大连理工大学

硕士学位论文

基于Layers架构模式的热轧计划编排系统实现

姓名：范圣冲

申请学位级别：硕士

专业：控制理论与控制工程

指导教师：王伟

20081201

摘　要

制造执行系统(MES)是位于上层计划管理系统与底层工业控制之间的面向车间生产的管理信息系统。在钢铁企业的 MES 系统中涉及到许多功能，生产计划编制是其中一个非常重要的环节。

现代钢铁企业的生产通常具有多品种，多规格，小批量大量订货合同的特点。因此，如何合理地编制轧制生产计划不但关系到合同能否及时兑现，而且关系到能否充分发挥设备能力及如何确保产品质量得到更好体现。根据现有的装备水平和组织方式进行生产计划的编制，发挥现有设备的能力，是一个非常关键的问题。轧制生产计划编制结果的优劣对中厚板轧制过程的生产成本有着巨大的影响。

本文以首钢热轧中厚板生产过程为实际应用背景，采用 Layers 软件架构模式将整个热轧生产调度系统划分为多个实现层次。在各层次间采用面向对象的程序设计思想进行程序设计，保证各层次间达到松散耦合的效果。在软件实现过程中使用成熟的.NET 程序设计技术，保证了系统具有很高的可维护性和可扩展性。在排产算法的实现过程中，采用基于规则的启发式方法，通过概率选择逐次对轧制调度方案进行寻优，保证了系统具有一个平均长度长，热装比高的排产结果。在 Layers 系统架构的基础上，本文设计了一个友好的人机交互界面，提供了排产数据的对比显示，以及排产结果的图形化拖动调整等功能，弥补了以往的工业应用软件人机交互功能粗糙，操作困难的缺点，进一步提高了系统的可操作性。

关键词：中厚板热轧计划；启发式算法；层次架构；.NET 架构

Design and Implementation of Hot-Rolling Scheduling System Based on Layers Structure Pattern

Abstract

Manufacturing Execution System (MES) is an information management system for the workshop production, which is located between the upper plan management system and the lower process control system. There are many functions in MES of iron and steel enterprises, and the production scheduling is one of the most significant parts.

The production process of modern steel enterprises usually has the features of multi-species, multi-standard, low-volume contract and a large number of orders. Therefore, how to prepare reasonably rolling production plan will not only affect the order delivery in time, but also has a big influence on fulfilling the production ability of the equipment and the products quality. The production planning according to the existing organization and equipment is a very key issue. In practice, the results of production planning in heavy plate rolling mill have a huge impact on the production costs.

Taking the hot rolling plate production process of Shougang as the background, this paper divides the whole system into several layers using Layers Architecture. The object-oriented programming idea is used to design all levels in the proposed system in order to ensure that all levels have been loosely coupled. And, the utilization of .NET technology ensures the high scalability and maintainability of this software system. In the implementation of the scheduling algorithm, this paper presents a heuristic-based approach to search the optimal solution through the probability-based selection. This method obtains the results of a long average length and a high hot charging. Furthermore, a friendly human-machine interface is implemented based on the layers architecture, which provides a comparison, drag adjustment and a graphical display to the scheduling results. This graphical interface overcomes the operating difficulties of the previous industrial application software and improves the system operability.

Key Words: Hot-rolled plate scheme; Heuristic Algorithm; Layers; .NET

大连理工大学学位论文独创性声明

作者郑重声明：所呈交的学位论文，是本人在导师的指导下进行研究工作所取得的成果。尽我所知，除文中已经注明引用内容和致谢的地方外，本论文不包含其他个人或集体已经发表的研究成果，也不包含其他已申请学位或其他用途使用过的成果。与我一同工作的同志对本研究所做的贡献均已在论文中做了明确的说明并表示了谢意。

若有不实之处，本人愿意承担相关法律责任。

学位论文题目：基于layers架构模式的热轧中厚板计划流程化实现

作者签名：[illegible]　　日期：2008 年 12 月 20 日

大连理工大学学位论文版权使用授权书

本人完全了解学校有关学位论文知识产权的规定，在校攻读学位期间论文工作的知识产权属于大连理工大学，允许论文被查阅和借阅。学校有权保留论文并向国家有关部门或机构送交论文的复印件和电子版，可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印、或扫描等复制手段保存和汇编本学位论文。

学位论文题目：基于Fagor数控系统的热补偿研究

作者签名：范圣帅　　日期：2008年12月20日

导师签名：王[illegible]　　日期：2009年1月12日

1 绪论

1.1 课题背景及研究意义

本文以首钢热轧中厚板生产过程为实际应用背景，采用基于规则的启发式方法通过概率选择逐次对轧制调度方案进行寻优，并结合面向对象的思想，开发了 Layers 软件架构模式下的热轧中厚板生产计划自动编制系统。本文的目的：通过对首钢热轧中厚板生产过程的深入分析，结合目前流行的先进软件设计思想，采用符合动态生产需求的生产计划编制技术，面向对象的编程语言与数据库相结合的设计方法，开发出具有实际应用价值的热轧计划编排系统。

1.1.1 研究背景

MES(Manufacturing Execution System)是美国管理界上世纪九十年代提出的新概念，MESA(MES 国际联合会)对 MES 的定义是：MES 能通过信息传递对从订单下达到产品完成的整个生产过程进行优化管理。当生产发生实时事件时，MES 能对此及时做出反应、报告，并利用当前的准备数据对它们进行指导和处理。这种对状态变化的迅速响应使 MES 能够减少企业内部没有附加值的活动，有效的指导生产运作过程，从而提高及时交货能力，改善物料的流通性能，提高生产回报率。MES 还通过双向的直接通讯，在企业内部和整个产品供应链中提供有关产品行为的关键任务信息。

生产计划与调度是 MES 的核心功能，在现代化工业生产中扮演着重要角色。它是针对某一任务，在一定的技术与资源的约束条件下，对现有的生产活动进行合理排序，确定某一时段所指派的任务，力求在设备可用的时间范围内，充分发挥其生产效率，缩短生产周期，降低生产成本。

对于钢铁企业而言，如果各工序自动化系统稳定可靠，产品质量可以保证，那么系统功能集成中遇到的最大难题便是如何面向多品种小批量的大量订货合同而组织生产，包括在不同钢种、规格和交货期需求下订单的组批；轧制计划、连铸计划与炼钢计划的各种平衡；如何将几个不同的工序视为一个整体，实现一体化管理，做到前后工序计划同步化，物流运行准时化；充分运用高温坯的潜热，取消或减少再加热过程，降低能耗，减少烧损，缩短生产周期，增加企业效益和市场竞争力。因此，建立完善的生产计划系统，对于提高钢铁生产的现代管理水平、降低企业产品成本、增强企业竞争力是至关重要的。

随着市场竞争的日益加剧，钢铁企业的用户对供应商的要求，已经在原来考虑价格和质量的基础上，又增加了对交货期方面的要求，并且份量越来越重。客户对交货期的

要求包括两方面：一方面是要求交货期尽可能短，另一方面要求交货准时。同时这两方面又是相互影响，随着交货期越来越短，准时交货变得越来越难。这种交货期对准时交货的影响主要表现在：随着交货期缩短，生产质量上的波动对准时交货的影响越来越显著；随着交货期缩短，资源的平衡从原来简单数量上的平衡到细节上(如工艺规程的要求)的平衡。

在此背景下，企业面临的问题是如何保证生产物流连续高效运作，使物流和信息流尽可能同步、有效的发挥热装热送和直轧工艺的作用，进而达到降低成本，提高产品质量，缩短产品生产周期的目的。此外，如何实现一体化管理，做到前后工序计划同步化，物流运行准时化，充分利用高温坯的潜热，取消或减少再加热过程，降低能耗，减少烧损，缩短生产周期，减少在制品库存，增加企业效益和市场竞争力，以及如何达到精确平衡物流，指导销售部门订货工作，为销售资源平衡系统和询单应答系统提供有效支撑，实现交货期的合理安排等作用，为最终全面实现按周交货提供保障是企业亟待解决的问题之一。

1.1.2 研究意义

钢铁产品的生产是一个相当复杂的过程，在整个生产过程中，钢铁企业的利润直接与生产产品的效率有关[1]。现代钢铁企业的生产大都是以销定产，企业首先要根据用户的订货合同和对市场需求的预测确定要生产的产品，并根据这些结果编制生产计划。在生产实施过程中，根据企业内外部的反馈信息不断地调整生产计划，并指导生产。这个编制计划——反馈——调整计划的过程贯穿于企业生产活动的全部过程中。

编制计划是企业管理的一项重要工作，计划编制的好坏直接影响到企业的资源是否得到合理配置，产品是否适应市场需求，企业能否获得最大利润。高质量的计划是管理现代化的需要，更是企业持续发展的前提。在当前的市场经济环境下，钢铁企业要提高企业的生产效益，增强企业的竞争力，就必须具有现代化的生产管理系统，必须做出高质量的生产计划和调度。

钢铁企业的生产过程兼有连续和断续两种生产过程，既有别于石油、化工的连续生产过程，也有别于机械行业的离散制造过程，是介于二者之间的混杂系统。由于炼钢、连铸、连轧这三大关键工序在生产过程中为顺序加工，不仅存在物流平衡和资源平衡问题，同时在高温作业环境下，也存在能量平衡和时间平衡问题。钢水根据计划提交连铸工序，最大限度的实现连铸，连铸高温坯的运送要与热轧计划有机结合，争取尽可能高的装炉温度和热装比。这就要求系列工序计划同步，物流运行准时，充分利用高温坯的

潜热，取消或减少再加热过程，降低能耗，减少烧损，缩短生产周期，减少在制品库存，增加企业效益和市场竞争力。

我国目前大部分钢铁企业采用的传统的手工调度管理方式无法很好的满足以上错综复杂的调度要求，更难达到系统优化的效果。随着计算机技术的普及应用和ERP(Enterprise Resource Planning)理念在企业中的展开，开发先进的生产计划自动编排系统具有很强的迫切性和现实意义。

1.2 热轧轧制计划问题的现状

1.2.1 热轧计划编制理论研究现状

关于对计划与调度问题进行研究的方法，最初是集中在整数规划、仿真和简单的规则上，这些方法多数难以解决复杂的问题。近年来，随着各种新的相关学科与优化技术的建立与发展，在生产调度方面也出现了许多新的优化方法，比如模拟退火法、遗传算法、禁忌搜索法等。生产调度问题的研究方法正在向多元化方向发展。生产调度问题的研究方法主要有如下几类[2]：

(l) 启发式方法

启发式方法针对调度问题的NP特性，并不企图在多项式时间内求得问题的最优解，而是在计算时间和调度效果之间进行折中，以较小的计算量来得到近似最优解或一个满意解。启发式方法通常称为调度规则[3]。

由于调度规则计算量小、效率高、实时性好，因而在生产调度研究中被广泛采用。

(2) 离散事件仿真技术

仿真方法是生产调度研究中最常用的方法。该方法通过对实际生产环境的建模来模拟实际生产环境，从而避开了对调度问题进行理论分析的困难。目前，仿真方法在生产调度研究中主要有以下两方面内容：

① 研究各种仿真参数对仿真结果的影响，以便在进行仿真实验时能做出恰当选择，从而使仿真所取得的结论更全面、更具说服力。Ramash[4]总结了大量的相关文献，对仿真时应考虑的参数及各参数的取值范围做了详细介绍。

② 将某些方法应用于某个仿真环境，通过仿真评价现有方法之间或新方法与现有方法之间的优劣，从而总结出各个方法的适用范围，或根据结论数据建立知识库或产生神经网络的训练样本。

唐立新[5-6]在分析轧钢厂生产作业计划的结构的基础上，建立了热轧精轧工序轧制批量调度的旅行商模型，这一模型能够适合与带钢、型钢和钢管的轧制生产批量调度问题。调度仿真方法的优点是它能够将在分析模型中不会考虑到的政策改变等因素的影响

映射到模型中去。另一个优点是帮助使用者有机会对做出的调度进行探查性的测试。仿真的实验本质是它的缺点。不过，仿真研究针对于具体实验设置而产生，所以对生产调度的理论研究贡献很少。运用仿真来产生生产调度很浪费，这不仅仅来源于产生调度所需要的计算时间浪费，而且来源于需要设计和运行仿真模型所作的人工脑力劳动浪费。仿真的准确程度取决于程序员的判断和技能，甚至高精确度的模型化也不能保证实验中一定能够发现优化调度或者好的调度。

(3) 专家系统

专家系统在生产调度研究中占有重要地位，目前已有一些较成熟的调度专家系统，例如 ISIS 和 OPIS 等。调度专家系统通常将领域知识和现场的各种约束表示成知识库，然后按照现场实际情况从知识库中产生调度方案，并能对意外情况采取相应的对策。

有效的领域模型和知识表示对于生产调度专家系统的设计十分重要。此外，约束在调度知识库中也占据重要地位，因为调度的好坏在很大程度上依赖于其对约束的满足程度。生产调度的决策参数具有很强的不确定性，为了有效地表示这种不确定性，许多学者选择了概率论，而应用模糊集理论则是一种更为有效的方法。在调度问题中应用模糊方法的优点在于，可为不精确约束的表示和应用提供丰富的表述语言和系统的框架，并且能对模糊目标进行评价[5]。

(4) 智能搜索算法

应用于调度问题的智能搜索方法包括模拟退火[8]、禁忌搜索[9]和遗传算法[10]等。目前在生产调度中使用最多的是遗传算法。遗传算法解决调度问题的优势在于它可以随机地从一个调度跳到另一个调度，从而可以解决其它方法易于使解陷入局部最优的问题。此外，它还具有计算速度快且易与其它算法相结合的优点，非常适合于解决生产调度问题。

应用遗传算法解决生产调度的文献较多，其中大多将遗传算法与其它方法结合使用。Lee 等[10]用遗传算法和机器学习来解决单件车间的调度问题，用机器学习来产生将工件下发到车间层的知识库，而用遗传算法在各台机器上分配工件。Jian 等[11]提出一种 MFS 的调度和重调度算法，该算法考虑了 4 种动态事件，即机器损坏、定单优先级提高、紧急定单下达和定单取消，用稳态遗传算法产生一个初始调度，当意外事件发生时，根据具体情况仅重新调度那些直接受影响的工序。JoneS 等[12]提出一种实时排序和调度算法，它集成了神经网络、实时仿真和遗传算法等方法，其中遗传算法主要用于对几个备选的调度规则进行优选。

(5) Mu1ti-agent 方法

Multi-Agent 通过在一系列分散的智能单元(Agent)间进行协调来解决问题。这些单元有各自的目标和自治的行为，并且可以有子单元。但是没有一个单元能够解决全局问题，因而它们之间必须进行协调。关于 Multi-agent 系统的结构，不同的人有不同的观点。例如 Kouiss 等[13]根据车间的物理布局来确定系统的结构，为每一个加工中心配备了一个 Agent，用于解决对应加工中心内的调度。另外还设计了一个全局 Agent，用于监视整个制造系统的状态，必要时为满足全局的需要可在各 Agent 间进行协调。Nof 等[14]提出了自适应/预测调度系统框架，将调度系统按功能划分成 5 个模块：调度器/重调度器、监视器、比较器、分辨器和调度恢复适应器。

关于每个 Agent 的结构，不同的人有不同的理解。但是每个 Agent 至少应有以下 3 个组成部分：

① 知识库，包含 Agent 执行其功能所必需的知识和数据；

② 控制功能，根据环境状态及与其它 Agent 间的相互作用，从知识库中提取知识来完成调度功能；

③ 通讯功能，用来与其它 Agent 和环境之间进行信息传递。

研究表明，Multi-agent 特别适用于解决复杂问题，尤其是那些经典方法无法解决的单元间有大量交互作用的问题。其优点是速度快、可靠性高、可扩展性强、能处理带有空间分布的问题、对不确定性数据和知识有较好的容错性；此外，由于是高度模块化系统，因而能澄清概念和简化设计。以上每一种方法或者策略都有自己的优点和缺点，在实际应用中应该具体问题具体分析，选择一种或几种最适合的使用。

1.2.2 热轧计划编制应用研究现状

近年来，国外许多钢铁企业如美国 LTV、日本的五大钢铁公司(新日铁、住友、川琦、日本钢管和神户钢铁)、韩国的浦项和鹿岛钢铁厂、德国的蒂森等，一直在致力于建立集成化的计算机生产管理系统[15-16]，其中包含核心模块——生产计划和调度，使生产管理合理化、在线化、集成化，能够对生产过程中的物质流、信息流和能源流进行综合管理，充分提高了大型设备的生产效率，减少了工序的等待时间，降低了物耗和能耗，从而降低了成本，提高了产品竞争力。

我国钢铁工业经过五十多年的发展，钢年产量位于世界前列，也逐步从资源密集、劳动力密集和追求大规模转向知识和技术密集、知识结构优化为特征的钢铁工业。目前，国内钢铁企业的生产管理远远落后于发达国家，尤其是管理水平需要进行系统变革，大部分的国内钢铁企业的生产计划和生产管理都是由人工来完成，不仅生产计划不精确，而且浪费人力，企业的自动化水平也受到很大限制。随着市场的发展，钢铁企业的生产

管理由订单组织生产，同时生产也多为多品种、小批量的生产，为了提高企业的竞争力，许多国内大型钢铁企致力于发展计算机集成制造系统(CIMS)。国内的一部分钢铁企业如首钢等在信息化和 CIMS、ERP 的建设上已经取得了初步成效。

总观生产计划与调度的发展与研究，在现代连续过程行业，生产计划和调度系统主要由生产计划模块和静态调度、动态调度、动态监控和统计报告功能模块组成[17]，近年来生产调度的发展逐步趋向借助计算机技术的帮助，利用面向对象编程方法，充分利用计算机的图形功能，建立生产系统的仿真系统。在仿真系统的基础上，对生产系统的生产状况进行预测规划，从而进行合理的生产调度。

1.3 本文工作

本文采用基于启发式的方法通过概率选择逐次对轧制调度方案进行寻优给出了一个基于.NET 平台的自动排产系统的实现。该系统采用面向对象的思想进行设计，采用 Layers 架构分层思想将系统分割成 4 层。每一层与它下面的各层保持松散耦合，任何一层的变化都可以很好地局限于这一层，从而保证了系统的可维护性，可扩展性以及系统的健壮性。系统业务逻辑层采用基于贪心思想的启发式算法进行计划的编制，保证了排产结果有足够的热装比和平均计划长度。在得到良好排产结果的同时，系统表示层设计了友好的人机交互界面，增强了系统的实用性与使用的灵活性。

本文各章的具体内容如下：

第 1 章是绪论，首先简单介绍了本文的选题背景，接着阐述了本文的研究与应用意义。然后概述了热轧计划编制的理论研究现状和应用研究现状。

第 2 章介绍了本文开发系统的系统架构和开发平台。首先介绍了架构和架构模式的相关概念，以及 Layers 架构模式的定义和特点。然后介绍了系统的开发平台，包括.NET Framework 的概述，公共语言，统一编程类库，及数据访问技术等。

第 3 章阐述了轧制调度系统的系统设计。分别介绍了系统的总体结构，系统各个层次的具体设计，以及系统用到的一些关键算法的设计等。

第 4 章是系统的软件实现。此章介绍了在系统设计中规划的各个层次的具体实现，并展示了系统的最终运行效果。

2 系统架构与开发平台

2.1 架构模式

架构和模式是在软件工程领域经常接触到的两个概念。现在的软件系统的规模之大以及它的复杂性是十几年前我们所无法想象的，软件生产过程中软件工程也就扮演了更加重要的角色，可以说是具有了举足轻重的作用；相应的，架构和模式在软件的设计和开发的过程中它们的作用也越来越重要，不可缺少。架构和模式这两个概念也就在软件生产过程中成为了相当时髦的词汇。

通常软件体系结构被称为架构，指可以预制和可重构的软件框架构[18]。架构在软件工程领域仍然处在发展期，对于它的定义，还未形成一个统一的认识。

软件体系结构是软件设计过程之中，超越算法设计和数据结构设计的一个层次。体系结构包括系统各个方面的组织和全局控制结构，通信协议、同步、数据存储，给设计元素分配特定功能，设计元素的组织，规模和性能，在各个设计的方案之间进行合理的选择并纳入了系统化的思维[19]。软件体系结构可看作：软件体系结构={构件(component)，连接件(connector)，约束(constraint)}。其中构件可以是一组代码，如程序的模块；也可以是一个独立的程序，如数据库服务器；连接件可以是过程调用、管道、远程过程调用等，用于表示构件之间的相互作用；约束一般为对象连接时的规则，或指明构件连接的信息和条件。

关于架构的定义还有很多其他观点，虽然各种定义关键架构的角度不同，研究对象也略有侧重，但它们核心的内容都是软件系统的结构，这些定义大部分是从构造的角度来审视软件体系结构。

模式是一种以固定格式得以保存和共享的文档，这些文档是一些经验总结后的记载，而这些经验是软件设计师和软件工程师在软件开发过程中解决某些重复出现的问题后所总结出来的、被证明成熟的经验。简单的说，模式就是问题—解决方案对，是对某种特定的场景下某个不断重复出现的问题的解决方案。

模式本身并没有任何的创新性，它仅仅是对于一些已经被证明为优秀的解决方法的归类、总结，并将这些解决问题的成熟方案记录下来，目的是为了重用该解决方案而又不用做重复的探索和试验，同时也用来指导和促进更好的设计实践。描述模式的格式有多种，但一般至少包括3个部分：语境、问题、解决方案，从而可以对特定环境中问题的解决方案作恰当的描述。

2.1.1 架构和模式的关系

架构和模式其实是一个相互涵盖的过程，但是总的来说，架构更加关注的是所谓的高层的设计(High Level Design)，而模式关注的重点在于通过经验提取的“准则或指导方案”在设计中的应用，那么在不同的层次上思考问题的时候就产生了相应的模式(Pattern)。模式的目标是，把共通问题中的不变部分和变化部分分离出来，不变的部分就构成了模式。例如，我们针对需要在代码中减少以大量的 if-else，或者 switch-case 语句来产生相关类所对应的对象以利于代码的修改和重用这一问题时，将变化的部分即产生不同的类的对象分离出来，提炼和抽象出产生不同类的对象的方法，那么我们就会形成在设计模式中称为创建型模式的模式来，例如 Factory pattern 和 Factory Method Pattern。再比如，我们为了提高代码的重用性，在各个类的组成结构中进行相应的设计，并且将这样的设计进行提炼和抽象，将会形成设计模式中的第二种类型的模式即结构型模式(Structural Patterns)。因此，模式是一个经验提取的“准则”，并且在一次一次的实践中得到了验证。总的来说：架构是 Height-Level Design，着眼于不同业务中共性的解决方案，而模式是侧重通用原理。

2.1.2 Layers 架构模式

层(Layers)体系架构模式就是把应用系统分解成子任务组，其中每个子任务组处于一个特定的抽象层次上。层架构模式组织成一个层次结构，每一层为上层服务(Service Provider)，同时也作为下层的客户端。在一些层次系统中，除了包含一些输出函数外，内部的层只对相邻的层可见。这样的系统中构件在一些层实现了虚拟机(在另一些层次系统中层是部分不透明的)机制。层的调用通过决定层间如何交互的协议来定义。这种风格支持基于可增加抽象层的设计。这样，允许将一个复杂问题分解成一个层堆栈的实现。由于每一层最多只影响两层，同时只要给相邻层提供接口，允许每层用不同的方法实现，因此为软件重用提供了强大的支持。

层结构是最成熟的软件体系架构模式，它起源于早期的系统设计，由开始的函数调用，作为函数库，供其他程序进行调用。一般在系统设计时，由一系列高层模块和底层模块处理构成，并且高层的模块依赖于底层。因此为了完成系统的设计必须要考虑以下因素：

(1) 源码的修改会影响整个系统，应该被限定在一个部件内部而不影响其他模块；

(2) 接口应当稳定，甚至要被规范化；

(3) 系统的架构应该灵活，可以更换；

(4) 系统的开发需要被划分为多个部分，比如团队开发或者异地开发。

从系统高层的观点来看，设计比较简单，它把系统分为几个层次并且把它们叠加起来，最下面的抽象层称为第 1 层，它是系统基础。依次类推，把 N 层放在第 N-1 层上。其结构如图 2.1：

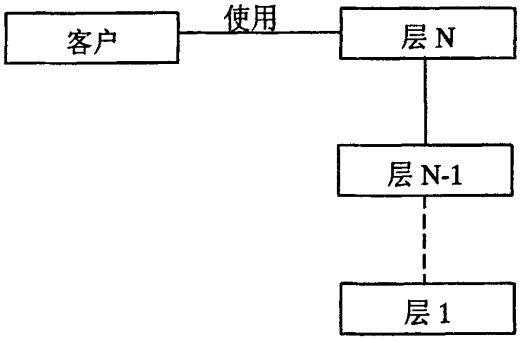

图 2.1 软件分层体系架构

Fig. 2.1 Software layers architecture

层架构模式是最常用的一种软件体系架构模式，从它的实现和结构图中，可以看出有如下优点：

(1) 层次的复用性，如果每个层次有很好的抽象接口，那么它可以被其他环境复用；

(2) 支持基于抽象程度递增的系统设计，使设计者可以把一个复杂系统按递增的步骤进行分解，使系统更容易模块化；

(3) 支持功能增强，因为每一层至多和相邻的上下层交互，因此功能的改变最多影响相邻的上下层；

(4) 可替换性，因为独立的层次设计很容易被功能相同的模块替换。

但是在实际的项目中，该模式也有相应的不足，缺点有：

(1) 低效率，分层结构通常要比单层结构的效率低。因为有时高层过分依赖底层的服务，因此必须穿过许多中间层进行数据的传送，甚至多次。

(2) 改变行为的连锁反映。

2.2 系统开发平台

一个好的开发平台对于系统开发来说是非常重要的。在微软公司发布的.NET Framework 技术体系中，引进了多项最新的技术，使得用微软的工具开发出来的软件在

便捷性、公共性、和其他技术的融合性方面得到了很大的提高，.NET Framework 技术无疑是企业级软件的开发利器。

2.2.1 .NET Framework 概述

.NET Framework 是一种新的计算平台，它简化了在高度分布式 Internet 环境中的应用程序开发。.NET Framework 主要有以下特点：

(1) 提供一个一致的面向对象的编程环境，而无论对象代码是在本地存储和执行，还是在本地执行但在 Internet 上发布，或者是在远程执行的。

(2) 软件越来越基于 XML，软件模块之间的交互通过 XML 语言完成，清除了不同软件之间的壁垒。

(3) 提供一个保证代码安全执行的代码执行环境。

(4) 提供一个可消除脚本环境或解释环境的性能问题的代码执行环境。

(5) 融合多种设备和平台，任何设备和平台，只要支持.NET Framework 就可以运行.NET 编写的应用软件。

(6) 按照工业标准生成所有通信，以确保基于.NET Framework 的代码可与任何其他代码集成[20-21]。

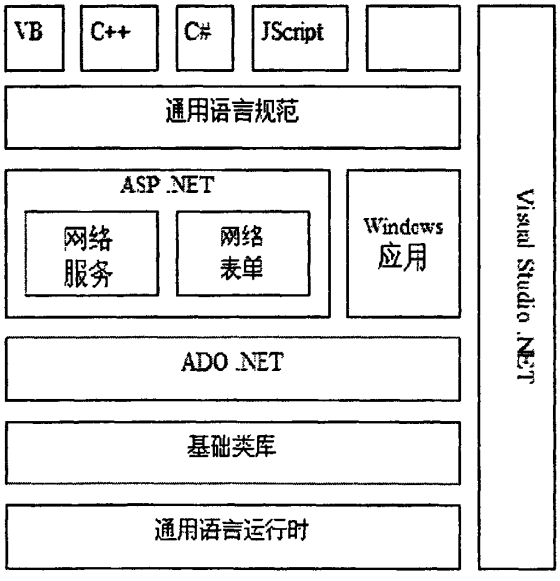

图 2.2 .NET 开发平台示意图

Fig. 2.2 Sketch map of .NET development platform

由图 2.2 可以看出，.NET Framework 主要有三个部分：公共语言运行库，.NET Framework 类库和高级版“活动服务器页面”(又名 ASP.NET)。

2.2.2 公共语言(CLR)

公共语言，又称为中间语言，是.NET Framework 的基础。无论在.NET Framework 中使用任何语言进行编写代码，程序经过编译之后，都会变成中间语言，因此不同语言之间的差别变得极其微小。公共语言运行库负责程序运行时的服务，比如语言集成、安全增强以及内存、子进程和线程的管理。众所周知，诸如组件的有效期管理、跨语言异常处理、组件的动态绑定等功能都是与应用软件密切相关的，但这些都是开发中比较困难的任务，以前往往都是要由有经验人员编写的。现在它们可交给运行库处理，最终的开发人员只要针对特殊的客户，编写代码以实现客户业务逻辑即可，这种方法大大地提高了应用软件的开发速度。

在.NET Framework 体系中，应用程序是由一个或多个可执行程序组成，每个可执行程序中都有元数据和托管代码。此种应用程序现在通常被称为程序集，一个程序集由一个或多个部署在一起的文件组成，程序集中保存了一份清单，该清单确定程序集标识，指定组成程序集实现的文件，指定组成程序集的类型和资源，并列举此程序集对其他程序集的编译依赖项，并指定为保证程序集正确运行所需要的权限集。在运行时，.NET Framework 使用此信息来解析引用，强制地实施版本绑定的策略，并验证已加载的程序集的完整性[22-23]。

2.2.3 统一的编程类库

统一的编程类库为开发人员提供了一个统一的、面向对象的、层次化的、可扩展的类库集。在.NET 框架下，这些类包括的内容从操作系统的底层操作到视窗界面的显示，从数据库开发到 Web 服务等，涉及面非常广，这些类都是建立在公用语言上，而且是被管理的、安全的代码，开发者能够放心地使用。此外，不同语言的开发人员面对的是同一套类库集，他们不再需要学习新的框架技术就能顺利编程。由于面对同一个公共跨编程语言 API 集，.NET 框架可实现跨语言继承性、错误处理功能和调试功能。从 Jscript 到 C++到 Visual Basic 到 C#的所有编程语言，都是相互等同的，开发人员可以自由选择理想的编程语言。在使用时，开发者只需在自己的应用中添加所需的基础类库的引用，然后就可以使用这个类库中的所有方法、属性等等。跟传统的 Windows 编程相比，使用和扩展基础类库都非常容易，这使得开发者能够高效快速的构建基于下一代互联网的网络应用[24-26]。

2.2.4 开发语言的选择

Microsoft.NET 开发框架支持多种语言，在目前的版本中已经支持 VB，C++，C# 和 Jscript 四种语言以及他们之间的深层次交互。而且微软支持第三方生产针对 Microsoft.NET 的编译器和开发工具，这也就是说几乎所有市场上的编程语言都有可能应用于 Microsoft.NET 开发框架。这样开发者可以任意选择自己喜爱的语言，这种开放和交互的特性正是开发者所热爱的。本系统选择 C#(C Sharp)作为开发语言。

C#是由 Microsoft 开发的一种新型编程语言，是一种类型安全的、现代的、简单的、并由 C 和 C++衍生出来的面向对象的编程语言，它牢牢根植于 C 和 C++语言之上，并可立即被的使用者所熟悉。C#的目的就是综合 Visual Basic 的高生产率和 C++的行动力。C#是微软.NET 战略的关键性语言，它是整个.NET 平台的基础。

2.2.5 数据库访问技术 ADO.NET

ADO.NET 是一种新的数据访问策略，它不只是 ADO 的改进版本。在许多方面它都采用新的思维方式，它涉及到我们通常使用“与数据源断开连接的”数据的领域。这种思想就是，在高度互联的世界中，应用程序中的数据可以有多个来源以及多种格式，当您取出数据后，可以在本地直接使用这些数据，而不需要保持到数据存储的连接。ADO.NET 提供对 XML 的内在支持，可以使数据传输通过防火墙的过程更容易[27-29]。

ADO.NET 提供了两种访问数据的基本方法：DataReader 和 DataSet。下面主要介绍这两个对象：

(1) DataReader

DataReader 在连接模式下运行，检索只向前的、只读的、且位于数据存储中的数据副本。它涉及用于快速数据访问，在这个过程中，你连接数据库并检索信息，然后关闭数据库连接。DataReader 不支持回滚、前滚或修改操作。它与 ADO 只向前的 RecordSet 最接近。DataReader 被推荐使用于下面的两种情况：

① 客户界面利用手写代码，或是没有用到数据绑定且数据的更新是利用手写的 SQL 语句或是存储过程，这种情况下，DataReader 将提供有效的访问数据的方法；

② 需要查看数据库状态，但不需要实现诸如自动更新的功能。

(2) DataSet

DataSet 是 ADO.NET 的核心，是数据在内存中的副本。与 ADO RecordSet 不同的是，它有多个表，每个表都来自不同的数据存储。可以通过关系将 DataSet 中表彼此相连。简而言之，DataSet 只是内存数据库，它没有与填充它的基本数据存储相连。DataSet 的工作原理如图 2.3 所示：

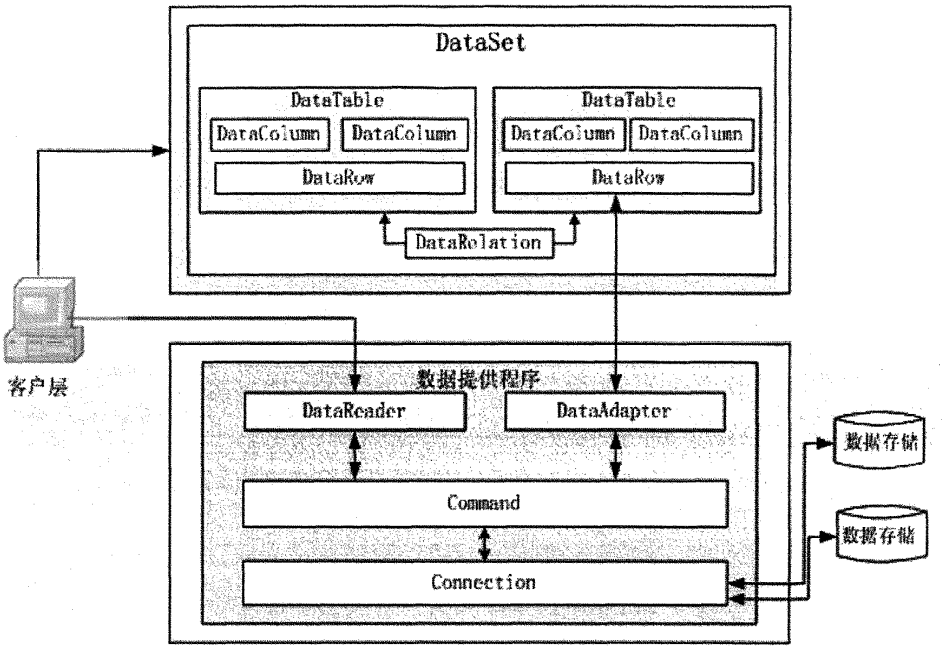

图 2.3 DataSet 工作原理

Fig. 2.3 Work principle of DataSet

如图 2.3 所示，DataSet 可以包含一个或多个 DataTable，利用数据存储的信息对它们进行填充。DataSet 中的每个 DataTable 都有一个或多个 DataColumn，它定义保存的数据列。它也有一个或多个 DataRow，它们是每个存储的记录的实际值。使用 DataRelation 描述 DataSet 中的表之间的关系。

在数据库应用程序的.NET 环境中，有数据提供程序(Data Provider)和数据使用程序(Data Consumer)。数据提供程序将连接到数据库执行命令并返回执行结果。在 ADO.NET 中，数据提供程序被称为托管提供程序(managed provider)，这只是因为它们是由 ADO.NET 托管的。数据使用程序就是那些使用数据提供程序用于操纵或检索数据服务的应用程序[31]。

图中客户可以通过两条不同的路径从数据存储访问数据：DataSet 或 DataReader。使用 DataSet 时，打开数据库连接并且 DataAdapter 通过 Connection 发送 Commands 从基本数据存储检索结果。然后，DataAdapter 使用值填充 DataSet 并返回到客户。ADO.NET 在后台使用 DataReader 将数据装入 DataSet。

如果使用 DataReader，打开数据库连接，DataReader 通过 Connection 发送 Commands 检索结果中的只向前的数据流。在 DataSet 和 DataReader 这两条路径中，数据提供程序促进与数据存储的通信和检索[32]。

3 轧制调度系统设计

本系统的设计主要包括两个方面。一方面是系统程序架构的设计，另一方面是系统的启发式排产算法设计。其中系统架构设计是系统的基础，实现高效优秀的排产算法是系统设计目的。启发式算法在此优化架构的基础上进行设计，从而可以保证算法有高的执行效率以及优秀的可扩展性和可维护性。

3.1 结构设计

3.1.1 系统总体设计

系统总体设计是系统设计的一个主要阶段，在此阶段需要确定整个系统的整体结构，以及实现系统功能的划分。本系统将应用在首钢热轧厂的实际生产当中，考虑到机组的生产改造等现场条件的变化，在系统设计阶段必须要保证系统具有一个好多架构体系。这样，才能很好的适应现场条件的变化，使系统具有更好的适应性和移植性。现场条件的变化可能发生在很多方面，如应用数据库类型的改变，排产规则的改变，约束条件的改变，用户接口的改变等多方面。为了适应可能的变动，采用多层程序架构思想，将本系统纵向划分为 4 个层次，分别为数据库层，数据存取层，业务逻辑层，表示层。

在每一个层次内根据面向对象的思想设计横向的类实现，本层的功能只在本层次内实现，仅留有很少的外部调用点，达到内部信息的透明化，以使各层次的功能表现出很好的封装性。各层间采用接口调用，保证各层次的高度可移植性。各层之间保持最少的信息交流，保证各层间具有松散的耦合性，以加强系统的扩展性和可维护性。各层的业务调用关系如图 3.1。

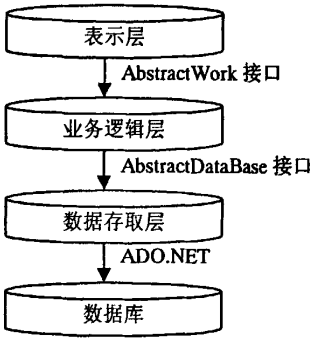

图 3.1 总体结构图

Fig. 3.1 The overall chart

系统运行在 C/S 架构模式下，系统表示层运行在客户端，业务逻辑层，数据存取层，数据库层运行在服务器端。

系统各层次只能调用其下面的层次，各层次也只能由其上层调用，不可以隔层调用。当某层次发生异常时，向其上层报告异常情况，逐层传递，直到最后由表示层报告给用户。系统中表示层实现显示逻辑，包括数据显示，结果显示，结果手动调整等功能。业务逻辑层是系统的核心功能层次，实现对待排产数据排产，形成排产结果。数据存取层实现数据库操作的抽象和屏蔽，使上层的数据库操作透明化，即无需考虑数据库类型以及数据的存储形式，只需直接使用接口调用相关函数即可获得所需数据。

系统顶部 3 层结构分别用 VS2005 的 3 个项目实现，分别编译成 3 个装配件，当系统某一层次变动只需改动相应项目即可，只要保证各层次间接口不变，其他代码无需改动。为提高系统的可移植性，在数据库层系统不实现任何的存储过程，将所有的业务逻辑都集中在系统的业务逻辑层。这样可以保证当系统的应用数据库类型发生变化时，只需修改其数据存取层部分代码即可，无需改动业务逻辑层，极大了提高了系统的可移植性。

系统具体各层设计如下文介绍。

3.1.2 数据存取层设计

数据存取层采用抽象工厂的设计模式来实现。抽象工厂模式向客户端提供一个接口，使得客户端在不必指定具体类型的情况下，创建多个产品族中的对象。抽象工厂模式面对的问题是多个产品等级结构的系统设计。抽象工厂模式是所有形态的工厂模式中最为抽象和最具一般性的一种形态。抽象工厂模式可以向客户端提供一个接口，使得客户端在不必指定具体产品类型的情况下，创建多个产品族中的对象。

抽象工厂模式设计到以下的角色：

抽象工厂角色：担任这个角色的是工厂方法模式的核心，它是与应用系统的商业逻辑无关的。通常使用接口或抽象类实现。在本文中设计一个抽象类 DataBase 来实现抽象工厂的角色。

具体工厂角色：这个角色直接在客户端的调用下创建产品的实例。这个角色含有选择合适的产品对象的逻辑，而这个逻辑是与应用系统的商业逻辑紧密相关的。通常使用具体的类实现。在本文中，具体工厂为由 DataBase 类派生的数据库访问类，包括 SqlServerDataBase，OracleDataBase，OldDbDataBase 等。

抽象产品角色：担任这个角色的类是抽象工厂方法模式所创建的对象的父类，或它们共同拥有的接口。通常使用接口或抽象类实现这一角色。本文中数据存取层的产品为

为 ADO.NET 的各种数据访问对象，包括 DataReader，DataCommand，DataSet 等。本文中的抽象产品为 IDataReader，IDataCommand 等接口。

具体产品角色：抽象工厂模式所创建的任何产品对象都是某一具体产品类的实例。这是客户端最终需要的东西。通常使用具体类实现这个角色。在本文中具体产品为具体的数据库访问类，包括 SqlDataReader，OracleDataReade，SqlDataCommand, OracleDataCommand 等。

在本系统中，业务逻辑层为数据存取层的客户。数据存取层提供的接口为抽象的 DataBase 类。客户在需要调用数据库功能时，直接使用 DataBase 类即可。DataBase 类返回抽象的产品类型，此返回结果用接口来表示。业务逻辑层在使用数据存取层的功能时，使用接口调用，这样可以很好的实现业务逻辑层和数据存取层之间的松散耦合。

所有数据库操作的逻辑都在数据存取层的内部实现。当数据库发生变化时，改动配置文件，实例化相应的数据库类即可[33-34]。数据存取层类设计如图 3.2。

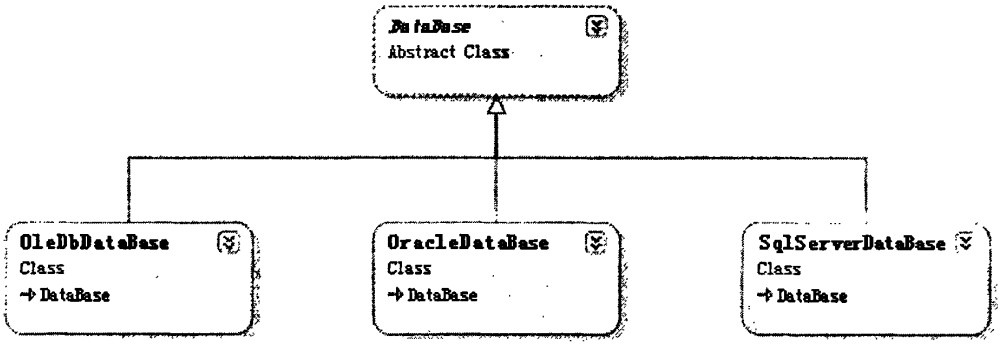

图 3.2　数据存取层类图

Fig. 3.2　Data access layer class diagram

如图 3.2 所示，系统数据存取层设计了一个抽象基类，用来定义数据库类的整体结构以及数据库类的访问接口，并实现一些各子类都需要使用的公共功能。由此抽象类共派生出 3 个派生类，分别用来访问相应类型的数据库。例如 OracleDataBase 用来访问 Oracle 数据库，SqlServerDataBase 用来访问 SQL Server 数据库。在使用数据库类的时候，统一使用 DataBase 类作为访问接口，具体的实例化工作在 DataBase 类的静态构造函数中实现。在此静态函数中读取配置文件，判断具体需要使用的数据库类型而实例化相应的类，从而实现在数据库类型发生变化时，可以很少改动甚至不改动代码，达到很高的可扩展性和可迁移性。

3.1.3 业务逻辑层设计

系统的业务逻辑层是整个系统最主要的一个层次，此层次涉及到具体排产算法的实现。计划排产是系统的一个主要功能，业务逻辑层实现的排产效果直接影响到系统的实用性，业务逻辑层是系统的核心层次。

本系统的业务逻辑层设计采用面向对象的思想，将现实世界中的实际物件抽象成类的概念。将类相关的属性和功能封装在类的内部，当某些功能和数据仅和类本身有关时，就降低其可见性，使其仅对此类的代码可见，对外透明。从而达到面向对象的类封装的效果。类与类之间仅存在很少的必要的联系，达到类与类之间的松散耦合。在类的设计中充分使用面向对象的多态技术的优点，利用运行时的一个动态绑定，达到不同计划编制方法的统一调用的效果。在本层次的类的设计中，很好的利用的面向对象思想的类的继承的技术，将一些子类共有的功能放于父类中实现，从而大大的减少了系统的代码量，而其当有改动时，仅需改动父类中的相应代码即可，充分的保证了程序的实现效率，也保证了程序具有一个很好的可维护性。

根据此设计思想，在系统的业务逻辑层共设计 3 个层次的类，分别为 Work（批次），Plan（轧制计划），Slab（板坯）。其中 Work 表示一次计划编排的结果，Plan 表示一个轧制单元，Slab 表示一块板坯。每个 Work 包含一个或多个 Plan 以及一部分没有排入计划的板坯，每个 Plan 包括一个或多个 Slab。业务逻辑层的运算结果即为一个满足所有约束的 Work。

考虑轧制单与排产批次可能有不同的种类，以及程序将会有一个很好的可扩展性，将上述 3 个层次的类全部声明为一个抽象类。具体的实现由其派生类来完成。类的继承关系如图 3.3，图 3.4。

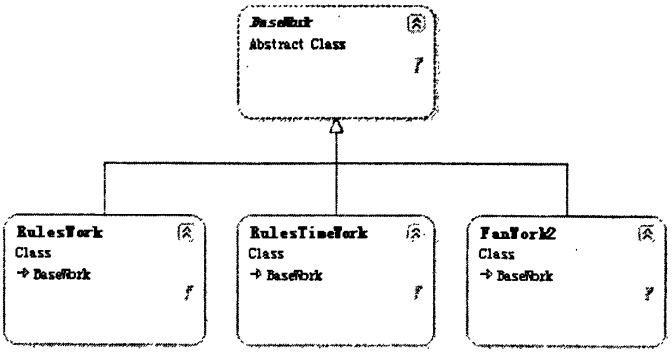

图 3.3 Work 类继承关系图

Fig. 3.3 Work Class inheritance diagram

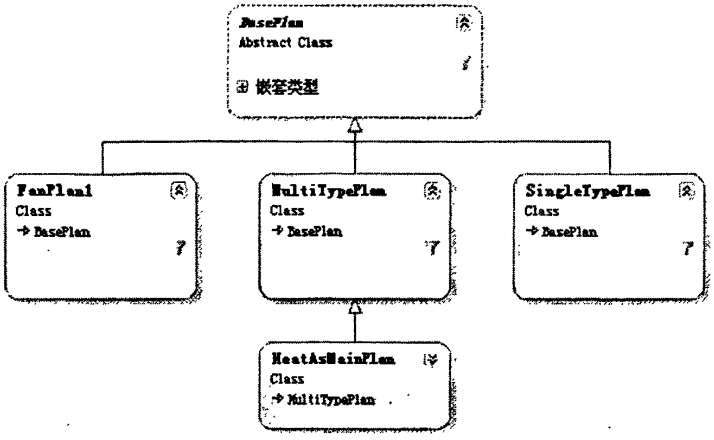

图 3.4 Plan 类继承关系图

Fig. 3.4 Plan Class inheritance diagram

根据每个类的实际含义，组织如图 3.5 的聚合关系。

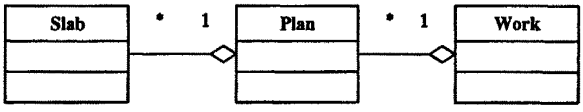

图 3.5 聚合关系图

Fig. 3.5 Polymerization diagram

根据实际生产的特点和现场工艺的要求，程序在进行优化排产时，分为两个阶段来进行。首先在 Work 层次内对排产数据进行初步优化，形成一个比较有利于下一步具体排产的初步的结果。Work 将此结果传入到 Plan 层次内，然后在 Plan 层次内运行排产算法，完成具体计划的编制。通过两次优化，可以减少程序设计的复杂度，达到将复杂的问题分布求解，可以使每一步的算法的实现都比较简单，也可以使排产算法的设计更加灵活，算法的调整和改进也更加容易。采用两次分阶段求解的方法后，程序的排产结果可以很容易的达到一个一个较理想的次优解，同时也增加了系统应对约束条件变化的灵活性。系统的可扩展，可维护性得到了近一步的加强。

图 3.6 描述了本系统的各个单元的主要的运行方式和调用关系。算法首先由 Work 类对数据进行一个初步的处理，产生一个初始的处理结果。在此结果的基础之上，采用基于规则的启发式算法不停的尝试建立 Plan 类。当剩余板坯已无法生成 Plan 类时，再

向每一个计划添加烫辊材。最后，尝试将没有排入计划的板坯插入到现有的计划当中以加大计划的平均长度。

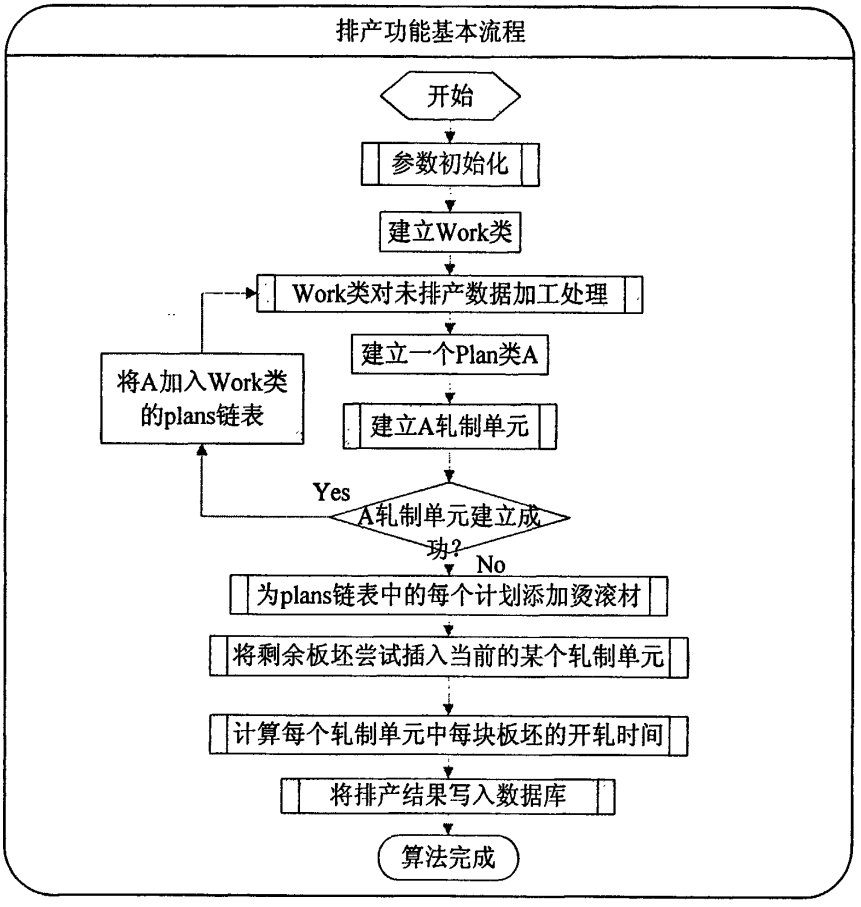

图 3.6　排产功能基本流程图

Fig. 3.6　The interface of planning function

3.1.4　表示层设计

本系统表示层的设计采用控件模块化的思想来实现。系统的表示层采用.NET 的 Windows 程序实现。将系统表示层以主要功能为单位，封装成 Windows 控件，控件的设计本着高内聚，低耦合的思想，保证了控件对其他控件的可见功能只是某些必须的接口功能。控件功能的具体的实现逻辑隐藏在控件的内部，对其他控件和模块来说是不可见的。在成功的实现了各控件的功能以及控件之间的接口之后，将各功能控件组合起来即形成了本系统的表示层。

采用模块化思想加强了系统的可维护性和可扩展性，当系统功能变化，仅需修改相应模块即可。同时，采用控件化的设计思想也极大的提高了系统的可移植性，当表示层结构有巨大的调整时，甚至需要重新开发一个用户表示层时，可以直接将编译好的控件转移到另一个模块中，只要保证模块的入口参数符合规则，即可以在其他某块中顺利的应用。系统表示层的各功能控件详细设计如下：

(1) 批次树控件：该控件继承自.NET 的 TreeView 控件，用来显示当前池中所有的已排批次，未排批次，下发批次。包括每一个批次的具体数据分布和详细信息，如共有多少个编制者对之进行编制，每个编制者的编制结果，热装比，剩余板坯数等。

生成树的算法如图 3.7 所示：

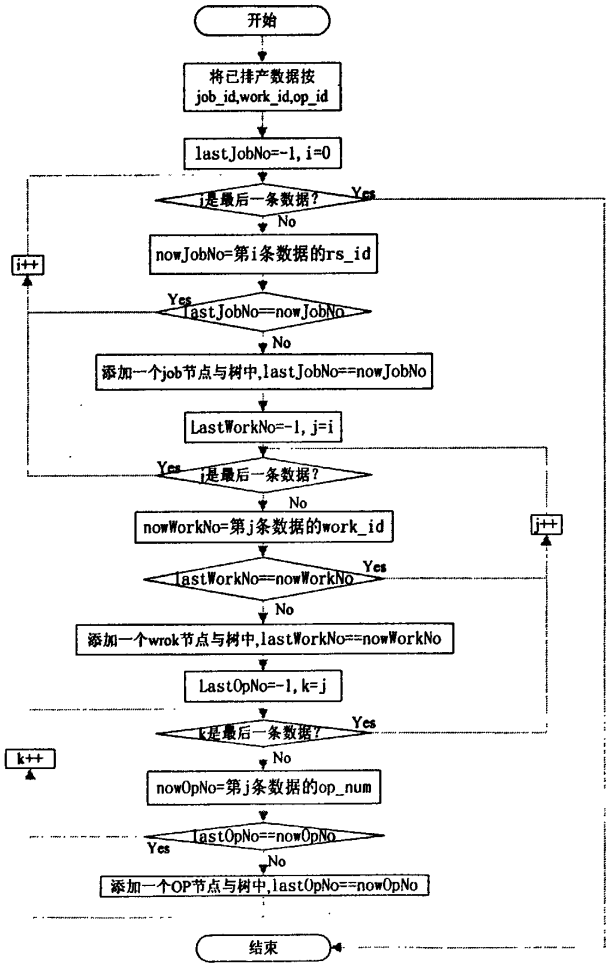

图 3.7　生成树算法

Fig. 3.7　Method to build tree

(2) 建池控件：池是系统中的一个逻辑概念，表示的是待排产的一批数据。在系统原料表中有很多待排产的数据，这些数据有的表示已经由连铸机组下线存储在板坯库中的板坯，有的表示目前还没有由连铸机组下线的板坯。原料表中所有的板坯都可以作为编制计划的原始数据。在计划编制的时候要考虑很多有限条件，比如交货期较早的板坯需要有限生产，有特殊要求的板坯要优先生产。建池控件的主要功能就是根据相关优先条件计算出原料表中所有板坯的优先级，并根据此优先级选择出一定数量的板坯作为算法一次排产的数据。

在计算优先级时，此控件可以允许用户输入一定的条件，控件根据此条件筛选出符合条件的板坯，再对筛选出的板坯计算优先级，最后选择出需要数量的板坯作为收池结果。在建池过程中，此控件提供两种计算优先级的方法，供用户选择。用户可以选择热装比优先的计算方法或是考虑综合优先级的计算方法。在得到建池结果后，控件还提供给用户修改建池结果的功能。

(3) 待排产数据显示控件：显示待排产数据的详细信息，包括板坯号，板坯宽度，板坯厚度切断时间等重要的板坯参数。

(4) 已排产数据显示控件：显示某一批次形成的多次排产结果，包括每次排产结果的数据批次号，编制者，热装比，优化次数，平均每个计划板坯数等。此控件显示所有排产结果的统计信息。该控件主要用来比较相同批次不同排产结果的比较。通过比较，用户可以选择一个好的排产结果，也可以分析出结果不好的原因，有助于下一次得到更好的排产结果。

(5) 排产结果详细信息控件：显示某次排产结果的详细信息。包括形成的每一个计划的总重量，总数量，总长度等。以及每一个计划的详细板坯信息，每一个计划的图形显示。此控件提供比“已排产数据显示控件”更详细的显示功能。此控件显示每一次排产结果的详细信息，包括形成的每一个计划的统计信息以及计划中的每一个钢卷的信息。此外，此控件还提供排产结果的图形显示功能，以图形化的方式显示形成计划的宽度，厚度等主要指标，还可以对计划中违规的板坯提供特殊的标示。

(6) 排产结果手动调整控件：提供对排产结果的图形化显示，以及对排产结果的手动调整。包括计划内的调整，计划间的调整，以及计划的动态增减。在进行计划调整时还可以同时选择是否现实未排产板坯。

此手动调整功能是系统表示层的最主要功能之一。以往的工业软件产品的用户接口往往都做得十分粗糙。以计划编制程序为例，往往仅提供一组表格数据，用户需要读取

表格分析数据才可以知道排产的结果，所有的调整也是针对表格进行。这种较为陈旧的设计方式要求用户要具有很多的经验，即使如此也需要花费很多时间和精力来分析表格数据，特别是对违规的判断，更是很难计算。

本文设计的排产结果手动调整控件完全实现了排产结果的图形化处理。用户可以直观的查看排产的图形化显示结果，对违规也给与图形化的显示，用户的使用非常方便。

3.1.5 数据库设计

本系统的开发采用了 Oracle 10G 关系型数据库管理系统。Oracle 完全支持所有的工业标准。采用完全开放策略；可以使客户选择最适合的解决方案对；开发商全力支持；适于海量数据；能在所有主流平台上运行并且与 Internet 高度集成，所以 Oracle 10G 是开发 MES 的一个极佳的选择。

数据库是数据库应用程序的核心。数据库设计，或称数据模型，是建立一个应用程序很重要的一步。一个好的数据库结构和文件设计可以使系统在已有的条件下，具有处理速度快、占用存储空间少、操作处理过程简单、查找容易、系统开销和费用低等特点。数据库设计一般经过需求分析与数据分析、概念设计、逻辑设计和物理设计 4 个步骤。

概念设计是指在数据分析的基础上，自底向上地建立整个系统的数据库概念结构，即先从用户的角度进行视图集成，最后对集成后的结构分析优化得到最终结果。

逻辑设计的任务是把概念结构转换为相应的逻辑结构。本系统选择的是关系型数据库管理系统。因而在逻辑设计中的工作是将概念结构转换为关系模式，并将关系模式进行规范化处理，得到系统中所需的关系表。

物理设计的目的是确定数据库的物理结构(存储结构)。关系数据库的物理设计比较简单，对于一般的微机的关系数据库系统来说，这一阶段的任务包括：

(1) 确定所有数据库文件的名称及其所含字段的名称、类型和宽度。

(2) 确定各数据文件需要建立的索引，在什么字段上建立索引等[35]。

按照系统功能，将所需数据库表分为两类，参数表和排产用表。下面逐一介绍各表的设计。

根据实际生产要求，系统运行需要大量可变参数的支持。其中包括各种约束条件参数以及算法运行参数等。为了保证参数使用的灵活性和可变性，本文在系统设计时没有把参数固化在代码之中，而是将所有参数存储在数据库中，系统每次运行都由数据库中动态的读取数据。当参数发生改变时，仅需将改动保存到数据库中即可，无需改动排产算法的代码。

系统各参数表作用如图 3.8 所示。其中 PARTABLE 表记录了所有的独立参数，这些参数和其他的实体没有直接的关联，即仅需定义一次的参数，如热装时间，定义完成后，在任何条件下，针对任意的计划，热装时间都是此设定值。PARTABLE 表中记录的都是一些独立的基本参数，包括计划最大最小限制，最大宽度跳跃，最大硬度跳跃等排产约束参数，以及热装时间，换辊时间，直接热装时间等工艺参数。由于记录参数的特点，在此表内只有一条记录。PARTHICK 表记录了计划编制的厚度变化约束参数。PAR_STOP 表记录了轧机的计划开停机时间。GROUPMIX 表记录了不同的钢种组的匹配规则。PARBYGROUP 表记录了一些钢种组相关的信息，如某一钢种组的同宽最大轧制长度，同宽最大轧制块数等约束。PARTIME 表记录了用来计算某一块板坯轧制时间的相关参数。

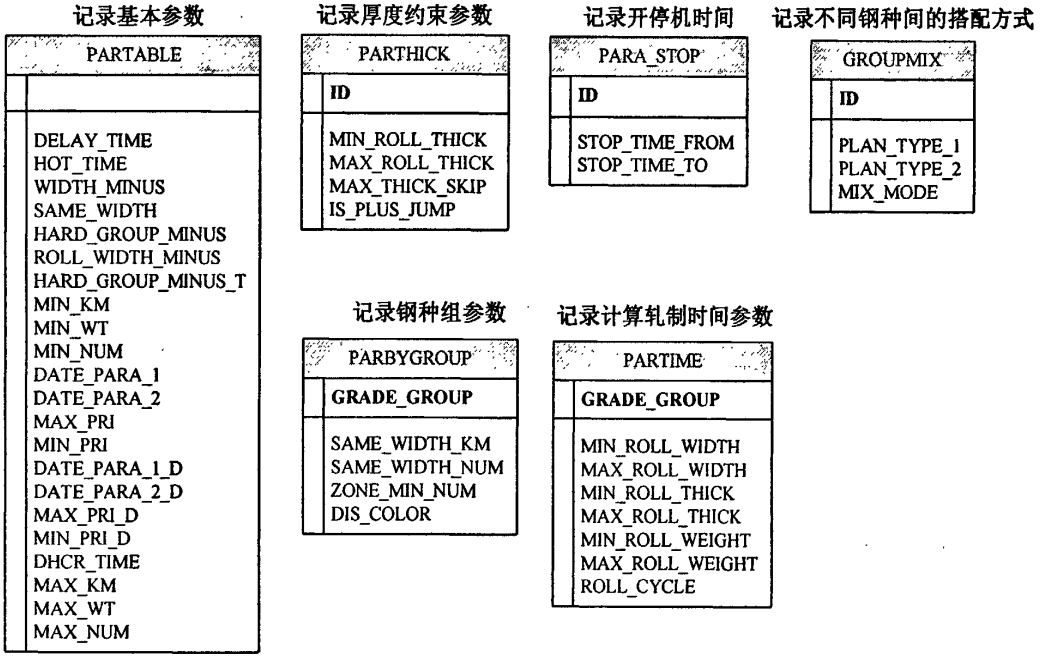

图 3.8　主要参数表

Fig. 3.8　Tables of main parameters

下面介绍各个参数表的字段设计：

(1) 基本参数表(PARTABLE)的设计。见表 3.1；

(2) 厚度参数表(PARTHICK)的设计。见表 3.2；

(3) 开停机时间表(PARSTOP)的设计。见表 3.3;
(4) 钢种搭配表(GROUPMIX)的设计。见表 3.4;
(5) 钢种组参数(PARBYGROUP)的设计。见表 3.5。
(6) 轧制时间规则表(PARTIME)的设计。见表 3.6。

表 3.1 基本参数表(PARTABLE)设计

Tab. 3.1 Design of table PARTABLE

字段名	数据类型	字段大小	允许为空	主键	默认值	说明
[DELAY_TIME]	[float]		否	否	60	传时间搁
[HOT_TIME]	[float]		否	否	16	热装时间
[MIN_KM]	[decimal]	(3, 0)	否	否		轧制单元最小长度
[MIN_WT]	[decimal]	(10, 0)	否	否		轧制单元最小重量
[MIN_NUM]	[decimal]	(3, 0)	否	否		轧制单元最小块数
[MAX_KM]	[decimal]	(3, 0)	否	否		轧制单元最大长度
[MAX _WT]	[decimal]	(10, 0)	否	否		轧制单元最大重量
[MAX _NUM]	[decimal]	(3, 0)	否	否		轧制单元最大块数

表 3.2 厚度参数表(PARTHICK)设计

Tab. 3.2 Design of table PARTHICK

字段名	数据类型	字段大小	允许为空	是否主键	默认值	说明
ID	int		否	是		
[MIN_ROLL_THICK]	[decimal]	(6, 3)				厚度下限
[MAX_ROLL_THICK]	[decimal]	(6, 3)				厚度上限
[MAX_THICK_SKIP]	[decimal]	(6, 3)				最大允许跳跃
[IS_PLUS_JUMP]	[float]					

表 3.3 开停机时间表(PARSTOP)设计

Tab. 3.3 Design of table PARSTOP

字段名	数据类型	字段大小	允许为空	是否主键	默认值	说明
ID	int		否	是		
[STOP_TIME_FROM]	[varchar]	(14)				停机开始时间
[STOP_TIME_TO]	[varchar]	(14)				停机结束时间

表 3.4 钢种搭配表(GROUPMIX)设计

Tab. 3.4 Design of table GROUPMIX

字段名	数据类型	字段大小	允许为空	是否主键	默认值	说明
[ID]	int		否	是		
[PLAN_TYPE_1]	[varchar]	(2)				前一类型
[PLAN_TYPE_2]	[varchar]	(2)				可匹配类型
[MIX_MODE]	[varchar]	(1)				匹配模式

表 3.5 钢种组参数表(PARBYGROUP)设计

Tab. 3.5 Design of table PARBYGROUP

字段名	数据类型	字段大小	允许为空	是否主键	默认值	说明
[GRADE_GROUP]	[varchar]	(2)	否	是		钢种组
[SAME_WIDTH_KM]	[decimal]	(6, 0)				最大同宽长度
[SAME_WIDTH_NUM]	[decimal]	(3, 0)				最大同宽数量
[ZONE_MIN_NUM]	[decimal]	(3, 0)				区间最小块数
[DIS_COLOR]	[varchar]	(32)				显示颜色

表 3.6 轧制时间规则表(PARTIME)设计

Tab. 3.6 Design of table PARTIME

字段名	数据类型	字段大小	允许为空	是否主键	默认值	说明
[GRADE_GROUP]	[varchar]	(2)	否	是		钢种组
[MIN_ROLL_WIDTH]	[decimal]	(5, 1)				宽度下限
[MAX_ROLL_WIDTH]	[decimal]	(5, 1)				宽度上限
[MIN_ROLL_THICK]	[decimal]	(8, 3)				厚度下限
[MAX_ROLL_THICK]	[decimal]	(8, 3)				厚度上限
[MIN_ROLL_WEIGHT]	[decimal]	(8, 3)				重量下限
[MAX_ROLL_WEIGHT]	[decimal]	(8, 3)				重量上限
[ROLL_CYCLE]	[decimal]	(4,01)				轧制时间

系统的排产用表包括如图 3.9 所示的各表。排产用表用来记录待排产原料数据，排产结果数据，以及排产中间结果等。

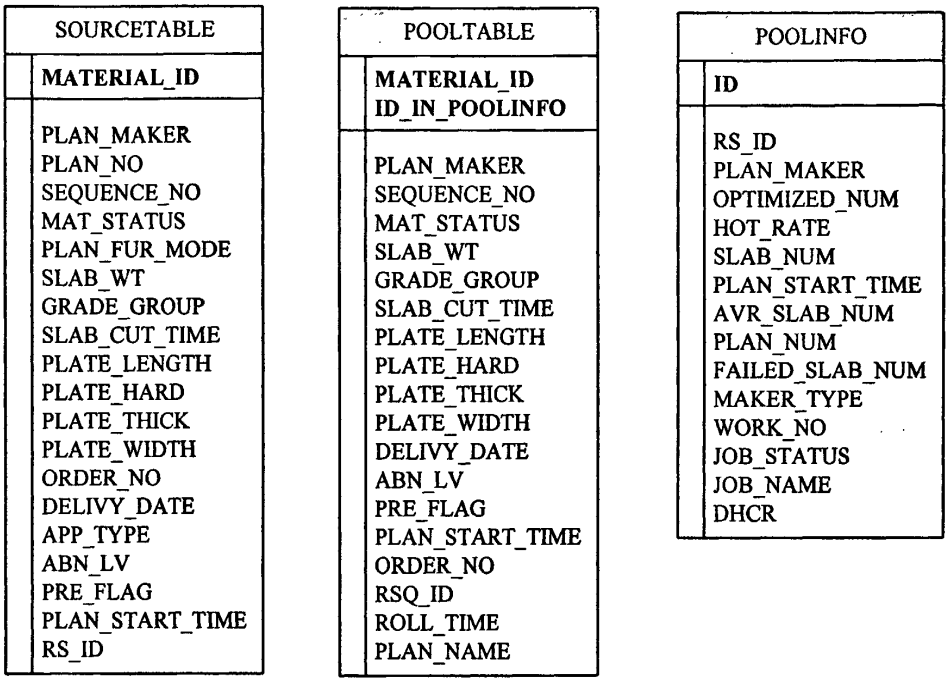

图 3.9 排产用表

Fig. 3.9 Tables used for planning

下面介绍各个排产用表的字段设计：

(1) 原料数据表(SOURCETABLE)的设计。原料数据表存储了待排产的初始数据。该数据描述了所有待排产板坯的详细信息，包括板坯的宽度，厚度，硬度，装炉方式等工艺参数，以及板坯的切断时间，板坯所在合同的交货期，计划加工日期等生产计划相关参数。原料数据表的详细设计见表 3.7；

(2) 排产结果表(POOLTABLE)的设计。排产结果表主要记录了排产的最终结果，包括手动调整前和调整后的结果。此表主要的字段为 RS_ID，PLAN_NO，SEQUENCE_NO，其中 RS_ID 表示排产的批次号，PLAN_NO 记录了该板坯所属的最后轧制单元编号，SEQUENCE_NO 记录了板坯在轧制单元内的序列号。排产结果表的详细设计见表 3.8；

(3) 排产结果统计表(POOLINFO)的设计。排产结果统计表记录了排产结果的详细统计信息，包括某一个排产批次共形成的轧制单元总数，轧制单元的平均长度，轧制单元的剩余板坯数等统计信息。此表的信息由 POOLTABLE 表统计而来，设计此表的原因是这些统计信息要在很多地方大量使用。在不设计此表的情况下，会重复的对 POOLTABLE 表进行统计，由于 POOLTABLE 表中含有大量的数据，对其进行统计查询将非常费时，程序整体的反应速度会明显下降。设计此表后，所有的统计工作仅进在

建立 POOLTABLE 表时执行一次，大大的提高了系统的反应速度。排产结果统计表得详细设计见表 3.9：

表 3.7 原料数据表(SOURCETABLE)设计

Tab. 3.7 Design of table SOURCETABLE

字段名	数据类型	字段大小	允许为空	是否主键	默认值	说明
[PLAN_MAKER]	[varchar]	(8)				计划编制人
[PLAN_NO]	[varchar]	(10)				计划编号
[MATERIAL_ID]	[varchar]	(20)	否	是		原料 ID
[SEQUENCE_NO]	[decimal]	(4, 0)				轧制序列号
[MAT_STATUS]	[varchar]	(2)				材料状态码
[PLAN_FUR_MODE]	[varchar]	(2)				装炉模式
[SLAB_WT]	[decimal]	(8, 3)				板坯重量
[GRADE_GROUP]	[varchar]	(2)				钢种组
[SLAB_CUT_TIME]	[varchar]	(14)				切断时间
[PLATE_LENGTH]	[decimal]	(6, 0)				板坯长度
[PLATE_HARD]	[decimal]	(2, 0)				板坯硬度
[PLATE_THICK]	[decimal]	(6, 3)				板坯厚度
[PLATE_WIDTH]	[decimal]	(5, 1)				板坯宽度
[PLAN_START_TIME]	[varchar]	(14)				计划开始时间

表 3.8 排产结果表(POOLTABLE)设计

Tab. 3.8 Design of table POOLTABLE

字段名	数据类型	字段大小	允许为空	是否主键	默认值	说明
[ID_IN_POOLINFO]	[float]		否	是		与 POOLINFO 关联的外键
[MATERIAL_ID]	[varchar]	(20)	否	是		原料 ID
[SEQUENCE_NO]	[decimal]	(4, 0)				轧制序列号
[MAT_STATUS]	[varchar]	(2)				材料状态码
[PLAN_FUR_MODE]	[varchar]	(2)				装炉模式
[SLAB_WT]	[decimal]	(8, 3)				板坯重量

字段名	数据类型	字段大小	允许为空	是否主键	默认值	说明
[GRADE_GROUP]	[varchar]	(2)				钢种组
[SLAB_CUT_TIME]	[varchar]	(14)				切断时间
[PLATE_LENGTH]	[decimal]	(6, 0)				板坯长度
[PLATE_HARD]	[decimal]	(2, 0)				板坯硬度
[PLATE_THICK]	[decimal]	(6, 3)				板坯厚度
[PLATE_WIDTH]	[decimal]	(5, 1)				板坯宽度
[PLAN_START_TIME]	[varchar]	(14)				计划开始时间

表 3.9 排产结果统计表(POOLINFO)设计

Tab. 3.9 Design of table POOLINFO

字段名	数据类型	字段大小	允许为空	是否主键	默认值	说明
[ID]	[float]		否	是		
[RS_ID]	[float]					批次号
[PLAN_MAKER]	[varchar]	(10)				编制者
[OPTIMIZED_NUM]	[float]					优化次数
[HOT_RATE]	[decimal]	(4,2)				热装比
[DHCR]	[decimal]	(4,2)				直接热装比
[SLAB_NUM]	[float]					板坯总数
[PLAN_START_TIME]	[varchar]	(14)				计划开始时间
[AVR_SLAB_NUM]	[float]					平均板坯数
[PLAN_NUM]	[float]					计划数
[FAILED_SLAB_NUM]	[float]					未排板坯数
[MAKER_TYPE]	[float]					是否自由轧制
[WORK_NO]	[float]					编制者编号
[JOB_STATUS]	[float]					是否已排产
[JOB_NAME]	[varchar]	(8)				批次名称

3.2 关键算法设计

3.2.1 问题描述

热轧计划编排描述：有 n 块板坯用来编制轧制计划，算法运行结束后形成 M 个轧制单元，其中 M 值不确定。形成的轧制单元要求符合一定的约束规则，算法需要考虑的约束条件主要有如下 4 条：

(1) 主体材板坯的宽度要由宽到窄变化且变化平缓，即相邻两块板坯的宽度跳跃值不能超过某一个数值。轧辊在轧过一块板坯后，板坯的边缘部分会在轧辊上留下一些划痕。如果用轧辊有划痕的区域来轧制下一块板坯必然会影响下一块板坯的产品质量。故要有计划编制时，必须保证板坯宽度按照由宽到窄排序，而且跳跃值不能超出一个设定的临界值。效果如图 3.10 所示。

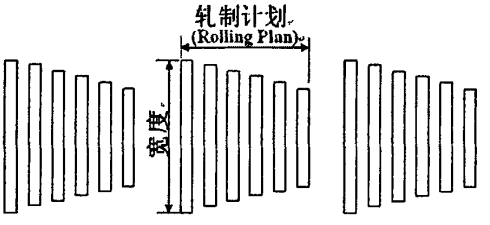

图 3.10 若干轧制计划单元

Fig. 3.10 A number of rolling unit plan

(2) 相邻板坯的厚度、硬度组要达到平滑过渡，不能有大的跳跃量。

(3) 轧制单元要满足同宽约束。算法设定一个同宽定义参数，两块板坯的宽度小于此参数则认为这两块板坯是同宽的。某板坯 A 不满足同宽约束的定义为：若与 A 依次相连的与 A 同宽的板坯总长度大于某一设定值，则认为 A 板坯同宽长度违规。

(4) 形成的计划必须满足一个最小限制。即形成的计划所有板坯的总轧制长度必须大于一个指定的总长度，总的轧制重量必须大于一个制定的重量，以及计划中的板坯数目必须大于一个最小板坯数参数。

文献[36-38]在轧制约束中还提到轧制宽度、厚度和硬度不允许同时跳跃的条件，随着钢铁生产设备自动化水平的不断提高，当前钢厂机组实际生产工艺能够做到对同时跳跃进行连续轧制，因此本文不考虑此约束条件，对轧制计划模型主要研究主体材的计划编制问题。

计划编制中的一个重要指标是轧制单元的总轧制长度(称为轧制公里数)。从成本角度考虑，应当使每个轧制单元的轧制公里数尽可能大，这样可以在满足合同要求的前提下减少换辊次数，达到降低生产成本的目的。

3.2.2 启发式算法设计

本系统的核心排产算法在系统的业务逻辑层的 Plan 类中实现。本系统采用一种改进的启发式算法来实现系统的排产功能。传统的启发式算法根据生产的规则约束进行判断，决定每一块板坯在整个计划中所在位置，算法多次运行的排产结果是固定的。热轧排产问题可以归结为一个 NP-Hard 问题，很难用启发式的规则决定一块板坯的最优位置。某一块板坯在排产结果中的位置不同可能会导致整个排产结果有很大的变化。根据启发式算法计算出某一块板坯的位置往往并不是一个最优的位置，当这个位置不当时，可能会导致最后排产结果的质量有很大的下降。

根据此思想，本文对传统的启发式算法进行了一个改进。当根据启发式规则判读某一板坯可以排在位置 X 时，并不一定将其安排在 X 处，而是以一个指定的概率 $a\%$来决定是否将其放于位置 X。即该板坯有 $a\%$的概率放于位置 X。在算法的最后，当 X 没有找到合适的位置时，再将其插入到位置 X。由于引入了随机数 a，算法每次运行的结果不尽相同。故每次排产时，多次运行算法，最后取一个最优解提供给用户。由于启发式算法具有速度很快的特点，此算法的运行时间也可以接受。

排产是本系统最主要的一个功能，排产算法的好坏直接影响着最后的排产结果。根据实际生产要求，将排产算法分为两种。一种为单排计划，令一种为混排计划。单排只针对单一钢种组进行排产，不允许在一个计划内存在两种以上钢种组。混排计划允许在一个计划内存在多种钢种组。混排计划除了要保证所有的主题材相关约束外，还必须考虑不同钢种组间的匹配规则。某一钢种组之后仅能衔接某些钢种组的板坯。每一中钢种组都必须有一个最小长度的限制，即计划中相邻的同钢种组的板坯数必须满足一个最小数量的限制。

单排算法包括如下几个步骤：

Step1：将待排产板坯按照板坯宽度和切断时间进行排序。令 i=0；

Step2：令当前板坯 A 为计划的第 i 块板坯。

Step3：判断将当前板坯 A 加入到计划的最后是否违反约束条件。如果违反约束则执行 Step5，否则执行 Step4。

Step4：生成一个随机数 x（x>0,x<1），若 x 小于 0.7 则将 A 加入到计划的最后。

Step5：判断 A 是否为最后一块板坯，若不是则令 A 为下一块板坯，转到 Step3。否则转到 Step6：

Step6：判断目前形成的计划总长度，总重量，总块数是否满足最小限制，若满足或者 i+1 大于待排产板坯的总数量，算法结束。否则，拆除当前计划，令 i=i+1，转到 Step2。

混排算法包括如下几个步骤：

Step1：将待排产板坯按照板坯宽度和切断时间进行排序。令 i=0；

Step2：令当前板坯 A 为计划的第 i 块板坯，令当前钢种组 P 为 A 的钢种组类型。

Step3：判断 A 的钢种组类型是否为 P，如果不是，则执行 Step8，否则执行 Step4。

Step4：判断将当前板坯 A 加入到计划的最后是否违反约束条件。如果违法约束则执行 Step8，否则执行 Step5。

Step5：生成一个随机数 x（x>0,x<1），若 x 小于 0.7 则将 A 加入到计划的最后。

Step6：由 A 开始向前计算钢种组为 P 的板坯数 k，若 k 等于 P 对应的最小区间块数则执行 Step7，否则拆除当前形成计划的最后一个钢种组为 P 的区间。令 P 为能和此拆除区间的前一个区间的钢种组匹配的下一个钢种组类型，执行 Step8。

Step7：令 P 等于能和 P 衔接的下一种钢种组。

Step8：判断 A 是否为最后一块板坯，若不是则令 A 为下一块板坯，转到 Step3。否则转到 Step9：

Step9：判断目前形成的计划总长度，总重量，总块数是否满足最小限制，若满足或者 i+1 大于待排产板坯的总数量，算法结束。否则，拆除当前计划，令 i=i+1，转到 Step2。

3.2.3 仿真结果

为了验证本算法的运行效果，用某厂的实际生产数据进行仿真试验。比较不同方法的对比结果如表 3.10，图 3.11 所示：

表 3.10 排产结果统计表(POOLINFO)设计

Tab. 3.10 Design of table POOLINFO

数据样本编号	1	2	3	4	5	6	7	8
手动排产结果	232	153	122	151	550	206	705	117
未引进随机数时剩余板坯数	173	109	101	143	501	141	673	87
引进随机数后剩余板坯数	54	46	37	94	254	109	490	55

如表 3.10 所示，与传统的手动排产方法相比，本文设计的排产方法可以保证有更多的板坯被编入到计划中，仅有少量板坯剩余。通过引入随即数后，算法的运行效果更是得到了进一步的优化，特别是在数据切断时间分布不合理的情况下，这种优化效果更加明显。

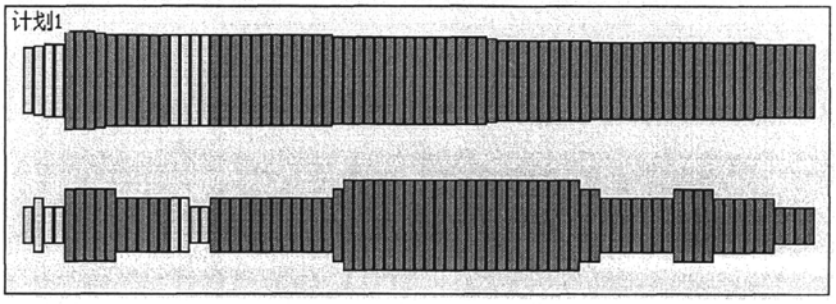

图 3.11　若干轧制计划单元

Fig. 3.11　A number of rolling unit plan

图 3.11 为算法运行的一个排产结果。图中上一个图形为计划的宽度变化趋势，下一个为轧制计划的厚度变化趋势。由图可知，在保证各种约束都满足的条件下，算法可以使计划的宽度和厚度变化尽可能的平缓，避免剧烈的波动。

4 软件实现

系统采用 C#语言，在.NET 2.0 平台下实现。系统采用.NET WinForm 程序设计架构开发出友好的应用程序界面。系统的核心算法封装成一个 dll 文件，由用户界面来调用。系统实现采用面向对象思想，将主要功能根据相关性关系封装在几个类中，从而提高了代码的可扩展性和可重用性。程序的总体运行界面如图 4.1 所示：

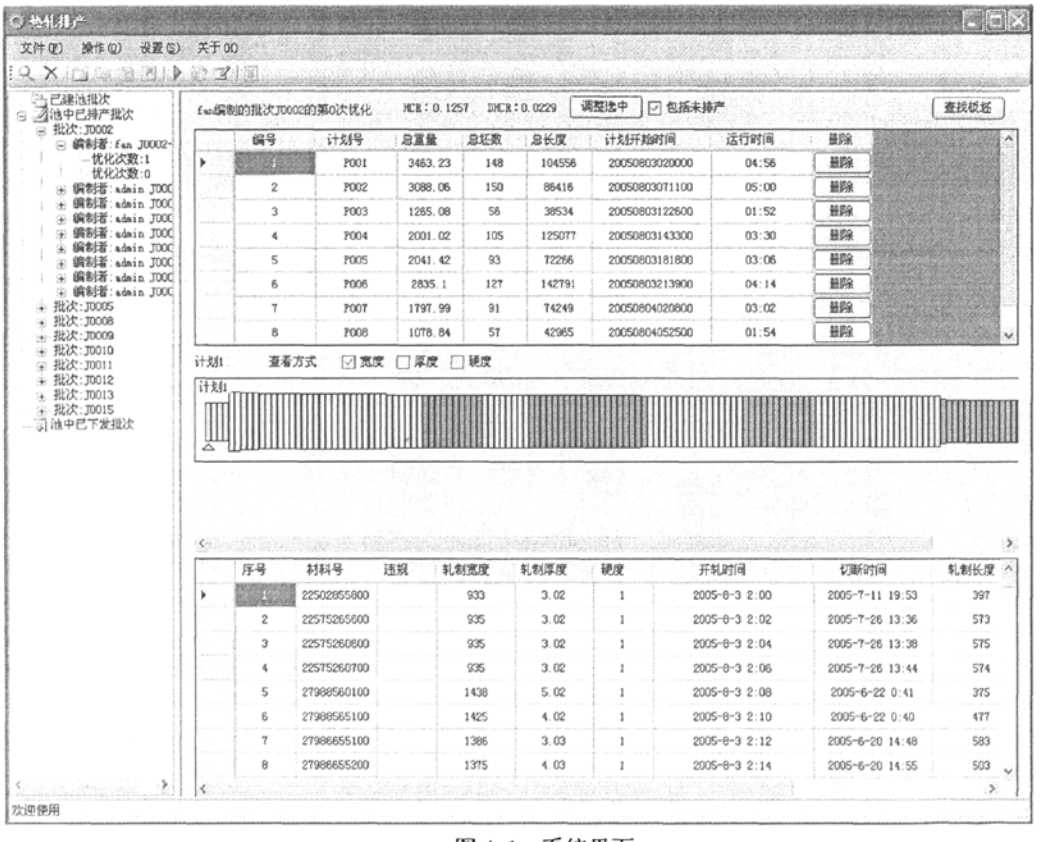

图 4.1 系统界面

Fig. 4.1 The interface of the system

4.1 数据存取层实现

数据存取层是整个系统的最底层，实现了和数据库交互的所有接口工作。考虑将来可能的数据库类型的变化，本文利用多态技术分别设计了针对不同数据的数据存取类。

考虑到目前系统应用的数据库类型为 Oracle 10G 本文实现了针对 Oracle 数据库的特定数据存取类。

DataBase 类主要成员包括返回 DataSet 数据类型的 DataBase.ExecuteDataset 函数，以及返回 IDataReader 接口的 DataBase.ExecuteReader 函数，以及返回 object 类的 DataBase. ExecuteScalar 函数等。所有的这些函数都经过多次重载，可以使用不同的参数类型来调用。其中最简单的调用方式只需要直接传递一个 SQL 语句即可，大大降低了数据库使用的复杂度。

通过数据存储层的封装设计，大大简化了系统的数据库操作。系统的数据库操作的代码量也大大地缩减。所有的数据库操作仅需一条语句即可实现。极大的减轻了开发用时，也缩短了开发时间，提高了开发效率。

4.2 业务逻辑层实现

4.2.1 BaseWork 类实现

BaseWork 是一个抽象的基类，表示的是一次排产。一次排产为一个 work，每次排产会形成多个计划，即一个 Work 中包含多个 Plan 对象。 BaseWork 提供了所有 Work 类的公共方法，和一些公共接口的定义，利用多态技术实现了不同 Work 类的统一管理。当系统需要扩展，比如要添加一种新的排产方法时，仅需增加一个 BaseWork 的派生类，将新的排产逻辑在派生类中实现即可。原系统无需改动，而且增加的新功能和原系统有一个很好的统一性。

BaseWork 类的主要需要实现的功能包括建立单排计划，建立混排计划，建立烫辊材做主题材计划，计算轧制时间，插入剩余板坯，将计划写入数据库能功能。下面将详细介绍各个功能的具体实现。

(1) AddHeatSlabToAllPlan

此函数的功能是向当前 work 所拥有的每一个 Plan 添加烫辊材。本系统的计划编制使用先编制主体材计划，后添加烫辊材的运行方式。在运行此函数之前，所有计划的主体材已经编制完成。此函数的功能是在已编制的主体材的基础上逐个向现有计划添加烫辊材。

(2) BuildHeatAsMainPlan

此函数实现将烫辊材用作主体才编制计划。在实际的钢铁生产中，可用作烫辊材的板坯数量都是相对较少。生产排产人员一般都将可用作烫辊材的板坯单独提出来，节省使用。但是在某些情况下可能有较多的烫辊材需要尽快轧制，本函数的功能就是在烫辊

材比较多的情况下，在已经成功地建立了主题材计划，并已经成功地添加了烫辊材的前提下，将剩余的烫辊材用作主题材编制计划。

具体的计划编制方法同混排计划，因为此时往往各种钢种组剩余的板坯都已经不多，使用单排算法很难成功形成计划，故使用混排方法，将剩余的烫辊材和剩余的主题材统一作为主题材使用，一同进行计划编制。

(3) BuildMultiTypePlan

混排计划是指在同一个计划内包含不同钢种组类型的板坯。在可能的情况下，需要将相同钢种组类型的板坯编制在同一个轧制单元呢，但是在某些特殊情况下，由于有计划最小长度限制，需要将不同钢种组类型的板坯安排在同一个轧制单元内。以达到使尽可能多的板坯编入计划的目的，此时需要使用混排功能。

混排功能工作在单排之后，在单排计划编制完成后，各种钢种组类型的板坯都会有一定数量的剩余。此函数以这些剩余板坯为入口参数进行计划编制。混排计划的编制方法具体和单排计划类似，但是需要多考虑一些因素。例如在混排计划中，每一个钢种组类型的板坯都要有一个最小块数的限制，当少于这个最小块数的限制时，不能紧接着安排生产其他钢种组类型的板坯。在混排计划内，各个钢种组衔接方式也有一定的约束条件。某一钢种组类型的板坯后仅能和制定的其他刚种组类型衔接，钢种组衔接约束是混排算法需要考虑的一个主要约束条件。

此函数在 Work 类中通过调用 Plan 类的功能来实现混排的功能。

(4) CalculateRollTime

此函数用来计算每一个计划的轧制时间细节。在计划编制时，需要知道板坯的轧制时间。此函数的功能是根据规则表的参数计算出每一块板坯的轧制时间。

此函数没有实现具体的计算时间的算法，而是统一的调用 Plan 级别中的计算时间函数来实现 Work 内所有板坯的轧制时间的计算功能。

(5) InsertElseSlab

在计划编制完成后，还会有一些板坯没有成功加入任何计划，主要原因是本文的算法要求严格服从各种约束条件，在实际生产中，各种约束条件非常繁杂，在满足所有条件的情况下总会有些板坯不能成功排入任何一个计划，除非某批数据板坯参数分布特别规整，但是这种情况在实际生产中是很少见的，在一个批次内总会有一些板坯无法编入计划。

此外，本文中还有一个原因导致部分板坯无法加入计划，那就是本文在排产算法中引入了随机排产因子，某一块板坯即使符合所有的约束条件，也有一定的概率不将其加

入计划，采用此算法的目的是保证更多的板坯能够排入计划，但带来的一个问题就是可能有一部分本来能排入计划的板坯没有排入。

此函数的功能是查找所有没有加入计划的板坯，并尝试将其加入现有计划，以弥补因为引入随机数后产生的计划平均长度变小的问题。通过此函数处理后，计划的平均长度会得到进一步增加。

(6) WriteResultToDataBase

此函数的功能是将排产结果写入数据库。为加快生产计划的编制速度，在计划编制的过程中，本文没有使用任何的数据库操作，所有的结果都存在内存数据表中。计划编制完成后，利用此函数将数据表中的排产结果写入数据库中。此函数具体的实现是通过调用所有计划的写入数据库函数来实现写入功能。

4.2.2 RulesWork 类实现

RulesWork 是 BaseWork 的一个派生类，实现了 BaseWork 的所有抽象方法，RulesWork 表示的物理意义是使用基于规则的启发式算法的一次排产。在本文的研究阶段曾经设计并实现了多个继承自 BaseWork 的派生类，不同的 Work 类使用不同的排产算法来实现排产功能。经过对比，最后保留了目前的 RulesWork 类来实现排产功能。下面详细介绍 RulesWork 的各主要函数的实现：

(1) BuildSingleTypePlan

此函数的功能是实现单排计划的编制。单排计划是指一个轧制单元中所有的板坯具有相同的钢种组类型。单排计划是计划编制中一种主要的编制方式。在 Work 级别的计划编制并不运行具体的编制算法，而是将可排产数据进行一定的处理，如剔出已排产或无法排产的数据，对板坯数据进行有利于排产的排序等。通过这些处理后，将处理后的结果传递给 Plan 级别的类。调用 Plan 级别类的具体的启发式排产算法再进行实际的排产操作。

(2) BuildWork

在此函数中实现计划编制的工作。此函数是 Work 类的对外接口函数，是排产算法运行的起点。此函数是一个高级别的排产函数，在此函数内运行一些方向性的排产逻辑。例如确定单排算法的运行次数，安排单排混排算法的运行方式等。通过此高层次的排产逻辑，可以为底层具体的排产算法创造一个比较好的运行大环境，而后更加有利于得到一个更好的排产结果。

4.2.3 BasePlan 类实现

BasePlan 是所有 Plan 类的抽象基类，表示一个具体的轧制单元。轧制单元有很多不同的种类，比如单排计划，混排计划等。但是所有的这些轧制单元都有某些基本的特性和基本的功能。所有轧制的单元共同拥有的属性和方法在本文中都有 BasePlan 类来实现，比如一些基本功能，如添加烫辊材，计算轧制时间，排产结果写入数据库等。下面将详细介绍各个功能的具体实现。

(1) AddHeatSlab

此函数的功能是向当前计划添加烫辊材，此函数根据 work 类传递进来的内存数据池实现向计划添加烫辊材的操所。由于本文主要考虑的是主体材的计划编制工作，在添加烫辊材时没有考虑过多的约束条件，仅考虑一些基本的如宽度跳跃等约束条件。

(2) InsertElseSlab

计划编制完成后，会有一些剩余板坯，此函数的功能是将剩余的板坯尝试加入到计划中，从而加大计划的平均长度。

具体的实现采用循环插入的方法。首先在未排产板坯中选择一块板坯，然后由最后一个计划的最后一块板坯开始依次查找插入位置，如果能插入到当前位置，不违反任何的约束条件，则将板坯插入当前位置。若不能插入，则寻找下一个位置，直到找到一个可以插入的位置或者没有可以插入的位置。一块板坯处理完成之后再继续找到下一块未排产板坯，进行插入处理。直到所有未排产板坯都处理完成。

在进行板坯插入时遇到较大的麻烦是违规的计算问题。在判断一块板坯是否可以插入到某一个位置时，都要计算在此位置是否满足各种约束条件。此计算需要较大的计算量，特别是考虑到同宽长度的计算，每计算一个位置都要计算很多块板坯的长度之和。针对此情况，本文采用利用空间换取时间的思想，将需要重复计算的许多信息用数组保存起来，下次需要计算时，直接使用即可。

(3) WriteResultToDateBase

此函数实现将排产的结果写入数据库中。在排产算法运行期间，为了提高运行速度，本文没有采取任何数据库操作，而是采用所有的操作都在内存中运行的模式。包括最后的排产结果，在算法运行完成之后都保存在内存之中。此函数的功能就是将这些排产的最终结果写入数据库之中。

此函数的实现比较简单，只需将排产结果根据内存信息依次写入数据库即可。需要注意的是为了防止重复写入，减少排产过程的运时间，需要判断此计划是否已经执行过此写入函数。

(4) CalculateRoolTime

此函数的功能是计算计划中的每一块板坯的开始和结束时间。在排产算法运行期间需要使用每一块板坯的具体的生产时间。根据此时间来确定板坯是否有切断时间违规以及安排每一块板坯的详细生产时间。每一块板坯的具体生产时间在排产时是无法得知的，只能根据板坯的厚度重量等参数根据一定的规则来推算此时间。此函数的功能就是根据板坯的厚度，重量，长度来具体的推测每一块板坯的轧制生产时间。

推测板坯生产时间的参数由用户设定，并保存在数据库中。每次此函数运行时，由数据库读取相关规则，包括一个重量范围，宽度范围，长度范围，以及满足这 3 个条件的板坯的生产时间，则所有满足这 3 个条件的板坯的实际生产时间即可以推测出来。

4.2.4 SingleTypePlan 类实现

SingleTypePlan 类继承自 BasePlan 表示的一个单排计划类型。在单排计划类型内，所有的板坯的钢种组类型都是相同的。SingleTypePlan 的大部分功能继承自 BasePlan 类，只是实现了针对单排类型的排产算法。下面详细介绍各个功能的实现流程：

(1) BuildPlan 方法

此方法是生成单排计划的启发式算法实现。在此函数内运行 3.2 节设计的基于规则的启发式算法，得到排产结果。

4.2.5 MultiTypePlan 类实现

MultiTypePlan 类继承自 BasePlan 表示的一个混排计划类型。在混排计划类型内，板坯的钢种组类型可以不同。采用混排是为了保证有尽可能多的板坯被排入计划。下面详细介绍各个过程的功能和实现流程：

(1) BuildPlan 方法

此函数是生成混排计划的启发式算法实现。在此函数内运行 3.2 节设计的基于规则的启发式算法，得到排产结果。

(2) GetNextMatchedType 方法

此函数实现根据当前钢种组获得下一个可以于之搭配的钢种组。根据当前板坯的钢种组类型，此函数找到能与之匹配的下一个钢种组类型。

4.3 表示层实现

4.3.1 建池功能实现

基于面向对象程序设计的封装思想，本系统将建池功能封装为一个独立的 WinForm 控件。当有移植功能的需要时，只需直接将此控件插入到新的项目中即可。

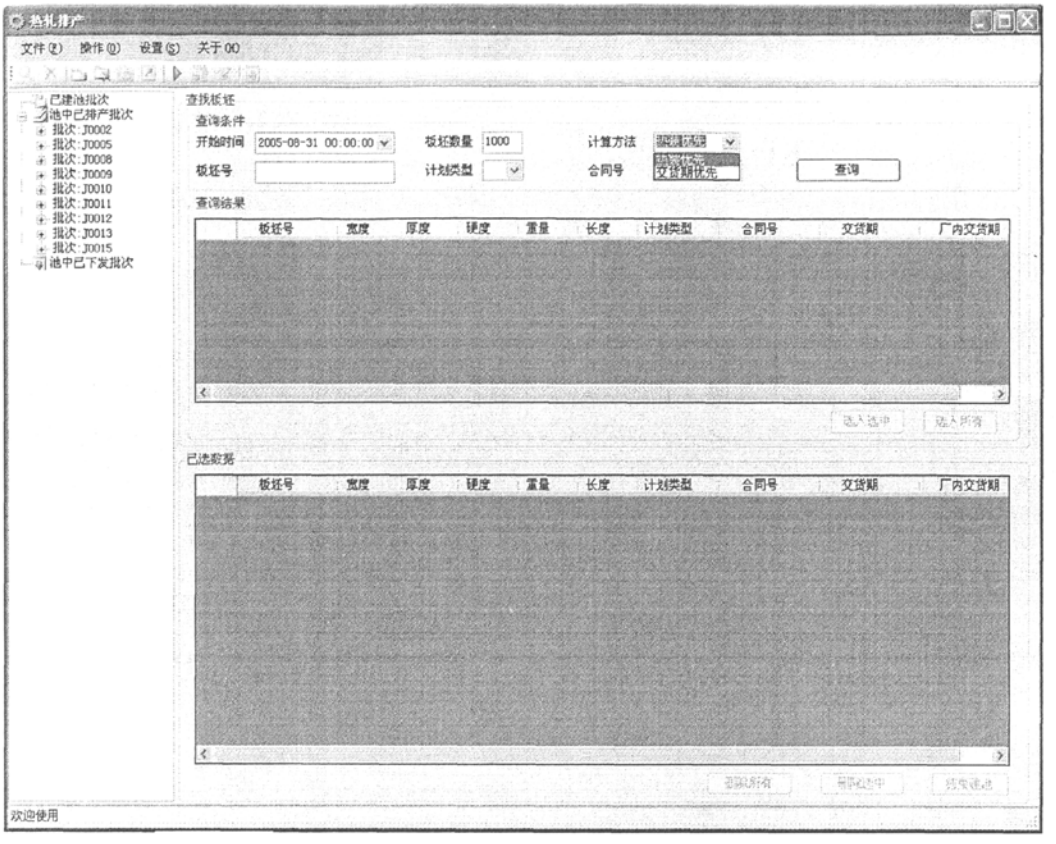

图 4.2 建池功能

Fig. 4.2 The function of building pool

建池控件的运行效果如图 4.2 所示：

在建池控件 ImportJobCtl 中，程序提供了两种建池方法，分别为综合优先级算法和热装优先级算法。两种方法分别对应不同的应用情况。当可排产数据较多，且交货期不很紧张的情况下，使用热装优先的建池方式，可保证排产结果有较高的热装比。在交货期很紧张的情况下，使用综合优先方法，可保证可以按时交货，并获得尽可能高的热装比。具体使用何种建池方法可由用户与程序界面处选定。

建池时，用户由界面输入相关查询条件，如计划开始时间等。点击查询按钮，程序会按照选定的计算方法计算符合条件的所有板坯的优先级，然后按照优选级由大到小选出指定数量的板坯显示在查询结果表中。

查询结果表中将显示选出板坯的详细信息，该结果为程序计算得到的一个比较好的建池方案，当用户对此结果不满意时，可以对查询结果表中的数据进行再一次的筛选。

点击下方的选入选中或选入所有，将符合用户要求的板坯选入下方的已选数据表中。在已选数据表中，用户可以进行一定的筛选，筛选结束后，点击结束建池按钮，则完成此次建池操作。建池数据会显示在左侧已建池批次中。

4.3.2 排产功能实现

排产功能是系统的核心功能，本文将排产功能实现在一个单独 VS2005 项目中。编译后得到与界面分离的单独的 DLL 文件，具有较高的可移植性。

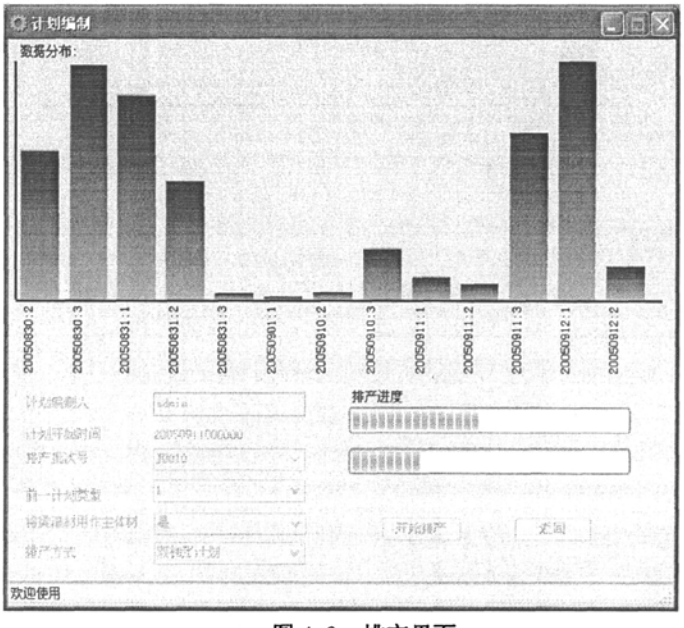

图 4.3 排产界面

Fig. 4.3 The interface of planning function

在系统的表示层，本文实现了一个用来调用排产功能以及显示排产进度对话框。如图 4.3 所示。在选择相关初始参数后，如排产批次号等信息后，点击开始排产按钮，算法则在后台运行。在选择批次号改变后，对话框上方的柱形图形也将随之改变。此柱形图形描述了选中批次的切断时间分布情况。根据此图形，用户可以对排产结果做出一个大概的分析，同时也有利于用户找到排产结果不好的原因。

在后台算法运行时，在排产对话框中将显示排产进度，其中上一进度条为排产子进度，下一进度条为排产总进度。此进度条采用多线程技术实现，保证算法和界面运行在两个不同的线程中，从而保证界面有较高的用户响应性。

4.3.3 排产结果显示功能实现

为方便移植与扩展功能，本文将排产结果显示功能也进行了模块化设计。如图 4.4 所示，ResultView 是用来显示排产结果的专用控件。

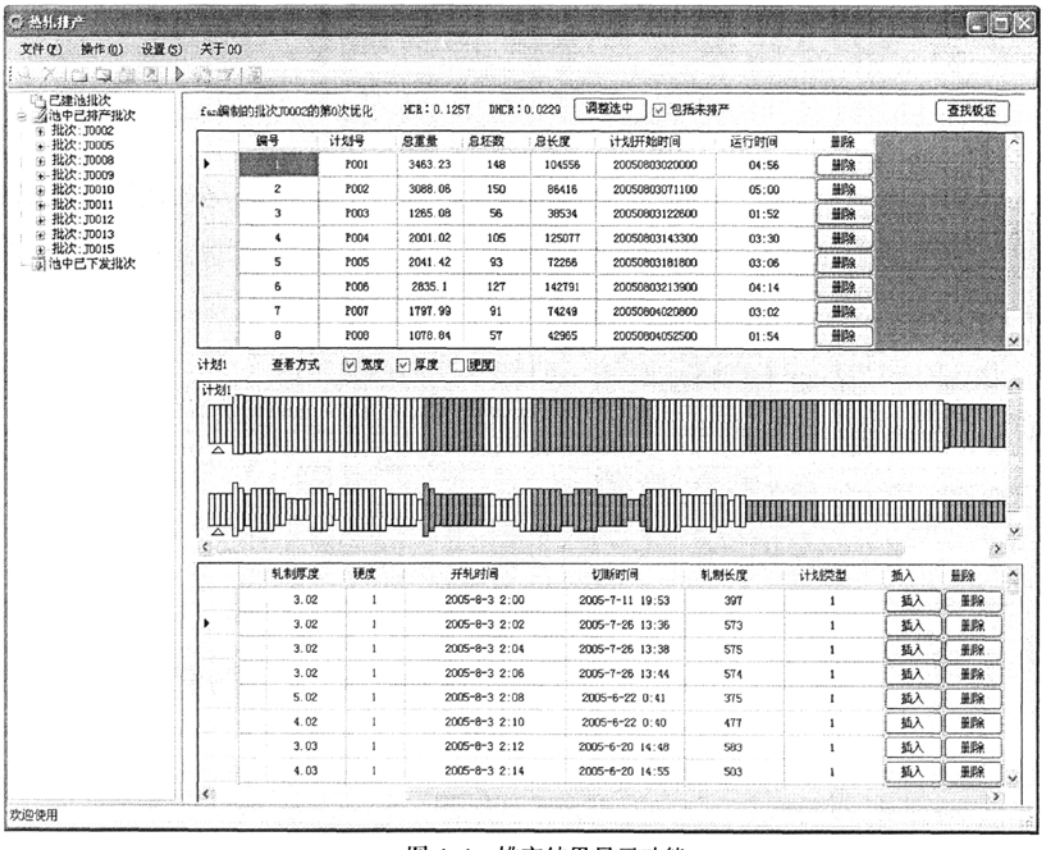

图 4.4 排产结果显示功能

Fig. 4.4 The function of displaying of result

在 ResultView 控件的上方是本次排产形成的所有的计划信息，包括每一个计划的编号，计划的总重量，计划的总板坯数，计划的总长度，计划开始时间等。可以通过点击删除按钮删除指定的计划。

在控件上方点击指定计划所在的行，该计划的详细图形信息将在控件中间的图形中显示。在此可以同时显示该计划宽度，厚度，硬度 3 个参数的变化情况。选择相应的选项，可以动态显示所需参数。

在控件的下方显示了计划的详细数据信息。当选中的计划发生变化时，此表格会显示当前所选计划中的每一块板坯的详细信息。包括板坯的宽度，厚度，等原料信息，以及板坯在计划中的编号，开轧时间等计划编制信息。在此处也可以进行简单的手动调整功能。包括在在指定位置插入板坯以及删除指定位置的板坯等。

4.3.4 手动调整功能实现

本文实现了友好的用户操所功能，其中最主要的是实现了板坯的手动拖动调整功能。在目前的实际应用中，计划的手动调整基本上操作人员面对的都是一个数据的表格，记录了每一个板坯的详细信息，操作人员需要针对此表格衡量排产的结果，再进行调整。对板坯违规情况的判断也很困难。往往需要进行大量的人工计算，浪费时间而且容易出错误。本文实现了一个图形化的拖动调整功能，操作人员只需直接在图形界面上拖动相应的板坯，即可实现计划的调整。当有板坯处于参数违规的状态时，程序会以红色边框标记，十分明显。手动调整功能的实现效果如图 4.5 所示：

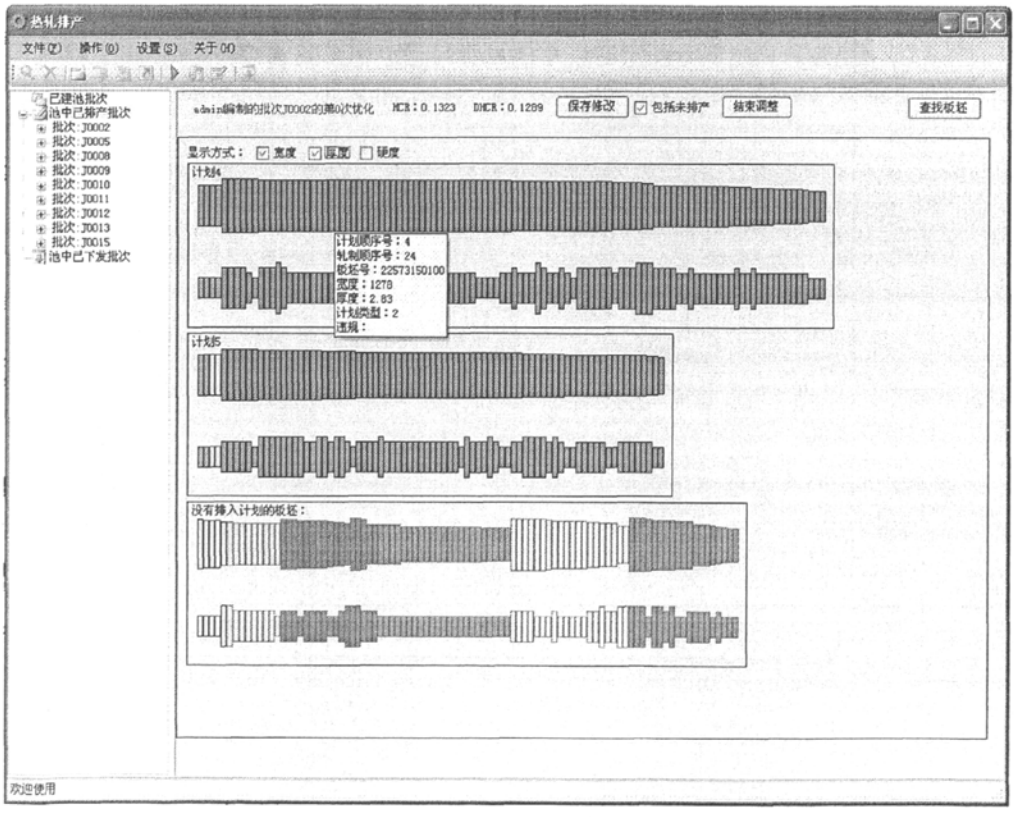

图 4.5 手动调整功能

Fig. 4.5 Manual adjustment function

在此控件中可以同时进行多个计划的手动调整功能。同时可以显示每一个待调整计划的宽度，硬度，厚度 3 个主要参数。当需要知道某块板坯的其他参数时，仅需将鼠标停留在板坯上方，则可得到板坯的其他详细信息，如图 4.5 所示。

在进行计划调整时，可以在不同的计划间任意拖动板坯，包括计划内板坯位置的任意调整以及计划间的板坯位置任意调整。当任意一块板坯的位置发生变化时，程序将重新计算违规情况，当发现板坯违规时，程序会将其边框置为红色。将鼠标停留在违规板坯的上会得到详细的违规信息，显示具体违规的参数。

在进行手动调整的过程中，可以查找某一特定板坯。点击右上方的查找板坯按钮，将会弹出一个查找功能对话框。

填写或选择查找条件，点击查询按钮，程序会给出符合查询条件的结果。在查询结果中选择需要显示的数据，双击或点击返回按钮，程序会将相应板坯以特定颜色显示。当查询结果不在正在调整的计划中时，程序将会提示是否将此计划显示在调整界面中。

结　论

本文主要研究了热轧生产计划编排系统的总体架构分析，软件设计以及实现。在软件开发的整个生存期中，用户的需求即使在开发的后期也可能经常有所变化，项目往往在整个开发的过程中都要处在不停的改动之中。采用 Layers 程序架构以及面向对象的程序设计方法可以很好的适应这种变动，同时保证了系统具有很高的可扩展性。尽管在程序设计的初始阶段要多做许多工作，甚至使系统变得更加复杂，但在整体的考虑下，采用此架构可以大大缩短开发时间，提高程序的健壮性。

本文在设计了一个合理的程序架构的基础之上实现了一个改进的启发式方法，以及一个友好的人机交互界面。此算法使程序得到了一个更好的排产结果，使排产结果的计划平均长度，热装比等综合指标得到了提高。良好的人机交互界面也改变了以往工业软件产品难用，接口不友好的特点，进一步加强了系统的用户满意度。

参 考 文 献

[1] 张涛. 基于 MTO 管理体系的钢厂合同计划和热轧生产调度的方法研究:(博士学位论文). 沈阳:东北大学, 2000.

[2] 钱晓龙, 唐立新, 刘文新. 动态调度的研究方法综述. 控制与决策. 2001, 16(2):141-145.

[3] Panwalker SS, Iskander W. A survey of Scheduling rules. Oper Res. 1977, 251:45-61.

[4] Ramash R. Dynamic job shop scheduling—A review of simulation research. OMEGA Int J Of Manag Sei. 1990, 18(1):43-57.

[5] 唐立新. 轧钢厂的精轧工序轧制批量调度的优化模型. 东北大学学报(自然科学版). 1998, 19(6):624-626.

[6] 唐立新. 热轧带钢轧制批量计划的实例应用. 东北大学学报(自然科学版). 1999, 20(3):267-269.

[7] Szelde E K, Roger M. Knowledge-based reactive scheduling. Prod Plan & Control. 1994, 5(5):124-145.

[8] 吴悦, 汪定伟. 用模拟退火法解任务的加工时间为模糊区间数的单机提前/拖期调度问题. 信息与控制. 1998, 275:394-399.

[9] 衣杨, 汪定伟. 并行多机成组工件调度的禁忌搜索方法. 系统工程. 2000, 20(S1):11-17.

[10] Lee C Y, Piramuthu S, Tsai Y K. Job shop scheduling with a genetic algorithm and machine Learning. Int J of Prod Res. 1997, 354:1171-1191.

[11] Jian A K, Elmaraghy H A. Production scheduling /rescheduling in flexible manufacturing, Int J of Prod Res. 1997, 35(1):281-309.

[12] Jones A, Rabelo K Y, Yuewhern Y. A hybrid approach of real-time sequencing and scheduling. Int J of Comp Integ Manuf. 1995, 8(2):145-154.

[13] Kouiss K, Pierreval H, Nasser M. Using multi-agent architecture in FMS for dynamic scheduling, J of Intel Manuf. 1997, 8(1):41-47.

[14] Nof S Y, Grant F H. Adaptive/Predictive scheduling review and a general framework. Prod Plan & Contr. 1991, 24:298-312.

[15] Lixin T, Jiyin L, Ayiing R et al. A Review of Planning and Scheduling System and Methods for Integrated Steel Production. European Journal of Optional Research. 2001, 133:1-20

[16] 刘晓强, 顾佳晨, 孙彦广等. 钢铁企业 MES 中的计划调度系统. 冶金自动化. 2004, 1:22-9

[17] 王军, 金以慧. 连续过程生产调度的研究策略. 系统工程理论与实践. 1998, 5: 40-46

[18] Paul C, Felix B, Len B et al. 软件架构编档. 北京:清华大学出版社. 2003

[19] 杰拉尔德, 温伯格. 系统化思维导论. 北京:清华大学出版社. 2003

[20] Kevin H, 汪钟鸣, Jeff G. .NET Framework 高级编程. 北京:清华大学出版社, 2002.

[21] 张志学. .NET 框架程序开发指南. 北京:清华大学出版社, 2002.

[22] Michael Platt. Microsoft 体系结构概述. The Microsoft Journal for Developers, 2001.

[23] Mary K. The Programmable Web Services Provide Building Blocks for the Microsoft .NET Framework. The Microsoft Journal for Developers, 2001.

[24] Middleware C. J2EE and .NET Application Server and Web Services Benchmark. 2002.

[25] Richard V. Constructing a web information system development methodology. Information Systems Journal. 2002, 12(3):247-261.

[26] Richard A, Brian F. ASP.NET 高级编程. 北京: 清华大学出版社, 2002.

[27] Paul D, Fabio C F. ADO.NET 高级编程—C#编程篇. 北京:清华大学出版社, 2003.

[28] Jason B, Tony C. ASP.NET Database Programming Weekend Crash Course. 2002.

[29] Brian D. Internet-based information systems use in organizations: an information studies perspective. Information Systems Journal. 2003, 13(2):113-132.

[30] Shawn W. ADO.NET 实用指南. 北京:清华大学出版社, 2003.

[31] Bahador G. Analysis, design and development model: a case study of an internet-based system for insert and parameter selection. Information Systems Journal. 2004, 14(2): 169-193.

[32] 求是科技. ASP.NET 信息管理系统开发实例导航. 北京:人民邮电出版社, 2005.

[33] 李建祥, 唐立新, 庞哈利等. 热轧无缝钢管排产计划研究. 控制与决策. 2002, 19(2): 238 -240.

[34] Martin F. 王怀民 周斌译. 企业应用架构模式. 北京: 机械工业出版社, 2004.

[35] 章立民. SQL Server 2000 中文版完全实战入门篇. 北京:中国铁道出版社, 2001.

[36] Tang L X, Liu J Y, Rong A Y, et al. A multiple traveling salesman problem model for hot rolling scheduling in shanghai baoshan iron & steel complex. European J of Operational Research. 2000, 124(2): 267-282.

[37] 张涛, 王梦光, 杨建夏. 不确定计划数的轧制批量计划的模型和算法. 系统工程学报. 2000, 15(1):54-60.

[38] 李耀华, 王伟, 徐乐江等. 热轧生产轧制计划模型与算法研究. 控制与决策. 2005, 20(3): 275-279.

攻读硕士学位期间发表学术论文情况

范圣冲,王伟. 基于 Layers 架构模式的热轧计划编排系统设计,大连理工大学网刊. 2008.

致　谢

值此论文完成之际，谨向导师王伟教授表示我最诚挚的感谢！导师渊博的学问、活跃的思想，严谨的态度以及宽厚的为人令学生获益菲浅；导师的谆谆教诲和无微不至的关怀将令我终生难忘；导师特有的人格魅力和可贵的精神将激励我不断地刻苦学习，努力工作。

感谢赵珺师兄，他不仅在项目的进程中给与我悉心的指导，在生活中也对我十分关心照顾，我论文的内容更是得到他细心指正。

感谢王利、刘颖、张晓平、王健、何平、郝源春、田苏洁、肖楠等所有师兄弟，他们的支持和鼓励使我在学习和研究中充满勇气和信心。同时也要感谢西门子中国研究中心 MES 项目组的有关工作人员，感谢他们对项目的大力支持与协助。

深深感谢含辛茹苦养育我，为我成长付出了无数心血，不断地鼓励和支持我上进的父母亲，是他们从物质和精神上给予我的支持才使得我的学业能够顺利完成。

感谢所有关心、帮助和支持我的人们!