UNIVERSITÉ DU QUÉBEC À CHICOUTIMI

MÉMOIRE PRÉSENTÉ À
L'UNIVERSITÉ DU QUÉBEC À CHICOUTIMI
COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN INFORMATIQUE

OFFERTE À

L'UNIVERSITÉ DU QUÉBEC À CHICOUTIMI
EN VERTU D'UN PROTOCOLE D'ENTENTE
AVEC L'UNIVERSITÉ DU QUÉBEC À MONTREAL

PAR

BAI HAI

THE IMPLEMENTATION OF

CRP (CAPACITY REQUIREMENTS PLANNING) MODULE

JUIN 2009
Motivated by a desire to make the results of its graduate students' research accessible to all, and in accordance with the rules governing the acceptance and diffusion of dissertations and theses in this Institution, the Université du Québec à Chicoutimi (UQAC) is proud to make a complete version of this work available at no cost to the reader.

The author retains ownership of the copyright of this dissertation or thesis. Neither the dissertation or thesis, nor substantial extracts from it, may be printed or otherwise reproduced without the author's permission.

L'auteur conserve néanmoins la propriété du droit d'auteur qui protège ce mémoire ou cette thèse. Ni le mémoire ou la thèse ni des extraits substantiels de ceux-ci ne peuvent être imprimés ou autrement reproduits sans son autorisation.

The author retains ownership of the copyright of this dissertation or thesis. Neither the dissertation or thesis, nor substantial extracts from it, may be printed or otherwise reproduced without the author's permission.
ABSTRACT

ERP (Enterprise Resource Planning) was originated as a management concept based on the supply chain to provide a solution to balance the planning of production within an enterprise. This article focuses on the CRP module in ERP, describes in detail the concept, content and modularization of CRP as well as its usefulness and application in the production process. The function of CRP module is to expand orders to production process to calculate production time, then to adjust the load or accumulative load of each center or machine accordingly. The function described in the article tries to load production time used as load to an unlimited extend, then to use auto-balance or normal adjustment to determine the feasibility of the production order planning.
I would like to thank Prof. Yu Chang Yun for letting me work on this topic although I study computer science. I have to thank for her patience, support and many answers to my questions.

Thanks to Mr. Zheng Gang and especially Mr. Yang Chuan Cai for their assistance and help in realizing the implementation.

I would like to thank everyone at Tianjin ZhuRi Software Co. Ltd., for sharing their ideas with me and helping me over the many decisions I had to make on the way.

Finally, I would like to thank all my friends and fellow graduate students here at the University of Technology and Science of Tianjin. The assistance and encouragement offered by them has been a great and unexpected aid in my Master’s work.
LIST OF TABLES ...................................................................................................................... vi
LIST OF FIGURES .................................................................................................................. ix
LIST OF ABBREVIATIONS ...................................................................................................... x
INTRODUCTION ..................................................................................................................... 12
CHAPTER 1 REVIEW THE EVOLUTION OF ERP ................................................................. 18
CHAPTER 2 THE REQUIREMENT ANALYSIS AND SYSTEM DESIGN.......................... 22
  2.1 LOGIN FUNCTION ........................................................................................................ 24
  2.2 DATA INPUT FUNCTION .............................................................................................. 26
  2.3 CALCULATE LOAD FUNCTION .................................................................................. 28
  2.4 AUTO-BALANCE FUNCTION ...................................................................................... 36
  2.5 MANUAL ADJUST FUNCTION .................................................................................... 45
  2.6 CHECK THE FEASIBILITY OF THE PLANNING ...................................................... 54
  2.7 OUTPUT THE NON-EXECUTABLE INFORMATION .................................................. 55
  2.8 DATA OUTPUT ........................................................................................................... 55
  2.9 LOGOUT ....................................................................................................................... 56
CHAPTER 3 THE CLASS DESCRIPTION ............................................................................. 57
  3.1 THE UML OF THE CRP MODULE ........................................................................... 57
  3.2 THE DATA INPUT CLASS ......................................................................................... 59
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3 LOAD CALCULATION CLASS</td>
<td>61</td>
</tr>
<tr>
<td>3.4 MANUAL ADJUST CLASSES</td>
<td>63</td>
</tr>
<tr>
<td>3.5 CRP TASK CLASS</td>
<td>65</td>
</tr>
<tr>
<td>CHAPTER 4 THE IMPLEMENTATION</td>
<td>69</td>
</tr>
<tr>
<td>CONCLUSION</td>
<td>73</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>75</td>
</tr>
<tr>
<td>APPENDIX A CLASS TABLE</td>
<td>77</td>
</tr>
<tr>
<td>APPENDIX B LOAD CALCULATION SOURCE CODE</td>
<td>80</td>
</tr>
</tbody>
</table>
LIST OF TABLES

Table 2.1 The following table shows the login input items ........................................... 24
Table 2.2 The following table shows the login input items ........................................... 25
Table 2.3 Login modes .............................................................................................. 26
Table 2.4 The input items of the data input function ..................................................... 26
Table 2.5 The output items of the data input function .................................................... 27
Table 2.6 The input items of the calculate load function ............................................... 28
Table 2.7 The output items of the calculate load function ............................................. 28
Table 2.8 The object to calculate load ........................................................................... 29
Table 2.9 The calculation of LFT (Backward scheduling logic) ..................................... 30
Table 2.10 The calculation of EST (Backward scheduling logic) .................................... 30
Table 2.11 The information introduced from the posterior planning (FCS) ................. 34
Table 2.12 The information deployed from the planned order ...................................... 34
Table 2.13 The input items of the Auto-balance function .............................................. 36
Table 2.14 The output items of the Auto-balance function ........................................... 36
Table 2.15 The operations of the manual adjust function ............................................. 45
Table 2.16 The input items of the transference of the load ........................................... 46
Table 2.17 The output items of the transference of the load ......................................... 47
Table 2.18 The information of the new work order of the division of the work order. .... 48
Table 2.19 The input items of the division of the work order ........................................ 48
Table 2.20 The output items of the division of the work order ........................................ 49
Table 2.21 The input items of the combination of the work order .................................. 50
Table 2.22 The output items of the combination of the work order .................................. 50
Table 2.23 The input items of the accession of the work order ....................................... 51
Table 2.24 The output items of the accession of the work order ....................................... 51
Table 2.25 The input items of the deletion of the work order .......................................... 51
Table 2.26 The output items of the deletion of the work order .......................................... 52
Table 2.27 The input items of the modification of the quantity of the work order .............. 52
Table 2.28 The output items of the modification of the quantity of the work order .............. 53
Table 2.29 The operations of the modification of the shift ............................................. 53
Table 2.30 The input items of the verification of the feasibility of the planning ............... 54
Table 2.31 The output items of the verification of the feasibility of the planning ............... 54
Table 2.32 The input items of the output of the non-executable information ...................... 55
Table 2.33 The output items of the output of the non-executable information ...................... 55
Table 2.34 The output items of the data output function.............................................. 56
Table 3.1 The private variable declarations in the ZRDatalnput class................................. 59
Table 3.2 The public function declarations in the ZRDatalnput class................................. 60
Table 3.3 Shows the private function declarations of the ZRDatalnput class...................... 61
Table 3.4 Shows the private variable declarations in the ZRLoadCalcul class.................... 62
Table 3.5 The public function declarations in the ZRLoadCalcul class.............................. 62
Table 3.6 The private function declarations of the ZRLoadCalcul class............................ 63
Table 3.7  The public functions placed in the interface ZRManualAdjust .............. 64
Table 3.8  Shows the private variable declarations in the ZRLoadTransfer class ... 64
Table 3.9  Shows the public variable declarations in the ZRLoadTransfer class .... 65
Table 3.10 Shows the private variable declarations in the ZRCRPTask class .......... 65
Table 3.11 Shows the public variable declarations in the ZRCRPTask class ......... 66
Table 3.12 The public function declarations in the ZRCRPTask class ............... 66
LIST OF THE FIGURES

Figure 2.1 The workflow of the CRP module .......................................................... 23
Figure 2.2 The calculation of EST and LFT ............................................................. 31
Figure 2.3 An example of the load ........................................................................... 35
Figure 2.4 The load of the each measure unit .......................................................... 38
Figure 2.5 The overload 20(Hr) is transferred to measure unit 4 ............................. 39
Figure 2.6 The overload 40(Hr) is transferred to the measure unit 3 ..................... 40
Figure 2.7 The overload 20(Hr) is transferred to the measure unit 1 ..................... 41
Figure 2.8 The load of the each measure unit .......................................................... 42
Figure 2.9 The load transference until the measure unit 2 ..................................... 42
Figure 2.10 Transfer the entire load of working procedure B to the measure unit 1 . 43
Figure 2.11 Transfer the overload to the measure unit 1 ....................................... 44
Figure 3.1 The UML of the classes ........................................................................... 58
Figure 4.1 Login window of the CRP module ......................................................... 69
Figure 4.2 The main frame of this CRP module ....................................................... 70
Figure 4.3 The menu items of the manual adjust operations ................................. 71
Figure 4.4 The order transference operation ............................................................ 71
Figure 4.5 The graphic output of the load ............................................................... 72
LIST OF ABBREVIATIONS

Enterprise Resource Planning  ERP
Materials Requirements Planning  MRP
Capacity Requirements Planning  CRP
Finite Capacity Scheduling  FCS
Object-Oriented  OO
Bills of Materials  BOM
Management Information Systems  MIS
Work In Progress  WIP
Master Production Schedule  MPS
Just in Time  JIT
Advanced Planning and Scheduling  APS
Graphical User Interface  GUI
Relational Database Management System  RDBMS
Fourth-Generation Language  4GL
Computer-Aided Software Engineering  CASE
Supply Chain Management  SCM
Customer Relationship Management  CRM
Product Data Management  PDM
Manufacturing Executions Systems  MES
Latest Finish Time  LFT
Earliest Start Time  EST
Unified Modeling Language  UML
INTRODUCTION

The CRP module discussed in this paper is the sub-module of the ERP system ZRERP version 1.1 developed by ZhuRi Software Co., Ltd. The version 1.0 of ZRERP is delivered at October 2000. But it did not include the CRP module. According to the user's feedback, version 1.0 is not completely fit the requirement of the modern production. An overused work center creates obvious problems: backup and delays, unanticipated and costly overtime, and loss of quality due to production pressures. After analyzing the load across work centers, complex functions analyze the MRP schedule and compare it against the current capacity of each work center. Using these inquiries, ZRERP version 1.1 is decided to add the CRP module. This is the motivation of my thesis.

This CRP module uses the output of the MRP system as the input data. It will deploy the manufacturing order into the manufacturing process, then to calculate the operation time of the process. This module can process two kinds of the plan unit for the load calculation: work centre or work machine. The measure unit of the plan can be day or week. When we calculate the load of the plan unit, we also use the planned orders outputted by the posterior plan function (Finite Capacity Scheduling –FCS) with the manufacturing orders, so that this load calculation is more accurate. After adding the load to the plan unit, we can the auto-balance operation or the manual adjust operation to verify the feasibility of
the plan of the tentative orders. If the plan is feasible, the module will output the manufacturing orders and produce the job orders. If the plan is not feasible, it will output the useful information to the anterior function to review the tentative orders.

In ZRERP version 1.0, the database design is completed. In my papers, I don’t care with this issue.

Through this CRP module is the sub-module of the ZRERP system, it can also be run independently. Following are the base knowledge and software tools used in my thesis.

1. The Development Environment

The hardware environment is:

Personal Computer with the frequency of 2G Hz, 512MB Memory, 80GB hard disk.

The Software environment is:

Windows 2K/XP, JBuilder 9.0, Oracle 8i.

2. Object-oriented method and approach

The object-oriented ("OO", for short) concept and approach have been used in many areas and for a myriad of applications, including software engineering, to name but one. OO approaches are now specifically considered as a useful alternative to the traditional approaches. The traditional approach models scheduling problems from two different
points of view, namely functional decomposition and related information (data-oriented). The OO approach, however, unifies functional decomposition with related information. Its principal aim is to blend together the functional approach and the data approach through the use of messages between objects; to divide real-world entities into classes and objects, et cetera; to represent classes and objects and their relationships and to describe the interconnectedness between abstraction and reality. The main advantages of the OO approach are realism, flexibility, re-usability and extensibility.

In this thesis, the design of the software is based on the principles of Object Oriented Programming (OOP). This allows for the development and testing of various parts of the code to be done independently. OOP also allows the code to be easily extended.

3. Java

Java is designed to meet the challenges of application development in the context of heterogeneous, network-wide distributed environments. Paramount among these challenges is secure delivery of applications that consume the minimum of system resources, can run on any hardware and software platform, and can be extended dynamically.

Java originated as part of a research project to develop advanced software for a wide variety of networked devices and embedded systems. The goal was to develop a small, reliable, portable, distributed, real-time operating environment. When the project started,
C++ was the language of choice. But over time the difficulties encountered with C++ grew to the point where the problems could best be addressed by creating an entirely new language environment. Design and architecture decisions drew from a variety of languages such as Eiffel, SmallTalk, Objective C, and Cedar/Mesa. The result is a language environment that has proven ideal for developing secure, distributed, network-based end-user applications in environments ranging from networked-embedded devices to the World-Wide Web and the desktop.

The Java system that emerged to meet these needs is simple, so it can be easily programmed by most developers; familiar, so that current developers can easily learn Java; object oriented, to take advantage of modern software development methodologies and to fit into distributed client-server applications; multithreaded, for high performance in applications that need to perform multiple concurrent activities, such as multimedia; and interpreted, for maximum portability and dynamic capabilities

4. Oracle

An Oracle database is a collection of data treated as a unit. The purpose of a database is to store and retrieve related information. A database server is the key to solving the problems of information management. In general, a server reliably manages a large amount of data in a multiuser environment so that many users can concurrently access the same data. All this is accomplished while delivering high performance. A database server also
prevents unauthorized access and provides efficient solutions for failure recovery.

Oracle Database is the first database designed for enterprise grid computing, the most flexible and cost effective way to manage information and applications. Enterprise grid computing creates large pools of industry-standard, modular storage and servers. With this architecture, each new system can be rapidly provisioned from the pool of components. There is no need for peak workloads, because capacity can be easily added or reallocated from the resource pools as needed.

The database has logical structures and physical structures. Because the physical and logical structures are separate, the physical storage of data can be managed without affecting the access to logical storage structures.

5. **JBuilder**

JBuilder is a programming compilation tool dedicated to Java which through it's extensive Component Palette contains many of the pre-coded classes emanating from the original Java Language but in a "drag and drop format". Unlike some other Visual Programming tools, not only does it support the reuse of library classes, it retains the ability, at the lowest level, to be amended by the introduction of Java source code as a means of constructing bespoke packages.
6. Structure of the thesis

Because the CRP part is a sub-module of the ERP system, knowing the history and evolution of ERP is essential to understanding its current application and its future developments. For this reason, Chapter 1 describes the evolution of ERP. Chapter 2 will discuss the requirement analysis and system design. In the chapter 3, I describe the most important classes of this module. The implementation of this module is shortly described in chapter 4 then followed by conclusion. The Appendices contain source code of some important classes developed for this thesis.
CHAPTER 1
REVIEW THE EVOLUTION OF ERP

Integrated enterprise resource planning (ERP) software solutions have become synonymous with competitive advantage, particularly throughout the 1990's. ERP systems integrate all traditional enterprise management functions like financials, human resources, and manufacturing & logistics. Knowing the history and evolution of ERP is essential to completing my thesis.

The history of ERP can be traced back to the first inventory control (IC) and manufacturing management applications of 1960s (Chung and Snyder, 1999, Gumaer, 1996). These first applications for the manufacturing were generally limited to IC and purchasing, which was due to the origins of these applications in the accounting software (Gumaer, 1996). The accounting, with its definition based around generally accepted standards, had been one of the first business functions to be computerized and the first applications for the manufacturing were created as by-products of accounting software driven by the desire of the accountants to know the value of the inventory (Gumaer, 1996). IC refers to the effort of maintaining inventory levels and costs within acceptable limits but includes also models for determining how much inventory to order and when to order as well as systems for monitoring inventory levels for management evaluation and decision making (Vonderembse and White, 1996, pp. 751-752). IC applications were the starting
point in the evolution process that led to the development of modern ERP applications (Kumar and Hillegersberg, 2000).

The next stage in the evolution of ERP following the IC and manufacturing management applications was the introduction of the concept of Material Requirements Planning (MRP) (Kumar and Hillegersberg, 2000). The concept of MRP, first introduced by Orlicky (1975), is based on an idea of a process that uses Bills of Materials (BOM), inventory records and the master schedule to determine when orders must be released to replenish inventories of parts or raw materials (Vonderembse and White, 1996, pp. 567). MRP system can be defined as a collection of logical procedures for managing, at the most detailed level, inventories of component assemblies, parts and raw materials in manufacturing environment and as an information system and simulation tool that generates proposals for production schedules that managers can evaluate in terms of their feasibility and cost effectiveness (Gass and Harris, 1996, pp. 380). MRP applications were introduced as a scheduling, priority and capacity management systems for the use of plant managers and their supervisory staff (Chung and Snyder, 1999) and typically included features for demand-based planning and algorithms for consumption-based planning (Klaus et al., 2000). The main benefits that enterprises sought with the implementation of MRP applications were the reduction of inventories, lead times, and costs and improvement of market responsiveness, control, organizational communication (Light et al., 2000) and customer service (Chung and Snyder, 1999).

During the 1970s, MRP packages were extended with further applications in order to
offer complete support for the entire production planning and control cycle (Klaus et al., 2000). This led to the next stage in the evolution of ERP, which was the introduction of the concept of Manufacturing Resource Planning (MRPII). The concept of MRPII, introduced by Wight (1984), emerged as a logical consequence of the development in earlier approaches to material control (Yusuf and Little, 1998). MRPII seeks to improve the efficiency of manufacturing enterprises through integration of the application of information and manufacturing technologies (Chung and Snyder, 1999). MRPII is an integrated decision support system that ties together departments such as engineering, finance, personnel, manufacturing and marketing via a computer-based dynamic simulation model, which works within the limits of an organization’s present production system and with known orders and demand forecast (Vonderembse and White, 1996, pp. 67).

The mainstream of the literature on the evolution of ERP, however, regard ERP as an extension of MRPII with enhanced and added functionality (Yusuf and Little, 1998, Gumaer, 1996, Kumar and Hillegersberg, 2000), encompassing functions that are not within the traditional focus of MRPII, such as human resource planning, decision support, supply chain management, maintenance support, quality, regulatory control, and health and safety compliance (Yusuf and Little, 1998). In the age of customized products and services, long-term forecasts are much less useful and production and distribution far too dynamic and unpredictable to be addressed solely through periodic planning approach of MRPII. However, despite of these shortcomings of MRPII, most ERP vendors still use the same basic model of MRPII for the manufacturing-planning portion of their systems (Gumaer, 1996).
During the last three years, the functional perimeter of ERP systems began an expansion into its adjacent markets, such as supply chain management (SCM), customer relationship management (CRM), product data management (PDM), manufacturing executions systems (MES), business intelligence/data warehousing, and e-Business. The major ERP vendors have been busy developing, acquiring, or bundling new functionality so that their packages go beyond the traditional realms of finance, materials planning, and human resources.

To circumvent MRPII's capacity planning limitations, planners turned to various ways of off-line capacity planning: either manually, with the help of spreadsheet programs, or with the help of new advanced planning and scheduling (APS) systems. APS systems are designed as bolt-ons with the idea of plugging into an ERP system's database to download information and then create a feasible schedule within identified constraints. The new schedule can then be uploaded into the ERP system thereby replacing the original MRP results. These APS systems typically offer simulation ("what if") capabilities that allow the planner to analyze the results of an action before committing to that action through the ERP system. Some of these systems go one step further by offering optimization capabilities. They automatically create multiple simulations and recommend changes in the supply chain within the existing constraints.
CHAPTER 2
THE REQUIREMENT ANALYSIS AND SYSTEM DESIGN

This CRP module uses the output of the MRP system as the input data. It will deploy the manufacturing order into the manufacturing process, then to calculate the operation time of the process. This module can dispose two kinds of the plan unit for the load calculation: work centre or work machine. The measure unit of the plan can be day or week. When we calculate the load of the plan unit, we also use the planned orders outputted by the posterior plan function (Finite Capacity Scheduling –FCS) together with the manufacturing orders, so that this load calculation is more accurate. After adding the load to the plan unit, we can do the auto-balance operation or the manual adjust operation to verify the feasibility of the plan of the tentative orders. If the plan is feasible, the module will output the manufacturing orders and produce the job orders. If the plan is not feasible, it will output the useful information to the anterior function to review the tentative orders.

The following figure 2.1 shows the entire work flow the CRP module.
Figure 2.1 The workflow of the CRP module.

MRP Function

Manufacturing Data

Table

Login

Data Input

Calculate Load

Load Balance

Manual Adjust

Check feasibility

Output the non-executable information

Non-executable Data

FCS Function

Manufacturing Data

Workcenter Schedule

Output the result

Logout
As showing in the figure 2.1, the CRP module is composed of 9 parts:

1. Login function
2. Data input function
3. Load calculation function
4. Load auto-balance function
5. Manual adjust function
6. Check the feasibility function
7. Output the non-executable information function
8. Data output function
9. Logout function

2.1 LOGIN FUNCTION

User must input user’s ID and password for login the CRP function.

2.1.1 Input

Table 2.1 The following table shows the login input items.

<table>
<thead>
<tr>
<th>Input Item</th>
<th>Explain</th>
</tr>
</thead>
<tbody>
<tr>
<td>User’s ID</td>
<td>The ID of the login user</td>
</tr>
<tr>
<td>Password</td>
<td>The password of the login user</td>
</tr>
<tr>
<td>Login mode</td>
<td>The mode of user’s login</td>
</tr>
<tr>
<td>Cipher File</td>
<td>The list of user and password</td>
</tr>
<tr>
<td>Plant table</td>
<td>According to the range of charge table, set the exclusive status in this table.</td>
</tr>
<tr>
<td>Workcenter table</td>
<td>According to the range of charge table, set the exclusive status in this table.</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Charge table</td>
<td>Get the responsible range of work center from this table.</td>
</tr>
<tr>
<td>Data source type</td>
<td>Specify the type of the data source</td>
</tr>
</tbody>
</table>

2.1.2 Output

Table 2.2 The following table shows the login output message.

<table>
<thead>
<tr>
<th>Output Item</th>
<th>Explain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Login success or not</td>
<td>Return the result of the login.</td>
</tr>
<tr>
<td>Type of the error</td>
<td>The number of the login error.</td>
</tr>
</tbody>
</table>

2.1.3 Detail Process

The process logic of the login function is described as:

The fields of user ID and password must be inputted.

Reading in the Cipher file, verify the user ID and password being correct or not.

If the user ID or password is not correct, the error occurs, return the type of error.

If the user ID and password are correct, then reading in the facility table and work center table and charge table from the specified data source.

From the charge table, get all responsible work centers of the user.

Verify the responsible work centers being used by other user. If it is the case, return an error.

If there is no the responsible work center being used by other user, set the status of the work center to exclusive.
According to the user's login mode, we can restrict the CRP functions which user can be used. The following table shows this limit.

Table 2.3 Login mode

<table>
<thead>
<tr>
<th>Login mode</th>
<th>Usable Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference</td>
<td>Data Output Function.</td>
</tr>
</tbody>
</table>

### 2.2 DATA INPUT FUNCTION

This module reads the manufacturing orders produced by MRP from the data source into memory. At the same time, it completes the same operation with the part table, work center table, Calendar table etc.

#### 2.2.1 Input

Table 2.4 Input items of the data input function

<table>
<thead>
<tr>
<th>Input Item</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manufacturing Order Table</td>
<td>Read into memory</td>
</tr>
<tr>
<td>Manufacturing Divided Order</td>
<td>Read into memory</td>
</tr>
</tbody>
</table>
Table 2.5 Output items of the data input function

<table>
<thead>
<tr>
<th>Output Item</th>
<th>Explain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data input success or not</td>
<td>Return the result of the Data input.</td>
</tr>
<tr>
<td>Type of the error</td>
<td>The number of the Data Input error.</td>
</tr>
</tbody>
</table>

2.2.3 Detail Process

Because the work center table and the responsible work center table are already read into memory when user logged into this module, all the rest tables are got into memory from the specified data source. Verifying the correctness of the tables must be processed at the time of reading of the tables. Database is as the default data source, if the user wants to read data from the other data source, it must be specified in the parameter file.
2.3 CALCULATE LOAD FUNCTION

The load calculation of the measure unit is taken by using the data read from the data source. There are two kinds of the work area for the load calculation: the work center and the work machine. We use the manufacturing time of each process deployed by the manufacturing order to calculate the workload. This function also uses the information of the process of the planned order in the posterior function (FCS) in order to achieve the high accurate calculation of the load.

2.3.1 Input

Table 2.6 Input items of the calculate load function

<table>
<thead>
<tr>
<th>Input Item</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manufacturing order table</td>
<td>Used for deploying the manufacturing process</td>
</tr>
<tr>
<td>Process sequence table</td>
<td>The object of the load</td>
</tr>
<tr>
<td>Work center table</td>
<td>Used for getting the calculational methods of Job Time</td>
</tr>
<tr>
<td>Producible work center table</td>
<td>Used for calculating the Job Time</td>
</tr>
<tr>
<td>Calendar table</td>
<td>Used for determining the unit of measure of the load object</td>
</tr>
<tr>
<td>Unit of measure</td>
<td>Used for calculating the load</td>
</tr>
<tr>
<td>Base time point of the planning</td>
<td>Used for calculating the load</td>
</tr>
<tr>
<td>The period of the planning</td>
<td>Used for calculating the load</td>
</tr>
</tbody>
</table>

2.3.2 Output

Table 2.7 Output items of the calculate load function

<table>
<thead>
<tr>
<th>Output Item</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information of the calculated load and the report of the load</td>
<td>The load information of every unit of the measure (each work center or each</td>
</tr>
</tbody>
</table>
The result of this function

<table>
<thead>
<tr>
<th>The type of the error</th>
<th>Return the number of the error to indicate its type.</th>
</tr>
</thead>
</table>

2.3.3 Detail process

First of all, we check the input manufacturing orders whether they are the object to calculate the load or not. If they are the object to calculate, then deploy the process of this manufacturing order and calculate the job time of each process according to the unit of measure. We assume that infinite capacity exists at each of these work centers to satisfy this calculated. We use the processes that are determined by FCS function for the high precision load calculation.

2.3.3.1 Verification of the manufacturing order

The LFT (Latest Finish Time) of the manufacturing order must be later than the base time point of the planning. The EST (Earliest Start Time) of the manufacturing order must be in the period time of the planning. The table of 2.8 shows the detail.

Table 2.8 The object to calculate load

<table>
<thead>
<tr>
<th>The range of the manufacturing order</th>
<th>The object to calculate load</th>
</tr>
</thead>
<tbody>
<tr>
<td>The LFT (Latest Finish Time) of the manufacturing order &lt; the base time point of the planning</td>
<td>No</td>
</tr>
</tbody>
</table>
The base time point of the planning \( \leq \) the EST (Earliest Start Time) of the manufacturing order \( < \) (the base time point of the planning \( + \) the period of the planning) \( \leq \) the EST (Earliest Start Time) of the manufacturing order

<table>
<thead>
<tr>
<th>The base time point of the planning ( + ) the period of the planning ( \leq ) the EST (Earliest Start Time) of the manufacturing order</th>
<th>Yes</th>
</tr>
</thead>
</table>

2.3.3.2 Deploying the manufacturing process

By using the process sequence table, the manufacturing process is deployed from the manufacturing order of the component. The EST and LFT of each process are generated according to the production time of the process. The following tables figure out the method of this calculation.

Table 2.9 The calculation of LFT (Backward scheduling logic)

<table>
<thead>
<tr>
<th>The manufacturing process</th>
<th>The calculation of LFT</th>
</tr>
</thead>
<tbody>
<tr>
<td>The last manufacturing process</td>
<td>The LFT of the last process</td>
</tr>
<tr>
<td>The rest manufacturing process</td>
<td>The LFT of the rest process is earliest time among all the subtraction of LFT with its production time (LFT - production time)</td>
</tr>
</tbody>
</table>

Table 2.10 The calculation of EST (Backward scheduling logic)

<table>
<thead>
<tr>
<th>The manufacturing process</th>
<th>The calculation of EST</th>
</tr>
</thead>
<tbody>
<tr>
<td>The earliest manufacturing process</td>
<td>The earliest start time of the first manufacturing process.</td>
</tr>
<tr>
<td>The rest manufacturing process</td>
<td>The EST of the rest manufacturing process is latest time among all the subtraction of EST with its production time (EST - production time)</td>
</tr>
</tbody>
</table>

The concrete demonstration of the calculation of EST and LFT is shown in figure 2.3.

In this figure PT means Production Time.
Figure 2.2 The calculation of EST and LFT.

The manufacturing processes of a component X

- manufacturing process A
- manufacturing process B
- manufacturing process C
- manufacturing process D
- manufacturing process E

The Calculation of EST and LFT of the component X

<table>
<thead>
<tr>
<th>EST of the component X</th>
<th>The component X</th>
<th>LFT of the component X</th>
</tr>
</thead>
<tbody>
<tr>
<td>EST of A</td>
<td>PT (A)</td>
<td>PT (A)</td>
</tr>
<tr>
<td>EST of B</td>
<td>PT (B)</td>
<td>PT (B)</td>
</tr>
<tr>
<td>EST of C</td>
<td>PT (C)</td>
<td>PT (C)</td>
</tr>
<tr>
<td>EST of D</td>
<td>PT (D)</td>
<td>PT (D)</td>
</tr>
<tr>
<td>EST of E</td>
<td>PT (E)</td>
<td>PT (E)</td>
</tr>
</tbody>
</table>
2.3.3.3 The calculation of production time of the manufacturing process

The production time of the manufacturing process equals the sum of the product of rated production time of the process and the mount of components and the prepare time. The formulation is:

\[ \text{The production time} = (\text{rated production time of the process}) \times (\text{the mount of components}) + \text{the prepare time}. \]

2.3.3.4 The different capacities and loads

The object to be calculated the load can be work center or work machine of the manufacturing area, at the same time, the measure unit of the load can be hour or day, this can be set in the parameter file. In the case of work center, the calculation of load is completed according to the measure unit defined in the process sequence table. In the case of work machine, the rules of selection the work machine to load must be followed are:

Select the machine in the top-priority work center.

In the same work center, select the top-priority machine. If it is overload, then select the next top-priority machine.

If all machines of the center are overload, then select the top-priority machine.
2.3.3.5 The calculation of the standard capacity

Using the following method to calculate the standard capacity of the different machines:

The standard capacity of the current measure unit = The sum of production time of every day according to the measure unit.

The production time of the current day = (∑(the shift production time of the current day of the machine) + append production time) * capacity coefficient

The standard capacity of the different work centers is shown as the formula:

The standard capacity of the current measure unit = ∑(the standard capacities of all machines that are belong to the current center).

2.3.3.6 The discrimination of manufacturing area

If the object to be calculated the load is work center, the load of different machines can not be identified. Whereas in the case of the calculation load of the machine, the load of every work center can be discriminated, because the load of the center equals to the sum of the loads of all machines that are belong to the current center.
2.3.3.7 Getting the information from the function of the posterior planning (FCS)

This calculation load function also gets the information from the function of the posterior planning (FCS). It uses the confirmed production orders of the posterior planning to calculate the load so that this calculation is more accurate. The table 2.11 shows the detail.

Table 2.11 The information introduced from the posterior planning (FCS)

<table>
<thead>
<tr>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>The ID of process sequence of the current process</td>
</tr>
<tr>
<td>The ID of manufacturing order which includes the current process</td>
</tr>
<tr>
<td>The start time of the process</td>
</tr>
<tr>
<td>The end time of the process</td>
</tr>
<tr>
<td>The ID of the machine of the current process</td>
</tr>
</tbody>
</table>

The ID of planned order introduced from the function of the posterior planning must be same as the ID of current manufacturing order. The ID of process sequence of the planned order must be the same as the ID of process sequence of the current manufacturing process deployed from the current manufacturing order. The following table shows the information:

Table 2.12 The information deployed from the planned order

<table>
<thead>
<tr>
<th>Information of the process</th>
<th>Setting information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Earliest start time</td>
<td>The start time of the process of the planned order</td>
</tr>
<tr>
<td>Latest finish time</td>
<td>The finish time of the process of the planned order</td>
</tr>
</tbody>
</table>
Production time | The finish time of the process of the planned order - the start time of the process of the planned order
---|---
The object to calculate the load | The machine which its ID is uniform to the ID of the machine of the current manufacturing order. Or the work center that the machine of the planned order belongs to. The ID of the machine is uniform to the ID of the machine of the current manufacturing order

2.3.3.8 The example of the loads

The figure 2.3 shows an example of the loads. Here MP means Manufacturing Process.

Figure 2.3 An example of the load.
2.4 AUTO-BALANCE FUNCTION

For the calculated load, the auto-balance function of the load is processed at the base of the standard capacity. The result of the adjustment of the manufacturing process of this auto-balance function can not be as the output of the CRP module.

2.4.1 Input

Table 2.13 The input items of the Auto-balance function.

<table>
<thead>
<tr>
<th>Input Item</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>The information of Calculate Load Function</td>
<td>Load information of the work center</td>
</tr>
<tr>
<td>Work Center Table</td>
<td>To get the calculate measure of the job time</td>
</tr>
</tbody>
</table>

2.4.2 Output

Table 2.14 The output items of the Auto-balance function.

<table>
<thead>
<tr>
<th>Output Item</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information of the calculated load and the report of the load</td>
<td>The load information after auto-balance function</td>
</tr>
<tr>
<td>The result of this function</td>
<td>If this function runs successfully, return success, else return an error.</td>
</tr>
<tr>
<td>The type of the error</td>
<td>Return the number of the error to indicate its type.</td>
</tr>
</tbody>
</table>

2.4.3 Detail process

This auto-balance load function will process and reschedule all open and planned
manufacturing orders using backward scheduling. Backward scheduling logic will calculate each operation backwards from the manufacturing order or planned order.

The algorithm of the auto-balance load function we used is simple. In the case of calculation of work center, the auto-balance load function is processed with the work center as measure unit. Whereas in the case of calculation of work machine, the auto-balance load function is processed with the work machine as measure unit. The loads can only be transferred between the machines in the same work center. For the complicate auto-load function, such as the transference of loads between machines in different work centers, we don't take into account.

The detail algorithm of auto-balance load function is illustrated by two examples shown in Figure 2.4-2.7.

These two examples are based on the assumption that the sequence of EST of every manufacturing process is the same as the sequence of the English letters used to express the process. So the EST of the manufacturing process A is the earliest, and that the EST of the process H is the latest. Moreover, The EST of each manufacturing process is earlier than the base point of the planning.

Example 1 of the Auto-balance, here MP means Manufacturing Process.
Figure 2.4 The load of the each measure unit.

As showing in the figure, because the capacity of the measure unit is over load compared with the standard capacity, it must be taken the auto-balance process of the load. Taking the latest measure unit 5 as the first object to be treated with, the over load 20(Hours) of the manufacturing process of the measure unit 4, as showing in the following figure.
Figure 2.5 The overload 20(Hr) is transferred to measure unit 4.

After this transference to the measure unit 4, the loads of the measure unit 5 is under the standard capacity, but the load of the measure unit 4 is over the standard capacity. So the transference of load begins from the earliest EFT of the manufacturing process F, the over load of 40 Hours moves ahead. This is shown in the following figure.
Figure 2.6 The overload 40(Hr) is transferred to the measure unit 3.

<table>
<thead>
<tr>
<th>Standard Capacity</th>
<th>Measure Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP(A) (40Hr)</td>
<td>ME (3)</td>
</tr>
<tr>
<td>MP(B) (40Hr)</td>
<td>ME (2)</td>
</tr>
<tr>
<td>MP(C) (60Hr)</td>
<td>ME (1)</td>
</tr>
<tr>
<td>MP(D) (40Hr)</td>
<td></td>
</tr>
<tr>
<td>MP(E) (40Hr)</td>
<td></td>
</tr>
<tr>
<td>MP(F) (40Hr)</td>
<td></td>
</tr>
<tr>
<td>MP(G) (60Hr)</td>
<td></td>
</tr>
<tr>
<td>MP(H) (20Hr)</td>
<td></td>
</tr>
<tr>
<td>MP(I) (100Hr)</td>
<td></td>
</tr>
</tbody>
</table>

Because the load of the measure unit 3 is not over the standard capacity, the measure unit 2 is treated directly. The over load of the measure unit 2 is 20 hours. So the transference of load begins from the earliest EFT of the manufacturing process B. Figure shows this case.
Figure 2.7 The overload 20(Hr) is transferred to the measure unit 1.

There is not measure unit ahead of the measure unit 1, so the auto-balance process finish.

Because the load of the measure unit 1 is under the standard capacity, so the planning of this work center is executable.

Another example, the case of the planning is not executable.
The procedure of auto-balance of the load in this example is similar with that is in the example 1 until the measure unit 2.
Because the load of the measure unit 2 is over the standard capacity, the transference of load begins from the earliest EFT of the manufacturing process B. In this case, the load of the manufacturing process B is less than the over part of the load of measure unit 2. So the total load of the manufacturing process B is transferred to the measure unit 1. The figure 2.10 shows this case.

Figure 2.10 Transfer the entire load of manufacturing process B to the measure unit 1.

The load of the measure unit 2 is still over load even though the entire load of manufacturing process B is transferred to the measure unit 1. So the over load 20 Hours of
the manufacturing process C must be moved to the measure unit 1.

Figure 2.11 Transfer the overload to the measure unit 1.

There is no measure unit ahead of the measure unit 1, so the auto-balance of the load finishes. Because the load of the measure unit 1 is over the standard capacity, in this case the planning of the work center is non-executable.
2.5 MANUAL ADJUST FUNCTION

It is required to offer the means of manual adjust function after the load calculation. The conceivable operations of the manual adjust function are listed in the following table.

Table 2.15 The operations of the manual adjust function.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transfer</td>
<td>Transfer the load from one manufacturing process to an other.</td>
</tr>
<tr>
<td>Divide</td>
<td>Divide one manufacturing order into two orders.</td>
</tr>
<tr>
<td>Combine</td>
<td>Combine two divided orders to one manufacturing order</td>
</tr>
<tr>
<td>Append</td>
<td>Append manufacturing orders. When the manufacturing orders are appended, the validity of MRP is not guaranteed.</td>
</tr>
<tr>
<td>Delete</td>
<td>Remove the manufacturing orders. When the manufacturing orders are deleted, the validity of MRP is not guaranteed.</td>
</tr>
<tr>
<td>Modify the quantity of the product</td>
<td>Modify the quantity of the product of the manufacturing order. When this amount is modified, the validity of MRP is not guaranteed.</td>
</tr>
<tr>
<td>Modify the schedule and the shift type</td>
<td>Modify the day-off or the overtime of the facility to adjust the standard capacity of the work center, so that the load is under the standard capacity.</td>
</tr>
<tr>
<td>Undo</td>
<td>Recover the current manual operation.</td>
</tr>
</tbody>
</table>

2.5.1 The transference of the load

The specified load of the manufacturing process is transferred to the appointed plan unit or work center. There are two kinds of the transference.
2.5.1.1 The transference of the load between the work centers

The possible moveable work centers are only those which are corresponding to the current work sequences defined in the producible work center table. This transference can not be performed to the work centers of the other facility. If the moveable status of the facility is set to false, an error will occur.

2.5.1.2 The transference of the load between the work machines

The load calculation is performed using the work machine unit. The possible moveable work machines are only those which are corresponding to the current manufacturing process defined in the producible work center table. This transference can not be performed to the work machines of the work center of the other facility. If the moveable status of the machine is set to false, an error will occur.

2.5.1.3 Input

Table 2.16 The input items of the transference of the load.

<table>
<thead>
<tr>
<th>Input Item</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID of the manufacturing order</td>
<td>The ID of the manufacturing order which includes the manufacturing process, its load is transferred.</td>
</tr>
<tr>
<td>ID of the divided manufacturing order</td>
<td>The ID of the divided manufacturing order which includes the manufacturing process, its load is transferred.</td>
</tr>
<tr>
<td>ID of the manufacturing process</td>
<td>The ID of the manufacturing process, its load is transferred.</td>
</tr>
<tr>
<td>ID of work center or machine</td>
<td>The ID of the work center or machine to be transferred.</td>
</tr>
<tr>
<td>Date</td>
<td>The date of manufacturing process to be transferred.</td>
</tr>
</tbody>
</table>
2.5.1.4 Output

Table 2.17 The output items of the transference of the load.

<table>
<thead>
<tr>
<th>Output Item</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID of manufacturing process</td>
<td>The ID of the manufacturing process after the transference.</td>
</tr>
<tr>
<td>The load information</td>
<td>The load information of the work center or machine after the transference.</td>
</tr>
<tr>
<td>The result of this operation</td>
<td>If this function runs successfully, return success, else return an error.</td>
</tr>
<tr>
<td>The type of the error</td>
<td>Return the number of the error to indicate its type.</td>
</tr>
</tbody>
</table>

2.5.2 The division of the manufacturing order

The manufacturing order is divided by the specified quantity of the order. At the same time as the manufacturing order is divided, all the manufacturing processes deployed from the current manufacturing order are divided the same quantity. The specified quantity the order must be ranging between $0 < \text{divided quantity} < \text{quantity of the order before division}$.

After the division, the information of the new manufacturing order is shown in the table.
Table 2.18 The information of the new manufacturing order of the division of the manufacturing order.

<table>
<thead>
<tr>
<th>Items</th>
<th>The manufacturing order 1 after division</th>
<th>The manufacturing order 2 after division</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID of the manufacturing order</td>
<td>Same as the manufacturing order before division</td>
<td>Same as the manufacturing order before division</td>
</tr>
<tr>
<td>ID of the divided manufacturing order</td>
<td>Same as the manufacturing order before division</td>
<td>The minimum value not used of the divided manufacturing order for the same manufacturing order ID.</td>
</tr>
<tr>
<td>ID of the component</td>
<td>Same as the manufacturing order before division</td>
<td>Same as the manufacturing order before division</td>
</tr>
<tr>
<td>ID of the work center</td>
<td>Same as the manufacturing order before division</td>
<td>Same as the manufacturing order before division</td>
</tr>
<tr>
<td>EST</td>
<td>Same as the manufacturing order before division</td>
<td>Same as the manufacturing order before division</td>
</tr>
<tr>
<td>LFT</td>
<td>Same as the manufacturing order before division</td>
<td>Same as the manufacturing order before division</td>
</tr>
<tr>
<td>Quantity</td>
<td>The specified quantity</td>
<td>The quantity before division – quantity.</td>
</tr>
</tbody>
</table>

2.5.2.1 Input

Table 2.19 The input items of the division of the manufacturing order.

<table>
<thead>
<tr>
<th>Input Item</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID of the manufacturing order</td>
<td>The ID of the manufacturing order to be divided.</td>
</tr>
<tr>
<td>ID of the divided manufacturing order</td>
<td>The ID of the divided manufacturing order to be divided.</td>
</tr>
<tr>
<td>Quantity</td>
<td>The quantity of the manufacturing order after division.</td>
</tr>
</tbody>
</table>
2.5.2.2 Output

Table 2.20 The output items of the division of the manufacturing order.

<table>
<thead>
<tr>
<th>Output Item</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manufacturing order 1</td>
<td>The manufacturing order 1 after division</td>
</tr>
<tr>
<td>Manufacturing order 2</td>
<td>The manufacturing order 2 after division</td>
</tr>
<tr>
<td>The load information</td>
<td>The load information of the work center or machine after the division.</td>
</tr>
<tr>
<td>The result of this operation</td>
<td>If this function runs successfully, return success, else return an error.</td>
</tr>
<tr>
<td>The type of the error</td>
<td>Return the number of the error to indicate its type.</td>
</tr>
</tbody>
</table>

2.5.3 The Combination of the manufacturing order

The combination of the manufacturing order is performed by two specified manufacturing orders. After combination these two manufacturing orders, all the manufacturing processes deployed from these two manufacturing orders must be combined together. The quantity of the manufacturing order after combination is the sum of two combined manufacturing orders. The ID of the divided manufacturing order is the minor value of these two IDs of the divided manufacturing orders. The other information except the quantity and the ID of the divided manufacturing order will inherit the same information of the manufacturing order. The combination can only be performed between the divided manufacturing orders which have the same manufacturing order ID.
2.5.3.1 Input

Table 2.21 The input items of the combination of the manufacturing order.

<table>
<thead>
<tr>
<th>Input Item</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID of the manufacturing order 1</td>
<td>The ID of the manufacturing order 1 to be combined.</td>
</tr>
<tr>
<td>ID of the divided manufacturing order 1</td>
<td>The ID of the divided manufacturing order 1 to be combined.</td>
</tr>
<tr>
<td>ID of the manufacturing order 2</td>
<td>The ID of the manufacturing order 2 to be combined.</td>
</tr>
<tr>
<td>ID of the divided manufacturing order 2</td>
<td>The ID of the divided manufacturing order 2 to be combined.</td>
</tr>
</tbody>
</table>

2.5.3.2 Output

Table 2.22 The output items of the combination of the manufacturing order.

<table>
<thead>
<tr>
<th>Output Item</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manufacturing order</td>
<td>The manufacturing order after the combination.</td>
</tr>
<tr>
<td>The load information</td>
<td>The load information of the work center or machine after the division.</td>
</tr>
<tr>
<td>The result of this operation</td>
<td>If this function runs successfully, return success, else return an error.</td>
</tr>
<tr>
<td>The type of the error</td>
<td>Return the number of the error to indicate its type.</td>
</tr>
</tbody>
</table>

2.5.4 The accession of the manufacturing order

The accession of the new manufacturing order is based on the specified information of the manufacturing order. After the new manufacturing order is created, all the processes of the manufacturing order are generated. At the same time, the load calculation of the manufacturing process is also performed.
2.5.4.1 Input

Table 2.23 The input items of the accession of the manufacturing order.

<table>
<thead>
<tr>
<th>Input Item</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>All the necessary information of new manufacturing order</td>
<td>Input the necessary information for creating the new manufacturing order.</td>
</tr>
</tbody>
</table>

2.5.4.2 Output

Table 2.24 The output items of the accession of the manufacturing order.

<table>
<thead>
<tr>
<th>Output Item</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manufacturing order</td>
<td>The new manufacturing order after the accession.</td>
</tr>
<tr>
<td>The load information</td>
<td>The load information of the work center or machine after the accession.</td>
</tr>
<tr>
<td>The result of this operation</td>
<td>If this function runs successfully, return success, else return an error.</td>
</tr>
<tr>
<td>The type of the error</td>
<td>Return the number of the error to indicate its type.</td>
</tr>
</tbody>
</table>

2.5.5 The deletion of the manufacturing order

Deleting a manufacturing order is performed by specified its ID, at the same time, all manufacturing processes deployed by the current manufacturing order are also removed.

2.5.5.1 Input

Table 2.25 The input items of the deletion of the manufacturing order.

<table>
<thead>
<tr>
<th>Input Item</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID of the manufacturing order</td>
<td>The ID of the manufacturing order to be removed.</td>
</tr>
<tr>
<td>ID of the divided manufacturing order</td>
<td>The ID of the divided manufacturing order to be deleted.</td>
</tr>
</tbody>
</table>
2.5.5.2 Output

Table 2.26 The output items of the deletion of the manufacturing order.

<table>
<thead>
<tr>
<th>Output Item</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manufacturing order</td>
<td>The manufacturing order to be deleted.</td>
</tr>
<tr>
<td>The load information</td>
<td>The load information of the work center or machine after the deletion.</td>
</tr>
<tr>
<td>The result of this operation</td>
<td>If this function runs successfully, return success, else return an error.</td>
</tr>
<tr>
<td>The type of the error</td>
<td>Return the number of the error to indicate its type.</td>
</tr>
</tbody>
</table>

2.5.6 The modification of the quantity of the manufacturing order

Modifying the quantity of a manufacturing order is performed by specified its new quantity at the same time, all manufacturing processes deployed by the current manufacturing order are also modified.

2.5.6.1 Input

Table 2.27 The input items of the modification of the quantity of the manufacturing order.

<table>
<thead>
<tr>
<th>Input Item</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID of the manufacturing order</td>
<td>The ID of the manufacturing order to be modified.</td>
</tr>
<tr>
<td>ID of the divided manufacturing order</td>
<td>The ID of the divided manufacturing order to be modified.</td>
</tr>
<tr>
<td>The Quantity</td>
<td>The new quantity of the manufacturing order.</td>
</tr>
</tbody>
</table>
2.5.6.2 Output

Table 2.28 The output items of the modification of the quantity of the manufacturing order.

<table>
<thead>
<tr>
<th>Output Item</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manufacturing order</td>
<td>The manufacturing order to be modified its quantity.</td>
</tr>
<tr>
<td>The load information</td>
<td>The load information of the work center or machine after the modification.</td>
</tr>
<tr>
<td>The result of this operation</td>
<td>If this function runs successfully, return success, else return an error.</td>
</tr>
<tr>
<td>The type of the error</td>
<td>Return the number of the error to indicate its type.</td>
</tr>
</tbody>
</table>

2.5.7 The modification of the schedule and the shift type

The objective of modifying the day-off or the overtime of the facility is to adjust the standard capacity of the work center, so that the load is under the standard capacity. There are four types of the modifications listed in the following table.

Table 2.29 The operations of the modification of the shift.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modify the shift type</td>
<td>Modify the shift type of the specified work center or machine to the other type of the shift.</td>
</tr>
<tr>
<td>Append the overtime</td>
<td>Append the overtime of the work center will change the overtime of all machines of this work center.</td>
</tr>
<tr>
<td>Append the day-off</td>
<td>Append the day-off of the work machine.</td>
</tr>
<tr>
<td>Delete the day-off</td>
<td>Delete the day-off of the work machine.</td>
</tr>
</tbody>
</table>
5.2.8 Undo operation

For the manual operations, the Undo mechanism is necessary. In one case, if the input parameters of the manual operation are not correct, the Undo operation must be performed. In the other case, when several manual operations are executed, we want to recover the one of them. So the Undo mechanism can satisfy this requirement.

The level of the Undo operation can be set in the parameter file.

2.6 CHECK THE FEASIBILITY OF THE PLANNING

This module is to check the load of the measure unit whether is under the standard capacity or not.

2.6.1 Input

Table 2.30 The input items of the verification of the feasibility of the planning.

<table>
<thead>
<tr>
<th>Input Item</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID of the work center (machine)</td>
<td>Check the feasibility of planning of the specified work center (machine).</td>
</tr>
<tr>
<td>The load information</td>
<td>Use the load information to check the feasibility.</td>
</tr>
</tbody>
</table>

2.6.2 Output

Table 2.31 The output items of the verification of the feasibility of the planning.

<table>
<thead>
<tr>
<th>Output Item</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>The result of this operation</td>
<td>If the planning of the work center is executable, return true, else return false.</td>
</tr>
<tr>
<td>Arrange the measure unit</td>
<td>General view of the non-executable measure unit.</td>
</tr>
</tbody>
</table>
2.7 OUTPUT THE NON-EXECUTABLE INFORMATION

If the load of the work center is overload, output the non-executable load information.

2.7.1 Input

Table 2.32 The input items of the output of the non-executable information.

<table>
<thead>
<tr>
<th>Input Item</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID of the work center (machine)</td>
<td>Check the feasibility of planning of the specified work center (machine).</td>
</tr>
<tr>
<td>The load information</td>
<td>Use the load information to output.</td>
</tr>
</tbody>
</table>

2.7.2 Output

Table 2.33 The output items of the output of the non-executable information.

<table>
<thead>
<tr>
<th>Output Item</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>The non-executable load information</td>
<td>If the planning of the work center is non-executable, save the manufacturing order to local file.</td>
</tr>
<tr>
<td>The result of this operation</td>
<td>If this function runs successfully, return success, else return an error.</td>
</tr>
<tr>
<td>The type of the error</td>
<td>Return the number of the error to indicate its type.</td>
</tr>
</tbody>
</table>

2.8 DATA OUTPUT

This CRP module finally generates the job order to reflect the result of the load adjustment and the update of the manufacturing order.

2.8.1 Input

None.
2.8.2 Output

Table 2.34 The output items of the data output function.

<table>
<thead>
<tr>
<th>Output Item</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>The manufacturing order</td>
<td>Reflect the adjustment of load</td>
</tr>
<tr>
<td>The job order</td>
<td>Reflect the adjustment of load</td>
</tr>
<tr>
<td>The shift</td>
<td>Reflect the adjustment of load</td>
</tr>
</tbody>
</table>

2.9 LOGOUT

Logout function exits the current CRP module and saves the necessary information. There is no input data and output data involved in this part.
CHAPTER 3
CLASS DESCRIPTIONS

This chapter will discuss the data input, the load calculation and manual adjust parts. Each part contains an associated class for which their behavior and data will describe. The class descriptions will include a statement of purpose and description of both the public and private declarations. The other parts (login, data output and Undo function) will not be discussed.

3.1 THE UML OF THE CRP MODULE

This CRP module contains 9 domains as showing in figure 3.1. Each domain comprises several classes. The Figure 3.1 illustrates the UML of these classes.

The Figure 3.1 The UML of the classes.
3.2 THE DATA INPUT CLASS

The data input class will be addressed first because it gets all necessary data into memory from the database for the later use. The corresponding ‘ZRDatalnput’ class allows us to dynamically build all tables in memory. The default data source is database. It also allows the user to read the data from other data source specified in the configuration file. This class contains all of the information needed to calculate the load of the work area.

Table 3.1 describes the private variable declarations in the ZRDatalnput class. Private variables are accessible only to member functions of the class.

Table 3.1 The private variable declarations in the ZRDatalnput class.

<table>
<thead>
<tr>
<th>Declaration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection conn</td>
<td>The connection with the data source</td>
</tr>
<tr>
<td>String logFileName</td>
<td>The file name of the log file</td>
</tr>
<tr>
<td>BufferedWriter bufWriter</td>
<td>Used to write file.</td>
</tr>
<tr>
<td>Vector vComponent</td>
<td>Data storage vector that holds the records data of the component table.</td>
</tr>
<tr>
<td>Vector vWorkingProcedure</td>
<td>Data storage vector that holds the records data of the Manufacturing process table.</td>
</tr>
<tr>
<td>Vector vSequence</td>
<td>Data storage vector that holds the records data of the Working Sequence table.</td>
</tr>
<tr>
<td>Vector vValidSequence</td>
<td>Data storage vector that holds the records data of the Working Sequence table.</td>
</tr>
<tr>
<td>Vector vMachine</td>
<td>Data storage vector that holds the records data of the Work Machine table.</td>
</tr>
<tr>
<td>Vector vWorkCenter</td>
<td>Data storage vector that holds the records data of the Work Center table.</td>
</tr>
<tr>
<td>Vector vShiftType</td>
<td>Data storage vector that holds the records data of the Shift Type table.</td>
</tr>
<tr>
<td>Vector vShift</td>
<td>Data storage vector that holds the records data of the Shift table.</td>
</tr>
<tr>
<td>Vector vDayOff</td>
<td>Data storage vector that holds the records data of the Day Off table.</td>
</tr>
</tbody>
</table>
Vector vMnfOrder  | Data storage vector that holds the records data of the Manufacturing order table.
Vector vMnfDividedOrder | Data storage vector that holds the records data of the Manufacturing divided order table.
Vector vCalendar | Data storage vector that holds the records data of the Calendar table.
Vector vPlannedOrder | Data storage vector that holds the records data of the Planned Order table.
Vector vFacility | Data storage vector that holds the records data of the Facility table.
Vector vCharge | Data storage vector that holds the records data of the Charge table.
Vector vUserInfo | Data storage vector that holds the records data of the User table.

Each vector contains the records data of the corresponding table. It uses the base class of the record. For example, the Vector vMnfOrder uses the MnfOrderRecord class which is corresponding with one record of the Manufacturing order table.

The following table shows the ZRDataInput public function declarations. These functions are the users interface to the load calculation function.

Table 3.2 The public function declarations in the ZRDataInput class.

<table>
<thead>
<tr>
<th>Declaration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int getAllTable</td>
<td>Read all record data from the data source into corresponding vector. If the reading operation runs successfully, this function return 0, else return a number to indicate the type of the error.</td>
</tr>
<tr>
<td>Vector getVectorComponent</td>
<td>Get the private variable vComponent of the ZRDataInput.</td>
</tr>
<tr>
<td>Void setVectorComponent</td>
<td>Set the vector to the private variable vComponent of the ZRDataInput.</td>
</tr>
<tr>
<td>The other getVectorXXX,</td>
<td>Get the vXXX vector from ZRDataInput or set the vector to vXXX.</td>
</tr>
<tr>
<td>setVectorXXX functions</td>
<td>corresponding to the vectors listed in Table 3.1</td>
</tr>
</tbody>
</table>
Table 3.3 shows the private function declarations of the ZRDataInput class.

<table>
<thead>
<tr>
<th>Declaration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int getComponentTable</td>
<td>Read the record data from the Component table into the corresponding vector vComponent. If the reading operation runs successfully, this function return 0, else return a number to indicate the type of the error.</td>
</tr>
<tr>
<td>int checkComponentRecord</td>
<td>When this function reads the Component data from the data source, it must be taken the verification of correctness of the Component record.</td>
</tr>
<tr>
<td>The other getXXXTable and checkXXXRecord functions corresponding to the vectors listed in Table 3.1</td>
<td>Read the record data from the XXX table into the corresponding vector vXXX. If the reading operation runs successfully, this function return 0, else return a number to indicate the type of the error.</td>
</tr>
</tbody>
</table>

3.3 LOAD CALCULATION CLASS

The ZRLoadCalcul class is the most important class in this module, because it performs not only the load calculation of the work area according to the measure unit, it also takes the task of auto-balance of the load. The load calculation is performed by using the work time of each manufacturing process deployed by the manufacturing order.

The auto-balance load function will process and reschedule all open and planned manufacturing orders using backward scheduling. Backward scheduling logic will calculate each operation backwards from the manufacturing order or planned order.
Table 3.4 shows the private variable declarations in the ZRLoadCalcul class.

<table>
<thead>
<tr>
<th>Declaration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZRDatalnput</td>
<td>The data input class</td>
</tr>
<tr>
<td>int intCapacityUnit</td>
<td>The unit of the capacity</td>
</tr>
<tr>
<td>Date datePlanBase</td>
<td>The base point of the plan</td>
</tr>
<tr>
<td>int intPlanDuration</td>
<td>The period of the plan</td>
</tr>
<tr>
<td>Date dateBeginTime</td>
<td>The start date of the plan</td>
</tr>
<tr>
<td>Hashtable htMachine</td>
<td>Data storage hashtable that holds all machines information.</td>
</tr>
<tr>
<td>Hashtable htWorkCenter</td>
<td>Data storage hashtable that holds all Work Centers information.</td>
</tr>
<tr>
<td>Hashtable htMachineWeek</td>
<td>Data storage hashtable that holds all machines information. The capacity unit is week.</td>
</tr>
<tr>
<td>Hashtable htWorkCenterWeek</td>
<td>Data storage hashtable that holds all work center information. The capacity unit is week.</td>
</tr>
<tr>
<td>Hashtable htAllProcedure</td>
<td>Data storage hashtable that holds all manufacturing processes information.</td>
</tr>
</tbody>
</table>

The following table shows the ZRLoadCalcul public function declarations. These functions are the users interface to the load calculation function.

Table 3.5 The public function declarations in the ZRLoadCalcul class.

<table>
<thead>
<tr>
<th>Declaration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZRLoadCalcul</td>
<td>Constructor function, in this function the initialization task is done.</td>
</tr>
<tr>
<td>loadCalculation</td>
<td>This is the core function in this CRP system, it will divide manufacturing order into manufacturing process, and do load calculation operation.</td>
</tr>
<tr>
<td>autoBalance</td>
<td>It is this function that does the auto-balance of the load. It compares the load of the current measure unit with its standard capacity to determine the under load or over load.</td>
</tr>
</tbody>
</table>
### Table 3.6 The private function declarations of the ZRLoadCalcul class.

<table>
<thead>
<tr>
<th>Declaration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>initWorkCenterByDay</td>
<td>Initialize the WorkCenter by day, create instances of ZRWorkCenter class and put them into Hashtable variable htWorkCenter</td>
</tr>
<tr>
<td>initWorkMachineByDay</td>
<td>Initialize the WorkMachine by day, create instances of ZRWorkMachine class and put them into Hashtable variable htWorkMachine</td>
</tr>
<tr>
<td>initWorkCenterByWeek</td>
<td>Initialize the WorkCenter by week, create instances of ZRWorkCenter class and put them into Hashtable variable htWorkCenterWeek</td>
</tr>
<tr>
<td>initWorkMachineByWeek</td>
<td>Initialize the Machine by week, create instances of ZRMachine class and put them into Hashtable variable htMachineWeek</td>
</tr>
</tbody>
</table>

### 3.4 MANUAL ADJUST CLASSES

As discussed in chapter 5, the manual adjust part consists of 8 operations. These operations are base on the interface ‘ZRManualAdjust’. Here I will discuss one class for illustration: the class ZRLoadTransfer. The other classes are similar to this class. You can find them in the Index C.
3.4.1 The interface ZRManualAdjust

The following table shows the public functions placed in the interface ZRManualAdjust.

Table 3.7 The public functions placed in the interface ZRManualAdjust.

<table>
<thead>
<tr>
<th>Declaration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Void setParameter</td>
<td>This interface sets the necessary parameters of the manual operation.</td>
</tr>
<tr>
<td>Vector getParameter</td>
<td>This interface gets the input parameters of the manual operation.</td>
</tr>
<tr>
<td>Hashtable[] getExistedRecords</td>
<td>This interface gets all necessary existed hashtable to perform the manual operation.</td>
</tr>
<tr>
<td>int changeExistedRecords</td>
<td>This interface performs the manual operation.</td>
</tr>
</tbody>
</table>

3.4.2 Load transfer class

ZRLoadTransfer class is corresponding to the load transference operation. There are two kinds of the transference: the load between the work centers and the load between the work machines.

Table 3.8 Shows the private variable declarations in the ZRLoadTransfer class.

<table>
<thead>
<tr>
<th>Declaration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZRDatalnput zrDatalnput</td>
<td>The class instance of class ZRDatalnput</td>
</tr>
<tr>
<td>ZRLoadCalcul zrLoadCalcul</td>
<td>The class instance of class ZRLoadCalcul</td>
</tr>
<tr>
<td>Boolean moveFlag</td>
<td>The flag of the transference operation</td>
</tr>
<tr>
<td>Vector vLoadTransferInfo</td>
<td>The vector stores the parameter list.</td>
</tr>
<tr>
<td>Hashtable[] existedOrderRecords</td>
<td>The existed manufacturing order hashtable.</td>
</tr>
</tbody>
</table>
Table 3.9 shows the public variable declarations in the ZRLoadTransfer class.

<table>
<thead>
<tr>
<th>Declaration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OperationObject oo</td>
<td>The class object is prepared for the Undo operation.</td>
</tr>
<tr>
<td>UndoList ul</td>
<td>The Undo list used for the Undo operation.</td>
</tr>
</tbody>
</table>

Because the class ZRLoadTransfer extends the interface ZRManualAdjust, the public functions of this class are the same as shown in the table 6.4.1.

### 3.5 CRP TASK CLASS

I design the ZRCRPTask class to provide the interfaces of all of the tasks described in the previous chapters. This is for two purposes. The first is that the CRP module will work as a whole to integrate to the ZRERP system. The second is that this module can work independently, so we can use this class to interact with the GUI classes.

Table 3.10 shows the private variable declarations in the ZRCRPTask class.

<table>
<thead>
<tr>
<th>Declaration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZRConnection crpconn</td>
<td>Open the connection with the data source</td>
</tr>
<tr>
<td>ZRDatalnput zrDatalnput</td>
<td>The class instance of class ZRDatalnput</td>
</tr>
<tr>
<td>ZRDataOutput zrDataOutput</td>
<td>The class instance of class ZRDataOutput</td>
</tr>
<tr>
<td>ZRLogin zrLogin</td>
<td>The class instance of class ZRLogin</td>
</tr>
<tr>
<td>ZRLoadCalcul zrLoadCalcul</td>
<td>The class instance of class ZRLoadCalcul</td>
</tr>
<tr>
<td>ZRLoadTransfer zrLoadTransfer</td>
<td>The class instance of the class ZRLoadTransfer to perform the load transference operation of manual adjust.</td>
</tr>
<tr>
<td>ZROrderDivide zrOrderDivide</td>
<td>The class instance of the class ZROrderDivide to perform the load division operation of manual adjust.</td>
</tr>
<tr>
<td>ZROrderAppend zrOrderAppend</td>
<td>The class instance of the class ZROrderAppend to perform the order append operation of manual adjust.</td>
</tr>
</tbody>
</table>
The class instance of the class ZROrderUnite to perform the order combination operation of manual adjust.

The instance of the class ZROrderQuantityChg to perform the modification of the quantity of the manufacturing order.

The instance of the class ZRShiftChg to perform the modification the type of the shift.

The instance of the class ZRDayOffAppend to append the day-off of the work center or machine.

<table>
<thead>
<tr>
<th>Declaration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UndoList undoList</td>
<td>The list to perform the undo operation.</td>
</tr>
<tr>
<td>int undoNum</td>
<td>The number of Undo times.</td>
</tr>
</tbody>
</table>

The table 3.12 shows the public function declarations in the ZRCRPTask class.

These functions are used by other modules or by the GUI functions.

Table 3.12 The public function declarations in the ZRCRPTask class.

<table>
<thead>
<tr>
<th>Declaration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int init()</td>
<td>This function will perform the necessary initialization of the ZRCRPTask class, such as preparing the database connection, reading the configuration file and initializing the private variables. When the initialization operation runs successfully, it returns 0, else an error number occurs.</td>
</tr>
<tr>
<td>int login</td>
<td>This function will do the login action as described in chapter 4. When user logs in successfully, it returns 0, else an error number occurs.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>int dataInput</td>
<td>This function will read all tables into the memory as described in chapter 4. When it runs successfully, it will return 0, else an error number occurs.</td>
</tr>
<tr>
<td>int loadCalculate()</td>
<td>This function will perform the load calculation as described in chapter 4. When it runs successfully, it returns 0, else an error number occurs.</td>
</tr>
<tr>
<td>int autoBalance</td>
<td>This function will perform the load auto-balance operation as described in chapter 4. When it runs successfully, it returns 0, else an error number occurs.</td>
</tr>
<tr>
<td>Vector getLoadInfo</td>
<td>This function will get the load information of the specified work center or work machine. When it runs successfully, it returns a Vector which contains the load information, else it returns NULL.</td>
</tr>
<tr>
<td>int orderDivide</td>
<td>This function will divide a specified manufacturing order into two manufacturing orders as described in chapter 4. When it runs successfully, it returns 0, else an error number occurs.</td>
</tr>
<tr>
<td>int orderUnite</td>
<td>This function will combine the specified two manufacturing orders into one manufacturing order as described in chapter 4. When it runs successfully, it returns 0, else an error number occurs.</td>
</tr>
<tr>
<td>int orderAppend</td>
<td>This function will append a new manufacturing order as described in chapter 4. When it runs successfully, it returns 0, else an error number occurs.</td>
</tr>
<tr>
<td>int orderDelete</td>
<td>This function will delete a specified manufacturing order as described in chapter 4. When it runs successfully, it returns 0, else an error number occurs.</td>
</tr>
<tr>
<td>int quantityChange</td>
<td>This function will change the quantity of the specified manufacturing order as described in chapter 4. When it runs successfully, it returns 0, else an error number occurs.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>int shiftChange</td>
<td>This function will change the shift type of the specified work center as described in chapter 4. When it runs successfully, it returns 0, else an error number occurs.</td>
</tr>
<tr>
<td>int dayOffAppend</td>
<td>This function will append the day-off of the specified work center or machine as described in chapter 4. When it runs successfully, it returns 0, else an error number occurs.</td>
</tr>
<tr>
<td>int getNonExecutableInfo</td>
<td>This function will get the non-executable information of the specified work center or machine as described in chapter 4. When it runs successfully, it returns a Vector which contains the load information, else it returns NULL.</td>
</tr>
<tr>
<td>int unDo</td>
<td>Calling this function one time to cancel the last operation in the unDoList.</td>
</tr>
<tr>
<td>int checkFeasibility</td>
<td>This function will check the feasibility of the planning, i.e. the load of the measure unit is under the standard capacity or not. If the planning is feasible, it returns 0, else either the planning is not feasible 1 or an error number (&gt;1) occurs.</td>
</tr>
<tr>
<td>int dataOutput</td>
<td>This function will write back all memory data into the data source, at the same time output the necessary information into the local file. When it runs successfully, it returns 0, else an error number occurs.</td>
</tr>
<tr>
<td>int logout</td>
<td>This function will do the logout action to exit this module. When user logs out successfully, it returns 0, else an error number occurs.</td>
</tr>
</tbody>
</table>
CHAPTER 4
IMPLEMENTATION

The current implementation of this CRP module supports two types of the data source. One is Oracle 8i and another is text format file (Comma Separated Value -- CSV). For the text format file, the CsvJDBC driver must be used. It is just like any other JDBC driver. The csvjdbc.jar file should be included in the application's classpath.

Because Java is the platform independent, so the program is designed to run on any platform. The following figure shows the login window of the CRP module.

Figure 4.1 Login window of the CRP module.
As described in chapter 2, when user logged in correctly, the data input function was performed automatically. So all the data of the data source is read into memory. Then the main frame of the CRP module was shown, as showing in figure 4.2.

Figure 4.2 The main frame of this CRP module.

As discussed in chapter 2, this CRP module contains load calculation, auto-balance, manual adjust and data output functions. Each function is corresponding to a menu item in the main frame. The menu items of the manual adjust are displayed in figure 4.3.
Figure 4.3 The menu items of the manual adjust operations.

Figure 4.4 The order transference operation.

Here we give an example of the order transferring operation for illustration.
For the load information output, the graphic display is more legible, visual and dynamic. Figure 4.4 is the graphic result of the implementation.

Figure 4.5 The graphic output of the load.
CONCLUSION

Materials Requirements Planning is a new information system program and its rather simple mission was to schedule manufacturing processes to meet customer needs as laid out in a Master Production Schedule (MPS). The MPS come from the individual product forecasts and is intended to ensure that customers get what they want in a timely fashion. A problem arose early on, however, the MPS might not be feasible with available capacity. The MPS might call for more units to be produced in a period than the factory could handle. The simple solution is to move some of the production to an earlier or later time period. This required an understanding of customer requirements and capacity restrictions. Critical customers would have to be moved to earlier periods to avoid a stock-out that could damage important relationships. Capacity Requirements Planning (CRP) is added to further enhance MRP and allow for complete and feasible scheduling of manufacturing lines.

In this paper I introduced the requirements analysis, the detail design and the implementation of the CRP module. The load calculation is performed by taking into account of the planned orders in the posterior planning function (FCS) to achieve the calculation more accurate. The load auto-balance function is designed to enable verifying the feasibility of the planning by the means of comparing the added load with its standard capacity of the measure unit. I used the backward load scheduling (infinite) method to
perform the load auto-balance function. As shown in chapter 4, I realized the graphic load
displaying. In chapter 2, I also described the manual adjust operations. The purpose of these
operations is to offer the manual means to adjust the planning.

The class descriptions in chapter 3 explicitly describe the functions involved in the
load calculation, the auto-balance and the manual adjust functions. Class declarations and
access are also listed here. Public and private functions and data members ensure that data
is not accidentally changed by some outside source.

Design of the CRP module is based on the principles of Object Oriented Design and
Object Oriented Programming (OOD/OOP). To further facilitate this all code is written
using Java. This language was also chosen based on the ease of extensibility of completed
code.

In conclusion I believe this implementation of the CRP module to be a big step in
ZRERP system. Many and varied ERP systems may be evaluated to do the same job.
REFERENCES


LANDVATER D V, GRAY C D. MRPII STANDARD SYSTEM. JOHN WILEY & SONS, INC., 1989

MANAGEMENT.


[10] LEARNING JAVA WITH JBUILDER, SCOTTS VALLEY, CA 95066-3249.

## APPENDIX A

### CLASS TABLE

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZRCRPTask</td>
<td>The interface class of GUI</td>
</tr>
<tr>
<td>ZRConnection</td>
<td>Connecting to data source</td>
</tr>
<tr>
<td>ZRLogin</td>
<td>Logging in this Module</td>
</tr>
<tr>
<td>ZRDdataInput</td>
<td>Reading the data into memory</td>
</tr>
<tr>
<td>ZRLoadCalcul</td>
<td>Calculating the load and doing the auto-balance</td>
</tr>
<tr>
<td>ZRManualAdjust</td>
<td>The interface to do manual adjust</td>
</tr>
<tr>
<td>ZRDdataOutput</td>
<td>Output the data</td>
</tr>
<tr>
<td>ZRLogout</td>
<td>Logging out</td>
</tr>
<tr>
<td>ZROrderDelete</td>
<td>Deleting the specified order</td>
</tr>
<tr>
<td>ZRLoadTransfer</td>
<td>Transferring the load</td>
</tr>
<tr>
<td>ZRShiftChg</td>
<td>Modifying the shift</td>
</tr>
<tr>
<td>ZROrderDivide</td>
<td>Dividing the order</td>
</tr>
<tr>
<td>ZROrderUnite</td>
<td>Combining the order</td>
</tr>
<tr>
<td>ZROrderAppend</td>
<td>Append the order</td>
</tr>
<tr>
<td>ZROrderDivide</td>
<td>Append the day-off</td>
</tr>
<tr>
<td>CalendarDB</td>
<td>The interface of CalendarDBImpl</td>
</tr>
<tr>
<td>CalendarDBImpl</td>
<td>Operating the calendar table</td>
</tr>
<tr>
<td>CalendarRecord</td>
<td>Storing the calendar record</td>
</tr>
<tr>
<td>ChargeDB</td>
<td>The interface of ChargeDBImpl</td>
</tr>
<tr>
<td>ChargeDBImpl</td>
<td>Operating the charge table</td>
</tr>
<tr>
<td>ChargeRecord</td>
<td>Storing the charge record</td>
</tr>
<tr>
<td>MachineDB</td>
<td>The interface of MachineDBImpl</td>
</tr>
<tr>
<td>MachineDBImpl</td>
<td>Operating the machine table</td>
</tr>
<tr>
<td>MachineRecord</td>
<td>Storing the machine record</td>
</tr>
<tr>
<td>FactoryDB</td>
<td>The interface of FactoryDBImpl</td>
</tr>
<tr>
<td>FactoryDBImpl</td>
<td>Operating the facility table</td>
</tr>
<tr>
<td>FactoryRecord</td>
<td>Storing the facility record</td>
</tr>
<tr>
<td>MnfOrderDB</td>
<td>The interface of MnfOrderDBImpl</td>
</tr>
<tr>
<td>MnfOrderDBImpl</td>
<td>Operating the manufacturing order table</td>
</tr>
<tr>
<td>Class</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>MnfOrderRecord</td>
<td>Storing the manufacturing order record</td>
</tr>
<tr>
<td>MnfOrderDivideDB</td>
<td>The interface of MnfOrderDivideDBImpl</td>
</tr>
<tr>
<td>MnfOrderDivideDBImpl</td>
<td>Operating the manufacturing order detail table</td>
</tr>
<tr>
<td>MnfOrderDivideRecord</td>
<td>Storing the manufacturing order detail record</td>
</tr>
<tr>
<td>PartDB</td>
<td>The interface of PartDBImpl</td>
</tr>
<tr>
<td>PartDBImpl</td>
<td>Operating the component table</td>
</tr>
<tr>
<td>PartRecord</td>
<td>Storing the component table</td>
</tr>
<tr>
<td>ProduceableTimeDB</td>
<td>The interface of ProduceableTimeDBImpl</td>
</tr>
<tr>
<td>ProduceableTimeDBImpl</td>
<td>Operating the producible time table</td>
</tr>
<tr>
<td>ProduceableTimeRecord</td>
<td>Storing the producible time record</td>
</tr>
<tr>
<td>SequenceDB</td>
<td>The interface of SequenceDBImpl</td>
</tr>
<tr>
<td>SequenceDBImpl</td>
<td>Operating the process sequence table</td>
</tr>
<tr>
<td>SequenceRecord</td>
<td>Storing the process sequence record</td>
</tr>
<tr>
<td>WorkOrderDB</td>
<td>The interface of WorkOrderDBImpl</td>
</tr>
<tr>
<td>WorkOrderDBImpl</td>
<td>Operating the work order table</td>
</tr>
<tr>
<td>WorkOrderRecord</td>
<td>Storing the work order record</td>
</tr>
<tr>
<td>WorkCenterDB</td>
<td>The interface of WorkCenterDBImpl</td>
</tr>
<tr>
<td>WorkCenterDBImpl</td>
<td>Operating the work center table</td>
</tr>
<tr>
<td>WorkCenterRecord</td>
<td>Storing the work center record</td>
</tr>
<tr>
<td>WorkingProcedureDB</td>
<td>The interface of WorkingProcedureDBImpl</td>
</tr>
<tr>
<td>WorkingProcedureDBImpl</td>
<td>Operating the process table</td>
</tr>
<tr>
<td>WorkingProcedureRecord</td>
<td>Storing the process record</td>
</tr>
<tr>
<td>ValidSequenceDB</td>
<td>The interface of ValidSequenceDBImpl</td>
</tr>
<tr>
<td>ValidSequenceDBImpl</td>
<td>Operating the valid process sequence table</td>
</tr>
<tr>
<td>ValidSequenceRecord</td>
<td>Storing the valid process sequence record</td>
</tr>
<tr>
<td>WorkCenterDay</td>
<td>Storing the information of the work center by day</td>
</tr>
<tr>
<td>WorkCenterWeek</td>
<td>Storing the information of the work center by week</td>
</tr>
<tr>
<td>MachineDay</td>
<td>Storing the information of the work machine by day</td>
</tr>
<tr>
<td>MachineWeek</td>
<td>Storing the information of the work machine by week</td>
</tr>
<tr>
<td>ZRException</td>
<td>The exceptions defined in this module.</td>
</tr>
<tr>
<td>CErorNumber</td>
<td>Contains all errors' number</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>ZRComm</td>
<td>The common class</td>
</tr>
<tr>
<td>ZRMachine</td>
<td>Storing the machine's information</td>
</tr>
<tr>
<td>ZRWorkcenter</td>
<td>Storing the center’s information</td>
</tr>
<tr>
<td>ZRParamFile</td>
<td>Reading the configuration file.</td>
</tr>
<tr>
<td>ZRProcess</td>
<td>Storing the information of a process</td>
</tr>
<tr>
<td>UndoList</td>
<td>Performing Undo operation.</td>
</tr>
</tbody>
</table>
/**
* the load calculation function
*/
public void loadCalcul() {
    Enumeration e = this.htMnfOrderNew.keys();
    while(e.hasMoreElements()) {
        MnfOrderRecord order = (MnfOrderRecord)(this.htMnfOrderNew.get(e.nextElement()));
        String partID = order.getPartlD();
        // get manufacture order details
        String orderlD = order.getOrderlD();
        MnfOrderDivideRecord orderDetail = (MnfOrderDivideRecord)(this.htMnfOrderDivid.get(orderID + "0");

        if (orderDetail == null) {
            System.out.println(logDate.toString() +":ManufacturingOrder Table :OrderId="+orderID+":There is no corresponding data in the ManufacturingOrder Detail Table.
        }
        Date orderEst = orderDetail.getEstDate();
        Date orderDeliverDate = orderDetail.getDeliverDate();
        Date planLastDay = new Date(this.datePlanBase.getTime() +
            this.intPlanDucration*3600*1000*24);

        if(!orderEst.before(this.datePlanBase) && orderEst.before(planLastDay)) {
            int number = orderDetail.getQuantity();
        }
    }
}
// get sequence and process ID
TreeSet sequencers = (TreeSet)this.htAssistSequence.get(partID);
Iterator i = sequence_s.iterator();

Vector procedures = new Vector();

// for each sequence, generate procedure
while(i.hasNext())
{
    SequenceRecord sequence = (SequenceRecord) i.next();

    // if the sequence is valid
    boolean isNotValid = true;

    String sequenceid = sequence.getSequenceID();
    TreeSet validSequenceTS = (TreeSet)this.htValidSequence.get(partID + sequenceid);
    Vector validSequenceVecotr = new Vector(validSequenceTS);
    // System.out.println(validSequenceVecotr.size());

    Date submitDate = orderDetail.getDeliverDate();
    int num = 0;
    boolean isFind = false;

    while (!isFind && num<validSequenceVecotr.size())
    {
        ValidSequenceRecord validSequence
            = (ValidSequenceRecord)validSequenceVecotr.elementAt(num);
        if(submitDate.after(validSequence.getValidStartDate()))
        {
            if(num == validSequenceVecotr.size() -1)
            {
                isNotValid = validSequence.getIsValid();
                isFind = true;
            }
            else
            {
                ValidSequenceRecord validSequenceNext
                    = (ValidSequenceRecord)validSequenceVecotr.elementAt(num+1);
                if(submitDate.before(validSequenceNext.getValidStartDate()))
                {
                    isNotValid = validSequence.getIsValid();
                }
            }
        }
    }
}
isFind = true;
}
} // end if reach end

} // end if after the current date
num = num + 1;
} // end while

if(isNotValid)
{
    System.out.println(logDate.toString() + "The process is not in the valid period.
"
    )
}

if(!isNotValid)
{
    /*
     // get work procedure information
     String workProcedureID = sequence.getWkProcedureID();
     WorkingProcedureRecord wkProcedure =
     (WorkingProcedureRecord) this.htWorkProcedure.get(partID + workProcedureID);
     String workcenterID = wkProcedure.getShopID();
     */
     // get producable table
     String sequenceID = sequence.getSequenceID();
     ProducibleTimeRecord producable =
     (ProducibleTimeRecord) this.htProducibleTime.get(partID + sequenceID +
     workcenterID);
    *
    // get producable table, new
    ProducibleTimeRecord producable = null;
    String workcenterID = null;
    String sequenceID = sequence.getSequenceID();
    Hashtable producableWorkcenter =
    (Hashtable) this.htProducibleTime.get(partID + sequenceID);
    if(producibleWorkcenter == null)
    {
        System.out.println(logDate.toString() + "Routing Table :RoutingId=" + sequenceID +":There is no corresponding workcenter in the ProducibleTime Table.
"
    }
    else
    {
    

Enumeration workcenters = produceableWorkcenter.keys();
boolean isFind2 = false;
while(workcenters.hasMoreElements() & & !isFind2)
{
    workcenterID = (String)workcenters.nextElement();
    TreeSet workcenterInfo = (TreeSet)produceableWorkcenter.get(workcenterID);
    Vector vWorkcenterInfo = new Vector(workcenterInfo);

    int num2 = 0;
    boolean isInThisShop = false;
    boolean isNotValid2 = true;

    while(!isInThisShop & & num2<vWorkcenterInfo.size())
    {
        ProduceableTimeRecord currentRecord = (ProduceableTimeRecord)vWorkcenterInfo.get(num2);
        if(num2 == vWorkcenterInfo.size() - 1)
        {
            if(!orderDeliverDate.before(currentRecord.getValidStartDate()))
            {
                produceable = currentRecord;
                isNotValid2 = currentRecord.get isValid();
                isInThisShop = true;
            }
            else
            {
                isInThisShop = false;
            } // end if the deliver date is after the last record's begin date
        }
        else
        {
            ProduceableTimeRecord nextRecord = (ProduceableTimeRecord)vWorkcenterInfo.get(num2 + 1);
            if(!orderDeliverDate.before(currentRecord.getValidStartDate()) & &
            orderDeliverDate.before(nextRecord.getValidStartDate()))
            {
                produceable = currentRecord;
                isNotValid2 = currentRecord.get isValid();
                isInThisShop = true;
            }
            else
            {
isInThisShop = false;
}
} // end if reach the last record
num2 = num2 + 1;
} // end while judge if in this shop
if(isInThisShop && !isNotValid2)
{
    isFind2 = true;
}
else
{
    
} // end while all workcenters
if(isFind2)
{
    produceable = null;
}
} // end if produceableWorkcenter == null

// calculate the procedure's total time
//
WorkShopRecord workcenter = (WorkShopRecord) this.htWorkShop.get(workcenterID);
if(workcenterID == null || produceable == null)
{
    System.out.println(logDate.toString() + "There is no corresponding WorkCenter in the WorkCenter Table. 
\n");
}
else
{

    WorkShopRecord workcenter = (WorkShopRecord) this.htWorkShop.get(workcenterID);

    char type = (char) workcenter.getCalculateType();
    // System.out.println("flaskjdfasdfsdfkasdf");
    double totalTime = 0.0;
    if(type == 'U')
    {
        totalTime = produceable.getLt();
        totalTime = totalTime + produceable.getAbundantTimeO;
    }
    else if(type == 'S')
totalTime = produceable.getStQ * number;
totalTime = totalTime + produceable.getAbundantTime();
}
else
{
    throw new ZRException(CErrorNumber.CALCULATE_TYPE_ERROR);
    System.out.println(logDate.toString()+":Calculate Type error.
}
// System.out.println(
// System.out.println("
ZRWorkProcedure pro = new ZRWorkProcedureCorder.getOrderlDO^rderDetail.getDivideNOQ^rder.getPartlDCorderDetailgetQuantity()5sequence.getWkProeediireID()3sequence.getSeqiienceID()?sequence.get
ZRWorkProcedure pro = new ZRWorkProcedure(order.getOrderByID(),orderDetail.getDivideNO(),order.getPartID(),orderDetail.getQuantity(),sequence.getWkProcedureID(),sequence.getSequenceID(),sequence.getSequece(),workcenterID,totalTime);
// System.out.println("flaskjdfasdjfkasdfs");
    procedures.add(pro);
    System.out.println("flaskjdfasdjfkasdfs");
}
else
{
    //the sequence is not valid
}
} //end search in all sequence belong to this order
//calculate LFT and EST
Date lft = orderDetail.getLftDateQ;
Date est = orderDetail.getEstDateQ;
int n = procedures.sizeQ;
if(procedures.size() == 0)
{
    throw new ZRException(CErrorNumber.PROCEDURE_SPREAD_FAILED);
    System.out.println(logDate.toString()+":Procedure spread error.
}
else
{
    ZRWorkProcedure pro = (ZRWorkProcedure)procedures.get(0);
    pro.setEST(est);
    for (int j = 1; j < n ; j++)
    {
        ZRWorkProcedure pro_prev = (ZRWorkProcedure)procedures.get(j-1);
        ZRWorkProcedure pro_curr = (ZRWorkProcedure)procedures.get(j);
        long prev_time = pro_prev.getEST().getTime();
        prev_time = prev_time + (int)(pro_prev.getTotalTime() * 1000);
Date curr_date = new Date(prev_time);
pro_curr.setEST(curr_date);
}

pro = (ZRWorkProcedure)procedures.get(n-1);
pro.setLFT(lft);
for(int j = n-2; j>=0; j--)
{
    ZRWorkProcedure pro_next = (ZRWorkProcedure)procedures.get(j+1);
    ZRWorkProcedure pro_curr = (ZRWorkProcedure)procedures.get(j);
    long next_time = pro_next.getLFT().getTime();
    next_time = next_time - (int)(pro_next.getTotalTime()*1000);
    Date curr_date = new Date(next_time);
    pro_curr.setLFT(curr_date);
}

//add load process
String unitID = "";
Date start = null;
Date end = null;
String taskID = "";
double totalTime = 0;
String mnfOrderID = "";
int divideNum = 0;
for(int j = 0; j < n; j++)
{
    pro = (ZRWorkProcedure)procedures.get(j);
    this.htAllProcedure.put(pro.getOrderID() + pro.getOrderDividNo() + pro.getProcID())
    pro.getProcID(),pro);
    this.htProForSearch.put(pro.getOrderID() + pro.getOrderDividNo() + pro.getSequenceID(),pro);
    String workcenterID = pro.getWorkcenter();
    if(this.intCapacityUnit == 1 && this.strPlanUint.equals("R"))
    {
        TreeSet tsMachine = (TreeSet)htAssistMachine.get(workcenterID);
        if(tsMachine == null) {
            System.out.println(logDate.toString()+"There is no machine in this workcenter: workcenterID=
            workcenterID="+workcenterID+" .
        }
    }
    int machineNum = tsMachine.size();
    Vector vMachine = new Vector(tsMachine);
    Date theDate = new Date(pro.getLFT().getTime() -


Date lastDay = new Date(this.datePlanBase.getTime() +
    this.intPlanDuration*24*3600*1000);
    if(theDate.after(lastDay) || theDate.equals(lastDay))
    {
        theDate = new Date(lastDay.getTime() - 24*3600*1000);
    }

    //get the latest unit, if standard capacity lower than current load
    EquipmentRecord lastMachine = (EquipmentRecord)vMachine.get(machineNum -1);
    String lastMachineID = lastMachine.getEquipmentID();
    ZRMachine lastZRMachine = (ZRMachine)this.htMachine.get(lastMachineID);
    MachineDay lastMachineDay = (MachineDay)((Hashtable)lastZRMachine.getDayHT()).get(lastMachineID +
    theDate.toString());
    if(lastMachineDay.getCurrentLoad() >
    lastMachineDay.getStandardCapacity())
    {
        //add the procedure to the first unit
        EquipmentRecord machine = (EquipmentRecord)vMachine.get(0);
        String machineID = machine.getEquipmentID();
        ZRMachine zrMachine = (ZRMachine)this.htMachine.get(machineID);
        MachineDay machineDay = (MachineDay)((Hashtable)zrMachine.getDayHT()).get(machineID +
        theDate.toString());

        pro.setMachine(machineID);
        Integer seq = new Integer(pro.getSequence());
        String sequence = seq.toString();
        String temp = "";
        for(int nu = 0; nu< 5-sequence.length();nu++)
        {
            temp = temp + "0";
        }
        sequence = temp + sequence;

        String strTaskID = pro.getPartsID() + "_" + pro.getMachine() + "_" +
        ZRComm.date2String(pro.getLFT()) + sequence;
        pro.setTaskID(strTaskID);
        machineDay.addLoad(pro);
// out put the info mation
unitID = machineID;
start = machineDay.getCalDate();
end = new Date(machineDay.getCalDate().getTime() + 24*3600*1000 - 1);
taskID = "undefined";
totalTime = pro.getTotalTime();
mnfOrderID = orderID;
devideNum = 0;
}
else
{
    boolean flag = true;
    Enumeration eMachine = vMachine.elements();
    while(eMachine.hasMoreElements() && flag)
    {
        EquipmentRecord machine = (EquipmentRecord)eMachine.nextElement();
        String machinelD = machine.getEquipmentID();
        ZRMachine zrMachine = (ZRMachine)this.htMachine.get(machinelD);
        MachineDay machineDay = (MachineDay)((Hashtable)zrMachine.getDayHT()).get(machineID + theDateioString());
        if(machineDay.getStandardCapacity() >= machineDay.getCurrentLoad())
        {
            pro.setMachine(machineID);
            Integer seq = new Integer(pro.getSequence());
            String sequence = seq.toString();
            String temp = "";
            for(int nu = 0; nu < 5 - sequence.length(); nu++)
            {
                temp = temp + "0";
            }
            sequence = temp + sequence;
            String strTaskID = pro.getPartsID() + "_" + pro.getMachine() + "_" + ZRComm.date2String(pro.getLFT()) + sequence;
            pro.setTaskID(strTaskID);

            machineDay.addLoad(pro);

            unitID = machineID;
            start = machineDay.getCalDate();
            end = new Date(machineDay.getCalDate().getTime() + 24*3600*1000 - 1);
taskID = "undefined";
totalTime = pro.getTotalTime();
mnfOrderID = orderID;
devideNum = 0;

flag = false;
}
else
{
    flag = true;
}

else if(this.intCapacityUnit == 1 && this.strPlanUint.equals("W"))
{
    ZRWorkcenter workcenter = (ZRWorkcenter) this.htWorkCenter.get(workcenterID);
    if (workcenter == null) {
        System.out.println(logDate.toString()+":There is no corresponding 
workcenter.\n\n");
    }
    Date theDate = new Date(pro.getLFT().getTime() - (pro.getLFT().getTime())%(24*3600*1000) - this.TIME_ZONE*3600*1000);
    Date lastDay = new Date(this.datePlanBase.getTime() + this.intPlanDuration*24*3600*1000);
    if (theDate.after(lastDay) || theDate.equals(lastDay)) {
        theDate = new Date(lastDay.getTime() - 24*3600*1000);
    }
    // System.out.println(theDate.toString() + pro.getLFT());
    Hashtable htWorkcenterDay = workcenter.getDayHT();
    Integer seq = new Integer(pro.getSequence());
    String sequence = seq.toString();
    String temp = "";
    for(int nu = 0; nu< 5-sequence.length();nu++)
    {
        temp = temp + "0";
    }
    sequence = temp + sequence;
String strTaskID = pro.getPartsIDQ + "_" + pro.getWorkcenterQ + "_" + ZRComm.date2String(pro.getLFT()) + sequence;
pro.setTasklD(strTasklD);

((WorkcenterDay)htWorkcenterDay.get(workcenterID + theDate.toString())).addLoad(pro);

//out put the infomation
unitID = workcenterID;
start = ((WorkcenterDay)htWorkcenterDay.get(workcenterID + theDate.toString())).getCalDate();
end = new Date(start.getTime() + this.intCapacityUnit*24*3600*1000 -1);
taskID = "undefined";
totaiTime = pro.getTotalTimeQ;
mnfOrderlD = orderlD;
devideNum = 0;
}
else if(this.intCapacityUnit == 7 && this.strPlanUint.equals("R"))
{
    TreeSet tsMachine = (TreeSet)htAssIstMachine.get(workcenterID);
    if (tsMachine == null) {
        System.out.println(logDate.toString()+":There is no machine in this workcenter: workcenterID=\" + workcenterID + \" .\n\n");
    }
    int machineNum = tsMachine.sizeQ;
    Vector vMachine = new Vector(tsMachine);
    long weekNum = (pro.getLFT().getTime()  - this.datePlanBase.getTime())/(this.intCapacityUnit*24*3600*1000);
    Date theDate = new Date(this.datePlanBase.getTime() + weekNum * (this.intCapacityUnit*24*3600*1000));

    Date lastDay = new Date(this.datePlanBase.getTime() + this.intPlanDucration*24*3600*1000);
    if(lastDay.after(lastDay))
    {
        weekNum = this.intPlanDucration/this.intCapacityUnit;
        theDate = new Date(this.datePlanBase.getTime() + weekNum * (this.intCapacityUnit*24*3600*1000));
    }

    //get the latest unit, if standard capacity lower than current load
    EquipmentRecord lastMachine =
    (EquipmentRecord)vMachine.get(machineNum -1);
    String lastMachineID = lastMachine.getEquipmentlDQ;
ZRMachine lastZRMachine = (ZRMachine)this.htMachineWeek.get(lastMachineID);

MachineWeek lastMachineWeek = (MachineWeek)((Hashtable)lastZRMachine.getWeekHT()).get(lastMachineID + theDate.toStringO);

if(lastMachineWeek.getCurrentLoad() > lastMachineWeek.getStandardCapacityO)
{
    //add the procedure to the first unit
    EquipmentRecord machine = (EquipmentRecord)vMachine.get(0);
    String machineID = machine.getEquipmentID();
    ZRMachine zrMachine = (ZRMachine)this.htMachineWeek.get(machineID);
    MachineWeek machineWeek = (MachineWeek)((Hashtable)zrMachine.getWeekHT()).get(machineID + theDate.toStringO);

    pro.setMachineCmachineID);
    Integer seq = new Integer(pro.getSequence());
    String sequence = seq.toStringO;
    String temp = 
    for(int nu = 0; nu < 5-sequence.length(); nu++)
    {
        temp = temp + "0";
    }
    sequence = temp + sequence;

    String strTaskID = pro.getPartsID() + "_" + pro.getMachine() + "_" + ZRComm.date2String(pro.getLFT()) + sequence;
    pro.setTaskID(strTaskID);

    machineWeek.addLoad(pro);

    //out put the information
    unitID = machine.getEquipmentID();
    start = machineWeek.getCalDateO;
    end = new Date(start.getTime() + this.intCapacityUnit*24*3600*1000 -1);
    taskID = "undefined";
    totalTime = pro.getTotalTimeO;
    mnfOrderID = orderID;
    divideNum = 0;
else
{
    boolean flag = true;
    Enumeration eMachine = vMachine.elements();
    while(eMachine.hasMoreElements() &amp;&amp; flag)
    {
        EquipmentRecord machine = (EquipmentRecord)eMachine.nextElement();
        String machineID = machine.getEquipmentID();
        ZRMachine zrMachine = (ZRMachine)this.htMachineWeek.get(machineID);
        MachineWeek machineWeek = (MachineWeek)((Hashtable)zrMachine.getWeekHT()).get(machineID + this.theDate.toString());
        if(machineWeek.getStandardCapacity() &gt;= machineWeek.getCurrentLoad())
        {
            pro.setMachine(machineID);
            Integer seq = new Integer(pro.getSequence());
            String sequence = seq.toString();
            String temp = "";
            for(int nu = 0; nu &lt; 5-sequence.length();nu++)
            {
                temp = temp + "0";
            }
            sequence = temp + sequence;
            String strTaskID = pro.getPartsID() + "_" + pro.getMachine() + "_" + ZRComm.date2String(pro.getLFT()) + sequence;
            pro.setTaskID(strTaskID);
            machineWeek.addLoad(pro);
            unitID = machine.getEquipmentID();
            start = machineWeek.getCalDate();
            end = new Date(start.getTime() + this.intCapacityUnit*24*3600*1000-1);
            taskID = "undefined";
            totalTime = pro.getTotalTime();
            mnfOrderID = orderID;
            devideNum = 0;
            flag = false;
        }
        else
        {
flag = true;

else if(this.intCapacityUnit == 7 && strPlanUint.equals("W"))
{
    ZRWorkcenter workcenter = (ZRWorkcenter)this.htWorkCenterWeek.get(workcenterID);
    System.out.println(logDate.toString()+":There is no corresponding workcenter.
\n");
    long weekNum = (pro.getLFT().getTime() - this.datePlanBase.getTime())/(this.intCapacityUnit*24*3600*1000);
    Date theDate = new Date(this.datePlanBase.getTime() + weekNum *
    (this.intCapacityUnit*24*3600*1000));
    Date lastDay = new Date(this.datePlanBase.getTime() +
    this.intPlanDuration*24*3600*1000);
    if(theDate.after(lastDay))
    {
        weekNum = this.intPlanDuration/this.intCapacityUnit;
        theDate = new Date(this.datePlanBase.getTime() + weekNum *
        (this.intCapacityUnit*24*3600*1000));
    }
    Hashtable htWorkcenterWeek = workcenter.getWeekHTQ;
    Integer seq = new Integer(pro.getSequence());
    String sequence = seq.toString();
    String temp = "";
    for(int nu = 0; nu< 5-sequence.length();nu++)
    {
        temp = temp + "0";
    }
    sequence = temp + sequence;
    String strTaskID = pro.getPartsID() + "_" + pro.getWorkcenter() + "_" +
    ZRComm.date2String(pro.getLFT()) + sequence;
    pro.setTaskID(strTaskID);
    ((WorkcenterWeek)htWorkcenterWeek.get(workcenterID +
    theDate.toString())).addLoad(pro);
unitID = workcenterID;
start = theDate;
end = new Date(start.getTime() + this.intCapacityUnit*24*3600*1000 -1);
taskID = "undefined";
totalTime = pro.getTotalTime();
mnfOrderID = orderID;
devideNum = 0;
}
else
{
}
}//end add load process

}//end if the order is in the term
else
{
   //the order is out of range
   System.out.println("The Order is out of range. orderID="+orderID+"\n");
}
}//end search in all the orders
}