An Approach to Factory Dynamics Based on Manufacturing Algebra

Enrico CANUTO, Francesco DONATI, Maurizio VALLAURI

Dipartimento di Automatica e Informatica, Politecnico di Torino,
Corso Duca degli Abruzzi 24, I 10129 Torino, Italy

1. AIM OF THE FACTORY DYNAMICS

The aim of a production process is to transform objects from raw materials, into semifinished and then into finished products. As a consequence, the mathematics which has been introduced to model a production process deals with transformations of elements into the *space of the object quantities.*

The *space of the object quantities* has been defined in the *Manufacturing Algebra* (see the companion paper [3]) as the Cartesian space having dimensions equal to the cardinality of the different types of objects and coordinate axes representing the quantities of the various object types. Then the Algebra deals with operations between elements of such a quantity space. To provide an example in terms of the Algebra, one can say that *assembling four wheels, one engine and one car body yields a car as a product.* No question is raised concerning *from where* the components will have to be drawn out, or *when* the assembling operation will have to be carried out. *The Algebra is formulated outside space and time.*

The Factory Dynamics aims instead at describing the production process as it is evolving in the factory space and during time. The term Dynamics has been appropriately introduced to recall the time and consequently the dynamic evolution of the production process during time.

The introduction of space and time does not modify what has to be dealt with, i.e. the transformations of object quantities, and the mathematical space where one has always to operate, i.e. the space of the object quantities already defined by the Algebra. What is new is the distinction of the objects, not only according to *types,* but also according to the *place* where they are located and to the *time when* they are at a certain place. In the language of *Factory Dynamics* one will say: *an assembling operation to be launched at time t_0 will require four wheels to be drawn out of store A at time t_1, one engine to be drawn out of store B at time t_2 and one car body to be drawn out of store C at time t_3; the operation will release one car at time t_4 at store D.*

Let us point out that the complete interchangeability of objects of the same type, which holds for the Algebra, does not hold for the Dynamics, where the interchangeability implies - beyond the property of belonging to the same type - also the property of being at the same place at the same time. According to the Dynamics, two objects of the same type but located at different places - or located at the same place but at different times - have to be considered as *different objects.* As a consequence, the cardinality of the objects increases as well as the dimension of their quantity space. In summary: a quantity can not be referred only to an object *type*, but also to the place *(space)* where the object is located and to the location *time.*

Therefore the *Factory Dynamics* will still apply the Algebra as it has been defined, with the only peculiarity of operating over an object quantity space of larger dimensions, since the cardinality of the objects under consideration increased, because the objects differ not only in type, but also in their locations in space and time. Besides this rather formal distinction, the *Factory Dynamics* differs from the Algebra because it aims at analysing the process evolution in time. To this purpose, a new concept comes out, which is totally missing in Algebra: the concept of *state.*

A *state* is the ensemble of information allowing a clear separation between past and future: indeed, a *state* summarizes all the effects the past history may have on the future. Given the state, the past can be forgotten. The dynamic approach which has been followed hereafter is strongly based on the *state* concept and, as such, it can fully meet the objective of writing the equations describing the time evolution of the factory state, under the

control actions and the effects of random disturbances.

2. THE FACTORY

2.1. Definition

The factory is a *tool* for implementing the manufacturing operations which have been introduced as elements of the Algebra. From a mathematical viewpoint, the factory appears as a network of operators (*storage units, transport units, manufacturing units*) which transform quantities of objects through manufacturing operations, under the management of a production control system.

Definition. A factory plant is the composition of four mathematical entities: *storage units, transport units, manufacturing units* and *control units.*

Storage, transport and manufacturing units can be interconnected to form a network corresponding to factory layout. Control units manage the factory production.

2.2. The time

The approach followed here is the one of the *discrete dynamic systems*; therefore the *time* is an independent integer variable $t \in \mathbf{Z}$, evolving naturally in the direction of the increasing integers with a finite step (called *sampling step*) taken equal to the unit of time.

The factory is modelled as a *causal* system, in which no cause can accomplish its effect at the same instant when acting. The minimum delay between a cause and the corresponding effect is assumed to be one sampling step, i.e. one unit of time.

2.3. Space-time arrangement of the objects in the factory

The concepts and the relevant mathematics of object sets and object quantities, which have already been used in the Manufacturing Algebra [3], are introduced also in the factory, with the peculiarity of assigning to objects an arrangement in the space-time dimension. Objects of the same type must therefore be considered different (that is, *non-interchangeable* from the viewpoint of the feasibility of manufacturing operations) when they are in different positions and/or at different times. The effect is equivalent to a multiplication of the number of the elements belonging to the object set.

The places in a factory where the objects can be arranged, are denoted as *storage units*. Their cardinality is assumed to be finite and equal to the integer n_i. Hence they can be ordered in a list and put in correspondence with the integer set $\mathbf{Z}\{1,n_i\}$, so that the space in a factory is described by the integers $i=1,2,...n_i$.

All the actions performed in the factory are assumed to produce their effects within a finite time delay, not longer than a well-defined and finite *time horizon.* Hence, having decided to model the time as a discrete variable, the time horizon will be defined as a finite number n_j of sampling steps, starting from the current time t.

The above assumptions on space and time imply that factory object quantities are described by a three-index vector $\boldsymbol{q}$, whose generic component $q(i, j, k)$ represents the quantity of the object k which, at time $t+j$, are present in the storage unit i. Having denoted with

- n_k the total number of objects potentially present in all the storage units,
- n_i the total number of the storage units,
- n_j the total number of time units in a time horizon,

the dimension of the quantity vector $\boldsymbol{q}$ equals $n_k(1+n_j)n_i$.

It must be however pointed out that the use of the triple of indices (i, j, k) is a simple convenience for a more immediate understanding. In fact, it is perfectly equivalent to consider $\boldsymbol{q}$ as an ordinary vector of dimension $n_k(1+n_j)n_i$ indexed by a single integer r, the latter being related to indices (i, j, k) by

$$r = i + (j+1)\, n_i + k\, n_i\, (n_j+1) \qquad (1)$$

2.4. The factory elements: storage units

A *storage unit* is a mathematical element introduced to model stores and more generally any area of the factory which is used to store a quantity of objects. A storage unit is a simple mathematical element consisting of a vector of object quantities, which is a function of the time t. This vector, indicated in the following as the state of the storage unit, describes the object quantities which are present in the storage unit at time t and the object quantities which, as a consequence of the commands already dispatched by the production control system, will be present (stored) at the successive instants $t+1,t+2,...,t+n_j$, where n_j defines the time horizon.

Storage and *drawing* operations can be carried out. In the case of storage, the state is incremented by the quantity of stored objects and, in the case of drawing, it is decremented by the drawn quantities. Constraints of physical realizability can be imposed on the stateof a storage unit, in terms of maximum and minimum (zero) quantities which it can contain.

Let us remark that the state at the current time t which, by definition, sums up the whole past history, must necessarily take into account all drawing and storage commands which at the same time t have been already received but not yet carried out. By knowing the

time when those commands will be carried out, one can predict the evolution of the object quantities in the stores; the prediction horizon corresponds to the maximum delay (time horizon) which can occur between the reception of commands and their performance. In this way the state of any store is the sum of the object quantities actually present at time t and the object quantities expected at the successive instants $t+1$, $t+2$, ... till to the horizon $t+n_j$..

The state of all the factory storage units is then denoted by the three-index vector $\boldsymbol{x}_t$ (function of time t), whose generic component $x_t(i, j, k)$ represents the quantity of the object k which, at time $t+j$, is expected to be in the store i, taking into account all commands dispatched at time t.

2.5. The factory elements: production units

Transport and manufacturing units

A *transport unit* interconnects two stores and is capable of performing specific transport operations (the identity operations defined by the Manufacturing Algebra), drawing the object quantities contained in the input vector (equal to the output vector) from a store and storing them in the other one.

A *manufacturing unit* interconnects two stores, and is capable of performing specific manufacturing operations, drawing the input object quantities from the input store and placing the output object quantities in the output store. Input and output stores can be the same.

Transport units are just like to manufacturing units, except for the peculiarity that they perform only identity operations in which objects are moved from one store to another one without any transformation. Therefore, in what follows, transport and manufacturing units will be treated as the same mathematical element and generically indicated as *production units*.

The cardinality of the production units of a factory is assumed to be finite and equal to the integer n_r. Hence they can be ordered in a list and put in correspondence with the integer set $Z\{1,n_r\}$, so that they can be described by the integers $r=1,2,...n_r$.

A production unit r is capable of performing only the manufacturing (or transport) operations has been programmed for; those operations are assumed to define the finite and countable set $\boldsymbol{E}_r$ of the *elementary admissible manufacturing operations* of the unit r. As a consequence, the production control system can only command a production unit r to perform an operation s belonging to $\boldsymbol{E}_r$. By considering all the production units of a factory, the admissible set $\boldsymbol{E}$ of the elementary manufacturing operations of the whole factory can be defined, where each elementary operation is associated to an integer s, $s=1,...,n_s$.

States of production units

A production unit is characterized by two logic states: the *waiting state* and the *working state*. When in the waiting state, it can receive from the production control unit the command of performing one of the admissible operations. Then it moves to the working state for a time interval depending on the command, i.e. on the manufacturing operation to be performed. As soon as the commanded operation is terminated, it switches back to the waiting state, ready to receive a new command.

Note that the time of permanence in the working state does not necessarily coincide with the manufacturing operation time. In fact, in complex units one should distinguish between:

- the *cycle time* $c(r,s)$, i.e. the time of permanence in the *working state*, defined as the minimum time interval between two successive commands (or between the productions of two successive objects),
- the *manufacturing time* $\tau(r,s)$, defined as the time interval between the command and the coming out of the finished object. Manufacturing times can be much longer than cycle times.

The cycle and manufacturing times are defined for each manufacturing operation of the admissible set; they can be deterministic or stochastic variables. In the latter case the factory model will be a stochastic system.

Admissible manufacturing operations

Each admissible manufacturing operation is described, in conformity with the Manufacturing Algebra, by two input and ouput quantity vectors $\boldsymbol{u}$ and $\boldsymbol{y}$ and the corresponding balance vector $\boldsymbol{b}$. Those vectors are expressed using the already explained three-index notation (i, j, k), where:

- i denotes the storage unit from which objects have to be drawn (or in which they must be stored),
- j denotes the time counted from the dispatching instant of the command, when drawing or storage will effectively take place,
- k the type of object.

It is worth pointing out that here the balance vector $\boldsymbol{b}=\boldsymbol{y}-\boldsymbol{u}$ does not suffer any more a loss of information with respect to input and output vectors as it happened in the Algebra, where operations were defined as independent of factory and time. Here, the arrangement in time and space of *drawing* and *storage* operations prevents reusable materials and some semifinished products from disappearing from the balance vector. For this reason, in the following any manufacturing operation will

be described by its balance vector.

Stochastic production units

Vectors $\boldsymbol{u}$, $\boldsymbol{y}$ and $\boldsymbol{b}$ can be assumed to be stochastic variables; in which case the model of the relevant production unit becomes stochastic. To this end, let us recall here what has been already said in the companion paper [3] about randomness and how it can concern three types of variables:

- the quantities of input objects (f.i. in repair operations),
- the quantities of output objects (f.i. in the quality control operations),
- the time j, when the drawing of the input objects or the storage of the output objects takes place.

The assumption of a deterministic or stochastic model is a choice to be made in connection with the type of process and the objectives of the study.

Inputs to production units

The *commands*, which at a given instant t are dispatched to all the production units of the factory, will be described by a two-index vector $\boldsymbol{u}_t$ whose component $u_t(r,s)$ can take the values *zero* or *one* depending whether or not the admissible operation s is commanded to the production unit r at time t.

In order to describe which *drawing* and *storage* operations follow a specific command, the multi-index matrix B, resulting from the composition of the balance vectors $\boldsymbol{b}$ of all manufacturing operations which can take place in the factory, is introduced; its generic element $b(i,j,k;r,s)$ represents the object quantity to be drawn (if negative) or to be stored (if positive) according to the following indices:

- i, index of the store where the operation is performed
- j, time, in number of sampling steps and counted form the dispatching time t, when drawing/storage will take place
- k, index of the object to be drawn or stored
- r, index of the production unit which performs the *drawing/storage* operations
- s, index of the operation requiring drawing/storage.

The relation

$$\boldsymbol{p}_t = B\,\boldsymbol{u}_t \qquad (2)$$

generates a multi-index vector $\boldsymbol{p}_t$ whose component $p_t(i,j,k)$ represents the total of *storages* (if positive) or *drawings* (if negative) of the object k, occurring in the store i at time $t+j$ as a consequence of the command $\boldsymbol{u}_t$ dispatched at time t.

2.6. The factory elements: the control units

Control units aim at managing a network of manufacturing, transport and storage units, which could in extreme cases be reduced to one single unit or be extended to the whole factory. For the time being, it is seen as a *black box* described only in terms of input-output, as it follows.

- It receives commands from a higher hierarchical control level in terms of manufacturing operations, which can potentially be performed by the network of units placed under its control. In other words, the command dispatched by the higher level refers to a manufacturing operation which can be the result of an algebraic composition (in terms of additions and multiplications) of the elementary operations which the production units under its control are capable of carrying out. Note that the algebraic composition of an operation corresponds to a plan including both the elementary operations to be carried out and their order of performance.
- It receives information from the field on the state of the stores, manufacturing and transport units.
- After having verified the availability of the input objects in the input storage unit, it dispatches the relevant commands to the manufacturing and transport units which are in waiting state. Those commands have been already introduced through the definition of the vector $\boldsymbol{u}_t$.

In conclusion, a control unit performs what usually is called a *real-time control*, which is nothing else than putting into operation a manufacturing program set up by a higher hierarchical level, without taking into account randomness and unpredictable events which may occur in the field.

The search for an efficient and robust control logic is within the objects of this research. It is just for meeting such requirements that, in the following chapters, the factory network will be described in terms of state equations.

3. THE FACTORY STATE EQUATIONS

3.1. The storage state equations

The time evolution of the state of the storage units is described by the following linear discrete dynamic model:

$$\boldsymbol{x}_{t+1} = A\,\boldsymbol{x}_t + B\,\boldsymbol{u}_t \qquad (3)$$

where:

- $\boldsymbol{x}_t$ is the three-index state vector of all the factory storage units,

- B is the 3×2-index balance matrix of all the admissible manufacturing operations which can be performed by all the factory production units,
- u_t is the two-index command vector dispatched by the production control system,
- A is a square 3×3-index matrix, forcing the state x_t to *shift forward in time*; its entries are defined by the following relations:

$$a(i,j,k;i,j+1,k)=1 \text{ for } j<n_j \text{ and any } i,k$$
$$a(i,1,k;i,1,k)=1 \text{ for any } i,k$$
$$a(i,j,k;l,r,s)=0 \text{ otherwise} \qquad (4)$$

3.2. Stability, controllability and observability

The factory state equations given by equation (3) are linear and time invariant. One can easily verify that they have the following properties.

Stability

The system is dynamic, of order $n_i(1+n_j)n_k$. Note that n_i, n_j, n_k are respectively the number of storage units, the number of sampling steps or lags defining the time horizon and the number of types of objects. It turns out that the eigenvalues of the matrix A are all 1 or 0: more precisely $n_i n_k$ eigenvalues are equal to 1, whereas the remaining $n_i n_j n_k$ are 0. Hence, the system is described by a network of *adder* and *delay* elements.

Controllability.

The controllability depends upon the rank of the matrix W_C defined by

$$W_C=| B\ AB\ A^2B\ \ldots\ldots\ A^nB\ | \qquad (5)$$

where n denotes the order of the system. The system is controllable if and only if such a matrix has full rank, i.e. rank n. From the physical viewpoint the controllability depends upon the manufacturing operations for which the production units are enabled. To this end, the transport system is particularly important; if the transport units make possible to carry out displacements of any object between any pair of storage units, then the system can turn out to be controllable, otherwise it is not said that the system will be fully controllable.

In case of non completely controllable system, the subsystem of the controllable state variables must be found out, and the system order will have to be reduced, limiting the study of the control strategies to the controllable subsystem alone.

The system controllability is essential to the control of the system state both in closed and in open loop, but it is not essential that all the state variables used for modelling the factory be controllable, for what concerns the production management. Instead it is essential that the only part of the system which is controllable be capable of attaining the production objects. In other words: the whole potentiality of a factory is not necessarily required for developing the production process, but the part which is necessary must be controllable.

Observability.

The observability of the system is necessary for the design of the closed-loop control, i.e. of the real-time control of the production process. Observability requires the availability of a set of measurements, and therefore the definition of the output variables through a convenient matrix equation to be added for completing the state equations (3). Consider the equation:

$$z_t = C\,x_t \qquad (6)$$

where:

- z_t is the output vector, that is the vector of the quantities assumed to be measured. As a rule, it should be a two-index vector of quantities of objects, whose component $z_t(i,k)$ represents the quantity of objects k which are in the storage unit i at time t .
- C is a matrix defining which state variables composing the vector x_t have to be measured.

Therefore the observability depends upon the rank of the matrix W_O defined by

$$W_O=| C\ CA\ CA^2\ \ldots\ CA^n\ |^T \qquad (7)$$

whose rank must turn out to be full, that is equal to the order n of the system.

Because of the peculiarity of the matrix A, it is easy to verify that the observability requires to measure all the quantities of the objects present in all storage units. It is not necessary to measure those quantities at every step, nor to measure them all at the same time, but it is necessary to measure all of them, otherwise the observability of the system state will be incomplete. The state variables which turn out to be not observable, cannot be subject to closed-loop control, but, if controllable, they can be handled in open loop.

3.3. Event-models and synchronous models with slow dynamics

The mathematical description of factory and production processes presented up to now does not imply the choice of specific informatic techniques for working out the models and implementing the production control. It is a formal mathematical description of the discrete manufacturing processes in which, by using a rigorous but very general language, it was attempted to avoid imposing *a priori* simplifying hypotheses, which could have limited the generality of the approach.

The only somehow important simplification is the modelling of time as a discrete variable, which however allows to approximate continuous phenomena with the desired resolution, if only a sufficiently small sampling step is assumed. Hence the mathematical model introduced is capable of applying various informatic techniques depending upon the actual needs.

In order to achieve the best accuracy in the description of the phenomena, it seems convenient to use techniques related with the succession of the *events*. In fact process dynamics is clearly characterized by the connection of *events* which follow one another. The first cause is the command, which the production control system dispatches to a production unit being in the waiting state. The command causes, in the shortest possible time (i.e. after one time unit) the transition of the unit into the working state and, after a certain time interval, the *drawing* from the input storage units of the objects necessary for achieving the commanded manufacturing operation. Such a *drawing* operation can take place - in case of more objects - even in successive times. After the *manufacturing time* is expired, the event occurs of *storing* in the output storage units the objects which are the manufacturing products. Even in this case, if there are more products, storage can occur at different times. The production unit moves back from the working state to the waiting state after the *cycle time* is elapsed: the *cycle time* can be different from, typically shorter than the *manufacturing time*.

Event-control logic seems therefore to be the most natural solution for real-time managing the progress of production processes. On the contrary, at least for certain purposes, the technique of discrete synchronous models may be sometimes convenient. That holds in particular whenever there is no interest in following the detailed succession of the events, but it is desired to analyze only phenomena concerned with the *slow* factory dynamics.

We already suggested in previous papers [1, 2] to make use of synchronous models with a constant sampling step, longer than the manufacturing times. The commands are then concentrated at the sampling instant and can correspond to the requirement of performing more operations which, within the model approximation, are is assumed to be carried out during the sampling step. Starting with all production units in the waiting state when commands are applied, one is back with all units still in the waiting state at the end of the sampling step and can strike the balance of the manufacturing operations achieved, updating in this way the state of the storage units.

Discrete synchronous model appears suitable in particular for the description of aggregated models with sampling steps of the order of one day.

3.4. From the manufacturing algebra to the production process in the factory

Note that the Manufacturing Algebra, and therefore also the concept of production cycle, do not take into account the existence of manufacturing and storing units, and of the factory layout. The manufacturing operation - as it is defined in the Algebra - models a technological manufacturing operation, without considering the machine which can carry it out and the stores from which the input objects can be drawn and in which the output objects can be stored. When in the Algebra a production cycle is introduced, one only establish the performance order of the operations and whether the output of one operation shall be put at the disposal of other ones (series) or shall not (parallel).

A manufacturing operation introduced in factory modelling is the implementation in the production plant of a technological manufacturing operation defined in the Algebra. Being an implementation, it is associated by itself with one specific production unit, and in particular input and output objects must be referenced to specific storing units. Furthermore a manufacturing operation has now a precise placement in time, so that objects drawings and storages are specified by the delay time of their taking place with respect to the command dispatching instant.

This different characterization of the operation in the algebra and in the factory has several consequences: the first and more immediate is that two algebraically equal manufacturing operations can turn out to be different when they are implemented in the factory. It is sufficient to implement them on production units operating on different storing units: they will turn out to be different, i.e. not any more interchangeable. It will not be possible to interchange them *sic et simpliciter* during the production cycle of the factory, since they imply a different objects handling.

It is now possible to conclude, that any production cycle defined in terms of the Algebra as an operation belonging to the set $A(S)$ including all the possible algebraic compositions of the elements of an independent set S [3], can be implemented in the factory if the production units are capable of carrying out the manufacturing operations of set S and if the factory is equipped with an adequate system of transport units which ensure a proper handling of the objects during the production cycle. Such implementation is not trivial and it could have several possible solutions, among which the best must be chosen according to pre-established optimality criteria.

Leaving aside the problem of the procedures for implementing a production cycle in a factory - what lies outside the limits of these pages - let us emphasize that the result of the implementation turns out to be an ordered sequence of commands to the manufacturing and transport units, corresponding to an ideal time evolution of the previously defined command vector, denoted as u_{ot}, for distinguishing it from the *actual* value u_t.

The vector u_{ot} represents an ideal sequence of commands which could lead to the actual accomplishment of the cycle if the mathematical model of the factory were deterministic and perfect. Since however, in the best case, some stochasticities in the manufacturing times have to be expected, there is no doubt that the commands must be adapted to the actual evolution of the phenomena. That is the task of the real-time control which - operating in closed-loop, i.e. receiving signals from the field - must ensure the proper performance of the cycle, notwithstanding the stochasticity of the real process.

4. AGGREGATION AND HIERARCHICAL ORGANIZATION

4.1. Introduction

One of the major drawbacks met when tackling the two basic problems, previously mentioned, is the high number of variables needed for modelling a production system. To avoid the difficulties caused by the complexity of the mathematical model, the most promising way seems to be hierarchical aggregation, which allows to tackle the problem at different aggregation levels, always in a relatively simple situation.

The concept is the following: given a network of storage units, transport units, manufacturing units with a corresponding production control system, the network can be transformed into a single equivalent production unit operating on two storage units, respectively input and output. In this way it is possible to aggregate into equivalent units production lines, departments, plants etc. up to considering the whole company as a single equivalent production unit.

The aspect which seems attractive is the perfect mathematical equivalence of the models obtained at the different aggregation levels, what makes possible the use of the same algorithms and control methodologies at all hierarchical levels.

It is now important to emphasize the following fact: a production unit as introduced when modelling the factory, is an *intelligent entity*, capable of carrying out - on the basis of a simple command - a manufacturing operation within a set of admissible operations, drawing the input objects from a special storage unit and giving back the output objects to a special storage unit. To aggregate a set of production units into an equivalent unit, the existence of a control system capable of managing the aggregated production units and of interfacing it with the external world is essential. The external world does not need to know how the aggregation was made up nor how it works; it must have the possibility of knowing the aggregate exactly as a production unit.

Therefore, if it is true that the conceptual design of a hierarchical system follows a top-down approach, from the highest toward the lowest levels, the design phase must follow a bottom-up approach, that is the control of the lowest hierarchical levels must be designed first, moving then up, little by little, toward the higher levels of the aggregation.

In the following paragraph the rules are given for designing the production unit equivalent to a production system consisting of more units.

4.2. The aggregation of a production system into an equivalent production unit

Spatial ggregation

Considere first a production system composed of storage units, transport units, production units interconnected and managed by a production control system. Then consider a set S resulting from the union of all manufacturing and transport operations which can be carried out by the units making up the system. Here it is assumed that the operations of the set S are *algebraic independent*. It results that the production system is potentially capable of developing all operations of the set $A(S)$ spanned by S, and defined in [3]. Inside $A(S)$, the set S_1 will be selected, consisting of the *minimal* production cycles of interest for the production plans of the company. The cycles belonging to S_1 will then be programmed in the production control system, which, in turn, will be enabled to receive, from a higher hierarchical level, the commands for their accomplishment.

Thus, the hierarchical aggregation is made. The complex production system appears outside as a single production unit capable of carrying out potentially all the manufacturing operations included in the set $A(S_1)$ and obtained from the algebraic composition of the independent operations belonging to S_1. For specifying the operations of S_1, it is necessary to compute, for each minimum production cycle, the balance vector $b_1=y_1-u_1$ resulting from the algebraic composition.

To this purpose it is important to point out that in the aggregation one must give up a detailed description in favour of the simplicity of the aggregated model. The details lost in the aggregation are:

- the structure of the system which is aggregated: the number and layout of its manufacturing, transport and storage units are lost. One moves to a single equivalent production unit, with two storage units, one for input and one for output,
- the detail of all the semifinished objects which are produced and employed during the production cycles of the set S_l. The higher hierarchical level does not need to know the existence of those objects.

To carry out the computation of the balance vector $b_l=y_l-u_l$ corresponding to one of the production cycles belonging to S_l, the command sequence u_t, $t=1,...,T$, where T is the time interval necessary to carry out the production cycle, must be first constructed.

Then, in order to move from the complex production system to the equivalent production unit, it is necessary to subdivide the stores of the production system into:

- input-output stores,
- intermediate stores, inside the production system.

The latter ones will disappear from the equivalent production unit, whereas the former ones will be preserved, possibly included in a single equivalent store.

The practical procedure to move from the production system into the equivalent production unit is the following:

- The multi-index vectors $B\ u_t = p_t$ are considered for t varying from 1 to T
- Then the vector p_{eq} is formed by applying the following relation:

$$p_{eq}(i,j,k)=\Sigma_t p_t(i,j-t+1,k)$$

- In the vector p_{eq} the elements corresponding to intermediate stores are omitted, or rather the index i is limited to the input-output stores. Then one obtains the vector b_l of the *drawing-storage* operations distributed in time and for the input-output storage units corresponding to the equivalent operation performed on the equivalent production unit.
- Repeating the procedure for all the operations belonging to the set S_l and composing the resulting balance vectors b_l the matrix B_l is obtained which allows to write the state equations of the aggregated system.

Time aggregation

In general it is likely that, when creating the equivalent production unit of a complex production system, the cycle time T will result fairly long. As a direct consequence the resulting equivalent vector of *drawing-storing* operations b_l would be characterized by a large number of *lags*, that is the dimension n_j could be fairly large.

Such fact is absolutely normal and simply shows that, when moving to a higher hierarchical level, it is convenient, as a rule, to increase the value of the time unit (i.e. of the sampling step) used to describe the system.

It is, indeed, quite normal to use different time units (sampling steps) for different hierarchical levels, increasing the resolution as one moves from the highest hierarchical level to the lowest one. Should a change (increase) of the time unit be decided, then the elements of the vector b_l which fall within the same *new* time unit should be aggregated.

It is important to remark once more, that the aggregation transforming a complex production system into an equivalent production unit is possible only because the production cycles of set S_l have been defined and the relevant production control systems exist.

5. REFERENCES

[1] CANUTO E., DONATI F., VALLAURI M.: "Factory Modelling and Production Control", *International Journal of Modelling and Simulation*, 13 (4,1993), p. 162-166.

[2] CANUTO E., DONATI F., VALLAURI M.: "Modelling Manufacturing Dynamics for Production Control", *2nd IEEE Mediterranean Symp. on New Directions in Control and Automation*, Chania (Crete, Greece), June 1994, p.142-149.

[3] CANUTO E., DONATI F., VALLAURI M.: "An Algebra for Modelling Manufacturing Processes", *3rd IEEE Mediterranean Symp. on New Directions in Control and Automation*, Cyprus, 1995.