，同时也可添加其它业务所必要的声明信息。

session认证会存在这样的问题：在用户通过应用进行认证后，程序都会在服务端记录一次，便于识别用户的下次请求。一般情况下，session均是存入内存，但当认证用户的数目的变多后，服务端消耗就会显著上升。服务端在用户认证后会做出相应的记录。若将认证记录保存到内存，则为了用户拿到授权资源下次还必须要请求在本台服务器上，如此在分布式应用上时，会对应地去限制负载均衡器的能力，同时限制应用扩展性。由于传统认证是基于cookie的, 若cookie被截取了，那么用户就会轻易地遭受攻击。但是基于token的认证机制与http协议类似，没有在服务端保留用户认证信息或者会话信息的要求。因此基于token认证机制的程序无需关心用户是在哪个服务器进行了登入，为应用程序的扩展提供了很大的便利。

JWT包含Header，Payload，Signature。由于有payload，那么JWT则能够在本身存其他的业务逻辑中所必要的一些不太敏感的信息。且JWT的构成简单、占用字节也小，因而它是很方便传输的。不用在服务端保存会话信息, 对于应用的扩展很容易。

## 2.4 Java Persistence API

Java Persistence API（JPA），顾名思义，即对数据库操作的数据模型对象的操作，是Sun公司在JavaEE 5.0规范里边提出的Java持久化接口[JPA,2015]。JPA在Play中扮演了重要的角色，帮用户优化了许多代码，且包含了Java持久化相关技术优势。在解释为何使用JPA的之前，这里先简单介绍一下ORM。

Object Relation Mapping（ORM）封装了JDBC，是对象持久化的核心技术[R Johnson, 2003]。同时它也解决JDBC的很多问题：

1. 复杂的代码问题：ORM创建了对象与数据库表的映射关系，使得开发人员无需写SQL语句，只对对象操作就可以，大大降低了工作量，帮助开发人员能够更加潜心于业务的实现。
2. 数据库对象连接：ORM通过数据库与对象映射关系，根据数据库表间的关系来创建对象之间的关系，并维持此关系有效且完整。
3. 系统架构的问题：JDBC在数据库访问层，当用JDBC进行编码的时候，开发者必须知道后端使用的是什么数据库、表、字段、索引、表间关系等问题，以及与数据库相关的其它信息。而通过ORM，开发人员不用知道后端采用的具体数据库及数据库设计相关问题，只用根据业务逻辑调用JavaBean方法，便可以实现对后端数据库的操作。
4. 性能问题：当使用ORM时，相关框架会先判断操作数据库对应的需求，然后进行自我延迟，并向后端发送数据库连接的请求，ORM会判断现实的状况，把数据库访问和操作进行合成，以尽可能地降低那些非必须的对数据库的操作请求[田珂, 2004]。

JPA除了包含ORM的一些技术点同时，它还会有自己独特的一些优势：

1. 标准化：对于任意有着 JPA 标准的框架都有相同的架构，并提供同样的API。
2. 容器级特性：JPA 相关框架能够支持事务、并发和大的数据集等。
3. 易学易用：JPA为提高开发者效率提供了非常简单的开发模型。当使用JPA相关框架时，创建实体没有约束和限制，只需要用Entity注解；程序员学习起来较容易；很方便与其它框架集成。
4. 查询效率很高：可以把JPA的查询语句看作与Hibernate HQL[孙卫琴，2005]是同等的。JPA中的一些定义使之可以操作实体对象而不是数据库表，不仅可以支持大量的更新和修改等一般只有 SQL语句才会提供的高级查询特性，还能还支持子查询等功能。
5. 支持OOP的高级特性：JPA中可以支持OOP的一些高级的特性、类之间繁琐的关系等，这些会让程序员在很大程度上使用面向对象模型去对企业中的应用进行设计，但却不用自己去处理这些特性的持久化任务。

## 2.5 基于粒子群算法的服务组合

近年来国内外有关 Web 服务组合的相关研究越来越多，研究者们提出了很多可以优化服务组合的算法。粒子群优化算法是服务组合领域一个典型的优化方法，相较于其它一些典型的算法（如遗传算法），粒子群算法具有参数少、收敛速度快等特点，且在很多优化问题上具有良好的搜索能力[温涛，2013]。

本文中的服务组合是指生产产品时的生产线组合，使用了Web服务组合研究中的方法。这里首先要清楚几个概念性的问题：首先什么是服务组合，有哪些研究方法，本文中提到的服务组合指的又是什么。

Web服务中有一个重点的研究方向—Web服务组合。Web服务组合根据用户的不同需求，通过利用网上现有分散的Web服务，将相对简单的服务按照指定规则组合，成为更加完善的的服务。并且最终在众多的组合方案里，能够快速地将符合需求的服务组合方案提供给用户[Petrie, 2016]。

Web服务组合的研究人员各有各的关注点和角度。Web服务组合方法可以分为语义驱动和过程驱动这两种。过程驱动的方法利用Web服务组合与过程的一致性，以及成熟的建模工具和文字对Web服务组合业务过程建模，采用替代的思维模式，把过程模型中的相关部件替换为具体的Web服务，从而得到较好的组合的方案[Charles, 2016]。与过程驱动的思想不同，语义驱动Web组合方法通过推理能够自生成Web服务组合方案[邓水光, 2008]。

根据工厂实际的问题，本文选择了基于工作流模型的Web服务组合方法，在具体流程执行前首先要选出具体的服务，用来替代模型中抽象的服务，以获得能够执行的服务组合方案。在本文的应用中把生产一个复杂产品分解为多个任务的工作流，这样的生产任务工作流可以看作是服务组合的一个特殊的应用场景。

粒子群优化算法（PSO）[Eberhart et al, 1995]是一个自适应的全局搜索为基础的优化技术。它是基于一组粒子在空间中移动并相互通信以确定最佳的搜索方向。PSO由于其简单和有效的广泛应用和较低的计算成本而被广泛应用。通过将PSO适应于特定的领域，已经成功解决了不同领域的许多问题。在PSO中，一些简单的粒子被放置在某个问题或函数的搜索空间中，并且每个粒子都在其当前位置处评估目标函数。每个粒子然后通过将其当前和最佳（最合理）位置的某些方面与一些随机变化相结合来确定其在搜索空间中的移动。在所有粒子被移动之后进行下一次的迭代，就像一群集体觅食的鸟类，最后整个群体可能会接近最佳效果。

本文提出的成本感知调度（Cost Aware Scheduling, CAS）算法是基于PSO算法并应用到了生产线的服务组合优化中，能够在尽量减少执行成本时，满足最后期限的限制。本文将在第四章中进行详细介绍。

## 2.6 本章小结

本章节是对生产线数据管理系统开发过程中所涉及到的相关技术做介绍，主要包括以下内容：快速全栈式Web开发框架Play框架、前端根据不同的业务需求用到了AngularJS和Play框架的CRUD模块，用于安全认证的JWT，以及持久层所用到的JPA，最后还有用来实现使成本最低的生产线组合方案的算法所涉及到的服务组合相关技术介绍。

# 第三章 工厂生产线数据管理系统的分析与设计

## 3.1 系统需求分析

### 3.1.1 项目总体规划

本系统由生产线机器管理，机器参数维护两大部分组成。其中，机器管理是系统的核心部分，要通过系统与实体机器进行关联，通过该部分用户可以在线上完成机器信息核对、生产线调整、机器参数查询等等操作；机器参数维护是由对工厂机器所有参数以及参数值的维护，由指定的工厂专家有针对性地维护他们所属领域的参数信息，且这些参数作为一种标准数据提供给各个工厂的维护人员参考，机器管理模块中的机器参数配置都是基于参数维护模块中的参数信息，比如某条生产线添加一台机器，该机器的参数配置必须从已有参数列表中选填，而不能有机器维护人员自己编写。

而生产线机器管理部分分为三个小模块，分别是工厂信息总览、生产线查询模块和数据导出模块。在这一章节中将会详细描述各个功能的具体需求、使用流程以及其它相关设计。

### 3.1.2 用户特点

本系统提供给博世公司拥有权限的员工用户使用，这些用户包括管理层、办公人员、工厂专家以及工人等，界面简单友好方便实用即可。根据不同的用户特点，赋予这些用户不同的权限职责。例如办公人员只关心工厂和生产线的机器数据等信息，但他们不具备机器生产过程的专业知识，因此普通的办公人员没有操作修改这些机器信息和参数的权限；而工厂专家是具备相关领域知识，所以他们可以维护自己所负责的相关参数维护工作。在本文的设计与实现部分将会详细介绍。

### 3.1.3 功能要求

1）工厂整体信息维护功能：

1．查询所有工厂生产线信息。该功能面向所有用户，任一用户可以查询其他工厂的生产线及机器信息作为参考。

2．对工厂中生产线的维护，即当一条生产线上加工工序有所改变时，工厂的协调管理人员可以对生产线上机器的顺序做调整，包括将一条生产线上的某台机器删除或添加一台新的机器。这里的机器信息等删除并不是真的删除了，而是改变状态字段作为一个历史的版本，便于追踪历史记录。

3．在指定生产线对机器信息进行维护。工厂的协调管理人员及时地机器参数进行调整。当一些机器需要满足一些新的生产条件时，管理人员可以为现有机器更新生产可选参数。

4． 发现有用户调整或修改产线信息时邮件通知相关用户，确保工厂之间的透明度，同时也使得相关用户能够及时地发现问题并解决。

2）搜索功能

1．按参数或者机器类型搜索包含的生产线。可以快速查找到包含指定参数或某个机器类型的生产线具体到产线下的机器。

2．生产线组合搜索。搜索可以生产复杂产品的成本最优的生产线组合方案。

3）参数维护功能

1．参数信息维护管理。有工厂专家通过最佳实践得到标准化的数据后对整体参数的一个维护工作，包括增加新的参数类型、修改已有参数的可选值、删除已有参数类型。

2．有新的标准参数添加时邮件通知所有工厂，保证各个工厂相关技术人员及时对新技术进行了解并投入应用。

4）数据导出功能

1．导出整体参数，有哪些生产线有该参数以及参数值便于相关工作人员清楚地了解各个工厂现状。

2．统计各个工厂相关数据并导出。包括各个工厂所对应的机器类型、指定工厂所对应的产线、工厂中使用某一生产参数的机器数以及该参数的具体值等。要求工厂与生产线之间关系对应，生产过程与机器类型，机器类型与机器参数之间关系的对应。

### 3.1.4 非功能要求

1）可用性（usability）

一方面是指系统所提供的功能必须是完备可用的，需要满足需求的基本功能；另一方面要保证使用者使用系统很容易，使用流程清晰简洁，不需要学习很多东西。

2）可靠性（reliability）

保证运行时间内，系统不发生故障的概率。当用户做一些非法的操作时，不会导致系统崩溃无法继续运行，而是返回给用户一个友好的提示等，并且本系统能够正常运行。比如在本系统中用户查询机器不存在时，提示“机器维护中”、“暂不能查询”等。

3）性能（performance）

延迟与处理时间在一定范围内。例如系统响应用户的操作时间<3s，系统查询生产线机器数据<3s。本系统通过创建索引并优化查询语句来提高查询效率。

4）安全性（security）。

**·**只有具备一定权限的用户才可以进行相关操作，分工明确

**·**工厂协调人员只能修改自己所在工厂的生产线数据

**·**当工厂数据被修改后会邮件通知工厂下的所有人，当专家对参数信息做调整后会邮件通知所有其他的专家以及所有工厂的协调人员。

5）可扩展性（Extensibility）

要求系统便于拓展，方便后续新功能的添加以及后期与其他系统的功能对接，即系统适应“变化”能力要强。在本系统中每个模块都是独立的，相互联系但互不影响，当系统有新的需求时，只需要添加新的开发模块。

6）其他非功能要求

所有系统错误必须要有日志记录。

### 3.1.5 系统用例

**1.系统用例图**

根据系统的具体需求，以及使用者不同的工作职能，给出如下的系统用例如图3.1所示：

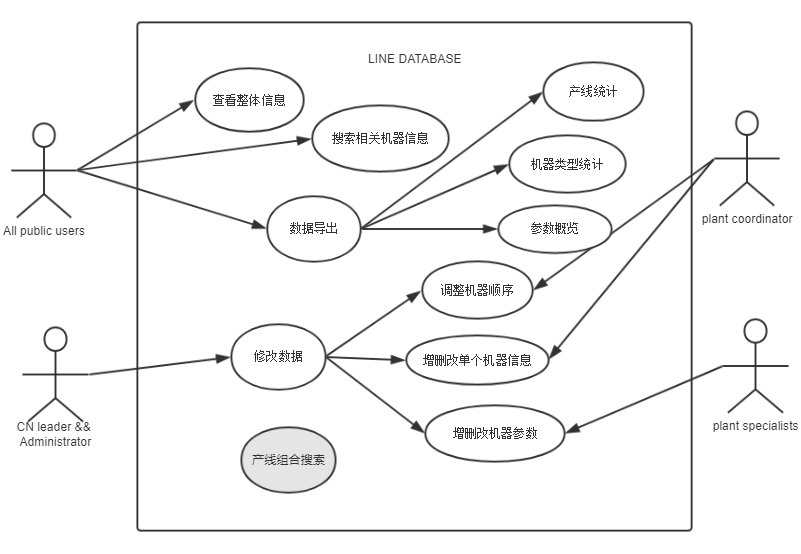


图3.1系统用例图

**2.用例说明**

1）查询整体信息：

**主要参与者**：所有普通用户

**涉众利益**：

* 所有的用户都可以看到工厂生产线的当前情况（以前都要打电话发邮件甚至去工厂确认）。
* 不同国家之间的工厂可以互通
* 汇总统计机器类型、产线、参数等信息

**前置条件**：

* 已有的相关数据信息已经录入系统
* 相关的用户已经能够被识别，且被授权。

**基本流程**：

1. 各个工厂将本工厂的生产线数据录入系统；

2. 用户登入系统；

3. 用户可以在首页点击需要了解情况的工厂图标进入工厂的详情页面。

4. 将当前工厂下包含的所有生产线展示给用户

5. 用户点击任一生产线即可查看该生产线的生产流程和机器参数信息。

**扩展流程**：

1a. 生产出产品的数据采集由机器来完成；

5a. 检测后得到的故障机器在离线机器区域。

2）搜索功能：

**主要参与者**：生产线搜索功能对所有用户开放

**涉众利益**：

* 所有用户都能够搜索到其他工厂相关生产线及机器的信息且无需电话沟通确认等繁琐流程；
* 不同国家工厂可以互通；
* 汇总统计机器类型、产线、参数等信息。

**前置条件**：

* 各个工厂的生产线的最新数据已经录入系统；
* 相关的用户已经能够被识别，且被授权。

**基本流程**：

1. 各个工厂将所在工厂的生产线、机器数据录入系统；
2. 相关用户登录到该系统；
3. 点击进入搜索查询页面，在最上方的搜索框内输入查询关键字；
4. 点击加号按钮添加查询条件，输入多个查询条件进行搜索查询；
5. 返回当前的查询结果，查询结果为所有符合要求的生产线；
6. 点击某一条产线则展开产线下的机器信息（包含查询条件的机器）。

**扩展流程**：

4a. 查询具体小产品，返回所有可以生产该产品的生产线；

4b. 查询多个小产品部件（可组装成一个复杂的产品），返回产线组合方案，能够使成本最优，并给出所需要花费的时间。

3）添加新机器：

**主要参与者**：工厂协调人员，相关管理者及领导

**涉众利益**：

* 工厂发布新增机器信息
* 所有人都可以收到新加机器的通知，了解新机器相关情况。
* 为其他工厂是否引入新机器作参考

**前置条件**：

* 相关工厂已引入实体机器并就位。
* 相关的用户已经可以被识别，且被授权。

**基本流程**：

1. 相关管理员用户登录系统；
2. 在首页选择所在工厂图标进入该工厂下所有生产线展示页面。
3. 点击具体生产线进入相关生产线详情页；
4. 点击编辑按钮，再点击创建按钮弹出新机器信息设置页面框；
5. 用户对机器进行设置以及相关参数选择对应的值并保存。
6. 保存后展示给用户的生产线上指定位置下多出新增的机器，并邮件通知所有工厂的相关用户。

4）机器信息修改：

**主要参与者**：工厂协调人员，相关管理者及领导

**涉众利益**：

* 所有人都收到机器更新通知，了解当前机器状态。
* 工厂对机器的修改只需更新信息即可让其他工厂知情；
* 增大工厂之间的透明度，提高效率

**前置条件**：

* 已有的相关数据信息已经录入系统
* 相关的用户已经能够被识别，且被授权。

**基本流程**：

1. 具有权限的相关用户登录到系统；
2. 点击自己所在工厂下的图标。进入工厂详情页面；
3. 选择需要修改信息的生产线点击进入；
4. 用鼠标选择机器进行拖动调整生产线机器顺序；
5. 点击编辑选中具体机器修改信息，修改完成后点击保存提示修改成功或失败并给出具体原因；
6. 邮件通知所有用户当前产线的具体机器信息被修改。

**扩展流程**：

1a. 生产出产品的数据采集由机器来完成；

5a. 与其他系统打通后，比如通过大数据分析系统检测到某个机器需要升级到新的版本便通知该系统进行自动更新信息。

5）添加新参数：

**主要参与者**：工厂专家，相关管理者及领导

**涉众利益**：

* 经过一定的研究后得到成果，工厂专家需要对新技术进行推广。
* 将新的标准纳入机器参数备选项中提供给工厂协调人员设置机器信息，并提供给工人使用。
* 各个工厂可以及时了解新技术参数的发布情况，保证工厂之间不会因为信息不透明而错过新的技术。

**前置条件**：

* 通过比较等维护活动并考虑相应的指标得到最佳实践
* 最佳实践已经被定义为一个标准，可以被应用。
* 相关的用户可以被识别，并且被授权。

**基本流程**：

1. 相关用户登入到系统；
2. 工厂专家打开系统的管理页面；
3. 可以对当前已有的参数进行检索，看是否存在或者有其它意图。
4. 点击新增参数按钮跳转至参数配置页面
5. 参数配置确认无误后点击保存。
6. 添加成功返回或停留在当前页面，参数添加失败则给出具体原因。
7. 参数添加完毕后自动发邮件提醒所有工厂的相关人员关注该信息。

**扩展流程**：

4a. 专家设置参数时，给出相关提示（模糊匹配），提醒专家已存在哪些类似的参数；

6a. 用户点击保存并返回后，页面返回到所有的参数列表中新增参数的所在页，参数是按用户指定的规则进行排序，并且刚刚新增的参数被高亮显示。

6）参数修改：

**主要参与者**：工厂专家，相关管理者及领导

**涉众利益**：

* 工厂专家需要对相关参数进行维护并推广。
* 工厂的协调人员需要使用相关信息进行机器配置设置，提供给工人使用。
* 各个工厂可以及时了解新参数值的修改发布，保证实时更新机器信息，不错过新技术标准。

**前置条件**：

* 系统中已存在当前需要修改的参数。
* 当前要修改的参数值又有新的技术标准，或者已经过时需要更新或删除相关值。
* 相关的用户已经能够被识别，且被授权。

**基本流程**：

1. 相关的管理员用户登录系统；
2. 在搜索框内搜索需要修改的参数并得到返回的结果；
3. 查询到参数后，点击所要修改的参数进行参数详情页；
4. 对参数值进行修改或添加新的可选项值或者删除当前的参数。
5. 若删除了当前参数则直接返回所有参数的列表页面
6. 若对参数做了修改，修改成功则返回或停留在当前的页面，修改失败则提示给出具体失败的原因。
7. 参数修改完后邮件通知到所有工厂的相关人员具体哪个参数有所改动。

**扩展流程**：

1a. 生产出产品的数据采集由机器来完成；

6a. 若是删除操作，则直接返回。修改操作点击保存并返回后，页面会返回到所有的参数列表中刚刚修改的参数所在页，所有的参数按照指定的规则进行排列，并且刚被修改的参数高亮显示使界面更加友好。

7）数据导出：

**主要参与者**：所有用户

**涉众利益**：

* 工厂生产计划人员使用数据进行估算工作。
* 增大工厂之间透明度，便于相互了解情况。
* 降低工厂内部的误报率。

**前置条件**：

* 系统中已存在当前需要修改的参数。
* 相关用户可以被识别。

**基本流程**：

1. 用户登录系统；
2. 选择数据导出功能；
3. 用户选择根据自己的需求选择所需工厂、生产线或者是机器类型等数据统计并导出。
4. 得到相关数据表

**扩展流程**：

3a. 可根据用户生产产品的需求，导出所有能够满足生产相关产品的生产线组合方案。

3b. 导出满足条件的组合方案后，并给出推荐的生产线方案。

8）生产线组合搜索：

**主要参与者**：所有用户

**涉众利益**：

* 一般用户了解复杂产品生产流程。
* 需要生产产品的用户获取成本最低的生产线组合方案。

**前置条件**：

* 用户选择相关产品可以被生产。

**基本流程**：

1. 用户选择想要生产的产品或者选择要组装的零部件；
2. 选择自己所在的地区；
3. 搜索得到生产线组合方案。

**扩展流程**：

3a. 得到多个生产线组合方案并按生产成本进行排序。

## 3.2 系统总体设计

### 3.2.1 系统模块设计

如图3.2所示，工厂生产线数据管理系统主要由工厂总览模块、生产线查询模块、数据统计导出模块、机器参数维护模块组成，每个模块下还有一些细分的功能点。

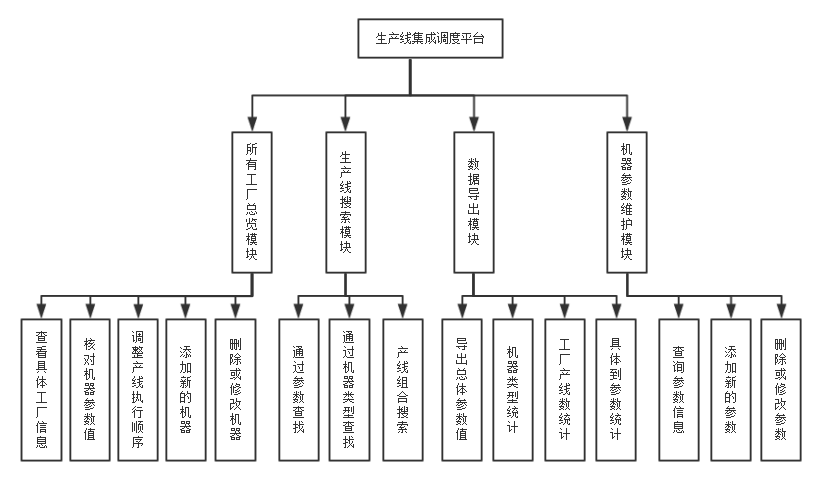


图3.2系统总体结构

### 3.2.2 系统类结构设计

如图3.3所示，是生产线数据管理系统的相关模块的类图，由于系统开发选择了Play框架，因此每个控制器类需要继承Controller类，包括用来处理工厂信息以及工厂生产线的LineDefineController类，查询生产线的QueryController类，处理机器相关信息的DeviceController类等等。

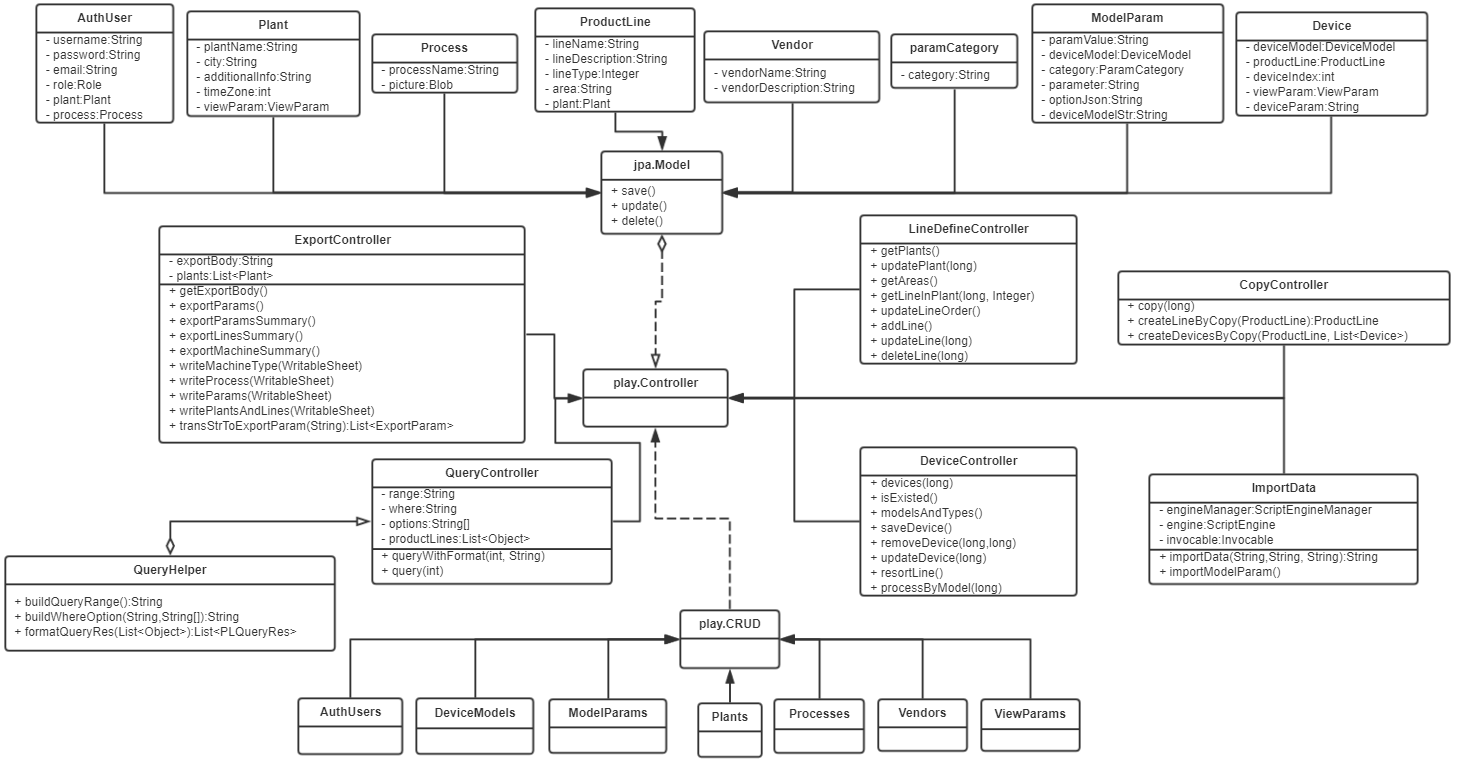


图3.3系统类图

相关实体类需要继承play.jpa.Model类，这样相关类就可以继承一些处理方法，这些实体类包括工厂实体Plant类，生产线ProductLine类，以及机器设备Device类等等。继承play.CRUD类的主要用于参数处理模块。

## 3.3 系统详细设计

本节通过时序图的方式对用户与系统之间进行交互的主要过程进行分析设计，介绍了系统的关键模块的使用流程。

### 3.3.1 工厂数据总览模块详细设计

在该模块中，普通用户可以查询工厂生产线、机器数据，相关管理人员可以在具体页面对生产线做调整、添加或删除机器、修改机器相关参数。

1. 所有用户（All public users）查询核生产线及机器参数信息流程如图3.4所示：

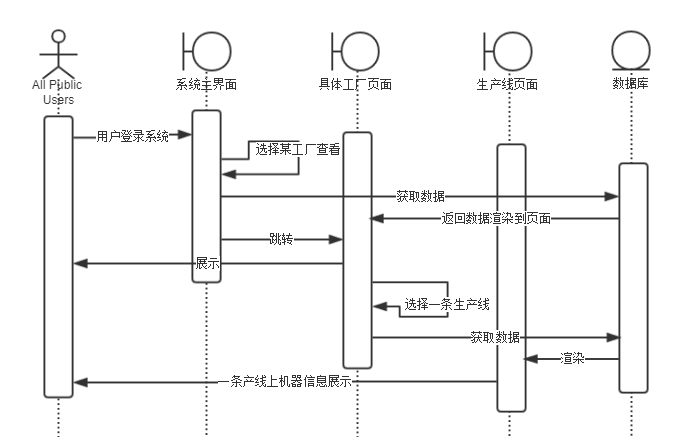


图3.4生产线查询时序图

2. 工厂协调管理人员调整机器在生产线中的顺序如图3.5，用户需要对某条生产线上机器顺序做调整时，对机器进行拖动。系统检测该用户是否具有操作权限，验证通过后可对生产线进行调整。

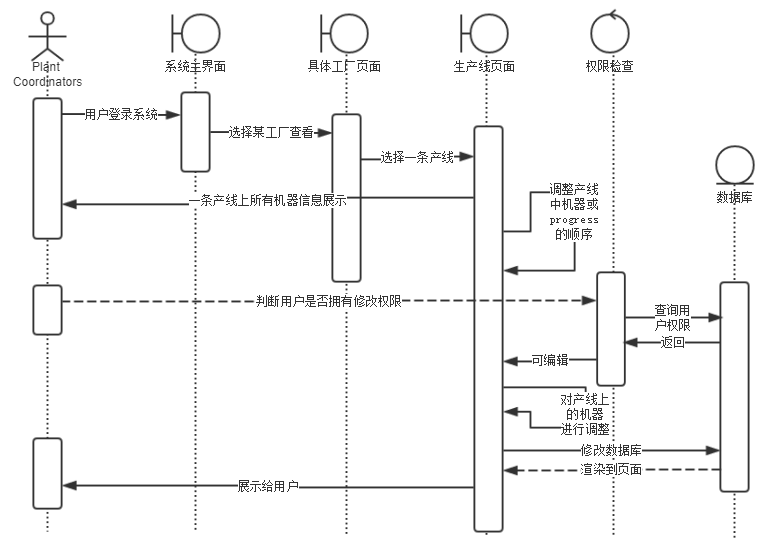


图3.5生产线调整时序图

3. 工厂协调管理人员添加新的机器见图3.6。根据工厂需要，当某条生产线需要添加新的机器时，管理人员验证身份后便可创建新的机器，并对这台新机器添加配置相关参数值。机器信息添加成功后更新到数据库并重新渲染到页面，再次查看该生产线时就会多出一台新机器，同时邮件提醒相关人员有新的机器被添加到具体的某条生产线，可以进行查看更新。

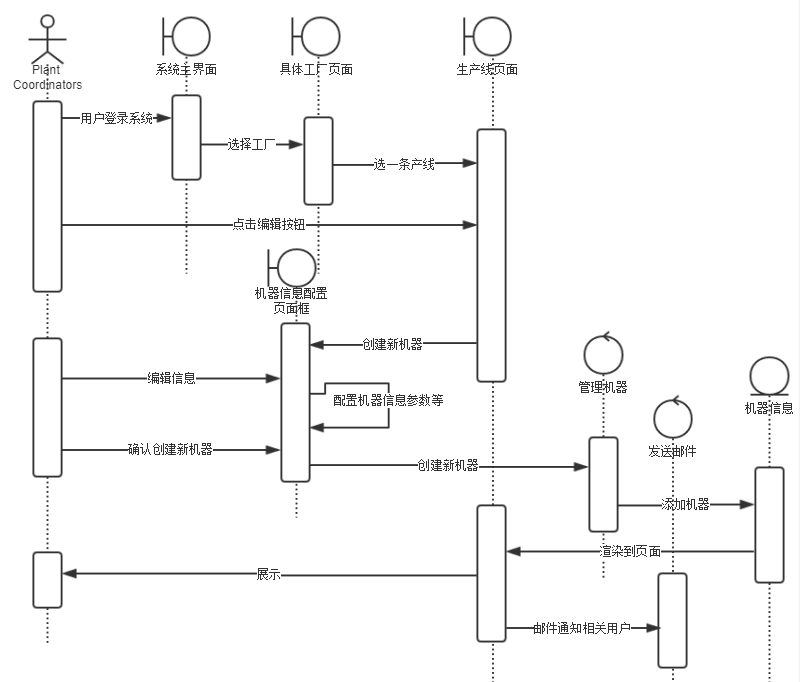


图3.6添加新机器时序图

其间还可能存在创建失败，不合规等情况，为了展示流程更加清晰，没有在图中全部画出。

4. 修改机器信息有编辑某一台机器的参数值，这里的参数值与参数维护模块的参数是不同的，这里修改的是对应到指定的机器下的参数，是每台机器所需要维护的参数，由每个工厂的协调管理人员来调整。而参数维护模块维护的是所有工厂所有机器所包含的参数，是由工厂专家经过一系列研究标准化后确定具体值提供给各个工厂机器参考。在图3.7中省略了一些验证及操作失败的流程。

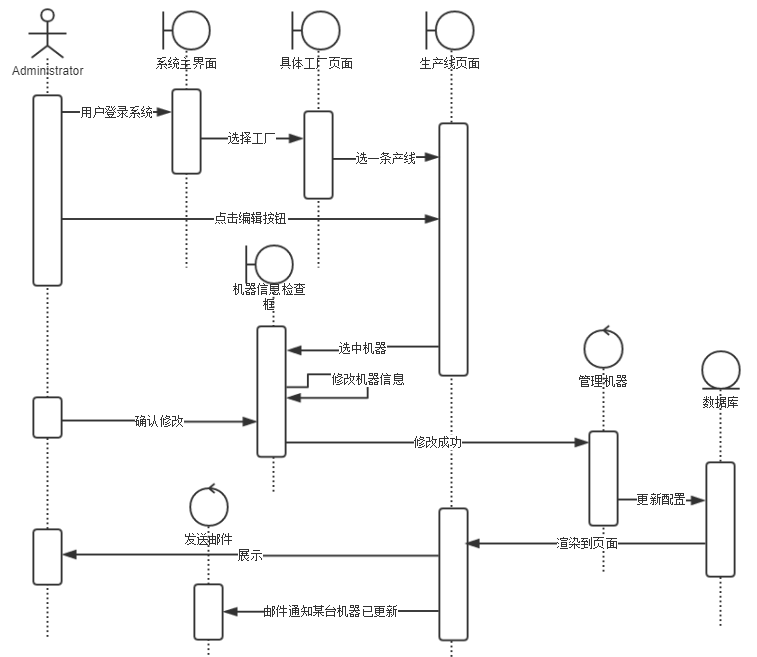


图3.7修改机器信息时序图

5. 管理人员删除机器的工作流程主要为，授权用户登入系统根据工厂->生产线->机器找到相关设备进行删除，确认删除且删除成功后会邮件通知相关用户，如图3.8流程所示：

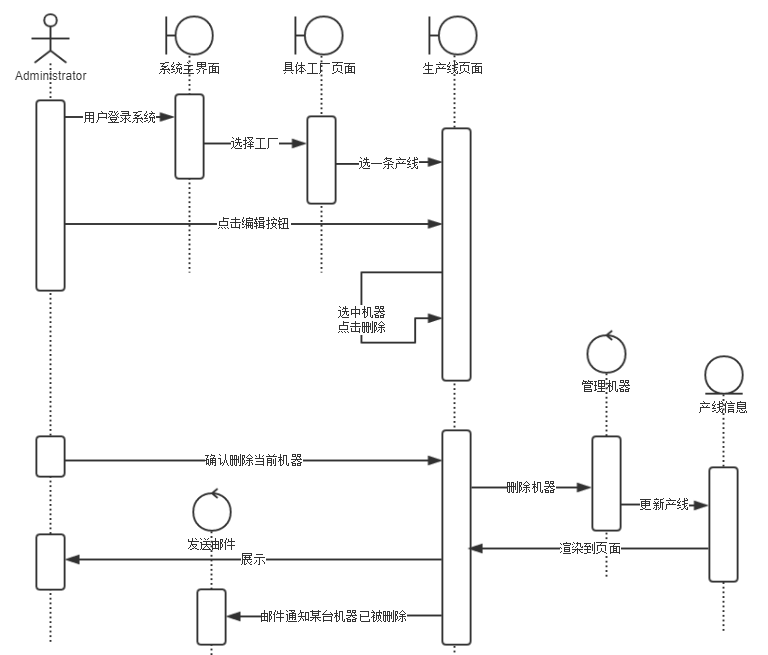


图3.8删除机器时序图

### 3.3.2 生产线搜索模块详细设计

1. 按条件搜索生产线，系统需要根据用户输入判断用户需要查询的数据类型，根据不同的类型查找到满足条件的生产线集展示给用户，如图3.9：

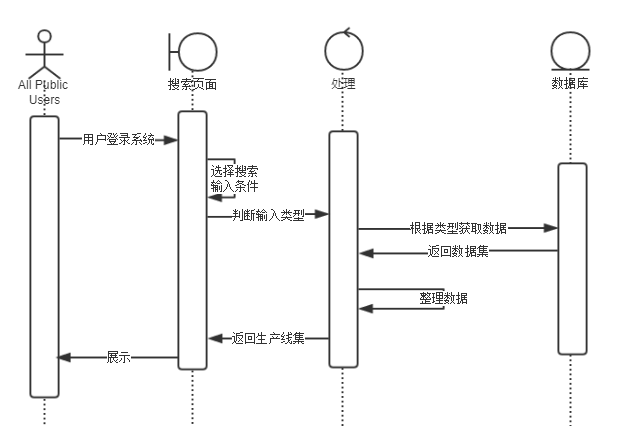


图3.9搜索生产线时序图

2. 生产线组合搜索功能是指搜索生产一个复杂产品的成本最低的生产线组合方案，通常一个复杂的产品是由多个简单部件组成，每个部件都会有对应生产的一批生产线。本文按照具体问题描述、问题建模、解决方案以及仿真实验结果给出算法的仿真实现，在第四章的最后一小节将会详解介绍。

### 3.3.3 参数维护模块详细设计

1. 工厂专家新增可选参数流程见图3.10，增加参数值提供给工厂机器选择，该模块主要是由工厂专家来维护，可对标准化的参数进行增删改查操作。

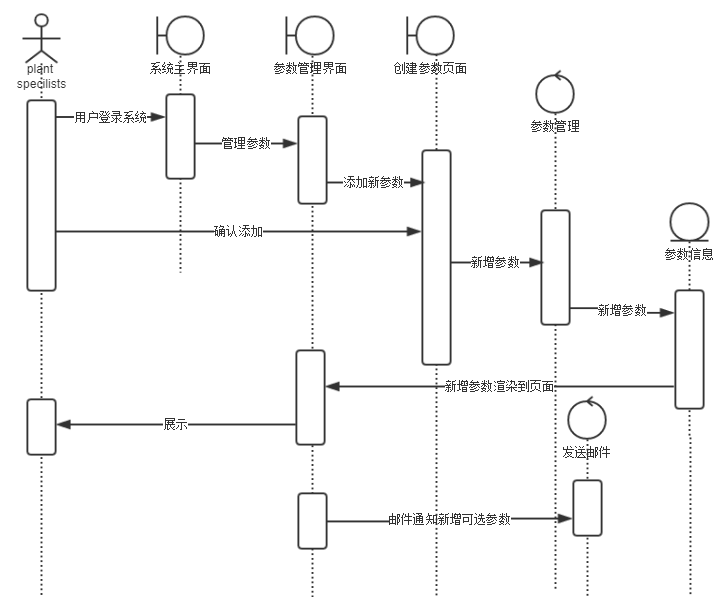


图3.10添加可选参数时序图

2. 工厂专家修改可选参数过程见图3.11，也可以是添加参数的可选值。

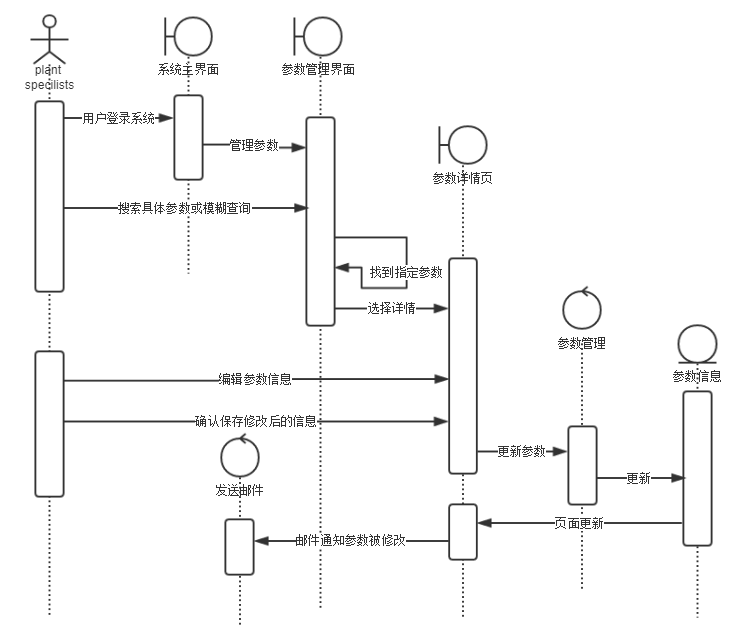


图3.11修改参数时序图

### 3.3.4 数据统计导出模块

1. 数据统计导出模块介绍如图3.12所示，用户选择要导出的数据类型，并选择需要数据有哪些具体的工厂，如果是统计机器类型和参数，还需要选择需要统计的机器类型与具体哪些参数。

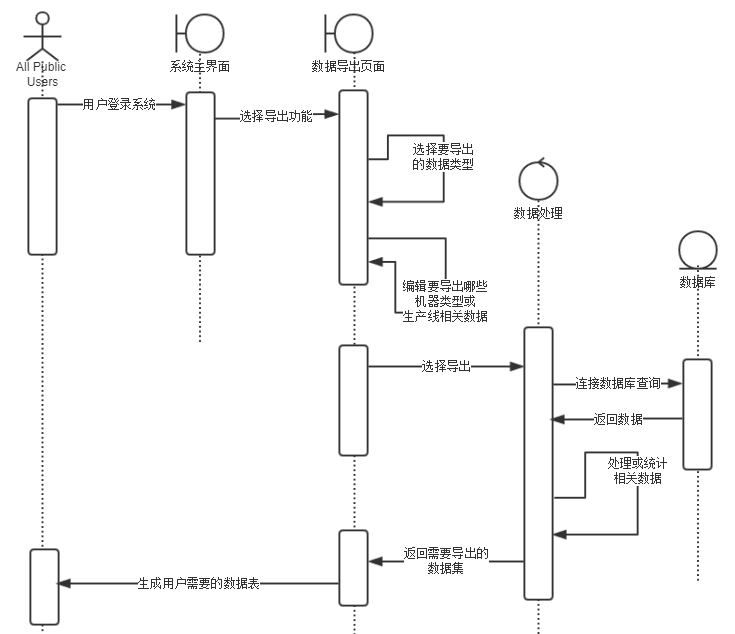


图3.12数据导出时序图

## 3.4 系统数据库设计

通过数据可以明确系统所要实现的具体功能以及很清晰地了解到各个功能之间的关系。由于系统使用的Play框架可以借助JPA来将对象持久化到数据库中，因此开发人员不用关心数据的具体关系和细节，只需要通过对象的方式来使用数据。

本系统中主要用到工厂表、生产线表、工序表、参数表、设备表等，在设计用户表之前，首先要明确使用系统的主要角色：

**角色描述：**

普通用户：可以对工厂机器数据进行读取、查询、导出这些基本操作。

工厂协调员：负责自己所在工厂的生产线于机器数据维护。

工厂专家：负责自己所在工厂下所负责的具体生产过程下的参数维护。

工厂领导：某一个生产过程负责人，可以维护该生产过程下的机器参数。

系统管理员：对整个系统进行维护故障排除，拥有系统的最高权限。

在图3.13中，可以看到与用户相关联表为工厂表和工序表。这是由于不同的角色是属于不同的工厂，每个人的分工不同，因此负责的process也不同。当用户只关心查询搜索功能时就不用关心工厂和生产过程，如果需要对数据进行维护工作则要考虑到用户在具体工厂具体生产过程下的不同权限。

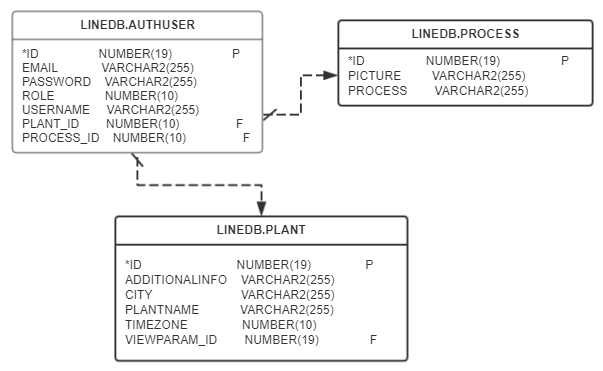


图3.13与用户相关联的表

图3.14展示了与生产设备所关联的表，以及这几个表之间的关系。Device设备表中存储所有生产线中所有的机器数据，这里的每一条数据都代表一台实体机器，可能会有多台同类型的机器出现在不同的生产线。

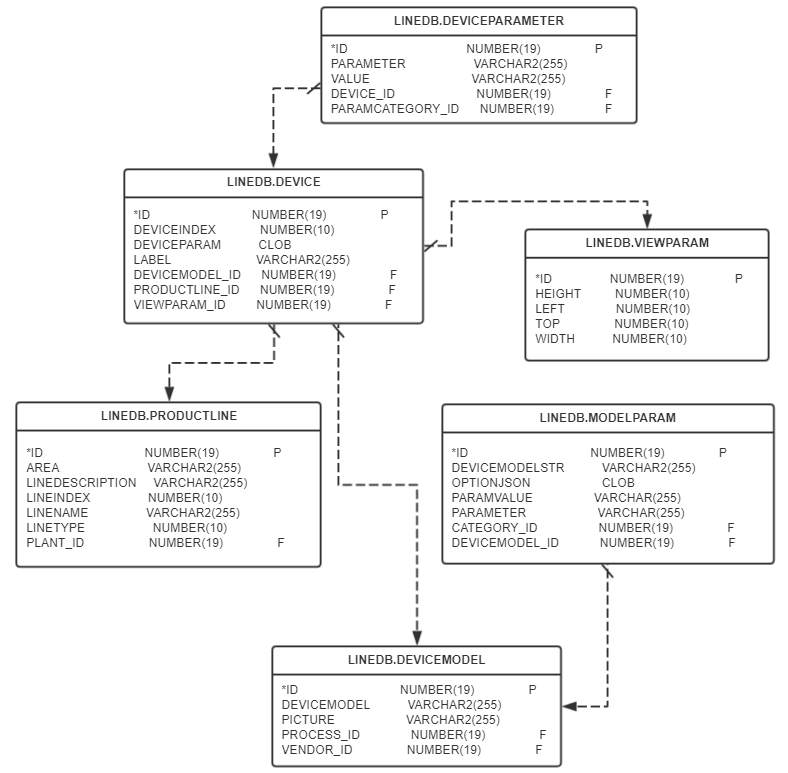


图3.14与机器关联的表

接下来通过数据流图来展示相关数据在生产线数据管理系统中的流动和处理过程如图3.15所示，在图中描述了系统中几个重要角色对工厂生产线、机器数据的处理、加工过程，以及这些信息被加工后流动提供给不同的用户使用的过程。这里以D2机器信息表为例，首先工厂管理人员对机器进行管理维护，操作的是D2机器信息表，配置机器参数需要从D3参数信息表中获取标准化的参数值进行配置。普通用户在查询、条件查找机器信息时，包括所在生产线、工厂等，导出机器统计相关数据也需要从D2机器信息表中获取对应的数据。

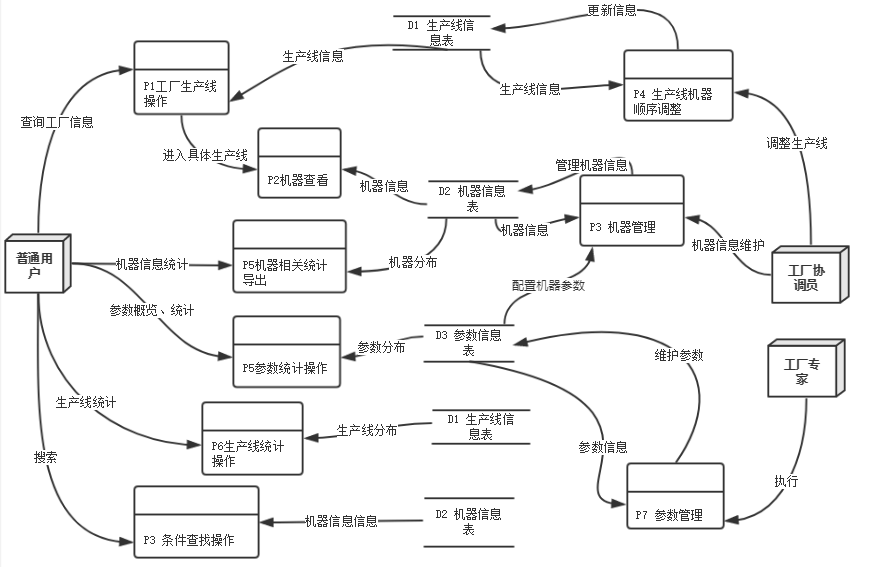


图3.15数据流图

## 3.5 本章小结

本章是对生产线数据管理系统的详细设计，包括对系统整体模块的划分设计，系统的用例图、用例描述，以及各个功能模块使用的顺序图，通过顺序图可以清楚地知道各个功能模块的使用流程。还有数据库设计，包括数据库表的设计、数据流图等。

# 第四章 工厂生产线数据管理系统的实现

本章将使用具体的技术将第三章中设计的系统进行实现，首先讲述了基于Play框架构建和部分功能模块的实现，最后基于该系统的拓展部分讲解了相关算法并给出实现方案。

## 4.1 数据连接及处理的实现

由于系统选择了Play框架完成开发，因此只需要在配置文件application.conf中添加如图4.1即可启用Oracle数据库：

db.default.url=jdbc:oracle:thin:@10.8.214.42:1521:STDFTEST

db.default.driver=oracle.jdbc.driver.OracleDriver

db.default.user=linedb db.default.pass=\*\*\*\*\*

图4.1 Play框架数据库连接配置

JPA用于对数据库操作的数据模型实体类操作，帮开发人员减少了许多臃肿的代码，下面举例子描述在本项目中是如何用JPA进行开发的。

首先实体类需要继承Model类，且要有@Entity注释，这里以DeviceModel类为例，如图4.2：

@Entity

public class DeviceModel extends Model{

@Required

@ManyToOne

public Process process; // 工序

@Required

@MaxSize(64)

public String deviceModel; // 设备型号

……

@OneToMany(mappedBy="deviceModel", cascade= CascadeType.ALL)

public List<ModelParam> modelParam;

public String toString(){

return this.deviceModel; }

public DeviceModel(){

super();

}……

}

图4.2数据模型实体类

然后通过.save(), .delete(), .find()等对实体类进行操作，图4.3为判断添加的机器设备是否已经存在：

public static void isExisted(){

RestRes res = new RestRes();

res.code = -1;

String body = request.params.get("body");

ParamTree params = new Gson().fromJson(body, ParamTree.class);

Process curprocess = (Process) Process.find("byProcess", params.process).fetch().get(0);

DeviceModel curDM = (DeviceModel) DeviceModel

.find("byProcessAndDeviceModel", curprocess, params.machineType).fetch().get(0);

……

List<ModelParam> mps = curDM.modelParam;

for(ModelParam mp : mps){

if(params.parameter.equals(mp.toString())){

res.code = 1;

}

}

renderJSON(res);

}

图4.3数据模型实体类操作

然后在routes文件中添加路由如图4.4：

POST /device/isExisted DeviceController.isExisted

图4.4配置操作方法路由

就可以通过/device/isExisted判断当前设备是否已存在，体现了REST风格。

## 4.2 工厂数据总览模块

在Play中，这些功能的实现主要涉及到它的三个组件—Router、Controller和Template engine。我们通过工厂数据总览模块的实现来描述三者的关系：

首先，用户在首页或其它页面选择查看所有工厂，然后Router将服务器接收到的http请求转换为对应的获取工厂信息的Action调用（每个Action对应Controller中的一个public静态方法），Controller接受并处理由Router转发过来的调用，根据请求的内容对模型对象进行访问或处理，Controller根据指定的模板动态地生成HTML、XML和JSON等文本格式的文档，最后系统返回给用户结果页面，工厂数据总览模块的实现流程如图4.5。

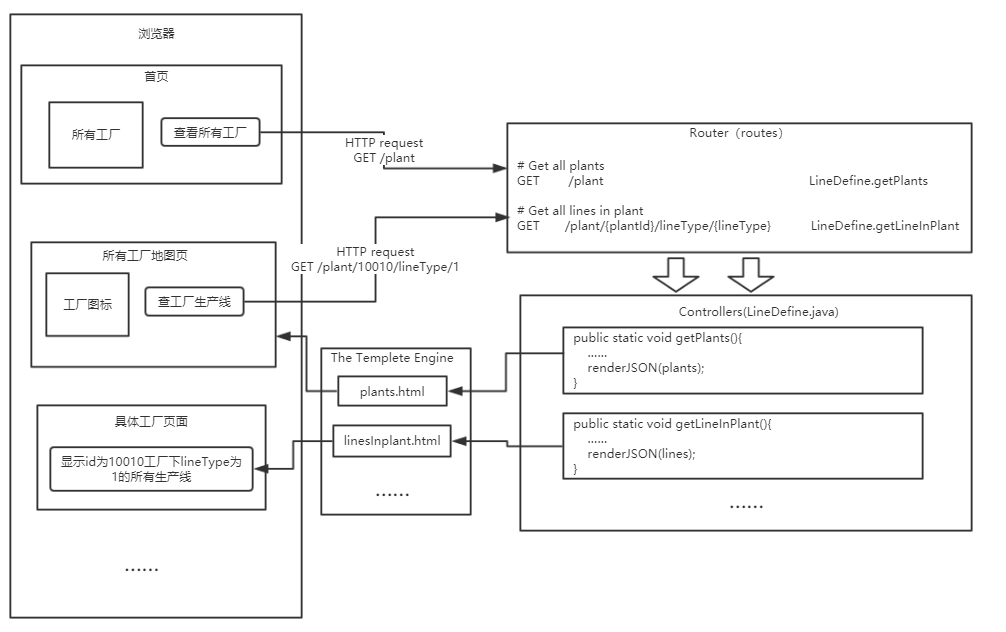


图4.5工厂数据总览模块的实现流程

用户点击其中一个工厂图标则进入该工厂的具体信息页面见图4.6：

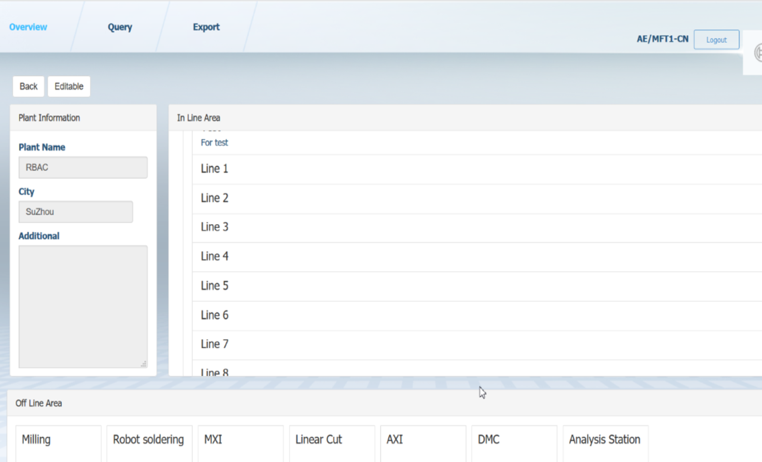


图4.6具体工厂详情页

根据用户的不同角色，拥有不同的用户权限，可以对该工厂下的生产线进行相对应的操作，比如说系统管理员，或者该工厂的管理者可以使用editable按钮对该工厂相关数据进行调整等。点击任一条生产线，这里以RBAC工厂下的line4为例见图4.7：

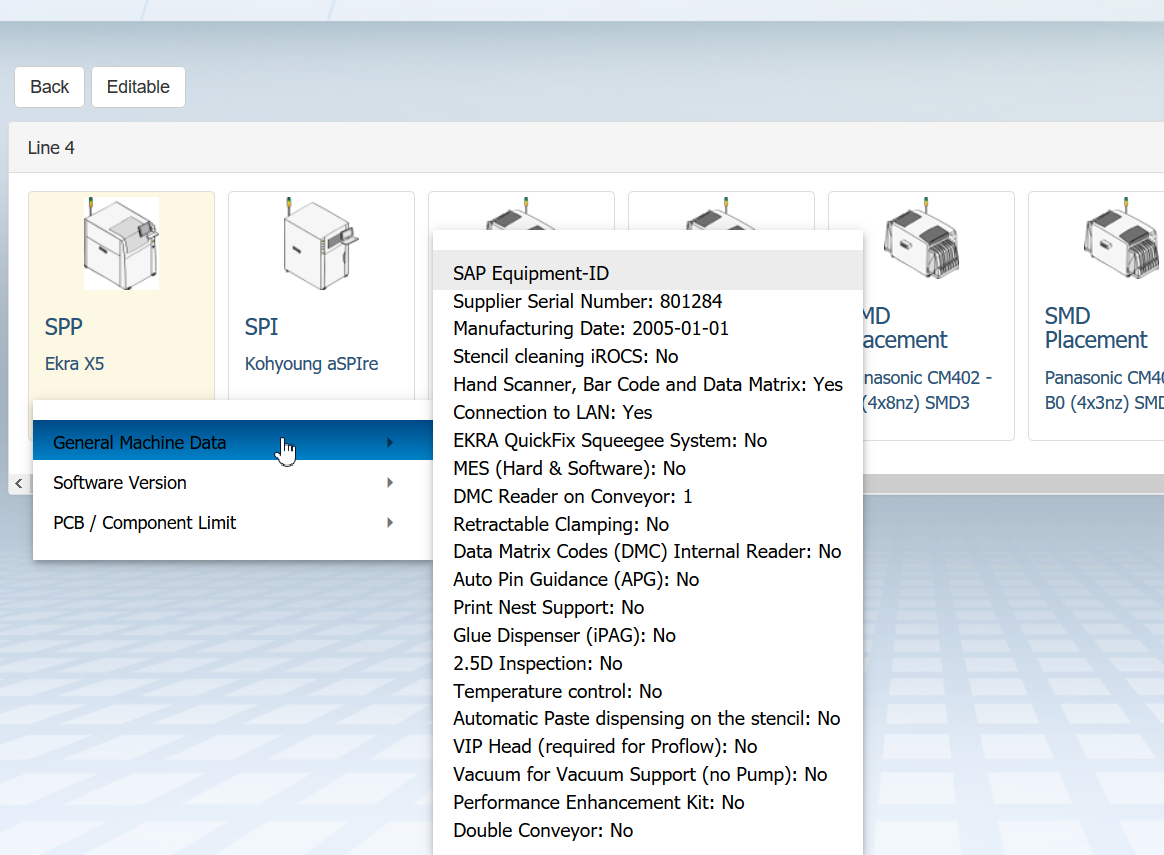


图4.7具体机器信息查看页面

用户可以看到产线的当前情况，很多时候比如像统计这些工作都要进行核对，以前都要打电话发邮件，甚至需要去工厂确认，现在用户就可以省去很多不必要的麻烦，而且不同国家的工厂就可以互通了。

点击editable,右上角则会多出一个create按钮，在上图中可以看到有的机器背景色不一样，这里不同的颜色代表不同的progress，截图上只显示了两种颜色，可能还有绿色、红色等，每种颜色都代表一个不同的progress，机器的顺序可以进行拖动调整，当然必须由拥有相关权限的用户来操作。点击create，拥有权限的用户则可以创建新的机器，也可以对已有机器的参数值进行调整修改，这里数据创建好存入对应device表中，这里为机器设定的参数值必须为系统中已存在且在维护的参数值里选，不可以由创建机器者自定义。

## 4.3 生产线搜索模块

生产线搜索模块用来检索满足条件的生产线，比如用户想要查看某个机器类型或者某参数都有哪些工厂包含，用户就可以通过该功能快速定位到具体工厂下具体的生产线，以及具体哪一台机器包含该参数。在本文第四章中生产线组合解决方案中，需要找到每个生产任务的搜索空间，这里就可以通过指定条件筛选出所有满足生产条件的生产线，得到搜索空间。

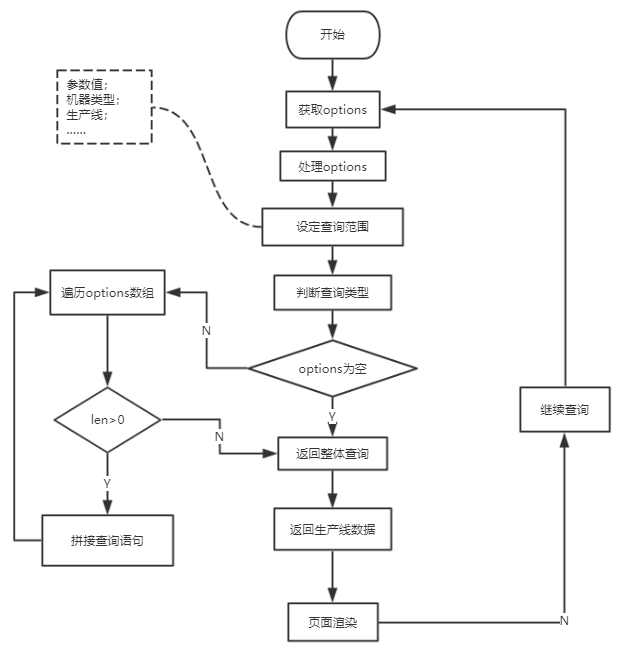


图4.8生产线搜索实现流程图

如图4.8是搜索的实现流程，上一小节主要关注的是系统整体的一个实现流程，这里是具体功能的实现，主要就是对用户要查询的内容进行判断，设定查询范围，根据条件拼接查询语句，最后返回结果给用户。如图4.9输入需要满足的参数条件或者其他信息，所有包含该信息的生产线及具体机器都会列出。

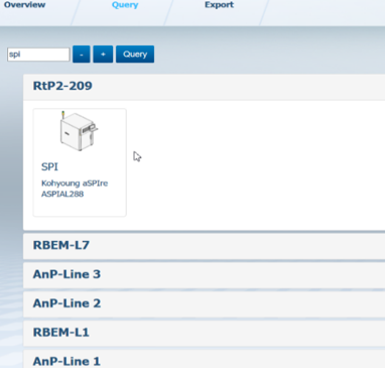


图4.9生产线查询模块

这里展开后用户可以知道具体是那一台机器包含该参数，若查询条件是生产工序，则查询结果就会列出所有可以完成该生产工序的生产线，以及各条生产线完成这道工序的机器。这里系统得到的生产线集合返回给前端，页面部分根据需求进行相关内容展示。

## 4.4 参数维护模块

参数维护模块的实现部分与前面有一些不同，我们选择Play框架的CRUD模块来实现。CRUD模块是Play提供便于原型管理的组件，该模块会检查Model类的内部结构，并自动创建简单的管理列表和表单，快速生成一个基本的参数管理区域。只要模型继承了Model类，就有save()，delete()等方法可以调用，通过操作实体类操作数据库表。在conf配置文件中添加相关配置就可以启动该模块了，需根据通用规则配置路由（第二章有配置方法）。另相关处理类需要继承play.CRUD类，如图4.10：

@Check("ADMIN&&CNLEADER")

@With(Secure.class)

public class DeviceModels extends CRUD {

}

@Check("ADMIN&&CNLEADER")

@With(Secure.class)

public class ModelParams extends CRUD {

}

@Check("ADMIN&&CNLEADER")

@With(Secure.class)

@CRUD.For(ParamCategory.class)

public class ParamCategories extends CRUD {

}

图4.10继承CRUD的相关类

如果系统有特殊要求，可以对相关方法进行重写，比如默认方法中用户操作完成后会返回首页而不是当前页面，因此在重写方法时可以讲页码等信息作为参数传递给方法，比如save(int pageN)，在最后渲染的时候返回具体页面。

图4.11展示了参数维护模块，工厂专家或是系统管理员可以根据职责在该模块对机器涉及到的所有的参数进行管理维护：

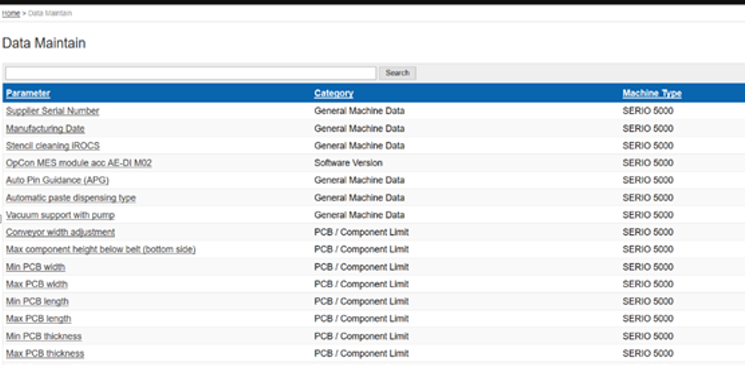


图4.11参数维护页面

这些参数值均为标准化后的数据。选择添加新的参数或选择已有的参数进行维护，这里管理的就是前面的ModelParam表中的数据，且对参数的编辑只能由工厂专家来执行相关操作。

## 4.5 数据统计导出模块

数据导出模块是根据用户需求对一些数据进行统计，功能并不复杂，但在导出表排版较麻烦，首先判断用户需要导出的数据类型，通过routes将请求转化为不同导出类型的action的调用然后根据不同分支的处理方法对数据进行处理、统计等工作，最后导出如图4.12，主要工作就是将工厂、生产线、机器类型、参数等信息按照横纵坐标写入表中，再对相应的数据进行处理统计找到对应的坐标并写入。如图4.13有export Data， machine Type Summary，parameter Summary，Lines Summary四个选项。

第一栏导出参数是对参数的一个概览，当用户需要调查一些参数信息时，可以通过导出具体参数，得到所有包含该参数的工厂及生产线，该生产线下该参数的值会在表中可以清晰地展示，表格可以提供给部门做相关分析研究。

第二栏是用来统计工厂下包含某个指定machineType即机器类型的数量，选择多个工厂加入对应的工厂列表，选择多种机器类型加入对应的机器类型列表，选好数据后导出得到各个工厂机器类型统计结果。

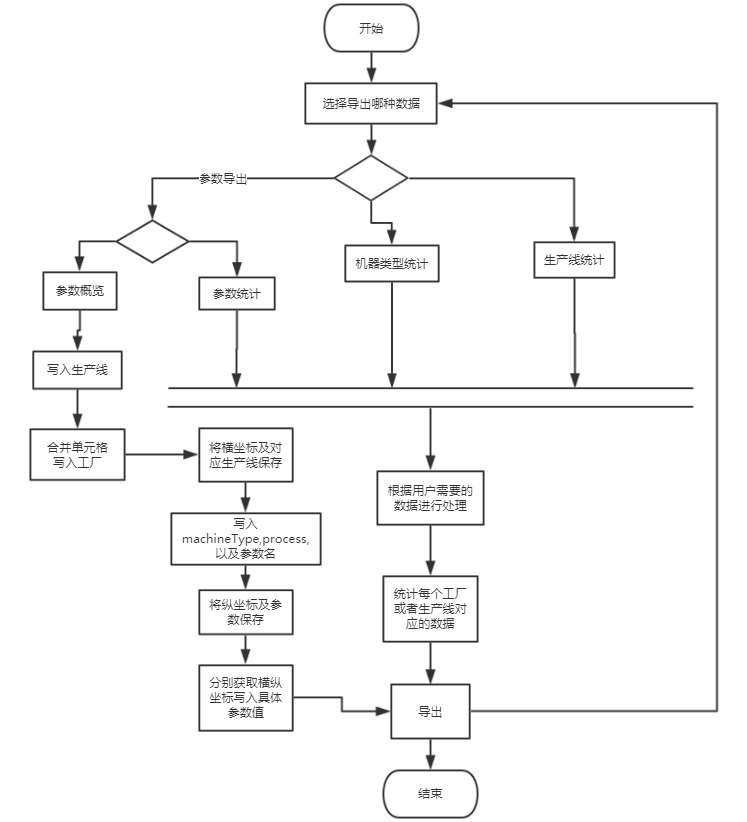


图4.12数据导出实现流程图

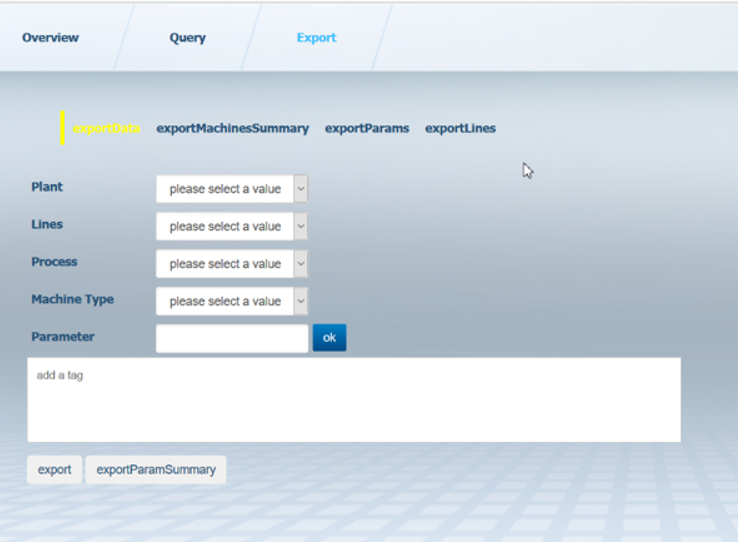


图4.13数据统计导出模块页面

机器类型统计可以为管理者或其他用户在做出某种决策时提供依据。比如某管理者现在需要给工厂拨款，可以通过该系统找到具体工厂并导出相关数据，很清楚地看到该工厂下的这些机器型号都有多少台，可以做出相对准确的预算。同时还有个好处就是增大了各个部门之间的透明度，提高了管理效率。

第四栏是生产线统计数据导出，这里统计各个工厂生产线的数量主要还是用来做一些预算，得到各个工厂生产线汇总信息。图4.14展示了参数整体数据导出的部分实现。

public static void export(){

String subStr = getExportBody();

List<ExportParam> eps = transStrToExportParam(subStr);

Set<String> proMacParamSet = new HashSet<String>();

……

while(epsItera.hasNext()){

遍历得到数据并做相关处理

……

}

try {

String tmpDir = Play.configuration.getProperty("linedb.tmpDir");

生成文件及路径，

创建并将容器数据按格式写入文档

//write MachineType

Map<String, Integer> proMachBeginCol = writeMachineType(sheet, proMachNum, processBeginCol, horizonFormat);

//write parameter

Map<String, Integer> proMacParamCol = writeParams(sheet, proMachBeginCol, proMacParamSet, cv);

//write lines

Map<String, Integer> plantAndLineMap = writePlantsAndLines(sheet, plantSet);

将容器存放的数值及坐标写入表，填充具体参数值。

下载文件流。

} catch ……

}

图4.14导出参数部分代码展示

后期可以再添加一项导出生产线组合方案的数据表。在此做一个简单的设想：用户可以选择所需要生产的产品，这里的产品系统会先提供一些备选项。如果用户没有找到需要的产品，可以自行选择组成该产品的各个部件，并对一些特殊的有具体要求参数进行设定并导出。用户就可以获取一组最佳的生产线组合方案。下一小节将讲述如何进行生产线组合。

## 4.6 生产线组合的实现

### 4.6.1 问题描述

一般情况下一个复杂的产品需要由许多简单的产品或是部件组成，而生产不同的产品需要由不同的生产线来完成不同的工作。通常情况下一个产品会有多条生产线可以生产，但这些生产线的生产效率和成本各不相同。而本文的最终目标是选出每一个任务对应的生产线，使得生产整个产品的总时间满足截至时间要求，并且生产成本最低。

这里以电子助力转向系统为例，一般情况下电子助力转向系统由转向传感装置、车速传感器、助力机械装置、助力转向电机及微电脑控制单元构成见图4.15。而传感器、机械装置、助力电机、微电脑控制单元等都有一批可选的生产线可供使用。

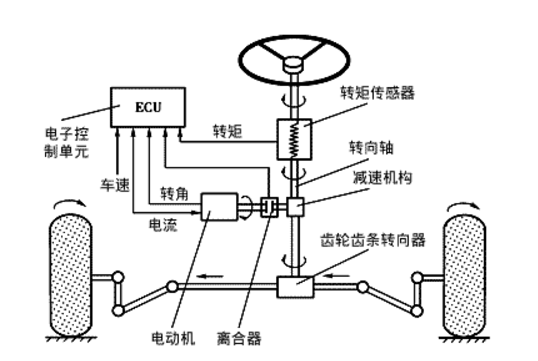


图4.15电子助力转向系统结构图

如果汽车不转向，电子控制单元就不会向电动机控制器发送指令，从而电动机也不运作。因此，当前假设转矩传感器必须在转向助力电机之前生产，电动机可以并行生产，以上两步完成再到微电脑控制单元，其间可与其他部件并行生产，最后组装。生产一个产品的过程可以通过一个工作流过程模型表达，生产电子助力转向系统的过程模型如图4.16。

其实找出产品的生产线组合方案是一个服务组合问题的特殊应用场景：每个生产线都至少可以生产出一种产品或一个部件，而一个部件可以有多条生产线满足生产要求，即每个部件对应着多条生产线可以满足生产需求，目标就是在这些生产线集中按一定的顺序选取出合适的产线，组合成可以生产出一个复杂产品的生产线组合并使得在规定时间内的成本最优。

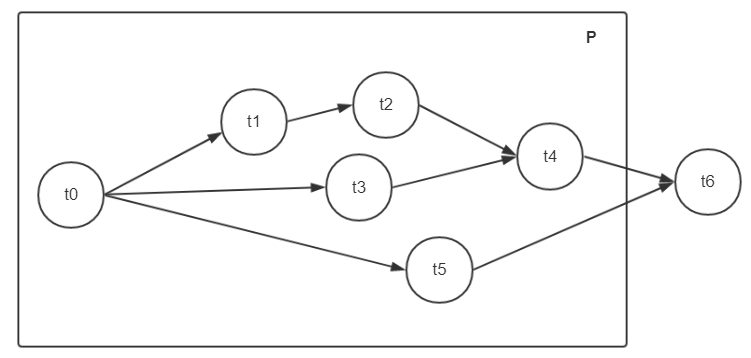


图4.16生产任务模型

运用改进的粒子群求解基本思想如下：首先按照一定的规则产生一定数量的粒子，该规则可以是根据需要生产产品选出一批满足条件的生产线，并且根据生产流程图确定好各个生产任务之间的关系，且要求在规定时间内完成等。每个粒子都是一个可能的生产线组合方案，然后粒子按照改进的算法过程进行迭代，产生具有更好适应值的粒子，直到算法终止。

### 4.6.2 问题建模

在本节中，描述了构成算法基础的工作流模型对各个部件的生产任务之间关系进行建模，生产任务执行过程分析，问题表述。表4.1介绍了在本文中使用的符号及其语义。

表4.1符号含义表

|  |  |
| --- | --- |
| **符号** | **定义** |
| *P* | 某个产品 |
| *ti* | 生产任务工作流*ti* |
| *S-alli* | 任务*ti*对应的所有可用生产线集合 |
| *Si* | 任务*ti*对应的搜索空间 |
| TD | 生产任务工作流截止时间 |
| *csk* | 单位时间内在生产线上生产成本 |
| *PLS* | 可以生产某一种产品的生产线集S |
| *plks* | 生产线集S系列中的一条k产线。 |
| ET（*ti* , *plks*） | 生产任务*ti*在plks上的执行时间。 |
| ST(*ti*) | 任务开始时间 |
| ET（*ti*） | 任务完成时间 |
| TT（*ti*） | 任务ti上的资源转移时间 |
| PT（*ti* ，*plks*） | 部件生产消耗总时间 |
| PST（*ti* ，*plks*） | 生产开始时间 |
| PET（*ti* ，*plks*） | 生产结束时间 |
| *ϕi* | 加速系数 |
| TEC | 生产总执行成本 |
| TET | 生产总执行时间 |
| *N* | 粒子的数量 |

本文使用有向无环图DAG（T, E）为生产任务工作流建模，其中T代表可以提供生产服务的生产线集合，T = {*t0*，*t1*，...，*ti*，...，*tn*-1，*tn*}是n+1个生产任务; E代表两个生产任务间的生产依赖关系。例如，边e（*i*，*j*）表示任务*ti*应该在任务*tj*开始之前完成其执行的约束条件。在这种情况下，任务*ti*表示任务*tj*的前驱之一，任务*tj*表示*ti*的直接后继者之一。*ti*的前驱任务集合表示为pre（*ti*），后继任务集合为succ（*ti*）。一个任务可以有一个或多个输入(这里的输入为生产线服务的参数，如灵敏度、可靠性、可用性、费用等)，一旦这些输入条件都满足就会触发任务的执行。假设任务*ti*的输入为*Ii*。当任务*ti*完成时，它产生输出*Oi*。在DAG中没有前置任务的任务称为入口任务*tentry*，没有任何后续任务的任务称为出口任务*texit*。在本文中，假设一个DAG只有一个入口任务和出口任务。一个简单的生产工作流模型如图4.16所示，每个生产线服务被表示为图中的一个节点。另外，每个工作流都有一个截止时间TD，它被定义为工作流执行的时间约束。

在图4.18中，一个任务对应所要生产产品的一个部件，首先从所有满足生产条件的产线*S-alli*中找出每一个部件对应的可用的搜索空间*Si*；这里简单介绍一下如何来确定搜索空间：当前假设事先知道哪些生产线可以生产同名产品，但是一些必要参数或者生产截止时间不能够满足的生产线需要被剔除，首先要知道生产的产品有哪些指标参数，然后根据这些指标来选择一批可用的生产线。以传感器为例，选择传感器需要从灵敏度、稳定性和精度等这几个方面去判断，这些性能都与传感器所选用的基板的制造工艺技术有关，因此在挑选电路板时需要考虑一些参数指标，表4.2列出几个参数作为参考，纵向代表参数名称，横向代表机器类型（每条生产线都有很多台机器），根据生产线中机器的参数值范围，就可以为每一个产品确定一个对应的搜索空间*Si*了。

表4.2产品部分参数表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Aspire2** | | **CM402 - A0 (4\*8nz)** | | **X7056 (AXI only)** | |
| **Max PCB thickness** | | 5 mm | |  | |  |
| **Max PCB weight** | | 5000 g | |  | |  |
| **Camera Type** | | 2M, 4M, 8M | |  | |  |
| **Warpage Sensor** | |  | | Yes, no | |  |
| **Chip Thickness Sensor** | |  | | Yes, no | |  |
| **Detector Type** | |  | |  | | 2D, 2.5D |

每一个部件的*Si*一般仍会包含多条产线，下一个问题是从每个部件的搜索空间中都得到一条生产线组成生产线集，以确保工作流的高效执行和成本最低；拟通过粒子群优化算法PSO算法解决产线组合问题。

**生产任务执行过程分析**

假设现已知某一产品可生产产线系列PL = {PL1，...，PLs，...，PLS}，每个系列包含一组满足指定部件生产线集PLs = {pl1s，pl2s，... ，plks }。 例如，电子助力转向系统各部件产线系列（转向传感装置、车速传感器、助力机械装置、提供转向助力电机等），本文用line1, line2, line3, line4来表示生产线。生产线集plks由部件所包含的相关参数是否满足条件标准来指定，包括传感器的额定载荷，扭矩大小范围等，单位时间内生产特定产品的数量，单位时间所需成本，部件使用寿命等等。此外，假设生产线成本计算按单位时间收费。

1. 假设同一条生产线上执行的两个任务传输时间为0，接下来，可以通过使用输入数据来执行任务*ti*，设在plks上的生产任务*ti*的执行时间：
2. 将*ti*所需资源从*ti-*1所在生产线转移到ti的产线的传输时间为：

3. 部件生产消耗总时间： （1）

**问题表述**

本章节着重于找到一个来执行前面所描述的工作流程的方案，使总执行成本最小化，并满足约定时间期限。首先定义一个生产线调度计划 schedule =（R, M, TEC, TET）分别表示：一组生产线资源，生产任务的资源映射，生产任务总执行成本，生产任务总执行时间等。 R = {*r0*, *r1*, ..., *rn*-1}是每个任务的调用生产线信息的集合，其中*ri*是一个三元组的形式*ri* =（plks(*ti*), PST(*ti*，*plks*), PET（*ti*，*plks*））。plks(*ti*)是生产任务调度的生产线，具有生产开始时间PST（*ti*，*plks*）和生产结束时间PET（*ti*，*plks*）。M表示映射关系，并且由m（*ti*，*plks*）=（*ti*，plks（*ti*），ST（*ti*），ET（*ti*））形式的元组构成。映射元组解释如下：任务*ti*为*plks*的一个调度任务，并且预期在ST(*ti*)时间开始，ET(*ti*)时间完成。方程（2）和（3）用来计算总执行成本TEC和总执行时间TET。

（2）

（3）

由于生产产品部件的产线是按单位时间的定价模式向用户收费，所以*csk*·（PET（*ti*，*plks*）- PST（*ti*，*plks*））的值是在*plks*上执行的任务*ti*的成本。目前先不考虑各个生产线之间调度运输等费用; 因此在计算生产工作流程的执行成本时也不考虑这个费用。

基于前面的定义，问题可以形式化定义如下：找到TEC最小，TET值不超过限定的调度计划，描述如下：

Minimize ： (4a)

Subject to： (4b)

### 4.6.3 解决方案

**PSO（粒子群算法）**

本文使用PSO（粒子群算法）搜索满足优化要求的生产线组合。在粒子群算法中每个粒子由一个三维向量表示，当前粒子位置代表优化问题的候选解。如果当前粒子位置比之前所发现的任何位置都好，则将其存储在矢量中。到目前最佳值被存储在变量*pbesti*中[胡珀, 2011]。三维向量表示为:

当前粒子位置;

历史最佳位置;

速度。

粒子群不仅仅是一个粒子集合。单个粒子是无法解决一个复杂的问题，只有粒子相互作用时才会发生进展。通过粒子间的相互作用，从粒子的个体行为中求解。每个粒子与其他一些粒子进行通信，并受其拓扑邻域任何成员的最佳点影响。向量*pg*表示向量*pi*的最佳邻居向量，最佳值表示为*gbest*，最终*gbest*即为群体的最优解。在这里使用整个粒子群的全局最优拓扑。在粒子群优化过程中，算法会一直进行迭代，直到满足停止条件。这里的停止条件通常是指达到最大迭代次数或者预定义的最佳值被认为足够好。在每次迭代中，粒子的速度和位置都根据方程式更新。

（5）

（6）

其中φ1和φ2是两个正数；rand1，rand2是两个在[0,1]范围内均匀分布的随机数。公式（5）（6）合称为标准的粒子群算法（SPSO）。

**PSO算法参数**

公式（5）包含影响算法性能的各种参数; 其中一些参数对算法的收敛性有显著的影响。参数w(weight)称为“惯性权重”。如果将*φ1* · *rand1*（*p⃗I - x⃗i*）+*φ2* · *rand2（p⃗g - x⃗i*）解释为作用于粒子的外力*f⃗i*，那么粒子速度的变化(即粒子的加速度)可以是写成△*v⃗i*=*f⃗i*- (1 –*w) v⃗i*。即常数1-*w*作为摩擦系数发挥作用， *w*可以理解为粒子移动的介质的流动性。因此，这个权重决定了以前速度对当前速度的影响大小。大的权重会使速度增加，从而有利于全局搜索；较小的权重值会使粒子减速，因此有利于局部搜索。所以平衡全局和局部搜索的*w*值建议随着迭代线性递减，以便算法收敛。参数*φ1*和*φ2*决定了*pi*和*pg*对新速度的影响程度，被称为加速度系数。较小会限制粒子的运动，较大可能会导致粒子发散。一般来说，这两个加速度常数被认为是一个单一的加速度常数*φ*=*φ1*+*φ2*> 4。

还有其他的参数不是速度定义的一部分而作为算法的输入。一个参数是群体的大小。这通常是根据问题的维度和感知困难经验设定的。比较常见的范围是20-50。其他两个参数是粒子的维度和它们被允许移动的范围，这些值由待解决问题的性质决定，以及如何对它进行建模以适应PSO[Zhongjin et al, 2016]。

**基于PSO的CAS算法**

本文的成本感知调度（CAS）算法是基于PSO的。伪代码如图4.17，下面的部分详细介绍了相关的实现步骤。

Begin

1 将粒子的数量设置为N，将粒子的维度设置为D.

2 设置全局最佳值gbest=0，p⃗g =0

3 初始化具有随机位置和速度的粒子群体

4 for each particle i=1 to N

5 随机初始化x⃗i和v⃗i

6 令p⃗i=x⃗i并计算pbesti的值

7 end for

8 while没有达到最大循环次数

9 for each particle i=1 to N

10 根据D的约束处理，基于工作流调度生成和违规值评估每个粒子的适应函数

11 将当前粒子适应性评估与pbesti作比较，如果当前值比pbesti更好，就将pbesti设置为当前适应值，p⃗i就为D维空间中的当前粒子位置x⃗i

12 end for

13 选择到目前为止所有粒子的适应值最好的粒子，并将其索引分配给变量p⃗g，最佳适应值表示为gbest, 即gbest=max{Pbest1…Pbesti,…, PbestN}。

14 for each particle i=1 to N

15 根据方程（5）（6）更新粒子的速度和位置

16 保持粒子在搜索空间内，以防超出边界。（避免生成不在有效搜索空间上的解决方案）

17 end for

18 计数器递增

19 end while

End

图4.17算法1

输入：粒子维度*D*，*N*个粒子的集合，适应度函数（图4.18算法2）。

输出：最佳粒子位置*p⃗g*，最佳适应值*gbest*。

这里的一个粒子表示一种生产线组合方案。维度*D*表示生产一个产品所包含的*D*个生产任务。现假设生产电子助力转向系统有多个子生产任务，分别为生产传感装置、生产助力机械装置、生产转向助力电机等。每一个小的生产任务根据产品相关要求有多条生产线可以满足生产条件，但只需要选择其中一条。假设传感装置由生产线*line1*来生产，助力机械装置由*line5*，转向助力电机由*line3*,……，那么{*line1, line5, line3,*……}就为一种生产线组合方案，代表一个粒子，而维度D就表示这些子任务的个数。

**编码策略**

为了定义问题的编码方案，需要确定一个粒子的含义和维度。本论文的目标是为每个任务选择适当的生产线，以最大限度地减少总工作流程执行成本，同时满足最后期限的限制。为了基于PSO解出方程(4a)，本文的编码策略设计如图4.17算法1所示：

Minimize:  （7a）

subject to: （7b）

公式（7a）中Γ(.)是与不同产线集相关联的总执行成本函数。对于这里提出的调度场景，粒子的数量*N*为满足生产条件的生产线组合数量，粒子的维数等于函数Γ(.)中的参数个数，即D = 1·n，这将决定用于定义空间位置的坐标系。



图4.18生产线调度方案

如图4.18所示，其中工作流程有7个任务，因此粒子是7维的，其位置由7个坐标定义。一个坐标属于一个任务。坐标的值可以在相应的搜索空间中变化。一般来说，每个生产任务坐标值对应一个索引，代表分配给该任务的生产线集。按照下图给出的例子，每个任务需要不同的实例系列，因此处于不同的实例系列空间。坐标0对应于任务*t0*，其值为1表明着该任务被分配给对应的生产线集1（这里每个生产任务能够满足生产条件的生产线可能存在重叠，即同一条生产线可以生产多种产品的情况）。同样的逻辑适用于其余的坐标和他们的生产任务。下图也给出了从生产流程到对应生产线集的可能调度方案。

**工作流调度生成**

粒子在当前位置调度的伪代码如算法2所示。最初，生产线集R，生产任务到产线资源映射集合M是空的，总执行成本TEC和执行时间TET设置为零。 然后开始解码粒子的位置并构建调度。为了达到这个目的，需要遍历位置中的每个坐标并更新R和M。首先要确定哪个任务以及哪个产线集与当前坐标及其值相关联。这就需要使用前面描述的编码策略来实现，其规定坐标i对应于任务*ti*, pos[i]，对应于可完成对应任务的产线。

首先计算任务的开始时间ST（*ti*）的值。开始时间ST（*ti*）基于两种情况。在第一种情况下，如果任务没有父任务，即可以尽快开始执行。在第二种情况是，如果任务有一个或多个父任务，在这种情况下，只有父任务完成后才能开始执行。

因为生产线生产基于单位时间定价，所以可能会存在很多空闲时间。例如，假设任务*ti*-1是任务*ti*的前驱; 如果任务*ti*-1的处理时间是PT（*ti*-1）= 80min，这里仍然使用两个小时。这样生产线将处于空闲状态40min，可以被任务ti重新使用。而且不需要将任务*ti*-1的产出配送到任务*ti*。因此对于可满足相关生产条件的产线进行重用可以降低工作流程的执行成本。在这种情况下，如果任务*ti-*1和*ti*已被映射到同一个产线集，即plpos [i]S（*ti*）= plpos [(i-1)]S（*ti*-1）。生产线复用的情况如图4.19所示。

如果产线被重用，则ET（*ti*）任务结束时间的值根据总处理时间和任务的开始时间来计算。处理时间PT（*ti*，plpos [i]S）由执行时间，产品配送时间组成，根据式（1）得到PT(*ti*, plks )。然后，通过向ST(*ti*) 添加PT（*ti*，plpos [i]S），得到ET（*ti*）的值，即：

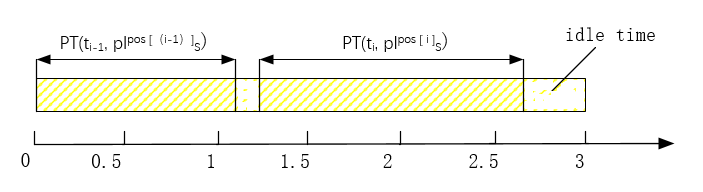


图4.19生产线复用时间示例图

接下来，只需要更新任务*ti*-1的结束时间，并将任务*ti*的开始时间和结束时间设置为零，防止重新计算生产线的生产成本。

如果任务*ti*不重用任务*ti-*1的生产线，则计算任务*ti*的处理时间PT(*ti*, plposs [I])和结束时间ET(*ti*)需要更多时间。然后只计算任务*ti*的生产开始时间PST（(*ti*, plposs [I])）和生产结束时间PET(*ti*, plposs [I])。

Begin

1 初始化调度元素R=M=∅ , TEC=TET=0

2 for each schedulable task ti in T

3 if ti没有父节点

4 设置开始时间ST(*ti*) = 0; //假设工作流开始时间为0

5 else

6 得到开始时间ST(*ti*) = max{ET(*tj*) | *tj*∈pre(*ti*)}；

7 end if

8 if plpos [i]S（*ti*）= plpos [(i-1)]S（*ti*-1）//产线被重用

9 那么任务*ti*-1到ti不需要传输时间;

10 根据方程（1）计算处理时间PT(*ti*，plpos [i]S);

11 计算任务结束时间ET（*ti*）= PT(*ti*，plpos [i]S) + ST(*ti*)；

12 更新任务ti-1的生产结束时间PET(*ti*, plposs [ I -1 ]) = ET(*ti*) ;

13 PST（(*ti*, plposs [I])）= PET（*ti*，plposs [I]））= 0//防止重新计算生产线的生产成本;

14 else

15 根据方程（1）计算处理时间PT(*ti*，plpos [i]s);

16 计算任务结束时间ET（*ti*）= PT(*ti*，plpos [i]S) + ST(*ti*)；

17 生产开始时间为PST（(*ti*, plposs [I])）= ST（*ti*）;

18 计算生产结束时间PET（*ti*，plposs [I]））= PT(*ti*，plpos [i]s) + PST（(*ti*, plposs [I])）

19 end if

20 添加m(*ti*, plposs [I])映射关系到M集合中，ri到R集合中。

21 end for

22 根据方程（2）计算TEC

23 根据方程（3）计算TET

24 记录Schedule=（R，M，TEC, TET）

End

图4.20算法2

现在已经算出M(*ti*, plposs [I]) = （(*ti*, plposs [I](*ti*)），ET（*ti*））和Ri=（plposs [I](*ti* )，PST(*ti*，plposs [I]），PET（*ti*，plposs [I]））。然后更新M和R。

一旦算法完成处理向量中的每个坐标，R将包含所有需要调用的产线资源以及启动和结束的时间。另外，资源映射元组的整个工作流将在资源映射集合M中。有了这些信息可以使用方程（2）（3）计算执行成本TEC和执行时间。在此之后，算法可以构造并返回与给定粒子位置相关联的计划。最后结合上述产生一个接近最优的调度。在图4.17算法1的步骤10中，不是计算粒子的适应度值，而是如图4.20算法2所述生成调度。算法的输入为生产任务集T={t0,……，t6}，输出为生产任务总执行成本TEC，生产任务总执行时间TET。

速度和位置更新方程（5）和（6）容易造成粒子超出可行区域的边界，并且必须修改PSO算法以使粒子保持在约束范围内。这样做可能对算法的性能有很大的影响，因为它影响了粒子在搜索空间周围移动的方式，当最佳决策变量值位于边界上或附近时，那么这个值就很重要。因此，当决策变量超出边界时，要做两件事：（1）决策变量取对应边界的值（或者是下边界或者上边界），（2）其速度乘以- 1，以便在相反的方向搜索。

### 4.6.4 仿真实现和结果

为了让实验比较的结果明显一些，这里定义了一个新的工作流模型（如图4.21）作为算法的输入。



图4.21实验的生产任务工作流模型

同时列出15条生产线，这些生产线可生产产品的品质不同，生产效率和成本也各不相同如表4.3，其中生产效率为单位时间的生产量，单价为单位时间的费用，均是相对值没有具体单位。

在任务*t*1到*t24*中，每一个任务的工作量为[1,32]内的一个随机整数，每个任务需要传送给下一个任务的运输量为[10，100]中的一个随机数，这个会影响任务的执行时间。为了简化应用PSO算法的复杂度，假设所有任务的生产线搜索空间是相同的。

表4.3生产线的效率与单价表

|  |  |  |  |
| --- | --- | --- | --- |
| 生产线ID | 产品质量等级 | 生产效率 | 费用单价 |
| 1 | 普通 | 1.7 | 0.06 |
| 2 | 3.75 | 0.12 |
| 3 | 7.5 | 0.24 |
| 4 | 15 | 0.45 |
| 5 | 30 | 0.9 |
| 6 | 中等 | 1 | 0.05 |
| 7 | 2 | 0.099 |
| 8 | 4 | 0.2 |
| 9 | 8 | 0.401 |
| 10 | 16 | 0.802 |
| 11 | 较高 | 1 | 0.07 |
| 12 | 2 | 0.14 |
| 13 | 4 | 0.28 |
| 14 | 8 | 0.56 |
| 15 | 16 | 1.12 |

仿真实验的其它假设包括：

（1） 所有不同产品的生产任务不会同时被调度，而是按照优先级顺序依次调度：每一个生产任务的所有单个任务的优先级是相同的。

（2） 每一次安排好生产任务后，生产线的开始时间会按照安排好的任务的执行时间进行更新：新生产任务在新的开始时间上进行调度。

（3） 如果有紧急任务插入已经安排好的生产任务队列中，则在当前任务执行完成后安排调度，并重新调度所有后续生产任务。

首先用PSO算法模拟了生产任务调度，生成了执行时间前50的成本-执行时间属于Parato解的50个最优调度方案。为了显示PSO算法的优点，同时使用Random方法和基于MOHEFT[Durillo et al, 2012]算法对生产任务进行调度。Random方法中随机执行了1000次调度，选出了执行时间前10的Parato[Wikipedia, 2018]解，MOHEFT则同样选择了执行时间前50的最优解。绘制成本-时间曲线如图4.22，可以看出PSO算法的执行结果在多数相同的执行时间内有着最低的生产成本。

图4.22实验结果比较

表4.4为分别给出了执行时间最短、最长和中值时的具体调度方案：

表4.4任务执行结果

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Task | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 最短时间 | 3 | 3 | 4 | 4 | 6 | 4 | 3 | 3 | 1 | 2 | 4 | 5 |
| 最长时间 | 3 | 3 | 4 | 4 | 6 | 4 | 2 | 2 | 1 | 1 | 3 | 4 |
| 中值时间 | 2 | 2 | 2 | 1 | 6 | 2 | 2 | 2 | 1 | 1 | 2 | 3 |
| Task | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| 最短时间 | 3 | 3 | 4 | 10 | 5 | 9 | 4 | 3 | 5 | 3 | 5 | 4 |
| 最长时间 | 4 | 3 | 2 | 3 | 3 | 1 | 5 | 3 | 3 | 3 | 4 | 4 |
| 中值时间 | 4 | 3 | 2 | 3 | 3 | 1 | 5 | 3 | 3 | 3 | 4 | 9 |

由于没有对生产产品的质量标准作要求，且质量较高的生产线在成本和时间上均不占优，因此在以时间和成本最优为目标的各个调度方案中均没有采用11-15的生产线。未来可以研究如何生成限定产品的生产质量的调度方案。

## 4.7 本章小结

本章节是对生产线数据管理系统实现整体的描述，包括对系统已经实现的功能的描述，部分代码展示，部分页面截取展示。本文把系统划分成四个小的模块，并分别对各个模块的实现进行部分的展示，在本章的第五小节，是基于系统的一个重要的应用，即生产线调度组合的实现，在第五小节中，对设计的算法应用的具体工业场景进行了充分的描述，包括举出具体所要生产的产品的例子，以及如何将产品拆解应用到工作流模型中，最后给出解决算法的伪代码。

# 第五章 总结与展望

## 5.1 总结

本文通过对企业工厂现状进行了详细地分析，阐述了在工业4.0背景下，如何管理利用好企业中机器设备相关数据，简单介绍了如何结合生产数据使得系统中的机器能够获得自我意识和自我预测性以实现及时维护，零停机等，让企业离实现工业4.0又近了一步。目前博世公司针对参数设备部分开发了一个基于Play框架的IT系统，为工厂之间沟通交流提供了极大地便利。本文介绍了开发过程所使用的相关技术背景介绍以及该系统的的设计与实现。在系统基础上，结合工业这个特殊的应用场景，为各个生产计划部门在决定生产一种复杂产品提供了一种生产线组合方案，能够使得在指定时间内生产成本最低，此方案结合了Web服务组合中的基于工作流模型的的研究方法，并以粒子群算法（PSO）为基础提出了一种成本感知调度算法（CAS）来提高服务组合的效率。

## 5.2 进一步工作展望

目前系统开始投入使用，但各个工厂数据还需要继续补充完善。关于得到生产复杂产品最佳的生产线集目前只是给出了生产线组合解决方案，如何在页面上以友好的方式展示并提供给用户使用也需要去考虑。另一方面，公司现已经有很多检测生产时数据实时监控机器是否出现故障等类似的独立系统。但这些系统与机器都是分离的，需要由工人参与进行管理，比如系统监测到故障并报警后，需要由工人来确认检查具体机器然后对机器运行系统进行更新升级，未来应该是完全省去人工操作，即机器数据管理系统与生产数据监测系统打通，当发现问题数据时，直接通知机器管理系统中的机器进行升级。

# 

# 参 考 文 献

[Bacon et al, 2013] Darwin, Peter Bacon, Pawel Kozlowski, AngularJS web application development, Packt Publishing, 2013.

[Barrow, 2013] Brad Barrow, Kickstart Your AngularJS Development with Yomen, Grunt and Bower, Auguest 23, 2013.

[B Fabian, 2012] Benjamin Fabian, Steffen Kunz, Marcel Konnegen , Sebastian Muller, Oliver Gunther, Access control for semantic data federations in industrial product-lifecycle management, 2012, 63(9):930-940.

[Charles, 2016] Charles J.Petrie, Stanford Logic Group, *Web Service Composition*, USA: Springer, 2016.

[Durillo et al, 2012] Durillo JJ, Prodan R, Fard HM, Moheft: a multi-objective list-based method for workflow scheduling, Cloud Computing Technology and Science, 2012:185-192.

[Eberhart et al, 1995] Eberhart R, Kennedy J, A New Optimizer Using Particle Swarm Theory, *Proceedings of the 16th International Symposium on Micro and Human Machine Science*, pages 39-43, 1995.

[JPA,2015] http://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html, Java Persistence API, maintained by Oracle Technology Network, 2015.

[JWT, 2017] https://jwt.io/introduction/, Introduction to JSON Web Tokens, maintained by JWT.IO official Website, 2017.

[Manuel, 2016] Manuel Bernhardt, Reactive Web Applications: With Play, Akka, and Reactive Streams, *Manning*, 2016.

[Odersky, 2013] http://www.scala-lang.org/what-is-scala.html, what is scala, written by M.Odersky, maintained by scala org, 2013.

[Petrie, 2016] Petrie, Charles J, *Web Service Composition*, Springer, 2016.

[Play, 2017] https://www.playframework.com/documentation, PlayFramework documentation, maintained by Play Framework Official Website, 2017.

[R Johnson, 2003] Rod Johnson, *Expert One-on-One J2EE Design and Development*, Wrox, 2003。

[Wikipedia, 2018] https://en.wikipedia.org/wiki/Pareto\_efficiency, Pareto efficiency, maintained by Wikipedia Website, 2018.

[Zhongjin et al, 2016] Zhongjin Li, Jidong Ge, Hongji Yang et al, A Security and cost aware scheduling algorithm for heterogeneous tasks of scientific workflow in clouds, *Future Generation Computer Systems (FGCS 2016)*, pages 140-152, 2016.

[邓水光, 2008] 邓水光，吴朝辉， Web服务组合方法综述，计算机科学，2008

[杜江淮, 2016] 杜江淮，SMT表面贴装技术工艺应用实践与趋势分析，电子技术与软件工程，2016(07):99-100。

[范凯, 2009] 范凯，Play! 一个颠覆JavaEE规范的web开发框架，*程序员*，2009。

[何焕春, 2017] 何焕春, 杨怿, 基于MVVM构架的Web前端框架研究, *电脑知识与技术*, 2017, 13(24):59-60。

[胡珀, 2011] 胡珀, 娄渊胜, 改进粒子群优化算法在服务组合中的应用,*计算机工程*，2011, 37(17):130-132。

[黄阳华, 2015] 黄阳华，德国“工业4.0”计划及其对我国产业创新的启示，*经济社会体制比较*，2015(02)：1-10

[李嘉, 2018] 李嘉, 赵凯强, 李长云, Web前端开发技术的演化与MVVM设计模式研究, *电脑知识与技术*, 2018, 14(02):221-223。

[缪学勤, 2014] 缪学勤， Industry4.0新工业革命与工业自动化转型升级，*石油化工自动化*，2014，50(1):1-5。

[孙卫琴，2005] 孙卫琴，*精通Hibernate：Java对象持久化技术详解*，电子工业出版社，2005。

[田珂, 2004] 田珂，谢世波，方马，数据持久层的解决方案，*计算机工程*，2004, 29(22):93-95。

[王华，2013] 王华，郭梅，从传统工厂到数字化、智能化工厂，*电子世界*，2013(20):205-206。

[王喜文，2013] 王喜文，工业4.0:智能工业，*物联网技术*，2013(12):3-6。

[温涛，2013] 温涛，盛国军，郭权，李迎秋，基于改进粒子群算法的Web服务组合，*计算机学报*，2013, 36(5):1031-1046。

[夏虹，2017] 夏虹，郭超等，面向工业的开放数据服务平台研究，技术报告，西安邮电大学大数据处理研究中心，2017。

# 致 谢

时光荏苒，我的硕士生涯已接进尾声.这段时光既漫长又短暂，其中充满了酸甜苦辣,更有收获和成长。感谢陪我一起度过美好时光的每位尊敬的老师和亲爱的同学，正是你们的帮助，我才能克服困难，正是你们的指导，我才能解决疑惑，直到学业的顺利完成.

本人的学位论文是在李传艺老师的殷切关怀和耐心指导下进行并完成的。衷心感谢恩师对我的淳淳教诲和悉心关怀。从课题的选择、项目的实施，直至论文的最终完成，李老师都始终给予我耐心的指导和支持，我取得的每一点成绩都凝聚着老师的汗水和心血，恩师开阔的视野、严谨的治学态度、精益求精的工作作风,深深地感染和激励着我，在此谨向李老师致以衷心的感谢和崇高的敬意。

感谢室友们与我一道分享她们青春的快乐！在此对平时开展相关工作中她们给予的支持和帮助一并表示感谢。感谢我的伙伴们付晔、黄靖、花霞等等。在论文写作过程中提供的热心帮助！他们不辞劳苦地为我提供无私的帮助，没有他们的帮助就没有这篇论文的顺利完成。

.感谢我的家人常年对我的支持和理解！他们是最爱我的人,也是我亏欠最多的人，他们默默的奉献是我坚持奋斗的不竭动力。同时还要感谢实习期间实习导师和同事们的关心与帮助，他们不仅在实习期间为我提供帮助、悉心指导我的工作，在我离职后也及时地为我提供帮助耐心解答一些问题。

最后,我要向百忙之中参与审阅、评议本论文各位老师、向参与本人论文答辩的各位老师表示由衷的感谢!人生的每个阶段都值得好好珍惜,这段美好岁月,因为有你们的关心和帮助，我很幸福。我会更加勤奋学习、认真研究。我会努力做得更好，我想这也是我能给你们的最好的回报。把最美好的祝福献给你们,愿永远健康、快乐!

# 版权及论文原创性说明

任何收存和保管本论文的单位和个人，未经作者本人授权，不得将本论文转借他人并复印、抄录、拍照或以任何方式传播，否则，引起有碍作者著作权益的问题，将可能承担法律责任。

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含其他个人或集体已经发表或撰写的作品成果。本文所引用的重要文献，均已在文中以明确方式标明。本声明的法律结果由本人承担。

作者签名：

日期： 年 月 日