MODELING BILL-OF-MATERIAL WITH TREE DATA STRUCTURE: CASE STUDY IN FURNITURE MANUFACTURER

Toni Prahasto
Jurusan Teknik Mesin
Universitas Diponegoro Semarang
Jl. Prof Sudarto, SH., Semarang
toni_prahasto@yahoo.com

Abstract

This paper presents a modeling of Bill-of-Material with tree data structure. The BOM represents wooden furniture products. The management of BOM is incorporated into an MRP software which is specially built for a furniture manufacturer. The tree data structure is approached with an object oriented programming to provide the creation and modification of the data. The tree object is designed so that a downstream programmer can create an application with high productivity, using the BOM object of course. Legality of the development is ensured by adapting open source resources, i.e. MySQL database engine, PHP server script, and client-side Javascript. The BOM object is used extensively in the MRP software that is being developed. A couple of screenshots are presented to demonstrate the ease of creation and manipulation of Bill-of-Material. The proper approach of modeling BOM with tree structure allows the programmer to reach high productivity during the development of the aforementioned MRP customized software.
Keyword : *Modeling, Bill of Material, Tree Data Structure*

INTRODUCTION

An owner of a furniture manufacturer, say Mr. X, approached the author to improve the department of Production Control and Inventory Control (PPIC) in his factory. The owner sensed that his factory was running inefficient and haunted by repeating delays. The delay caused a significant increase of complaints from the customer, leading to loss of dollars due to penalty and decrease of good will from the customer. The inefficiency raised unnecessary expenses and reduced productivity. In short, Mr. X was afraid that all of these caused his factory being forced out from the business.

Mr. X and the author agreed that the department of PPIC of the factory was not functioning properly, if not at all. An in-house customized MRP system will be introduced to remedy all of the problem within the department of PPIC.

This paper presents a part of the development of the MRP for the factory above. The content is focused and limited to the modeling of the Bill-of-Material using the Tree data-structure.

MATERIAL REQUIREMENT PLANNING

Material Requirements Planning (MRP) is a method (read software) based production planning and inventory control system used to manage manufacturing processes. Although it is not common nowadays, it is possible to conduct MRP by hand as well. An MRP system is intended to simultaneously meet three objectives:

1. Ensure materials and products are available for production and delivery to customers.
2. Maintain the lowest possible level of inventory.
3. Plan manufacturing activities, delivery schedules and purchasing activities.

Manufacturing organizations, whatever their products, face the same daily practical problem - that customers want products to be available in a shorter time than it takes to make them. This means that some level of planning is required.

Companies need to control the types and quantities of materials they purchase, plan which products are to be produced and in what quantities and ensure that they are able to meet current and future customer demand, all at the lowest possible cost. Making a bad decision in any of these areas will make the company lose money.

MRP is a tool to ensure the realization of powerful PPIC in the organization. MRP provides answers for several questions:

1. What items are required?
2. How many are required?
3. When are they required?

MRP are applied both to items that are purchased from outside suppliers and to sub-assemblies, produced internally, that are components of more complex items. The data include:

1. The end item (or items) being created. This is sometimes called Independent Demand, or Level "0" on BOM (Bill of materials).
2. How much is required at a time.
3. When the quantities are required to meet demand.
4. Inventory status records. Records of net materials available for use already in stock (on hand) and materials on order from suppliers.
5. Bills of materials. Details of the materials, components and subassemblies required to make each product.

MRP provides two outputs reports: (1) the "Recommended Production Schedule" which lays out a detailed schedule of the required minimum start and completion dates, with quantities, for each step of the Routing and Bill Of Material required to satisfy the demand from the MPS, and (2) the "Recommended Purchasing Schedule". This lays out both the dates that the purchased items should be received into the facility AND the dates that the Purchase orders, or Blanket Order Release should occur to match the production schedules. Figure 1 shows the pieces of information that form the MRP software that is being developed. This paper will focus on the Bill-of-Material and its data structure.

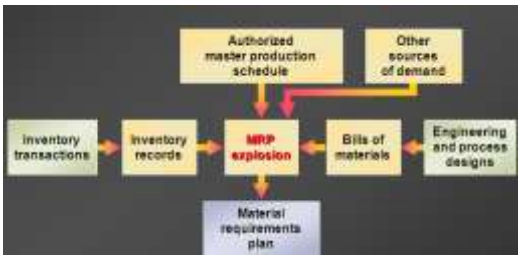

Figure 1. MRP and Bills of Materials

BILL OF MATERIAL

A manufacturing bill of material (MBOM) is a type of bill of material reflecting the product as planned by manufacturing engineering, also referred to as the "as manufactured" or "as built" bill of material. This type of Manufacturing Bill of Materials may and most times does differ from an engineered Bill of Materials. The As built structure is meant to reflect how an end item is manufactured on the shop floor. It is a list of the parts, materials, documents and tools required in the manufacture of a product. This BOM includes material that is contained in drawing notes and typically also includes an indicator of whether manufacturing intends on buying or making the indicated material. The MBOM includes any synthetic part numbers and is the driver for all manufacturing requirements in MRP and ERP Systems. Figure 2 shows a typical part list of a chair. The BOM of the chair is shown in Figure 3.

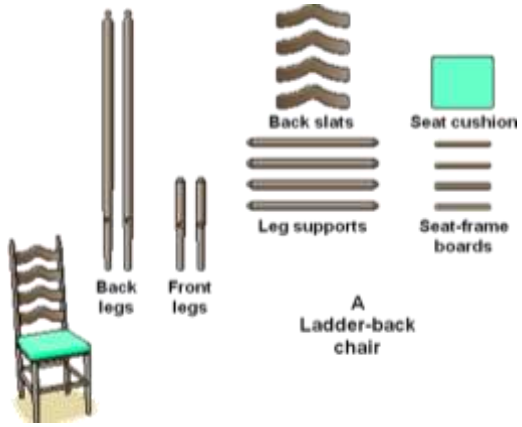

Figure 2. Part List of A Chair

Figure 3. BOM of Chair

TREE DATA STRUCTURE AND ABSTRACTION

Figure xx indicates that the structure of BOM resembles a tree. Therefore, the data structure of the BOM is best represented by the tree data structure. A tree structure is a way of representing the hierarchical nature of a structure in a graphical form. It is named a "tree structure" because the graph looks a bit like a tree, even though the tree is generally shown upside down compared with a real tree; that is to say with the root at the top and the leaves at the bottom. In graph theory, a tree is a connected acyclic graph (or sometimes, a connected directed acyclic graph in which every vertex has indegree 0 or 1). An acyclic graph which is not necessarily connected is sometimes called a forest (because it consists of trees).

A **node** may contain a value or a condition or represent a separate data structure or a tree of its own. Each node in a tree has zero or more child nodes, which are below it in the tree (by convention, trees grow down, not up as they do in nature). A node that has a child is called the child's parent node (or ancestor node, or superior). A node has at most one parent. The height of a node is the length of the longest downward path to a leaf from that node. The height of the root is the height of the tree. The depth of a node is the length of the path to its root (i.e., its root path). The topmost node in a tree is called the **root node**. Being the topmost node, the root node will not have parents. It is the node at which operations on the tree commonly begin (although some algorithms begin with the leaf nodes and work up ending at the root). All other nodes can be reached from it by following edges or links. (In the formal definition, each such path is also unique). In diagrams, it is typically drawn at the top. In some trees, such as heaps, the root node has special properties. Every node in a tree can be seen as the root node of the subtree rooted at that node. Nodes at the bottommost level of the tree are called **leaf nodes**. Since they are at the bottommost level, they do not have any children. An **internal node** or inner node is any node of a tree that has child nodes and is thus not a leaf node.

A **subtree** is a portion of a tree data structure that can be viewed as a complete tree in itself. Any node in a tree T, together with all the nodes below it, comprise a subtree of T. The subtree corresponding to the root node is the entire tree; the subtree corresponding to any other node is called a proper subtree (in analogy to the term proper subset). There are two basic types of trees. In an **unordered tree**, a tree is a tree in a purely structural sense — that is to say, given a node, there is no order for the children of that node. A tree on which an order is imposed — for example, by assigning different natural numbers to each edge leading to a node's children — is called an edge-labeled tree or an **ordered tree** with data structures built upon them being called ordered tree data structures. Ordered trees are by far the most common form of tree data structure. In graph theory, a tree is a connected acyclic graph. A rooted tree is such a graph with a vertex singled out as the root. In this case, any two vertices connected by an edge inherit a parent-child relationship. An acyclic graph with multiple connected components or a set of rooted trees is sometimes called a forest.

Stepping through the items of a tree, by means of the connections between parents and children, is called walking the tree, and the action is a walk of the tree. Often, an operation might be performed when a pointer arrives at a particular node.

A walk in which each parent node is traversed before its children is called a pre-order walk; a walk in which the children are traversed before their respective parents are traversed is called a post-order walk. Common operations for tree structures are: Enumerating all the items, Searching for an item, Adding a new item at a certain position on the tree, Deleting an item, Removing a whole section of a tree (called pruning), Adding a whole section to a tree (called grafting), and Finding the root for any node .

PHP SCRIPTS OF TREE STRUCTURE

The implementation uses PHP scripts for reasons of open-source environment and availability of scripts in the internet. The principal data member is shown below. The member $parent refers to the parental BOM object. The member $qtyforsubassy stores the number of part required to by the parent.

```
class BOM
{var $bomno;
var $parent;var,
$qtyforsubassy = 1;}
```

Methods for the BOM object in this development are:

BOM	getParentName
doQuery	add
edit	delete
isLeaf	getChildString
getChildNode	getStructure
printingStructure	printingStructure2
mrpExplosion	defaultleadtime
getTotalTime	

MRP SOFTWARE BEING DEVELOPed

The PHP script and the object definition of BOM are implemented in the software. Figure 4 shows a screenshot of the software interface that deals with the BOM of a furniture product. Figure 5 shows a screenshot of interface to modify the tree-structure of the BOM.

Figure 4. GUI of Management of BOM

Figure 5. GUI of Pruning and Moving Subtree

ANALYSIS AND CONCLUDING REMARK

The major problem with MRP systems is the integrity of the data. If there are any errors in the inventory data, the bill of materials (commonly referred to as 'BOM') data, or the master production schedule, then the outputted data will also be incorrect. Most MRP software recommend at least 99% data integrity for the system to give useful results. ***However, the actual integrity of our case is less than 60%***!

Another major problem with MRP systems is the requirement that the user specify how long it will take a factory to make a product from its component parts (assuming they are all available). Additionally, the system design also assumes that this "lead time" in

manufacturing will be the same each time the item is made, without regard to quantity being made, or other items being made simultaneously in the factory.

This means that other systems in the enterprise need to work properly both before implementing an MRP system, and into the future. For example systems like variety reduction and engineering which makes sure that product comes out right first time (without defects) must be in place.

Production may be in progress for some part, whose design gets changed, with customer orders in the system for both the old design, and the new one, concurrently. The overall ERP system needs to have a system of coding parts such that the MRP will correctly calculate needs and tracking for both versions. Parts must be booked into and out of stores more regularly than the MRP calculations take place. Note, these other systems can well be manual systems, but must interface to the MRP. For example, a 'walk around' stocktake done just prior to the MRP calculations can be a practical solution for a small inventory (especially if it is an "open store").

The other major drawback of MRP is that takes no account of capacity in its calculations. This means it will give results that are impossible to implement due to manpower or machine or suppler capacity constraints.

REFERENCES

1. Jay H. Heizer., (2000), *Principles of Operations Management*, Pearson Prentice Hall.
2. Krajewsky., (2000), *Operations Management*, Gale Group.
3. Michael Moncur. *Sams Teach Yourself JavaScript in 24 Hours*, 4/e, Sams Publishing, 2006.
4. Mark A. Weiss., (2002), *Data Structure and Problem Solving Using c++*, Pearson Addison Wesley, 2002.
5. Luke Welling and Laura Thomson. (2005), *PHP and MySQL Web Development*, 3/e, Sams Publishing, 2005.
6. Wikipedia, www.wikipedia.org.