Analyzing IT Supported Production Control by Relating Petri Nets and UML Static Structure Diagrams

Henk Pels

Analyzing IT supported production control by relating Petri Nets and UML static structure diagrams.

Dr. ir. Henk Jan Pels

Technische Universiteit Eindhoven
h.j.pels@tue.nl

Abstract.
A method to model the interaction between a production control process and an information system is presented. Colored Petri Nets are used to model the process and UML static structure. When the tokens in the internet are modeled as objects in the data model, the transitions in the process model can be specified as formal expressions over the data model. Thus the model verifies the consistency between the process and the information system and can be used as formal specification for e.g. an ERP implementation.

Keywords: process modeling, data modeling, ERP implementation.

1 Research Problem

Understanding the interaction between a business process and an information system has always been a difficult issue. When implementing ERP systems the users do not understand the logic of the Information System and the system integrators do not understand the business process. This problem calls for a formal technique to model, understand and discuss the interaction between the business process and the information system. Such a technique should enable to discuss the essential requirements for production control software to support the business strategy.

2 Approach

Petri Nets [Petri, 1962] provide a formal process modeling language. UML Static Structure diagrams [OMG, 2005] are a formal language to model the information that is relevant for a specific business process. In colored Petri Nets [Jensen, 1992] the tokens have attributes, which enables to consider them as objects. If a UML static structure model is used to specify the types of tokens and their state space, the transitions can be specified as pre- and post-conditions over this state space. Section 3 explains the modeling principles using the example of a simple assemble to order production situation with a one-level bill of material. In section 4 we demonstrate the suitability for more complex situations on a manufacture to order with a multi-level BOM. Then in section 5 we analyze the well-known MRP planning situation with

multilevel BOM and stock of parts. In section 6 the results of the research will be discussed.

3 Assemble to order process

3.1 Petri Net

We introduce the modeling approach with a relatively simple situation: the Assemble to Order process. Figure 1 shows the AtO process modeled in a Petri Net, using the CPN notation [Jensen, 1992]. Places are denoted by ovals, labeled with a unique name. The label below a place is the class of the tokens in the place. Places essentially model delays in the process. Consequently their names correspond to states of tokens. Transitions are modeled as rectangles which correspond to decisions in the business process. The label on the arrow is used as identifier for the selected or created tokens in the pre and post conditions. Transitions model decisions to be taken in the process.

The basic principle of Petri Nets is that a transition fires if a token is present in each of its input places. This means that transitions with multiple input arrows model synchronization points. If more than one token is present in a place, the selection is arbitrary. In colored Petri Nets the firing of a transition also requires that the selected set of tokens satisfies the precondition. The post-condition specifies the change of state of the system. In particular it specifies which output arrow a token will follow. Every token is always in one and only one place.

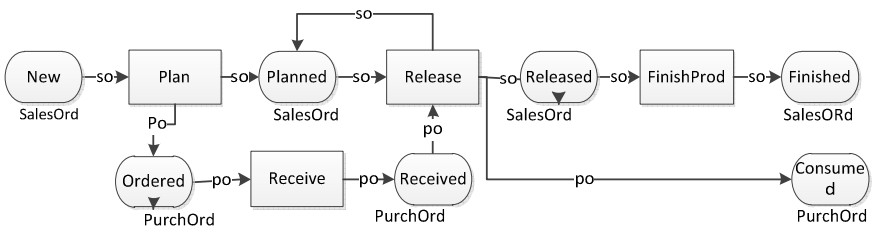

Fig. 1. Petri Net for Assemble to Order process.

The process in figure 1 starts in the place New. Transition Plan moves the sales order to the Planned place. At the same time for each material as required for assembling the ordered product, a purchase order is created and moved to the Ordered place. The Release transition fires if it finds a Received purchase order that matches a Planned sales order. Release checks whether all required materials for the sales order have been received. If so it moves the sales order to place Released, else it places the sales order back in the Planned place. The purchase order is moved to place Consumed. Transition Produce takes orders from place Released and moves them to place Finished. Note that the transition Finished models the decision to accept an assembled product as finished. During actual production the sales order remains in state 'Released". The Petri Net models the process control decisions and the delays between them, not the process activities, like e.g. the assembly work.

3.2 The data model

The purpose of the model is to analyze the control decisions to be taken to create the desired behavior of the business process. The evolving state of the system is recorded as the arrival periods in the places. Figure 2 shows the UML static structure diagram (further referred to as data structure) for the AtO process.

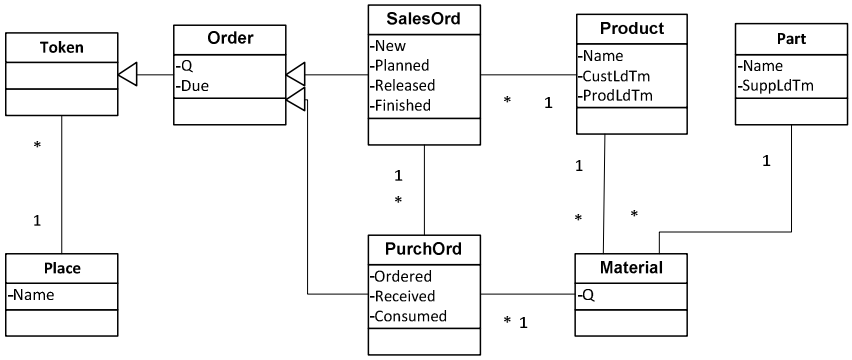

Fig. 2. Data structure for Assemble to Order process.

Relationships are represented as links, which are in fact attributes with objects as value. For the precise interpretation of the data model we refer to [OMG, 2005] and [Pels, 2006]. We just mention the derived attribute, indicated with a '/' before the name, meaning that the value of this attribute can be calculated from the current state of the total set of objects. The formula for calculation is specified outside the diagram.

In specifying constraints, pre- and post-conditions the dot notation from UML constraint language allows readable formal expression. If e.g. s1 is a SalesOrd object, then s1.Product is the ordered product.

In specifying constraints, pre- and post-conditions the dot notation from UML constraint language allows readable formal expressions. If e.g. s1 is a SalesOrd object, then s1.Product is the ordered product.

Object class Product models the products that can be ordered. Attribute ProdLdTm records the production lead time. Part models the parts used in the products. Each part has SuppLdTm for supplier lead-time. Material specifies the use of a specific part in a specific product. Attribute Q is the quantity used. The common properties of sales orders and purchase orders are modeled in class Order as generalization of SalesOrd and Purchord. Q is the quantity ordered, New the period of creation and Due the period due to be delivered. SalesOrd and PurchOrd have similar attributes for the period where phases start. Classes Token and Place connect the data structure to the Petri Net. Since they are common for this way of process-data modeling, they will be supposed to be implicitly specified in all further data structures.

3.3 The transitions

Transitions are specified in terms of pre- and post-conditions. The precondition specifies which tokens are selected from the input places. The post-condition specifies the state changes caused by the transition. Since the post condition must reason over the old as well as over the new state, some special operators must be added to standard predicate logic:

- The <- operator is used to move tokens. The left operand specifies the output place, the right operand the token to be moved. If the right operand is not the label of one of the input arcs, a new token is created,
- The := operator is used to specify that in the new state the left operand has the value resulting from the right operand. All variables in the right operand refer to the new state, unless they are preceded by a ~, in which case they refer to the old state,

Below we specify transitions Plan and Receive in terms of the data model.

```
Plan %for each material create a purchase order%
POST   Planned <- so ∧ ∀ m ∈ so.Product.Material [Ordered <- po ∧
po.SalesOrd := so ∧  po.Material := m ∧ po.Q := m.Q * po.SalesOrd.Q ∧
po.Ordered := so.New ∧ po.Due = so.New + m.Part.SupLdTime];
Receive %purchase orders are processed in order of delivery%
PRE po.Due = Min(n.Due: n ∈ Ordered.Token)%Ordered is a Place object,
so Ordered.Token is the set of tokens in this place%
POST Received <- po;
```

The formulas above show how the full behavior of the process can be specified in formal language. It verifies that process and data model are consistent. Note that the names of products and parts appear to be not relevant for the process. However, they are relevant in the user interface.

3.4 Discussion of the modeling approach

The colored Petri Net brings a process model that clearly shows the main characteristics of the process. It shows two parallel process lines for the sales order and the purchase order, that come together in the release transition. The data model enables an abstract, complete and unambiguous specification of control decisions in terms of information needed. So a consistent model verifies that the data model fits the process. If the systems integrator is ordered to implement the software system consistent with the data model, it is guaranteed that it will support the business process. Using this approach can reduce the risk and the cost for ERP implementation dramatically.

4 Manufacture to Order Process

4.1 MtO Process Model

The process model in Figure 3 shows that the MtO process has a third process line: the manufacturing process. A new sales order generates a manufacturing order for the product. The Plan transition not only creates purchase orders, it also creates manufacturing orders for non-purchased materials. A manufacturing order is released when all required materials, purchased and manufactured, are available. Production will start only when sufficient capacity is available. If not the InWork period is incremented and the manufacturing order is fed back to the Released place. Finished manufacturing orders are fed back to Received, unless their product is a customer ordered product. In that case it proceeds to Delivered. The sales order waits in Accepted until the end product has been manufactured and is then expedited to the customer.

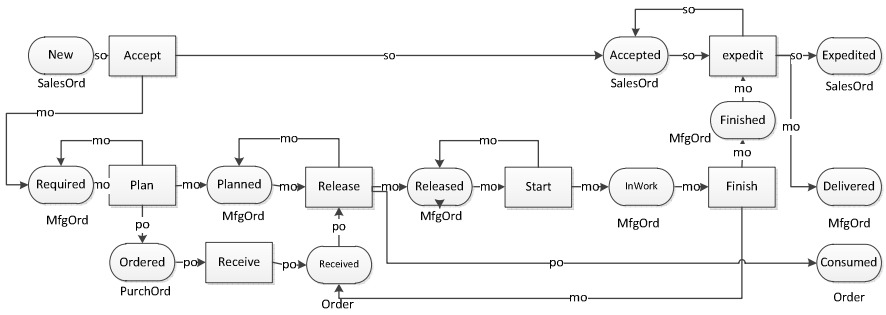

Fig. 3. MtO process

4.2 MtO Data Structure

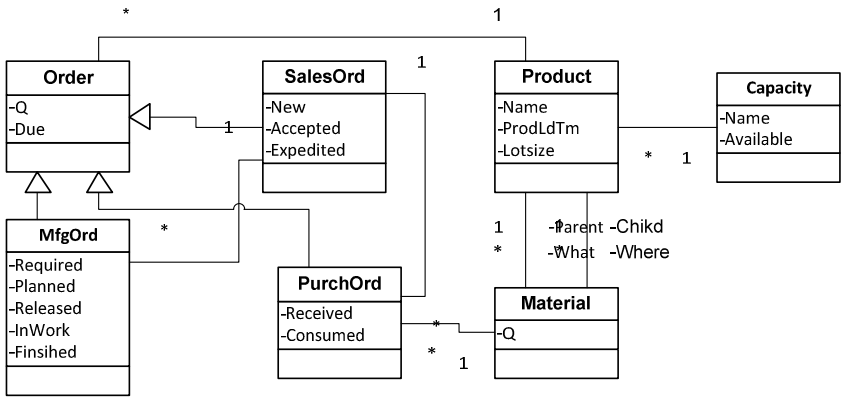

Fig. 4. MtO data structure

In figure 4 object class MfgOrd represents the manufacturing orders with their specific states. Material now records the multi-level BOM by connecting the Child Product to the Parent Product. The set of Material objects linked to the same Parent is the What (used), while the set of material objects linked to the same Child is the Where (used). New is the class Capacity, to specify capacity limits.

The MtO process as modeled above is perfectly just in time: materials are produced or purchased only when needed for a sales order and production and purchase orders are always uniquely linked to a sales order. The question now is what happens if the concept of stock is introduced is discussed in the next section.

5 MRP Process

5.1 MRP Process Model

Material Requirements Planning was introduced by APICS as a concept for computer aided production control. Manufacture to stock was still the most usual way of production, making stock control the central issue of production planning. APICS explained the MRP principles using the concepts of Net Requirements, Gross Requirements, Scheduled Receipts, Available Balance, Planned Orders Due and Planned Release [Bertrand, Wijngaard, Wortmann, 1990]. These concepts are rather abstract and difficult to understand, so let us try to achieve better understanding by extending the MtO model with stock. For simplicity we use unlimited capacity.

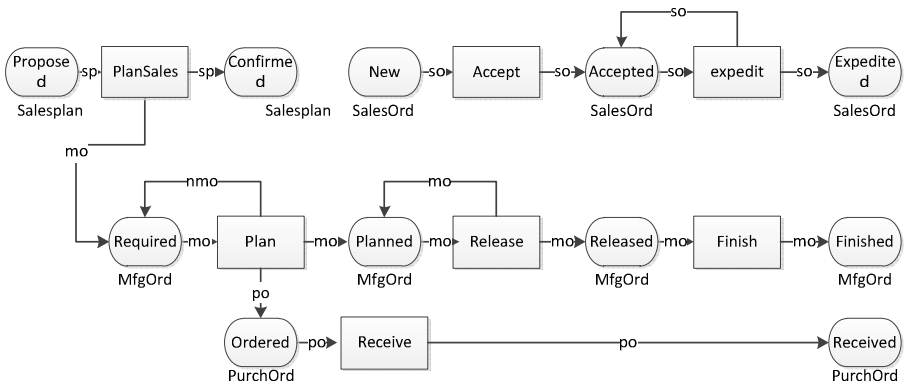

Fig. 5. MRP process.

Figure 5 shows the process model. Sales planning transforms proposed sales plans into confirmed and generates a manufacturing order for the product. When MfgOrd's are planned the BOM is exploded, PurchOrd's are generated for purchased materials and new Required MfgOrd's are generated for other materials. Planned MfgOrd's are released for production when sufficient stock of materials is available. New SalesOrd's are accepted unconditionally and expedited if sufficient stock of the end

product is found. If not the expedited period is incremented and the order is returned to accepted.

5.2 MRP Data Structure

The datamodel In figure 6 shows the MRP concept as class MRP.

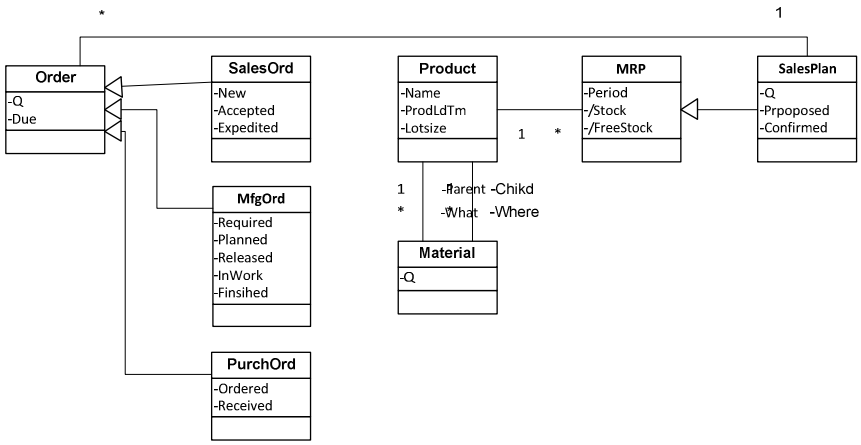

Fig. 6. MRP data structure

Each MRP object specifies two stock-levels for each period: Stock for the stock level that results from the planning and FreeStock the stock level taking sales orders into account. If sales orders are different from sales plan FreeStock differs from Stock. Stock levels are derived attributes since they can be calculated as inflow minus outflow.

5.3 MRP Transitions

The transition Plan is most specific for MRP, since it generates manufacturing or purchase orders only if the stock would get below zero. Also orders are planned backward, meaning that order period is the plan period minus the product lead-time.

```
Plan
POST Plan <- mo ∧ mo.Planned := mo.Required  ∧ ∀ m ∈ mo.Product.What:
IF ~mo.Stock < mo.Q THEN IF m.Child.What = {} %purchase part% Then po <-
Ordered ∧ po.Q = mo.Q * m.Q ∧ po.Due := mo.Due ∧ po.Ordered := mo.Due -
m.Child.ProdLdTm ELSE %mfg part% MfgOrd <- nmo ∧ nmo.Q := mo.Q * m.Q ∧
nmo.Due := mo.Due ∧ nmo.Ordered := mo.Due - m.Child.ProdLdTm ENDIF
ENDIF;
```

5.4 MRP analysis

Now comparing the MRP process with the MtO process, the striking difference is that any synchronization between the three process lines has disappeared. This is because stock eliminates the direct links between the different order types, as can be seen in the data structure. As a result an expensive process is needed to control stock: the FreeStock levels must be monitored frequently in order to adapt plans to actual sales. However, stock causes longer lead-times and thus delay between a change in plan and the change of stock. From control theory it is known that such a delay may cause oscillations, increasing uncertainty and again increasing stock levels. This is a very strong argument to outsource the stock control problem to the suppliers and to apply MtO where possible.

Another surprise is that that original APICS concepts are not explicitly in the data structure, but boil down to just Stock and FreeStock. This means that they are not essential for understanding, indicating that our modeling approach leads to easier understanding of the problem. This does fear that MRP software designers, systems integrators and users may have very different and even conflicting understandings of the logic, without being able to discuss or detect these conflicts.

6 Results

The contribution of the modeling approach is demonstrated in that it enables us to explain and discuss production control processes in relation to the IT support. Even a complex process like MRP can be explained with a relatively simple process and data structure. A surprising result is that the analysis generates critics to a long established and extensively published mechanism like MRP. The method has been applied in ERP selection and implementation projects and appeared to contribute considerably to smooth and effective implementation.

References

1. [Aalst ea, 2002] W.M.P. van der Aalst and K.M. van Hee. Workflow Management: Models, Methods,and Systems. MIT press, Cambridge, MA, 2002.
2. [Bertrand, Wijngaard, Wortmann, 1990], J.W.M. Bertrand, J. Wijngaard, J.C. Wortmann, Production control: A structural and design oriented approach, Elsevier, 1990.
3. [Jensen, 1992] K. Jensen, Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. EATCS, Monographs on Theoretical Comp. Science, Springer, Berlin, 1992
4. [OMG, 2005] Object Management Group (OMG), OCL 2.0 Specification. OMG document ptc/2005-06-06, June 2005.
5. [Pels, 2006], "Classification hierarchies for product data modelling", Production Planning and Control, vol. 17, nr. 4, pp.367-377, june 2006.
6. [Pels ea, 2007] H.J. Pels, J. Goossenaerts, A conceptual modeling technique for discrete event simulation of operational processes, in: Advances in production management systems; (APMS2007) september 17-19, Linköping, Sweden, 2007, pp. 305-312,
7. [Petri, 1962] C.A. Petri. Kommunikation mit Automaten. PhD thesis, Institut für instrumentelle Mathematik, Bonn, 1962.