Chapter 4 Architectural and Detail Design

4.0 Design Model

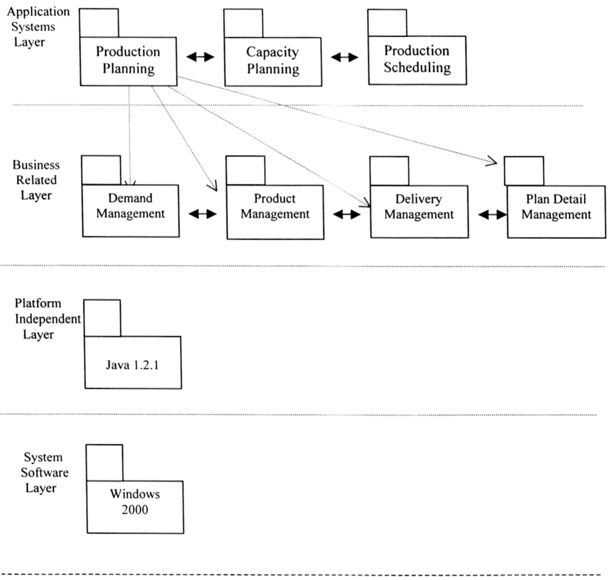
The design model is a higher-level view of solving the problem and building a solution. It consists of detailed design object classes, which correspond more closely to the final system implementation and source code in the implementation model. During design, decisions are made about how the customisable production planning tool will be solved, first at a higher level, then at increasingly detailed levels. The design model is organised into subsystems, in which lower-level subsystems and design classes are grouped. The system architecture is the overall organisation of the system into components known as the subsystems. The architecture also provides the context in which more detailed decisions are made in later design stages.

In the transition from the analysis types to the design classes, more details related to the implementation phase are revealed and incorporated. A design class represents a class at a more detailed, but still high level in the system's implementation.

4.1 System Architecture

The design model of the planning system begins with the system's layered architecture [Jacobson et al., 1997] as shown in Figure 4.1. The analysis model gives shape to the production planning system's architecture. This model is made up of the analysis types and subsystems including their relationships through traces. The architecture of this production planning tool is made up of layers of production elements that interact with each other. Each layer is built on top of another more general layer below it and providing the basis of implementation for the ones above it. A subsystem knows about the layers below it, but has no knowledge of the layers above it. Figure 4.1 below illustrates the layered architecture of this customisable planning tool with emphasis on production planning application systems.

Figure 4.1 Layered Architecture of the Customisable Production Planning System



The topmost layer, which is the application system layer, consists of different application systems in the manufacturing domain. The manufacturing domain is made up of production planning, rough cut capacity planning, production scheduling, material requirement planning, production control, statistical quality control, etc. Three such application systems are illustrated here in the development of this software. These application systems make up the family of applications within the manufacturing domain. Integration of this family of applications is a very difficult task as these applications at present are not designed to work together. This is due to the manufacturing environment which is so complex and diverse in nature not withholding the different technologies used within the production industry itself.

However, if there is a uniform architecture represented in each stage of the manufacturing process, integration can be achieved more successfully. Here, the focus is the production planning architecture in detail.

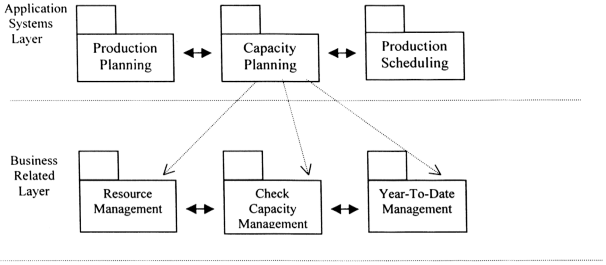
Below this application system layer is the business related layer, which consists of component systems such as demand management, product management, plan detail management and delivery management. The business related layer is made up of the component systems related to the production planning system above. These business related components exports components required by the application system which is the production planning system via the façade or interface of the component systems. These components can be reused by developers by reusing their classes and objects. The business related components are built on the platform independent layer, which is a layer below it.

The platform independent layer offers component systems that provide utility classes and platform independent services, for example, distributed object computing found on different platforms. In this production planning tool, Java 1.2.1 is used as the middleware layer because of its interoperable distributed object computing which is independent of the operating system platforms used. This will enable the created production planning tool to be more resilient to technology changes in the manufacturing industry.

The bottom layer, which is the system software layer, contains the distributed system platform that has IP connection for the networking infrastructure, commonly found in most business environments. There is actually no clear distinction between the third and the fourth layers.

A good software architecture is important for the creation of a concrete system that is tolerant to change with the most desired system functionality and smooth performance, not mentioning a minimal level of maintenance. Figure 4.2 shows the layered architecture of this customisable planning tool with emphasis on capacity planning (RCCP) application system. Under the layer on which RCCP application system is residing, is the business related layer that consists of component systems such as *Resource Management, Check Capacity Management* and *Year-To-Date Management. Resource Management* deals with the control of manpower, number of machines and total work hours whereas *Check Capacity Management* deals with available or existing capacity versus the actual capacity needed for production. *Year-To-Date Management* is the calculation of the total plan production and the corresponding production yield from the start of production to the present date, which will be shown in a line chart.

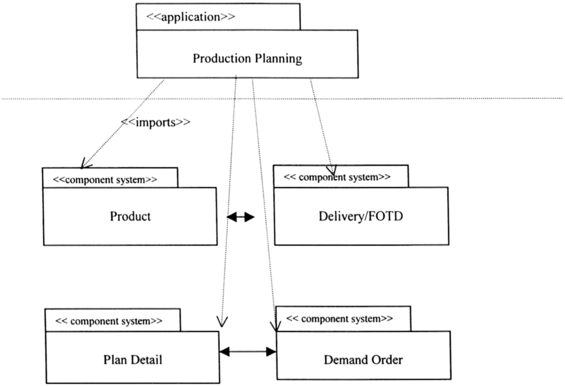
Figure 4.2 Layered Architecture of the Capacity Planning System



4.1.1 Production Planning Application System

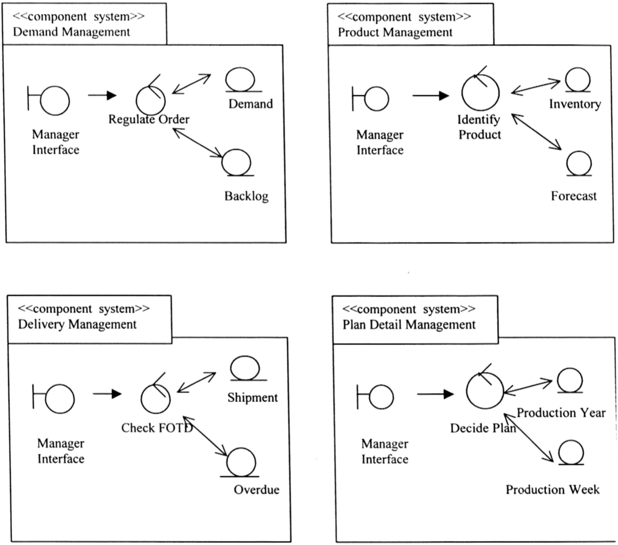
As mentioned in the beginning of this chapter, the development life cycle is iterative and incremental as the engineering process becomes clearer and takes shape when more details are added. Now that the system architecture has been defined, the design details will be shown in the form of the application systems and component systems. The application system imports through the facades from the component systems as shown in Figure 4.3. Each component in a component system will have its own objects that will form the classes when implemented and these components can be reused during the development of application systems. These components systems consist of *Product, Delivery/FOTD (Factory-On-Time-Delivery), Plan Detail and Demand Order* components.

Figure 4.3 Component Systems of the Production Planning Application System



Based on the layered architecture of the production planning, these component systems represent the business related layer. In the production planning application system, the design of the component systems can be illustrated by the boundary, control, entity, (BCE) diagrams as shown in Figure 4.4.

Figure 4.4 BCE diagram for Demand, Product, Delivery and Plan Detail Management Component Systems



4.1.2 Capacity Planning Application System

The capacity planning application system also applies the same design as the production planning application system. It is also based on the layered architecture whereby the application system also imports from the component systems. Figure 4.5 shows the component systems of the capacity planning application system. They consist of *Rough Cut* (RCCP), *Check Capacity* and *Year-To-Date* (YTD) component systems. The component systems in turn are also represented by the BCE diagrams as shown in Figure 4.6.

Figure 4.5 Component Systems of the Capacity Planning Application System

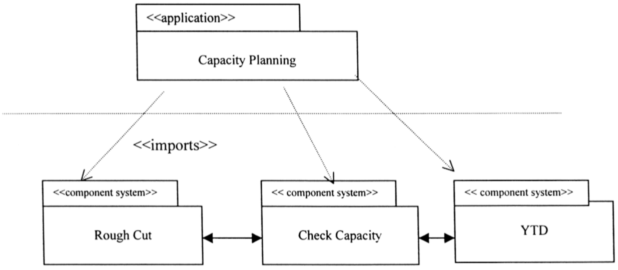
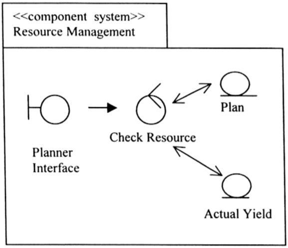
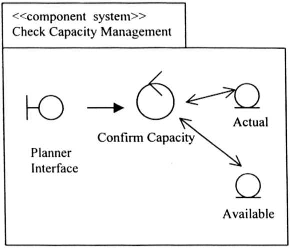
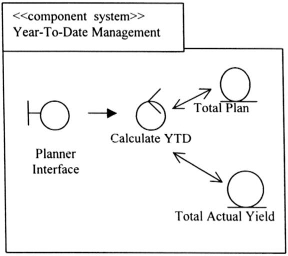


Figure 4.6 BCE diagram for Resource, Check Capacity and Year-To-Date (YTD) Management Component Systems







4.1.3 Production Scheduling Application System

The production scheduling is a straight forward application system that also imports from the product and production week component systems. The schedule is based on the actual amount planned for each production week with its respective type of product and quantity to be produced. This is the final part of this customised production planning tool whereby the production plan is actualised and products are manufactured according to plan. Figure 4.7 shows the component systems of the production scheduling application system and Figure 4.8 shows the BCE diagram for *Product Management* and *Production Week Management* component systems.

Figure 4.7 Component Systems of the Production Scheduling Application System

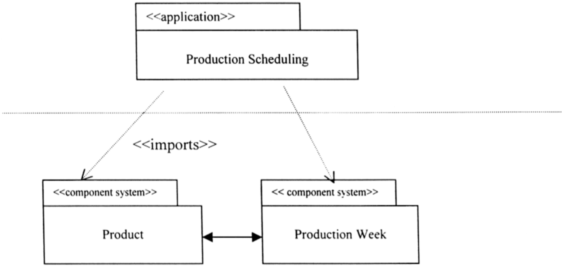
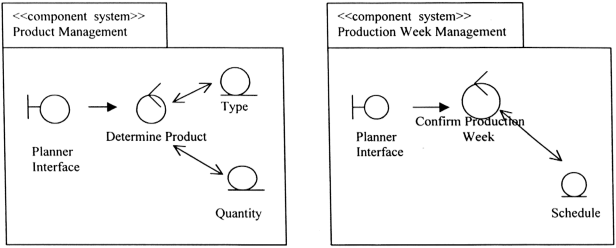


Figure 4.8 BCE diagram for Product and Production Week Management Component Systems



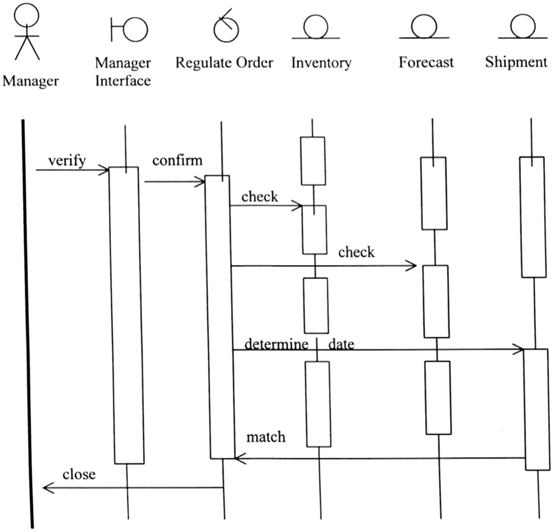
4.1.4 Sequence Diagrams

When the top-level component systems are identified, information flow among these subsystems are more clearly seen through sequence diagrams. A sequence diagram shows more detailed message flow as well as operation details before implementation. These diagrams reflect the behaviour of the objects and their activities carried out between those objects. In this customisable planning system, sequence diagrams are used to further describe how the objects perform each use case instances. Each use case is further traced through the BCE diagram and the details of the operations can be illustrated by using these sequence diagrams. The three use cases, **plan production, check capacity and schedule production** are now traced through to their respective activities in the sequence diagrams.

a. Plan Production Sequence Diagram

Figure 4.9 below shows a sequence diagram with more detailed message flow and operation activities of the plan production use case. The boundary object is the manager interface and the control object is *Regulate Order*. The entity objects are *Inventory*, *Forecast* and *Shipment*. The manager first verifies himself to the system to access it. Then he confirms the customers' orders and their order details. Inside the order also contains information about when shipment is due and the quantity to be delivered at the expected delivery date. Before the order can be executed, the system will check how much is the available inventory and the forecast for the respective products and whether the amounts are sufficient to meet those orders. If the orders are more than what is available, then the system will generate the plan to produce more quantity to meet those orders. Thus the *Regulate Order* object controls the production planning process. Once the shipment of goods are delivered, the order is then closed and the cycle continues on with new orders.

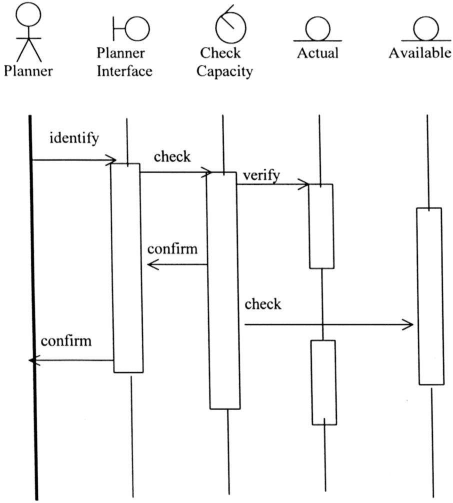
Figure 4.9 A sequence diagram describes how the objects perform the plan production use case



b. Check Capacity Sequence Diagram

Figure 4.10 below shows another sequence diagram of the capacity planning use case. The planner first identifies himself by interacting with the system through the planner interface. Then he checks on the capacity plan for the planning period via the control type, *Check Capacity*. The plan then must be verified against the *Actual Capacity* which is represented by the entity type. In the mean time, the *Capacity Plan* must be checked against the *Available Capacity* needed to proceed with the production. This *Available Capacity* is represented by another entity type. Once the *Actual Capacity* matches with *Available Capacity*, then production can start and the planner then confirms that there is sufficient capacity for the production to kick off. When the capacity plan is firmed up, then the system will confirm the production plan by indicating to the planner about whether the plant is operating under capacity, full capacity or over capacity. This is important so that the planner will be able to know the planning capacity and make the necessary adjustments to the plan.

Figure 4.10 A sequence diagram describes how the objects perform the capacity planning use case

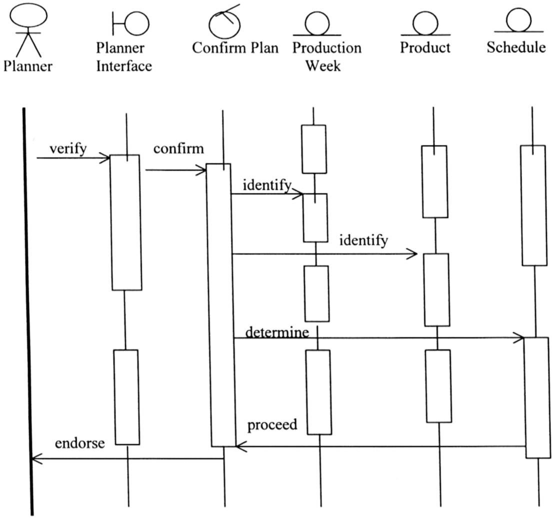


c. Schedule Production Sequence Diagram

Figure 4.11 below shows a sequence diagram of the schedule production use case. The boundary object is the planner interface and the control object is *Confirm Plan*. The entity objects are production week, product and schedule. The planner interfaces with the system to access it. He confirms the plan and identify the production week and the products to be produced for the respective weeks. The plan now controls the implementation of the production schedule as all production details must adhere to the production plan thus the plan determines the schedule.

Production scheduling will then proceed according to the plan. The plan is therefore endorsed when the product schedule is being executed. At this stage any amendments to the plan can still be accommodated before production scheduling rolls off. This will help to weather any last minute cancellation of customers' order or sudden surge in demand as scheduling has not started. Once scheduling proceed, any excess will increase production cost and shortage will erode customers' confidence as production will be behind schedule.

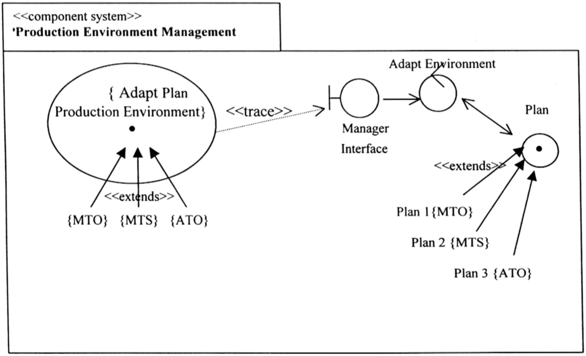
Figure 4.11 A sequence diagram describes how the objects perform the schedule production use case



4.1.5 Customisable Components

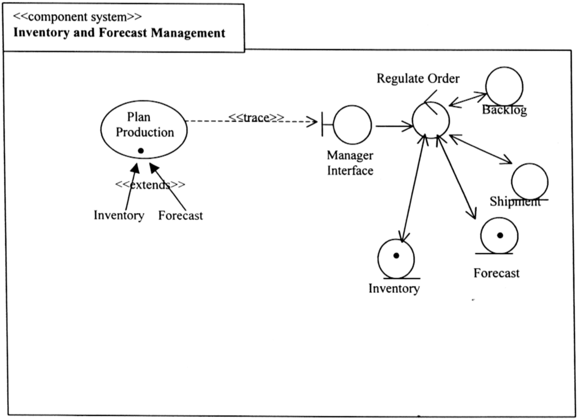
The customisable components consists of the production environment and the calculation of inventory and forecast. The production environments are Make-To-Order (MTO), Make-To-Stock (MTS) and Assemble-To-Order (ATO). The user can choose any of these environment and set accordingly for any production period at any time of the calendar year. Figure 4.12 shows the variation point that is **Adapt Plan Production Environment**, which is traced to the BCE diagram that shows in detail the plan component with three variants denoting the three types of production environments. The variability mechanism used is the extention point to express variants attached at the variation point in the plan use case and the plan object component [Jacobson, 1997].

Figure 4.12 Customised Component System for Production Environment



As for the customisation of inventory and forecast, the user can formulate his own calculation of the two entities according to their own calculation which will be set at the start of the production year. Figure 4.13 shows the variation point under the *Inventory* and *Forecast Management* component system, which is traced to the BCE diagram that shows in detail the plan production components which include inventory and forecast as two variants. The variability mechanism used here is also the extention point to express variants attached at the variation point in the plan production use case. These two variants allow the user to calculate his own inventory and forecasting methods.

Figure 4.13 Customised Component System for Inventory and Forecast



4.2 Storage Management for Production Planning

The production planning software uses the relational database management system (RDBMS) to store and manage data. The RDBMS used is Microsoft Access for storing and managing the production planning data. A RDBMS is used because of the following reasons :

- most manufacturing organisations are still using relational database due to its popularity in contrast to object-oriented database though the latter are more powerful with better performance.
- the structural and data independence allows the examination of the model's logical structure without considering the physical aspects of data storage and retrieval thus data management becomes much easier.
- the relational model provides a minimum level of controlled redundancy. Redundant data can lead to data inconsistency, data anomalies and a lack of data integrity.
- coding is reduced with the use of the powerful and flexible Structured Query Language (SQL).

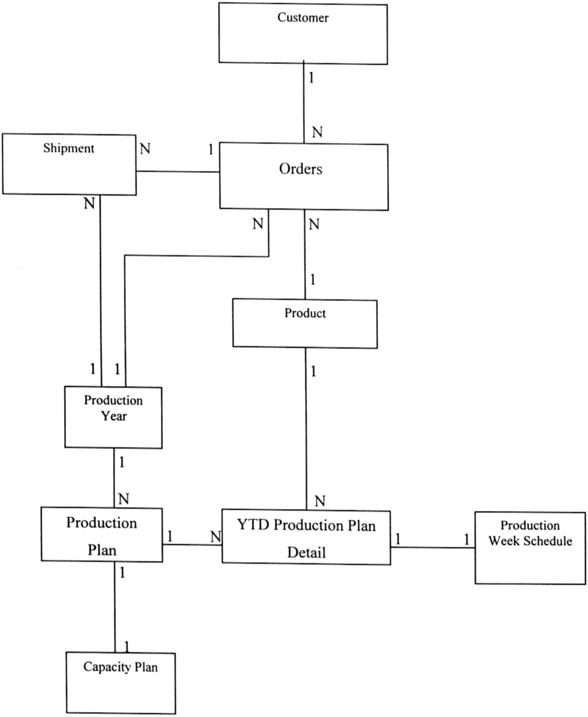
The database design starts with the use of the most common data model, which is the entity-relationship (E-R) model, the core of the database design. As the structure of a good database begins with a good database design technique, E-R diagram is the first step towards data modelling. A good model yield a good database design that will yield a good application which is why this model is used to develop the customisable production planning tool. For connectivity to the data source, the planning system will use the Java

Database Connectivity-Open Database Connectivity, JDBC-ODBC connection with SQL to query and fetch the results from the database.

4.2.1 Entity-Relationship Diagram

The database design of this production planning tool uses the entity-relationship (E-R) modelling method to design the structure of the database. The design begins by identifying appropriate entities and attributes and the relationships among those entities. These entities are derived from the component systems found in the system's analysis and design mentioned in the earlier sections. Figure 3.20 shows the entities which are identified as *Customer, Order, Shipment, Product, Production Plan Detail, Production Week Schedule, Capacity Plan, Production Year and Production Plan*. The relationships which show the association between entities are denoted by the One-To-One (1:1) and One-To-Many (1:N) relationships. The tables are related to each other and they can be traced through these relationships, for example, each production year has many production plans and each production plan has many production plan details. However, one production plan has only one capacity plan. A customer can placed many orders for any amount of whatever products he is ordering, but an order belongs to only one customer. There are many orders for one product and one order has many shipments and each production plan detail is related to one production week schedule. The details of the relationships can be seen in Figure 4.12.

Figure 4.12 The E-R Diagram for Production Planning



4.2.2 Storage Management Structure

The database contain nine tables which are the entities in the E-R diagram. The tables contain the respective fields which are the attributes in the E-R diagram. The tables have all been normalised by enforcing referential integrity in each of the relationships. The relationships of the tables are traced by the primary and foreign key fields which are underlined in the tables. The tables and fields are :

1. CUSTOMER (Cust_ID, Cust_Name, Cust_Add, Tel_No, Fax_No, E_Mail, Contact_Per)
2. PRODUCT (Prod_ID, Prod_Name, Prod_Desrpt)
3. ORDER (Cust_ID, Prod_ID, Ord_Qty, Ord_Date, Exp_Del_Date,)
4. SHIPMENT (Cust_ID, Prod_ID, Qty_Deld, Date_Deld)
5. PRODUCTION WEEK SCHEDULE (Prod_Yr_ID, Prod_Wk, Prod_ID, Qty)
6. YTD PRODUCTION PLAN DETAIL(Prod_Yr_ID, Prod_Wk, Prod_ID, Invent, Forecast, Yield)
7. CAPACITY PLAN (Prod_Yr_ID, Prod_Wk, No_of_Mac, Manpower, Opr_Hrs)
8. PRODUCTION YEAR (Prod_Yr_ID, Start_Date, End_Date)
9. PRODUCTION PLAN (Prod_Yr_ID, Prod_Wk, Desrpt)

4.2.3 Security Access Login

The login access into the database is restricted and controlled by the planning manager or administrator. Access to the database is restricted through access privileges set by the manager. Managers can grant and deny permissions to perform any action in the system by the planners, clerks and general staff. This security feature is to governed unauthorised access into the planning system as well as to safeguard data manipulation in the database. There are two tables which are *User* and *Access* with a 1 : M relationship.

The manager can set limited access into the database whereby only valid users are authorised to access the part of the database that is applicable to his or her area of work. The table names and fields for this login access into the planning tool are :

1. USER (User_ID, User_Name, Password, Role)
2. ACCESS (Role, Access_Mode, Module)

4.3 Summary

The analysis and design of the customisable production planning tool is based on the object-oriented software engineering (OOSE) modelling technique developed by Ivar Jacobson [Jacobson et al., 1992]. The system's requirements are captured through the use cases which show what the customisable tool will do in the abstract level. While the use cases describe what the tool is supposed to do, the various object models describe how the customisable tool is architectured and designed for a smooth implementation so as to meet use requirements. The development of this tool is done through the iterative and incremental approach using the BCE types to show the interaction of objects of the various classes as well as the information flows among these objects. The objects that perform the use case instances are presented in collaboration or sequence diagrams for each of the use cases. It then proceeds with the design model based on a layered architecture which forms the groundwork for the construction of the engineering process. The architecture also saw the software taking shape with the development of the application systems which import from the component systems. The use cases can be traced to the analysis model through to the design model which defines the implementation of the planning system.

The customisation features allow the user to choose from three production environments. These production environments can be set at any time of the year following the production year. This is represented by three variant points in the plan use case model. In addition, the user can also customise their calculation of inventory and forecasting methods that make use of variability mechanisms which are the extention points, that is actually a form of <<generalisation>> [Jacobson *et al.*, 1992; Griss, 1995d]. The customisation variants are denoted by extension points in the use case and object components. The mechanism provides a powerful way of making components more generic, thus enhancing reusability for future enhancement.

The storage management of the production planning software uses the relational database management system to store and manipulate data using the SQL. The storage design used the most common data model, which is the entity-relationship (E-R) model, the core of the database design. The entities formed the nine tables with their respective fields. Besides, login access into the database is controlled by the manager or the administrator. He can set the various levels of access for the different categories of staff. The design model is traced to the analysis model and serves as a blueprint for the implementation of this customisable production planning tool. The next chapter looks into the implementation model of this customisable production planning tool.