An analysis of production control tasks is given, in order to derive requirements for a language for programming production controllers. The programming of the controllers is divided into two related tasks: the specification of the controller environment (the description of the capabilities of the agents that the controller can use to make products and the description of how products must be produced), and specification of the internal control tasks like scheduling and command dispatching. Three temporal operators are introduced to describe the ordering of processing steps.

Keywords: Programming languages, Computer integrated manufacturing, Reference model, Assembly, Production controller, Production.

1. Introduction

The programming of flexible controllers is a difficult and complex task. There exist many robot programming languages [3] and also some languages for higher level controllers [1,4,6]. However, little research has been done on specifying requirements for these languages. This paper presents an analysis of the control tasks in an automated manufacturing system. This analysis is used to derive requirements and concepts for a general production control language.

2. A Reference Model

A reference model represents a system as a set of cooperating components and defines the tasks of these components. The analysis of the production control system (PCS) is based on the reference model of Biemans [2]. This reference model has two main components: the PCS and its Management. The PCS is concerned with the manufacturing of products, Management is concerned with the design of products, monitoring of the performance of the production system, design of machines etc. In this paper we will concentrate on the PCS. The PCS is divided into six hierarchical layers as shown in Fig. 1. Each layer executes its tasks using the services of the layer underneath it.

The Factory Controller gets a command to produce a certain amount of products before a certain point of time and to deliver the products to a customer or an internal stock. The Factory Controller determines the time slot wherein the products must be made and leaves the actual production to the Workcell.
<table>
<thead>
<tr>
<th>Control Level</th>
<th>Control Service</th>
<th>Task of Controller</th>
<th>Examples of Commands</th>
<th>Examples of Reports</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company Controller</td>
<td>DISPATCH PRODUCTS AT DUE DATES</td>
<td>DETERMINE FIRM AND FORECAST PRODUCTION TARGETS</td>
<td>DISPATCH RADIOS D AT DAY E TO F.</td>
<td>RADIOS C DISPATCHED TO D.</td>
</tr>
<tr>
<td>Factory Controller</td>
<td>PROCESS ITEM IN ALLOTTED TIME SLOT</td>
<td>PLAN PRODUCTION WHICH ITEMS TO PROCESS IN WHICH TIME SLOT TO DISPATCH PRODUCTS TIMELY</td>
<td>PROCESS BOARDS G BETWEEN DAY H AND I.</td>
<td>E BOARDS F MADE.</td>
</tr>
<tr>
<td>Cell/Line Controller</td>
<td>EXECUTE OPERATION ON PART</td>
<td>SCHEDULE WHEN, WHERE WHICH OPERATIONS ARE EXECUTED ON PARTS</td>
<td>PUT COMPONENTS H, I, J ON BOARD K.</td>
<td>QUALITY BOARD F IS OK.</td>
</tr>
<tr>
<td>Workstation Controller</td>
<td>EXECUTE PROCESSING STEP ON OBJECT</td>
<td>COORDINATE THE EXECUTION OF PROCESSING STEPS ON OBJECTS TO REALISE OPERATIONS</td>
<td>MOVE OBJECT M TO N, MAX PRESSURE O.</td>
<td>OBJECT WITH DIMENSIONS G, H AT I.</td>
</tr>
<tr>
<td>Automation Module Controller</td>
<td>FOLLOW JOINT PATH</td>
<td>ISSUE CONTROL SIGNALS SO THAT JOINT VARIABLES ARE SERVOED BY PHYSICAL PARAMETERS</td>
<td>CHANGE JOINT ANGLE P TO Q.</td>
<td>LINE SEGMENT AT K WITH VELOCITY J.</td>
</tr>
<tr>
<td>Device Controller</td>
<td>MODULATE PHYSICAL PARAMETERS</td>
<td>SENSING TASKS, SERVICES OMITTED.</td>
<td>CONTROL SIGNAL TO MOTOR.</td>
<td>VIDEO FRAMES.</td>
</tr>
</tbody>
</table>

Fig. 1. Overview of the reference model.

The Workcell Controller must ensure that the parts are produced efficiently. It schedules sub-tasks of the production job on the executing agents of the lower level: the Workstations.

The Workstation Controller must coordinate the processing steps on a part.

The Automation Module Controller is responsible for the processing of objects. In order to process an object, the environment has to change in a specific way. The Automation Module Controller defines the trajectory in the environmental parameter space. Therefore it has to coordinate different pieces of Equipment, each of them in control of one environmental parameter, so that their combined actions result in the given trajectory.

The Device Controller has to ensure that the environment exactly follows the given parameter setpoints.

The Sensors and Actuators form the lowest level in the hierarchy. Sensors transform physical properties of the environment into information and Actuators modulate physical parameters according to given information.
3. The PCS Environment

A PCS always operates in an environment consisting of machines, parts, tools, designs, scheduling heuristics, etc. The PCS has to interact with this environment, therefore it must be described before the PCS can be analyzed.

A case study of the manufacturing of television sets will be used throughout this paper to illustrate concepts. The factory has two departments: one where electronic components are mounted on printed circuit boards (PCBs) and another where the PCBs together with a tube are mounted in a case.

In the initial design phase of a product there exist only functional specifications and drawings. This information is not enough to make the product; information must be added specifying which components must be used and in what order operations must take place. This description of how the TV set must be made is called the Bill of Material (BOM). In constructing the BOM, knowledge of available machines and operations is added to the design information.

A BOM should be complete, i.e. it should list every operation and every component. A structured BOM can be generated by decomposition of the operations into sub operations and decomposition of parts into sub assemblies and components. A well-structured BOM should map the structuring of machines etc. into larger processing units, because then a part of the BOM can be given to a part of the factory as a description of how an operation of that processing unit must be executed. In our example the task of making TV sets is decomposed into: mount components on PCBs and assemble PCBs, tube and case.

The BOM alone gives not enough information to enable a controller to execute its task. The controller must have knowledge of the capabilities and capacities of the lower level too. This description is called the Bill of Process (BOP). The BOP can be structured just like the BOM. For each level the BOP describes the lower level agents and their interconnection. For each lower level agent there exists again another BOP, specifying its sub agents. On higher levels the capacities will be given in statistical terms like averages and standard deviations, whereas on lower levels the capacities can be known exactly. Compare the drilling of a number of holes in a PCB with the processing of several batches of PCBs simultaneously on several drilling machines.

The available space for products in the system must be defined too. On Factory and Workcell level this amounts to specifying the capacity of stocks. On Workstation level this can be done by specifying the available locations for parts. Note that locations do not need fixed coordinates; in_robot_gripper might be a valid location even though the robot hand is moving.

The possibility to define named locations is based on the assumption that the environment is structured, so that products can be on a limited number of predefined locations only. However most production environments allow this assumption.

4. A Formal Description of a BOM

In this section a concept for specifying the temporal order of operations in a BOM is given.

There are three different types of temporal relations:
- \( \text{Op}_1 \) before \( \text{Op}_2 \), e.g. mount_components before assemble_tv_set;
- \( \text{Op}_1 \) parallel with \( \text{Op}_2 \), e.g. mount_components parallel make_tv_tube; (Parallel does not mean that the operations must run in the same timeslot, only that the starting times of the operations may occur in any order.)
- \( \text{Op}_1 \) or \( \text{Op}_2 \), (either \( \text{Op}_1 \) or \( \text{Op}_2 \) is executed but not both) e.g. make_wooden_case or make_plastic_case.

Parallel does not mean that the operations must run in the same timeslot, only that the starting times of the operations may occur in any order. Operations can be grouped using parentheses. An example BOM is shown in Fig. 2. The first line of this BOM specification associates a name with it and also specifies the required input parts and the produce_tv_set(in components, PCB, tube, case; out tv_set)

\[ : \begin{align*}
\text{mount_components} \\
\text{parallel} \\
\text{make_tv_tube} \\
\text{parallel} \\
\text{(make_wooden_case or make_plastic_case)} \\
\before \text{assemble_tv_set}
\end{align*} \]

Fig. 2. Example BOM.
resulting product. In a similar way the mentioned operations, like assemble_tv_sets, can be seen as names of lower level BOMs. BOMs can thus be seen as procedures.

Since all operations are binary, a BOM can be represented as a binary tree, see Fig. 3. All operations are in the leafs of the tree and the temporal operators are associated with the nodes of the tree.

5. Decomposition of a Factory

After the structuring of the PCS environment, the PCS itself will be analyzed. The analysis is done top-down starting from the Factory level.

A Factory is decomposed into a Factory Controller and a number of workcells. The Factory Controller is concerned with determining stock levels, in the example the number of television tubes, television cases etc. Two conflicting demands influence the stock levels; on one hand the factory wants to deliver products when the customer wants them. When all products are available in stock it is possible to deliver the products immediately, but this requires large stocks. On the other hand stock levels should be as low as possible because products in stock represent capital which is not profitable. Also products in stock require space which might be used for other more profitable products.

The Factory Controller’s task is to plan in which timeslot workcell operations should be executed. The structuring of Workcells depends on when new parts are needed. In the example we assume that the mounting of components on PCBs occurs in one Workcell and the final assembly in another. This makes it possible to start the production of PCBs before the tubes and cases have arrived at the factory. On the other hand the final assembly is done in one cell even though the final assembly operation can be subdivided into:

assemble_tube_case before assemble_PCIcases

The underlying assumption is that the PCBs are needed so shortly after the tubes that it is not economical to make this division at Factory level. Similarly one does not define one Workcell for each insertion of a component in the PCB. However when PCBs are always processed in large batches, it might be economical to define a resistor insertion Workcell, a transistor insertion Workcell, etc.

After definition of the BOM and the Workcells, the internal control task of the Factory Controller is analyzed. Basically it consists of deriving a temporal ordered set of Workcell commands (this set is called a production plan) and a set of temporal ordered material requests from a command to produce a given amount of products before a given point of time. The Factory Controller needs the following information to execute its task:

(1) A set of factory level BOMs (one BOM for each type of product) describing the temporal order of the workcell operations. The required input parts and resulting output parts are associated with each operation.

(2) A set of workcell descriptions, giving each cell’s capabilities (i.e. the operations it can execute), capacities (i.e. how many and how fast operations can be executed) and also setup and switching times for operations.

(3) The maximum stock capacity.

(4) The current production plan.

(5) The current stock levels.

(6) The current time.

The items 1, 2 and 3 can be seen as the Factory Controller’s static view of the world, its static world model. The information is not likely to be changed in the time span needed for the completion of one factory command. The static world model is updated only by systems management. Items 4 through 6 comprise the Factory Controller’s dynamic world model, this is constantly

![Fig. 3. Binary tree representation of the example BOM.](image-url)
plan_production(BT, P, Td; return Pn, Tstart) :=
if is_leaf(BT)
  find_latest_starting_time(operation_of(BT), P, Td; return Tstart)
  add_operation(operation_of(BT), Tstart, Td, P; return Pnew)
  return(Pnew, Tstart)
else case operator_of(root_of(BT)) of
  before : plan_production(right_tree_of(BT), P, Td; return Pnew, Ts)
    plan_production(left_tree_of(BT), Pnew, Ts; return Pn, Tstart)
  parallel : plan_production(right_tree_of(BT), P, Td; return P1, Tsl)
    plan_production(left_tree_of(BT), P1, Td; return P2, Ts2)
    return(P2, earliest(Tsl, Ts2))
  or : plan_production(right_tree_of(BT), P, Td; return P1, Ts1)
    plan_production(left_tree_of(BT), P, Td; return P2, Ts2)
    if Tsl later than Ts2 return(P1, Ts1)
    else return(P2, Ts2) endif
  else return(P2, Ts2) endif
endif

Fig. 4. Algorithm for the generation of a production plan.

during the execution of a command. The dynamic world model is updated after reception
of commands, messages from Workcells etc.

The generation of a production plan can be
described as:
(1) Select the BOM associated with the requested
products from the set of BOMs;
(2) Associate timeslots with the cell commands
defined in the BOM and add the results to the
current production plan so that the constraints
provided by the temporal specification of the
BOM, the capacity of the Workcells and stocks
are met;
(3) Derive material requests from the production
plan.

For the solution of step 2, many algorithms and
heuristics are possible. It is not necessary to select
a particular one, but it must be possible to de-
scribe them in a factory language. An example
algorithm is given in Fig. 4 to show the necessary
concepts for the language. In this example the
Factory Controller tries to execute operations as
late as possible. The algorithm takes as input a
BOM (in tree representation as in Fig. 3) BT, a
current production plan P and a delivery time Td
and it computes a new plan Pn and a starting time
Tstart. The algorithm is recursive, so it starts at
the top node of the tree and goes down to the
leaves.

From this example we can derive the following
language requirements:
(1) Operations for set manipulation: add, remove,
retrieve elements, sorting operations (find_latest_starting_time can be described with sort-
ing all operations in P that must be executed
on the same workcell);
(2) Tree manipulation operations;
(3) Structured data types;
(4) Procedures with recursion;
(5) Conditional statements;
(6) Arithmetics.

6. Decomposition of a Workcell

A Workcell is decomposed into a Workcell
Controller and a number of Workstations. The
task of a Workcell Controller is to schedule oper-
ations on Workstations. Scheduling means de-
termining on which Workstation and when an
operation must be executed. A Workcell Con-
troller can have different goals to optimize the
schedule like maximizing capacity, maximizing
Workstation utilization, or minimizing the make
span.

The Workcell level BOM must contain the nec-
essary information for constructing a good sched-
ule. The temporal order of Workstation operations
can be specified with the same operators as re-
quired for the factory level, but now more informa-
tion must be associated with each operation.
The information associated with an operation
consists of:
(1) the required and resulting products;
(2) the setup, processing and removal time.
Not all this information will be necessary for all
production situations. The setup, processing and
removal time can be combined into one operation
if this does not diminish the quality of the sched-
ule.

The Workcell Controller also needs informa-
ton on transport times of parts between Worksta-
tions and on the interconnection of Workstations
(i.e. which Workstations can exchange products).
A typical Workcell consists of several processing
Workstations all connected to one transport
Workstation, which transports parts from one
Workstation to another.
The transport commands cannot be specified in the BOM, because if there are several Workstations in a Workcell capable of executing a certain operation, then the Workcell Controller decides during production which one will be selected. Therefore it cannot be known at the time that the BOM is made to which Workstation the parts must be sent.

A schedule is again a set of Workstation operations with a temporal ordering. Its construction can be described similarly to the construction of a production plan at Factory level.

7. Decomposition of a Workstation

A Workstation can be decomposed into a Workstation Controller and a number of Automation Modules. The Automation Modules will execute the technologically different processing steps of the operation. In our example a Workstation might consist of a robot and a component insertion machine, where the robot can be used to load and unload PCBs in resp. from the insertion machine. Both the robot and the insertion machine can be viewed as Automation Modules. The Workstation Controller has to coordinate the processing steps and the exchange of parts. The Workstation Controller also has to manage its resources being: Automation Modules, space, tools and parts.

Given the task of a Workstation Controller, processing steps should be defined so that upon completion all resources are released; even if a robot can only be commanded in terms of open_gripper, move etc., a processing step still would be move_object(from, to) which would be executed on the robot as: open_gripper; move(from); close_gripper; move(to); open_gripper; move(home). Although a processing machine and tools can be released after completion of a processing step, the location occupied by the part cannot. Therefore we need a special primitive at Workstation level: TRANSPORT(FROM, TO), which is necessary for a good management of locations. Before this step is executed the Workstation Controller first reserves the TO location so that there will be no collisions. Upon completion of the TRANSPORT step the FROM location is released.

TRANSPORT commands should not be specified by the BOM since they can be derived from the operations; when an operation has to be executed on a certain Automation Module, the necessary parts and tools should be transported to the Automation Module.

A Workstation level BOM should associate the set of necessary tools, the set of required parts and the set of resulting parts with an operation. This allows the Workstation Controller to derive which parts and tools should be moved to the processing Automation Module. Tools and parts can be treated similarly; the only difference is that parts are changed by operations whereas tools are not, although the associated tool information, like tool use time, might be updated after an operation.

Workstations exchange parts, therefore the working spaces of connected Workstations must overlap. The transfer of parts takes place in this common area. Since parts occupy locations, the transfer of parts involves the control transfer of locations in the common area. There are two different protocols for exchanging parts: one where the sender initially has control of the transfer location and one where the receiver has control of the transfer location. Below the second protocol is given.

Sender:
SEND(Product_Transfer_Request(product_id, transfer-location))
RECEIVE(Product_Transfer_Granted(product_id, transfer-location))
MOVE(from,transfer-location)
SEND(Product_Transfer_Completed(transfer-location))

Receiver:
RECEIVE(Product_Transfer_Request(product_id, location))
SEND(Product_Transfer_Granted(product_id, transfer-location))
RECEIVE(Product_Transfer_Completed(transfer-location))
MOVE(transfer-location,stock-location)

This example shows that the sending and receiving of messages must be expressed. Receiving a message implies that the associated process waits until the message has arrived.

Another problem arises when a Workstation executes several operations in parallel. Then it must have some kind of reservation mechanism with queuing to ensure that all operations will get their resources eventually.
8. Decomposition of an Automation Module

At Automation Module level the environment can no longer be described with discrete parts, processing steps and locations, now the environment consists of continuous trajectories, surfaces, contours etc.

Trajectories can be specified explicitly, for example by an NC program, or implicitly, for instance in a move command where only the starting and ending point are given, possibly with some forbidden regions.

The task of the Automation Module Controller is to determine the trajectory associated with the processing step. This means either finding the associated coordinate list or the actual computation of a trajectory. Another task is the transformation of this trajectory in the world coordinate system into the coordinate system of the available equipment. For instance following a straight line with an anthropomorphic robot hand must be transformed into rotation speeds for the robot joints. The two tasks of finding a trajectory and coordinate transformation are not independent; it can be advantageous to first transform the problem of finding a trajectory into the coordinate space of the Devices [5]. In this way trajectories requiring impossible Device operations are avoided. Although trajectory computation problems can be very complex, they only require arithmetical capabilities of the language. An extended arithmetical capability allowing the definition and manipulation of matrices and vectors might be helpful, especially for coordinate transformations.

Another problem at Automation Module level is that the knowledge of the world and the own state is known only imperfectly. Therefore information provided by Devices on their own functioning or special Sensor Devices on external conditions is necessary to guide the execution of the processing step. Robot languages like SRL and AL [3] already have instructions for using sensor information in the execution of a processing step. An example in AL is given of a processing step where the robot moves a grinding device with a force of 5 N over the surface of an object:

```
MOVE grinder to left_side WITH FORCE (XHAT) = 5*N;
```

A decomposition of Devices will not be given because it does not present any new language requirements. Mostly Device functionality consists of control loops and signal processing which can be described with arithmetical statements.

9. Error Recovery

Error recovery or exception handling is the most difficult part of an automated controller program. Some authors incorporate error recovery routines in their state table representation of a BOM [1,4]. However we think, together with Meijer et al. [6] that a BOM should specify only the necessary constraints for making a product and that the controller itself should exploit the freedom of alternative making sequences to recover from errors. Therefore error routines should be triggered either by error messages from the lower level, indicating that the lower level could not solve the problem, or by internal exceptions. An example of a failing lower level is a failing robot arm in a Workstation. The Workstation is now unable to load or unload parts from a processing machine, so the Workstation fails too. Finally at Workcell level the Workcell controller recovers from this error by sending all parts to other Workstations capable of executing the operation. An example of an internal exception is a Factory controller which derives a production plan starting before the current time. A recovery might be found in dividing the batch into two batches with each half of the products, this allows to execute the jobs more in parallel.

The programming of exception handling routines consists of:

1. Specification of a lower level exception message or an internal exception state;
2. Specification of recovery steps.

Note that the specification of recovery steps can use internal controller information, for instance to determine which tasks are affected by the exception. Also internal exception states can be detected best by control tasks themselves, when they detect that they did not achieve their goal. This is a more efficient routine than defining a watchdog routine looking for forbidden states.
10. Conclusions

Formal description of the behavior of production controllers can be divided into two main parts: the description of the environment and the description of the internal control tasks. The description of the environment consists of descriptions how products (BOMs) must be made and descriptions of the capabilities and capacities of the subordinates of the controller (BOPs). Both can be defined in a structured way, so that the BOM of one level is specified in terms of the subordinate operations each of which having its own lower level BOM. Similarly a BOP describes the subordinates and each subordinate has its own lower level BOP. Processing tasks are under constrained, therefore a language describing these tasks should not introduce artificial constraints, but should allow the controller to exploit the maximum freedom to find optimal operation sequences or recover from errors. The temporal operators: before, parallel and or provide a good basis for the construction of a process task description language.

Set and tree manipulations are a convenient way to describe control tasks at Factory, Workcell and Workstation level. At Automation Module level it is necessary to be able to specify the use of sensor information in the execution of a processing step to compensate for imperfect knowledge of the world.

References