MRP II-based Production Management Using Intelligent Decision Making

I. Hatzilygeroudis, D. Sofotassios, N. Dendris,
V. Triantafillou, A. Tsakalidis, P. Spirakis

Computer Technology Institute (CTI), Hellas (Greece)
&
University of Patras
Depart. of Computer Engineering & Informatics
Hellas (Greece)

Abstract: This paper presents an extended MRP II-based production management system (PMS), which improves the traditional MRP II paradigm by attaching an intelligent decision supporting system (IDSS) to it. The IDSS is attached to the lowest level of the PMS, namely the production activity control (PAC) subsystem, and includes a simulator, that imitates real system behaviour, a knowledge-based component, that imitates expert reasoning, and a real-time database manager, that acts as the data pool and the communication gate between them. The IDSS is capable of performing off-line and on-line rescheduling, thus resulting in more realistic short-term plans. Analysis of the related case problem and implementation of the system are also discussed.

1. Introduction

Currently, the *Manufacturing Resources Planning II* (MRP II) methodology [4] appears to be the most publicised approach adopted in manufacturing management. MRP II extends the primitive Material Requirements Planning (MRP) features [22]. A *production management system* (PMS) in general deals with all levels of production management, such as the strategic, tactical and operational level. An MRP II-based PMS supports manufacturing functions at all those levels in a hierarchical fashion.

At the lowest level (operational level) in the hierarchy, the *Production Activity Control* (PAC) subsystem resides. PAC concerns production control at the shop floor [2] and operates in a time horizon of between a month and real-time. Its most important function is scheduling. The output of the PAC system is a plan indicating the sequence of the orders to be executed in a production period, by specifying their release and due times. A serious drawback of an MRP II-based system is that the production plan produced by the PAC level is rather unrealistic, because it cannot take into account the real state of the production environment. Hence, the system cannot follow the large number of shop floor events to make real-time decisions. On the other hand, it does not provide any serious support to the production manager to revise the unrealistic plans and carry out the complex task of production control at the shop floor. Production control mainly deals with scheduling/rescheduling, which is really a complex task, since it involves decision making by taking into account a large number of conflicting factors or constraints [26]. Poor production control may cause serious problems to a firm's ability to meet production requirements and constraints.

Current research tends to attack the above problems mainly with structural solutions (e.g.[10, 15, 21, 25]): they provide generic PMS frameworks for future manufacturing systems rather than integrate new components to the existing ones to extend their functionality, as we do. To make plans more flexible and realistic and help the production manager to his task, we have attached *an intelligent decision support system* (IDSS) to the PAC system that uses real-time information. An IDSS is a decision support system that combines conventional and knowledge-based technologies, and where the user remains part of the decision cycle [9]. In our case, *simulation*, a conventional technique, is used to imitate the real system behaviour and the *expert systems* approach, a knowledge-based technique, to imitate expert reasoning. Given the introduction of knowledge-based technology and its capabilities, the computer-based system is more actively involved in the decision making process, in

contrast to its passive role in a traditional DSS. There have been a number of systems based on this point of view [28, 19, 16, 13].

The paper is organised as follows. Section 2 describes the case problem and its analysis process. Section 3 deals with the system architecture and the decision making processes. In Section 4, the simulation based component of the system is presented. Section 5 deals with the knowledge-based component, and finally Section 6 concludes.

2. Case Problem Analysis

2.1 Methodology

The source problem for our system design concerns production control at the shop floor of a yoghurt plant. The analysis of this problem was two-fold. On the one hand, it was related to conventional software systems analysis and, on the other hand, to knowledge acquisition, as in knowledge-based systems development. We specified three aspects of the problem analysis:

- *production process*: It concerns the production flow, i.e. the layout of the production lines, the workcenters, the operations in each workcenter etc.
- *production management*: It concerns the activities of the production manager during the production control process.
- *problem solving*: It concerns the ways the production manager reasons and acts when solving problems using his experience in cases of abnormal situations.

To achieve the above goals, we used a mixture of methods: questionnaires, interviews and observation, alongside printed material about the plant. In summary, we constructed over 10 structured questionnaires, of 50-60 questions each, and had over 10 semi-structured and structured technical meetings, of two or more hours long, with the production management staff of the plant. The major part of questionnaires, the printed material, observation and a small part of the meetings concerned information about the production process and the production management. The results were used for creating a model for the

shop floor and specifying the tasks and subtasks of the production manager. The major part of the meetings and a small part of the questionnaires were used for knowledge elicitation, mainly from the production manager and the senior staff, to realise their problem solving procedures. To this end, we also constructed Gantt-like charts of simplified real-like production plans and discussed solutions on occurrences of abnormal events with the production management staff [8].

In the following two subsections, we present the basic results concerning the first two aspects of the case problem analysis. These results were used to design the system architecture and functionality as well as the shop floor model of the plant. The results of the third aspect, namely problem solving, were used to construct the problem solving model of the system, so they are there presented (Section 5.6).

2.2 Production Process

The shop floor of a yoghurt plant is a flow shop environment consisting of a number of production lines, that may or may not be interconnected (i.e. have common work-centers), each producing various alternatives (flavours) of a basic type of yoghurt. There are some different basic types of yoghurt, like e.g. SET and STIRRED, each having various alternatives. An example (simplified) production line of the plant, with its interconnections to other lines, is depicted in Figure 1.

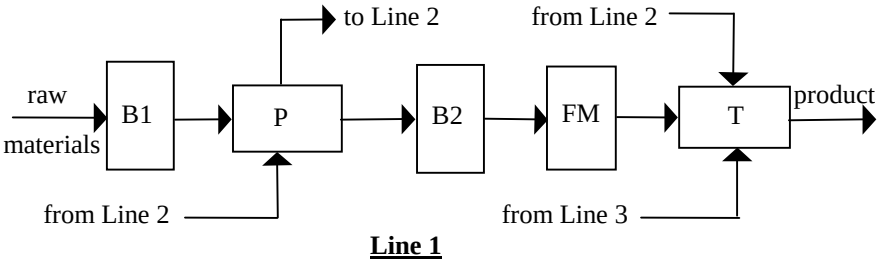


Figure 1. A Production Line in a Yoghurt Plant

The primary raw material is milk, which mixed with other materials, such as cream and protein, constitutes the mixture of a basic type of yoghurt. The mixture is initially created and stored in a buffer (B1), where it remains for a certain time in order a chemical process to be completed. Then, the mixture feeds, in a continuous flow mode, the pasteuriser (P), where it is pasteurised by increasing its temperature to 95°C. The pasteurised mixture is then stored into another buffer (B2). So, there is a continuous flow from B1 to B2 via P. After the mixture has been stored in B2, it is led to the filling machine (FM), where it is distributed into cups. Cups are put in palettes. Each palette is moved on to the cooling tunnel (T), where the mixture in the cups is being cooled passing through a number of stages. After it is sufficiently cooled, palettes are moved on to the inventory where they remain for 2-3 days, time necessary for the yoghurt to be ready for delivery to the market. Again, there is a continuous flow from FM to inventory via T. This kind of flow-shop production system have the following characteristics:

- There are no specific customer orders, but the orders are determined by the stock demands which in turn are determined by the market demands.
- Each work-center consists of one machine that performs only one operation.
- The operations that constitute an order should normally be executed successively, without any waiting, although waiting is possible in specific buffers (e.g. B2 in Fig.1), if required.
- There are no alternative process routes in a production line.
- The type of control applied is event-driven.

Although these characteristics result in a relatively simpler flow shop system than usual, the problem is still complex enough to be sufficiently handled by analytical methods and it still results in a large cognitive load to the production manager.

2.3 Production Management

The production plan period is a week. The production plan (schedule) for a week is more or less fixed (:it is not reconstructed every week). It is however periodically revised by the production manager depending on the time of the year, changes to the market demands and introduction of new or removal of old flavours. In this way, between two revisions the production manager knows for any week which flavours of which type of yoghurt should be produced in which day, in what sequence and in what quantities. Refinements of such a schedule concern only changes to the quantities, within certain limits.

Under these conditions, the task of the production manager is two fold. First, at the end of the current week he makes the appropriate refinements or modifications to the schedule for the forthcoming week, based on the stock requirements provided by the inventory control department, and the real condition of the factory. Once required changes have been made and the schedule is fixed (frozen), it is ready for execution.

On the other hand, during the real production (schedule execution) the production manager is responsible for reacting to any abnormal event(s) that may occur, like e.g. a machine breakdown, a high scrap or a rush order. This requires that first some immediate preventive actions, such as which work-centers production flow should stop at, are taken, and then some kind of reactive scheduling, i.e. on-line changes to the production schedule which is currently under execution should be made. The response time of the production manager to an event may sometimes be crucial, since e.g. milk products are time-sensitive. This is actually the main task of the production manager.

To make decisions, the production manager has to take into account a large number of constraints. For example, there are different setup times required for different breakdown intervals of a work-center. Also, there are certain yoghurt types that can be simultaneously passing through a certain work-center, whereas others cannot, depending on the compatibility of their temperatures. Furthermore, when a breakdown occurs and recovery time exceeds a certain limit, the quality of the product should be checked before proceeding.

The production manager does not use any computer-based tools to accomplish his task, but only his experience and he is no familiar with any analytical methods. Thus, sometimes his decision process seemed to be quite simplified, since he had no means to quickly explore various alternatives. Constraints such as cost-effectiveness and machine utilisation are only implicitly taken into account. Due dates are not considered as very hard constraints as product quantities are.

3. Extended MRP II-Based PMS

3.1 System Architecture

The architecture of the extended MRP II system is depicted in Figure 2. Extension consists in attaching an IDSS to the PAC subsystem, where the production manager (PM) is considered part of it. PM represents anyone who is responsible for making decisions for production control. The IDSS consists, apart from PM, of three major components: a real-time database manager (RTDM), a simulator (SML) and a decision maker (DM). These components can accomplish a variety of tasks and co-operate in a variety of ways.

PM acts like a controller that specifies each time the kind of the task and the co-operation activity to be performed. RTDM is the means for storing and managing available information. SML is a representative of the conventional DSS technology that allows PM to make simulation runs of a candidate production schedule, whereas DM is a representative of the knowledge-based technology that helps PM to evaluate such simulations and revise candidate schedules. RTDM includes the system database and is the only means for the communication between SML and DM. RTDM is a relational database management system implemented in INGRES, a tool for developing large database management systems.

3.2 Decision Making Processes

The functionality of the system reflects the production manager's two main tasks. Thus, it can operate in two independent modes, called the off-line and the on-line mode, that correspond to the off-line and on-line task of the production manager, respectively.

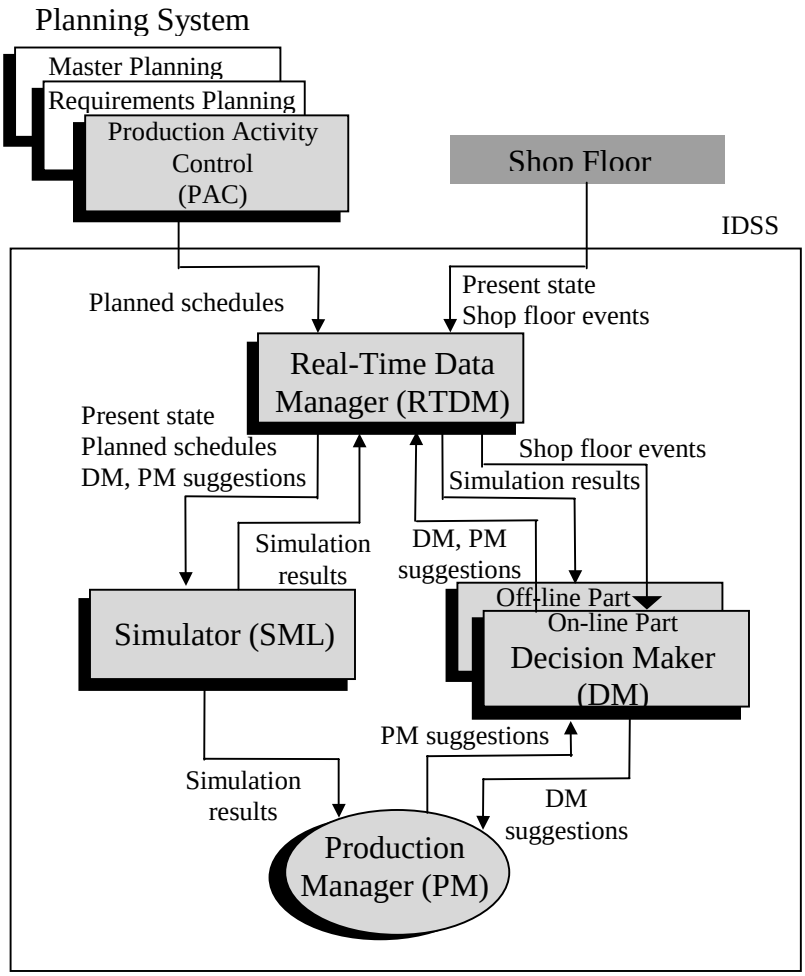


Figure 2. The Extended MRP II-Based PMS Architecture

The interactions between the components of the system in the *off-line mode* are depicted in Figure 3a. According to that, the planning system generates a schedule for a production period

which is stored in the system database. The execution of this planned schedule can be afterwards simulated by SML, according to PM's desire, using real data about the shop floor model from the database.

After a simulation run is completed and its results are available in the RTDM data pool, these results can be evaluated. This is done either directly by PM himself, or via DM. In the latter case, DM first decides on the degree of acceptability of the schedule by computing its total deviation from the planned production. Alternatively, revision rules are used to propose changes to the schedule. Finally, PM is the one who decides on which of the changes will be applied to the planned schedule, by approving some or all of DM's suggestions or by introducing his own. These suggestions are recorded in the RTDM data pool, and are taken into account by SML in the next simulation run.

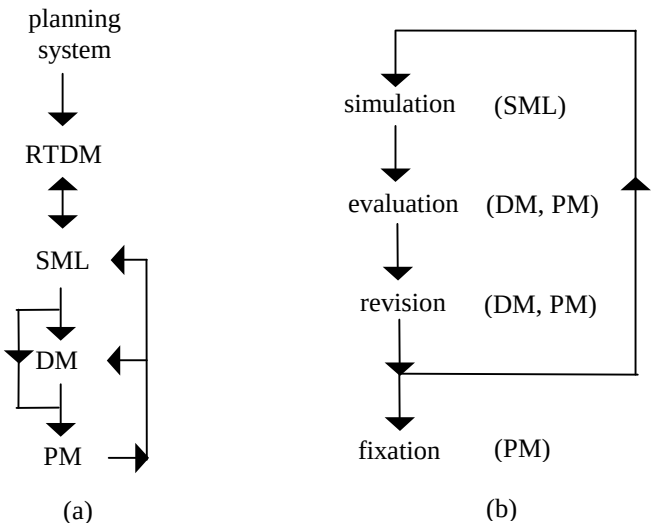


Figure 3. Off-line mode: (a) component interactions (b) decision cycle

Subsequently, PM can run as many simulations of the schedule as required to reach an acceptable plan for the forthcoming period. In conclusion, the off-line process can be described by the following decision cycle: simulation-evaluation-revision-fixation

(Fig. 3b). This reflects PM's subtasks in his off-line task. The final, acceptable schedule is recorded in the database.

For the *on-line mode*, the interactions between the components are depicted in Figure 4a. The on-line process is performed in two stages. In the first stage, as soon as an abnormal event is detected, the system proposes to PM a set of preparatory actions, to deal with the situation in short-term, like e.g. to interrupt production before a workcenter. Another set of actions is proposed to PM just after the recovery from an abnormal situation, e.g. after a machine repair has been completed. This type of action mainly concern workcenters setup.

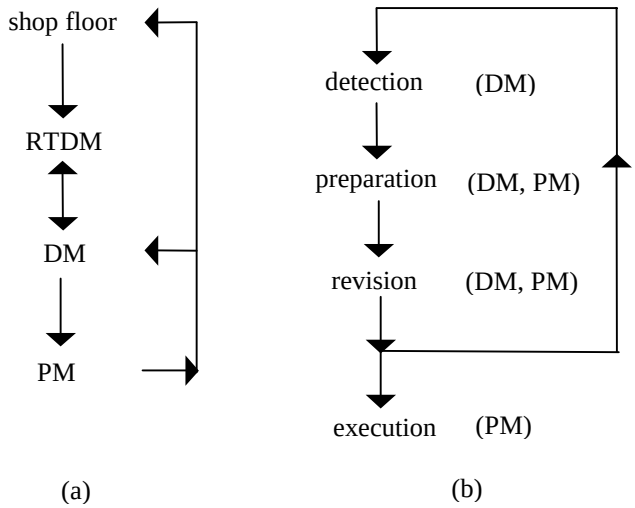


Figure 4. On-line mode: (a) component interactions (b) decision cycle

In the second stage, reactive scheduling is actually taking place. DM investigates possible changes to the executed schedule, to minimise the consequences of the disruption. The result is a number of changes to the current schedule to continue the disturbed production. Accordingly, PM either approves DM's modified schedule and takes the required actions or makes his

own decisions, or even interacts with DM for further investigation. So, the decision cycle performed is: detection-preparation-revision-execution (Figure 4b). This again reflects PM's subtasks in his on-line task. The final changes to the schedule are recorded in the database.

Given the above correspondence between the functions of the system and the tasks, subtasks of the production manager, we introduce the term *task-oriented architecture* to characterise the architecture of the IDSS.

4. Simulation Based Component

Simulation is used to imitate the behaviour of the real production system leading to useful inferences about its short-term production performance [6]. SML provides the necessary technological support for investigation of "what-if" questions and determination of the scheduling policy for the forthcoming production period. The operation of simulation corresponds to a pre-production step which extends the capabilities of the manager allowing him to manage production efficiently in a proactive manner [1].

4.1 Data Environment

The simulation process is concerned with the following classes of data:

- *model description data*: This data is related to the manufacturing profile of the system [2, 3]. It includes resource data, inventory data, routing information, released orders, historical data etc.
- *experimental data*: It includes the simulated time horizon, initial conditions, end-of-run criteria and a number of operational parameters used for the validation of the simulation model.
- *real system data*: This data is used to predict the state of the system at the beginning of the simulated horizon. When a simulation run starts, the system is unlikely to be in a zero state

condition (i.e. no active orders and all machines available). Moreover, to produce a more realistic run, real values should be used for various model variables instead of estimates (i.e. real capacities, present inventory levels).

- *output data*: The output of the simulator is a set of attribute variables and a number of statistics which are of interest to the production manager. The statistics can be presented in different ways, such as textual form, tables and Gantt charts.

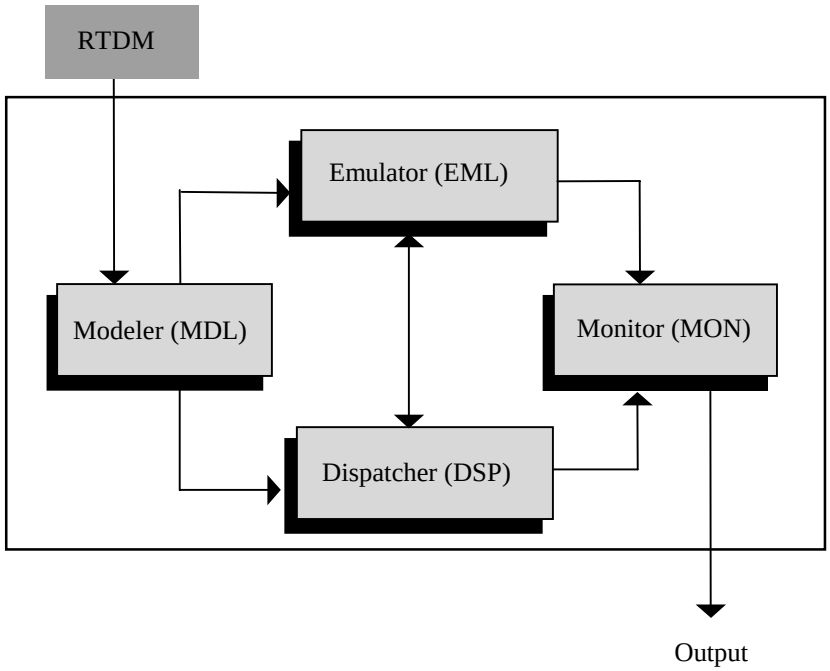


Figure 5. The Architecture of the Simulator

4.2 Main Building Blocks

SML is an event-driven simulator and consists of four building blocks (Fig. 5), namely *modeler* (MDL), *emulator* (EML), *dispatcher* (DSP) and *monitor* (MON). MDL retrieves all the required information from the system database to construct the system model. This information includes released orders, route sheet structures, various capacity data, real system data and correction

suggestions. Real system data is used to assess the initial state of the system in a simulation run. The basic model entities are released orders, machines, buffers and operators.

The main function of EML is the execution of the events occurring at the shop floor in a timely manner. Such events are an operation start/end, a machine breakdown, a maintenance start/return, an operator's work start/end etc. Probably, the most important task performed by EML is the generation of new events as a result of the execution of the current event. These events are dynamically created, given a time stamp and then stored in a sophisticated data structure, called the *event calendar*. EML executes the events in an order specified by DSP. DSP locates the next event to be executed, each time it is called by EML, and returns control to EML. The events waiting for execution are stored in the event calendar according to their time stamps. The event calendar drives the simulation and is updated every time DSP is called.

MON interacts with the other two modules and collects various process data. Data collection is carrying out whenever an event execution finishes. When a run terminates, MON is called again to make statistical calculations. The produced *statistical information*, that constitutes the output of SML, includes machine utilisation levels, average buffer loads, operation processing/waiting times, work-in-progress (WIP) levels, scrap quantities, problems identified (e.g. dropped orders), total shop output etc.

4.3 Modelling Stochasticity

Uncertainty in the real system is represented via the *stochastic parameters* of the model. Machine breakdowns are truly stochastic events that happen at non uniform time intervals, and historical data are necessary to predict this behaviour. A stochastic parameter is used to model the time of a breakdown via the Monte Carlo method [6, 29]. Two other stochastic parameters concern scrap quantities and the time an operator works. Both parameters are represented by proper probability distributions, such as the Gaussian, the Weibull, the Gamma and the Pearson distribution [6], given their average value and its variance.

Breakdown data is calculated off-line, before a simulation run. It is then stored in an appropriate model entity and is used by EML during the run. Breakdown data may remain unchanged between successive simulation runs after the schedule revisions have been made or a new breakdown data pattern may be generated before a new run begins. This capability helps manager in making comparisons between successive runs. Scrap quantities and operator times are computed by EML, during a run. So, by default, this data changes from one run to another, but the values obtained from each run are expected to be close enough to each other due to their small variances. The values produced in a run can be stored for use in later runs.

Finally, the times for machine setups and operation processing are considered by the simulation model as deterministic. Although this assumption seems to be quite reasonable for factories with high degree of automation in the flow line, in a more general setting both parameters have to be modelled with appropriate probability distributions, too.

4.4 Implementation Issues

In the integration of the SML with the PMS, flexibility of the simulation model is the key factor [12, 17]. There are several different classes of data provided by different data sources. All these data classes are stored in the system database. Consequently, SML

operation is driven from RTDM tables, while RTDM assures consistency of the simulation data. The main advantage of this approach is that the user does not need to have any simulation expertise, as SML runs off a database. Moreover, the internal structure of SML needs not to be changed when new products and processes have entered the system.

All building blocks of SML are built in C for two main reasons. First, SML can easily communicate via C with the other two technological platforms, a relational database and an expert system shell. Second, C allows very efficient RAM implementations in both time and space. An example is the event calendar structure. Event calendar has been implemented as a priority queue [20] which supports fast location of the next event to be executed. Another example is the use of dynamic list structures to store the model description data. Lists structures were chosen instead of fixed arrays [3] because they allow better memory management.

5. Knowledge-Based Component

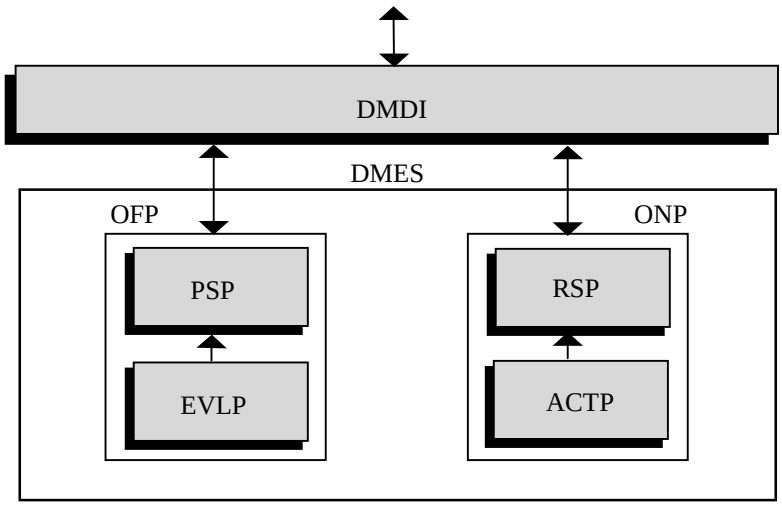
In the design of DM, the expert systems (ES) approach (see [27] for an account of the existing approaches) was mainly followed. This approach tries to mimic the reasoning process of an expert. Rules integrated with procedures are used for knowledge representation. There are a number of rule-based systems designed for production scheduling [5, 18, 24]. A difference of DM with existing scheduling systems based on the ES approach is that they are rather autonomous systems, that is the user is not actually involved in the decision process, and most of them do not deal with reactive scheduling.

5.1 Functional Architecture

DM includes three modules: DM expert system (DMES), DM user interface (DMUI) and DM data interface (DMDI) (Fig. 6). The main core of DM is DMES. DMES consists of two main parts, called *off-line part* (OFP) and *on-line part* (ONP).

OFP deals with making improvements to planned schedules. It consists of two parts, *evaluation part* (EVLP) and *predictive scheduling part* (PSP). EVLP is responsible for assessing the simulated schedule; it makes an estimation of the acceptability of the planned schedule. PSP actually proposes improvements to the schedule, if it is not acceptable. It makes suggestions about changes to the schedule in terms of the following *revision actions*: 'shift order', 'remove order', 'insert order', 'change quantity' and 'change rate'. Thus, OFP mainly deals with what is called *predictive scheduling* (see e.g. [7]). OFP can be activated by PM during the off-line operation mode.

ONP is concerned with making decisions on corrective actions in response to disruptions due to abnormal events occurring during the real execution of a schedule. ONP also consists of two other parts, namely *action part* (ACTP) and *reactive scheduling part* (RSP). The first gives advice to PM about preparatory actions that should be taken just after an abnormal event has occurred, or necessary actions after the recovery from an abnormal situation. The actions may refer to the work-center related to the event or to other active work-centers. RSP suggests on-line rescheduling scenarios in an abnormal situation. So, ONP deals with what is called *reactive scheduling* (see e.g. [7]). ONP is automatically activated during the on-line mode as soon as an abnormal event occurs and is recorded in the database.



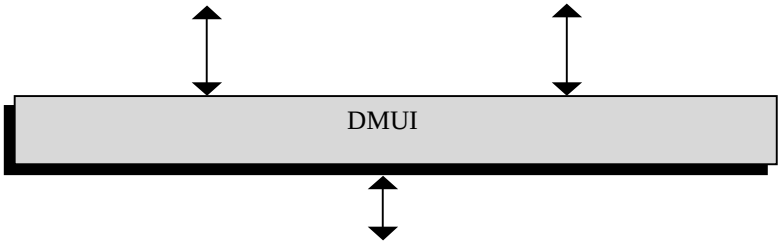


Figure 6. The Main Modules of the Decision Maker

Again, given the correspondence between the tasks of the production manager and the DM modules, we characterise the architecture of DM as *task-oriented*.

DMUI is the means for interacting with the user. The user can ask for alternatives and explanations. The output of DM, apart from suggestions, includes graphical representations of a schedule, via a Gantt chart, and of production lines layout, viaa schematic diagram. DM communicates with RTDM and (via RTDM with) SML via DMDI (see Section 5.3).

5.2 Implementation Tool

DM has been implemented in the Gensym's G2 Real Time Expert System Shell [12], which influenced its design. G2 is an object-oriented expert system development tool, where everything is defined as an object. An *object hierarchy* is used for static knowledge representation and *rules* integrated with *procedures* are used for representation of the dynamic (problem solving) knowledge. Apart from expert knowledge, a rule may also encode control knowledge via *control actions*. G2's real-time capabilities have facilitated watching production execution in real-time and making fast inferencing. Moreover, its graphical display environment is used for Gantt chart and production lines graphical representation. Finally, its capability for external communication according to GSI (Gensym Standard Interface) protocol was very important for implementing interactions with the other components. GSI protocol allows a G2 application to exchange data with other applications via RPCs (Remote

Procedure Calls). Arguments may be passed from the calling application to RPCs and RPCs may return data to the application.

5.3 Setup Subsystem

Before DM is ready for operation, a setup process is necessary. This is performed by the *setup subsystem* (Fig. 7). It consists *of G2 inference engine, DMDI* (the *bridge*), *system rule base (SRB), system procedure base (SPB)* and *shop floor model base (SFMB).*

DMDI (also referred to as *the bridge*) is developed in C. It uses embedded SQL calls combined with calls to the Gensym's interprocess protocol GSI. DM can call functions of DMDI (RPCs) as well as exchange data with them. DMDI can then perform the necessary transactions with RTDM, and store or retrieve data on behalf of DM. Also, via a polling technique (see below), DM activates event checking routines of DMDI to detect events at the shop floor or signals from SML.

SRB contains rules for initialising the static data of DMES. SPB contains the RPCs and other setup procedures.

Once DMES is loaded, the setup subsystem takes over and performs the following actions:

- Activates the bridge process and waits until the bridge has been successfully connected to the system database.
- Starts the procedures which retrieve the shop floor model from the system database and create the corresponding objects.
- Activates the bridge polling mechanism (see below).

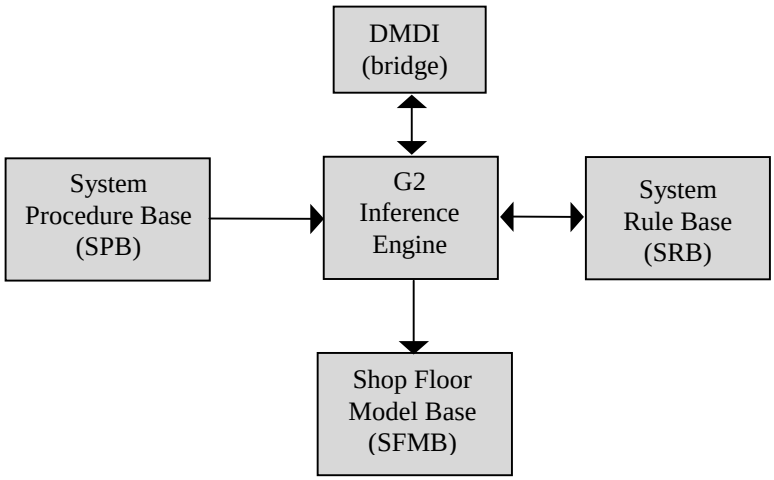


Figure 7. The DMES Setup Subsystem

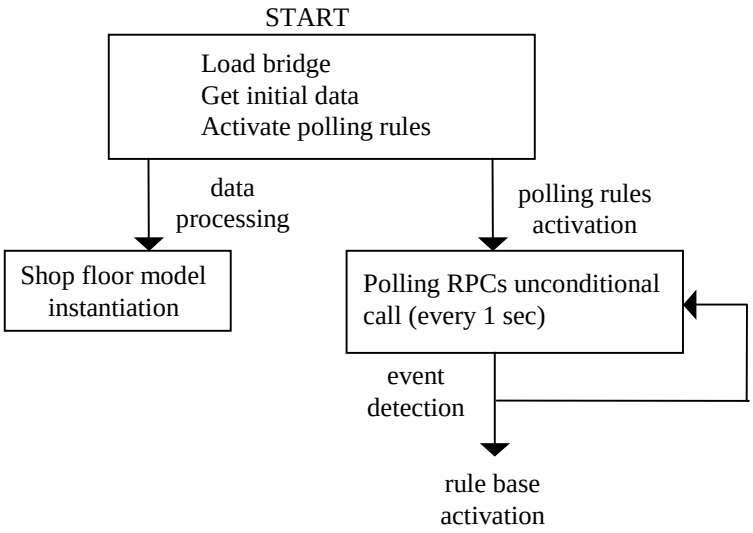


Figure 8. The DMES Setup Functions

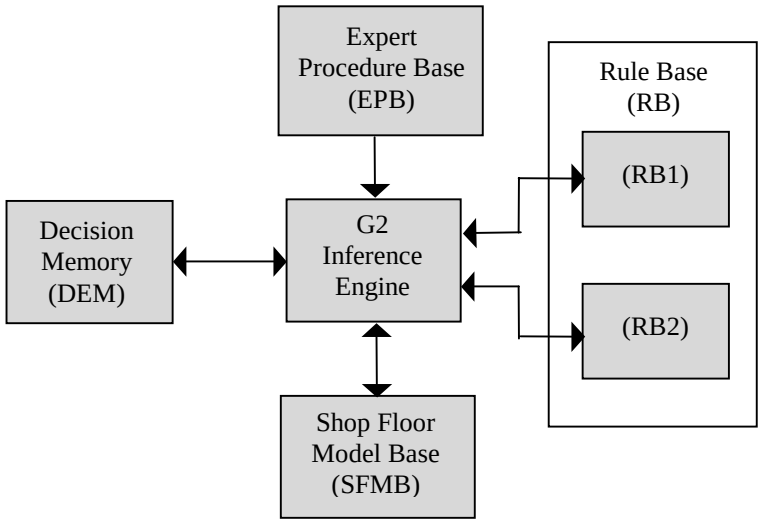
After the bridge has been activated, two RPCs are called once every second. The first deals with timing, whereas the second checks whether any event has occurred or any signal from the simulator has been received. So, if, at any time, the bridge detects some event at the shop floor or some signal from the simulator, then within a second this is reported to the expert system via the RPCs. This way of signalling is known as a “busy waiting” technique or “polling technique”. The setup functions are depicted in Figure 8.

5.4 Problem Solving Subsystem

The problem solving architecture of DM, that is the architecture of DMES, is depicted in Figure 9. It involves five modules: *G2 inference engine*, *shop floor model base (SFMB)*, *rule base (RB)*, *expert procedure base (EPB)* and *decision memory (DEM)*. RB has two instances, namely *off-line rule base* (OFRB) and *on-line rule base* (ONRB), that are not simultaneously used; OFRB is used in the off-line mode, whereas ONRB in the on-line mode.

Rules in OFRB can be triggered whenever statistical data is made available by a simulation run. Execution of these rules results in suggestions for off-line changes to the simulated production plan. OFRB consists of two parts, *evaluation rule base* (EVLRB) and *predictive scheduling rule base* (PSRB), related to EVLP and PSP of OFP respectively. EVLRB decides on the acceptability of the schedule (acceptable, non-acceptable). PSRB deals with off-line rescheduling of the planned schedule and is activated when the schedule has been assessed as non-acceptable. ONRB contains rules that are activated whenever an abnormal situation arises during the execution of a production schedule. It consists of two parts, *action rule base* (ACTRB) and *reactive scheduling rule base* (RSRB), related to ACTP and RSP of ONP respectively. So, ACTRB deals with actions required just after an event has occurred or after its recovery and RSRB with reactive scheduling.

Rules in a rule base are organised in *rule categories*, each concerning some decision aspects. The contents and structures of OFRB and ONRB are presented in Section 5.6.



Off-line mode: RB=OFRB(RB1=EVLRB, RB2=PSRB), EPB=OFPB
On-line mode: RB=ONRB(RB1=ACTRB, RB2=RSRB), EPB=ONPB

Figure 9. DMES Problem Solving Architecture

EPB also has two instances, namely *on-line procedure base* (ONPB) and *off-line procedure base* (OFPB), not simultaneously used. The first contains procedures that may be called by rules in ONRB, whereas the second those to be called by rules in OFRB.

G2 Inference Engine, which is the heart of the system, performs rule-based inferencing based on the expert and control knowledge stored in the rule and procedure bases. The distribution of inference knowledge in partial rule bases and further in rule categories and the guidance of inference, through control actions, reduces the search space and makes decision making more efficient. A forward chaining strategy is mainly used.

In SFMB, basically the shop floor model is represented. An object-oriented representation is used for that purpose. Classes representing various types of the real entities (e.g. machines, operations etc.) involved in the production process are organised

in a hierarchy. Each class is specified by a number of attributes. Instances in the hierarchy represent specific entities. Triggered rules may result in changing values of the attributes and/or creating new or deleting old (transient) objects. Constraints are represented either as attributes of or as relations between objects. The content and structure of SFMB is presented in Section 5.5.

Finally, DEM is the place where intermediate or final decisions are stored. DEM contains data items that are used to keep values of *expert parameters* during inferencing, such as delay-type, rescheduling-type etc. Expert parameters are variables used to represent notions of the decision making process. Thus, DEM reflects at any time the decision making progress.

5.5. Shop Floor Model

5.5.1. Classes and Objects

We use object classes to represent the entities of real world objects met in the problem. In our hierarchy (Fig. 10), all classes have as superior class the most general class `DELTA_OBJECT`. This class has three subclasses, `PROD_LINE_OBJECT`, `ON_LINE_OBJECT` and `OFF_LINE_OBJECT`. The classes (with their attributes) in the hierarchy in conjunction with the relations and connections (see below) constitute the shop floor model in our application. This model includes the necessary entity types with their attributes and entity relations to represent a specific instance of a flow shop environment, that of a yoghurt plant. The model is created during the setup of the system by taking data from the corresponding system database tables via the bridge (DMDI) (Section 5.3).

The subclasses of `PROD_LINE_OBJECT`, that are classes related to the production line layout, are:

- *workcenter*: This class describes a workcenter. A workcenter is a set of one or more similar machines that can be considered as one operational unit. In our case each workcenter is a single machine or a buffer. The most important attributes of ‘workcenter’ are: ‘max-capacity’, ‘current-capacity’, ‘normal-

setup-time', 'special-setup-time', 'low-critical-time', 'high-critical-time', 'status' (idle, setup, active, broken, repair), 'affected-machines'.

  - *machine, buffer*: Since a workcenter can be either a machine or a buffer, workcenter has two subclasses, 'machine' and 'buffer'. They inherit their attributes from workcenter, however they have a few own attributes.

- *buffer-tank*: A buffer may consist of one or more buffer tanks. Therefore, another class, 'buffer-tank', has been introduced with as main attribute 'max-capacity'. Each instance of buffer-tank is connected to an instance of buffer.

The subclasses of `ON_LINE_OBJECT`, that are classes related to the real production process, are:

- *operation*: This class describes an operation. An operation is the execution of a job in a machine or a buffer that leads to an intermediate or a final product. Some of its most important attributes are: 'workcenter' (where the operation is carried out), 'order' (which the operation is part of), 'setup-time', 'start-time', 'available-time', end-time'.
- *event*: It represents an event. An event is an abnormal change in the current state of the production process (e.g. a breakdown) that disturbs, in some way, the normal shop floor production flow. Some of the most important attributes of 'event' are: 'event-time' (when the event occurred), 'repair-time' (taken from occurrence to recovery), 'repair-type' (related to rescheduling decisions),'product-condition' (after the event recovery), 'current-scrap' (due to the event), 'schedule-type' (related to rescheduling decisions).
  - *breakdown*, *scrap-order*, *bottleneck*, *rush-order*: They represent various types of an event, therefore 'event' has these classes as subclasses. A breakdown refers to any problem occurring in a workcenter that sets it out of operation. A scrap order mainly refers to bad product quality. A bottleneck is the condition in which more than one operation are waiting for execution in an already busy workcenter. Finally, a rush order is resulted when a stock

out situation occurs. Each of these classes inherit the attributes of event and has its own as well. Whenever DM detects an event (e.g. a breakdown), an instance of the corresponding class is created.

The subclasses of `OFF_LINE_OBJECT`, that are classes related to off-line aspects of the shop floor model, are:

- *workcenter-gantt*: It is a record of the corresponding table of the system database that includes information about the off-line gantt chart. It has attributes such as: ‘start-time’, ‘ ‘end-time’ (of an operation), ‘order-number’ (of the operation), ‘workcenter-name’ (of the operation).
- *order*: It represents an order. Its attributes include ‘start-date’, ‘due-date’, ‘order-number’, ‘order-product’, ‘order-quantity’, ‘order-status’. It has the following two subclasses that inherit its attributes.
  - *out-of-date-order*, *dropped-order*: They represent orders that in the simulation run were completed after the planned date or dropped, respectively.
- *workcenter-statistics*: Describes the statistics resulted from a simulation run. Its main attributes, are: ‘workcenter-utilisation’, ‘average-operator-time’, ‘actual-operation-time’.
- *dm-suggestion*: Describes the suggestions given to the production manager in the off-line decision process. Its main attribute is ‘status’ (active, inactive). Since there are a number of types of suggestions, ‘dm-suggestion’ has a number of subclasses:
  - *capacity-change*, *quantity-change*, *day-reorder*, *order-merge*, *order-delete*, *order-shift*: They represent revision actions to be applied to the planned schedule.

To create instances of the above classes, DM submits corresponding requests to the bridge, supplying (if required) necessary query data. The bridge executes the query, and stores the data in a local space. It then reports back to G2 the number of records retrieved from the system database. Subsequently, G2 issues the correct number of data requests to the bridge, and retrieves the attribute values for the objects to be created.

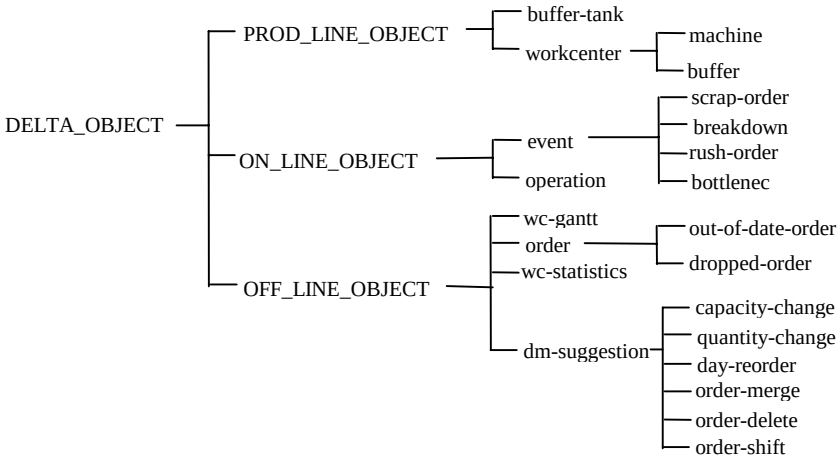


Figure 10. The Shop Floor Model Class Hierarchy

5.5.2. Connections and Relations

Apart from classes and their attributes, a number of *connections* and *relations* between the objects (instances) that reflect relationships of the corresponding real-world entities are also used to specify the shop floor model.

Connections are graphical relations established (drawn):

- between two workcenters, if the first workcenter is after the second in the shop floor model of the plant.
- between a buffer and a buffer-tank, if the buffer-tank belongs to the buffer.
- between an event and a workcenter, if the workcenter is the one which the event occurred at.

Also, relations have been established among objects in the hierarchy and are used during inferencing. Such are:

- *the-next-order-of*: defines the sequence of orders in a schedule. An order has only one next-order.
- *the-next-operation-of*: defines the sequence by which the operations that correspond to an order are executed.

- *the-current-operation-of*: defines a relation between workcenters and operations that indicates the operation that is currently being executed by a workcenter.
- *the-order-of*: defines a relation between orders and operations that specifies which order corresponds to an operation.
- *compatible-with*: defines a relation between operations. It specifies which other operation can be simultaneously executed in the same workcenter with an operation.

5.5.3 Shop Floor Model Base

Shop floor model base (SFMB) consists of three parts: permanent model base (PMB), schematic model base (SMB) and temporary model base (TMB). PMB contains all classes (types) of objects related to the shop floor model. SMB contains the specific objects (instances) that constitute the certain shop floor layout. Each object is displayed by a special icon and its connections to other objects. TMB contains temporarily created objects (instances), such as the current operations, which are later destroyed.

5.6 Problem Solving Knowledge Representation

Problem solving knowledge was extracted from the (experience of the) production management staff of the yoghurt plant via the methodology explained in Section 2.1. That knowledge was enhanced with simple analytical processes, like computing the consequences of an order shift or the amount of the reduction to the quantities of a number of orders etc., to improve the problem solving capabilities of the system. Expert knowledge was translated into rules, stored in the rule bases, whereas analytical methods into procedures, stored in the procedure bases, which are called from within the rules. That is, any computation required during a decision making process is performed via those procedures.

5.6.1 Off-Line Rule Base (OFRB)

The off-line rules are designed to evaluate simulation runs and make suggestions to improve or overcome the problems of the simulated production plan. They are organised into the following categories:

- *schedule evaluation*: It aims to evaluate the simulated run. Based on the number of out of date orders, the amount of their delays as well as the number of bottlenecks occurred, it classifies the simulated schedule into: normal, short-delayed, medium-delayed, long-delayed or ultra-long-delayed schedule.
- *action selection*: A normal schedule is accepted, whereas an ultra-long-delayed is rejected by the rules of this category. The other types of schedule cause calls to one of the rest rule categories.
- *short-delay*: The rules in this category try to revise a short-delayed schedule. First, the schedule deviation is computed as

  $$\sum_{m} (delay_i * quantity_i)$$

  where $delay_i$ and $quantity_i$ are the delay and the quantity of the i^{th} out of date order. If it does not exceed a threshold, the schedule is accepted, otherwise a number of rescheduling strategies are applied to it, such as: reordering orders in the same day by product type, decreasing the quantities of orders of the same product, unifying two or more consecutive orders of the same product, deleting the shorter order or increasing the capacities of some workcenters.
- *medium-delay*: It deals with the medium-delay schedules. First, the day with the maximum product quantity to be produced is found. If that (actual) maximum quantity is exceeds the maximum quantity allowed, some orders are shifted to other days so that each day's overall quantity remains below the maximum allowed. If the maximum quantity is less than the maximum allowed, it starts looking for sequences of medium-delayed orders with a short-delayed at their beginning. If found, we delete or reduce some or all of the short-delayed ones. If there are no such sequences, it increases the capacities of the workcenters involved, where possible.
- *long-delay*: It deals with the worst case of schedules, long-delayed schedules. First, sequences of the above type are looked

for and the same procedure is followed. Furthermore, if there is accumulation of orders in the last day of the week, reductions or deletions of orders take place.

From the above rule categories, 'schedule-evaluation' constitutes EVLRB, whereas the rest belong to PSRB.

5.6.2 On Line Rule Base (ONRB)

The on line rules handle cases of abnormal events, such as a breakdown, a scrap order, a bottleneck or a rush order, that occur during real production. Abnormal events result in delays of the scheduled production. Delay amounts to at least the time required to bring production back to normal operation, e.g. a broken workcenter back to order. Also, an abnormal event may affect the quality of the product in process. Therefore, whenever abnormal events occur, need for recomputing the production plan (rescheduling) may arise, especially when the recovery time is high or the product quality is affected. This task is carried out by ONRB. ONRB contains the following rule categories.

- *event detection*: The rules in this category first check whether any abnormal event has been occurred (by checking the values of certain variables, which are periodically updated by the RTDM via the bridge). If it has, they display messages about the type of the event and the required actions by the production manager to handle the case as well as they set initial values to the event related attributes and call corresponding rule category(ies). To avoid complexity, in the sequel we refer to rule categories related only to breakdowns, which are the most interesting case of abnormal events and subsume a significant part of the problem solving knowledge for the other types of events.
- *repair-time classification*: The time required to repair a broken (stopped) workcenter is classified in low-short, low-long, fuzzy-short, fuzzy-long or high, based ona comparison of it to the low critical and high critical times of the workcenter, on the one hand, and the maximum delay time, on the other hand. Low critical and high critical times are related to the condition of the product processed in a workcenter, and maximum delay time is

related to the maximum delay the schedule in the current day can tolerate without any serious problem. Classification of repair time in fuzzy-short or fuzzy-long classes requires testing the product, before proceeding.

- *preventive actions*: The rules of this category propose preventive actions whenever a breakdown occurs, mainly related to holding one or more operations.
- *breakdown evaluation*: Based on its repair time classification and the product condition, a breakdown is evaluated and either a decision is reached (no changes case) or other rule categories are called for further processing.
- *affected machines*: These rules propagate the breakdown consequences of a breakdown to workcenters affected by the broken workcenter. Based on these consequences, use of a substitute machine, if available, is examined. Factors that influence such a decision are the relation of the overall break time with the low and critical times of the affected machines and the relation of the overall scrap created due to the breakdown with the capacity of the affected machines and the maximum allowed overall scrap.
- *reactions*: These rules decide whether rescheduling is required or not, based on the relation between the overall scrap created by a breakdown and the maximum allowed scrap, on the one hand, and the relation between the overall delay due to the breakdown and the maximum allowed delay time for the current day, on the other. If rescheduling is required the next rule category is invoked.
- *rescheduling*: The rules here recalculate the production plan by invoking the rules in the next category for every order, starting from the current order of the broken workcenter. They do so until those rules detect no more conflicts in the schedule.
- *conflicts*: It is invoked by the rescheduling rules and focus on a single order. It checks all pairs of operations which are executed in the same workcenter, and consists of an operation of this order and an operation of its next order. It identifies possible overlaps, which are called *conflicts*. For each conflict it detects, shifts the conflicting order to the future for an appropriate time period,

unless the conflicting order is the last order of the next day, in which case the quantity of the order is reduced accordingly, or the order is cancelled.

From the above rule categories, ‘event detection’, repair-time classification’, ‘preventive actions’, ‘breakdown evaluation’ and ‘affected machine’ belong to ACTRB, whereas ‘reactions’, ‘rescheduling’ and ‘conflicts’ to RSRB.

5.6.3 Inference Flow

Inference in the system can be seen as a flow from rule category to rule category until a conclusion (decision) is reached. In each rule category a local inference takes place, which either leads to a conclusion or to a call to another rule category, via control actions. In the local inference several procedure calls may take place to assist making a decision. Procedures are called from the procedure bases.

Inference flow between the rule categories in the off-line and the on-line mode is depicted in Figures 11 and 12, respectively. A bold arrow indicates a conclusion (decision) draw that results in a flow stop, whereas a dashed bold arrow indicates proposition of some actions to be taken by the production manager. A circle over arrows indicates conjunctive flow, which otherwise is disjunctive.

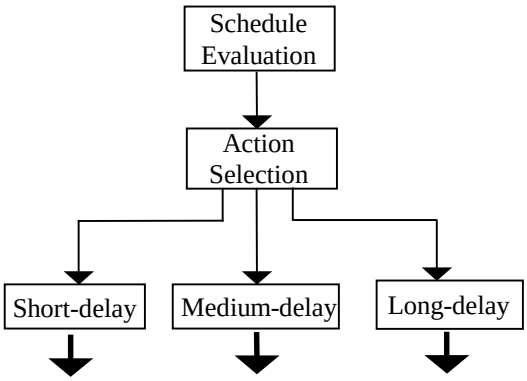


Figure 11. Inference Flow in Off-Line Mode

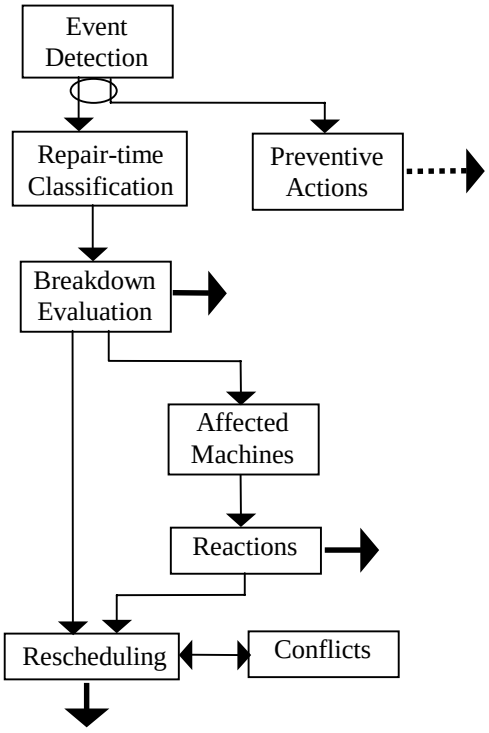


Figure 12. Inference Flow in On-Line Mode

6. Conclusions

In this paper a PMS based on an improved MRP II model is presented. Improvement concerns production control at the PAC level. An IDSS is attached to the PAC system that extends its functionality. The IDSS uses simulation and knowledge-based decision making, by including an event-driven simulator (SML) and a knowledge-based decision maker (DM). The IDSS extends the PAC system in two respects. First, it provides mechanisms for revising the often unrealistic planned schedules through a simulator that uses real data for its initialisation. Second, it is capable of performing on-line rescheduling (or reactive scheduling), via a knowledge-based decision maker (expert system) that makes decision on-line as well as off-line. The system performs in two main modes, the off-line and the on-line mode, corresponding to production manager's two main tasks. SML and DM communicate via a relational real-time database manager.

In the off-line operation mode the production manager can make a simulation run of the production schedule via SML, evaluate and decide on revisions, with DM support, and apply the revisions via SML. The production manager can repeat this cycle as many times as required to reach a realistic and cost-effective plan. In the on-line mode, the system responses to abnormal event occurrences by giving advice to the production manager, via DM, for corrective actions in the production lines.

This improvement was basically developed in the context of the CEC ESPRIT Hellenic Special Actions project DELTA-CIME and a prototype of the system applied to a yoghurt plant with a flow shop environment in Greece [8, 13, 14].

Although what is presented here is a specific application, one can easily abstract from the details and see the extended architecture of a classical MRP II-based system. This architecture can be used as the basis for the development of an improved MRP II-based tool for production management. This requires a parameterisation of the system to be able to accommodate different types of manufacturing environments. This

parameterisation mainly concerns the components of the IDSS and constitutes a direction for further work.

References

[1] Alting L., Bilberg A., “When Simulation takes Control”, Journal of Manufacturing Systems, Vol.10, No.3, 1993, 179-193.

[2] Bauer A., Bowden R., Browne J., Duggon J., Lyons G., “Shop Floor Control Systems”, Chapman and Hall, 1994.

[3] Browne J., Davies B. J., “The design and validation of a digital simulation model for job shop control decision”, Int. Jour. Prod. Res., Vol.22, No.2, 1984, 335-357.

[4] Browne J., Harhen J., Shirman J., “Production Management Systems”, Addison-Wesley, 1988.

[5] Bruno G., Elia, Laface P., “A rule-based system to schedule production”, IEEE Computer Vol.19, No.7, 1986, 32-40.

[6] Carrie Al., “Simulation of Manufacturing Systems”, John Wiley and Sons, 1992.

[7] Collinot A., Le Pape C., “Adapting the behaviour of a job-shop scheduling system”, Decision Support Systems, Vol.7, 1991, 341-353.

[8] Dendirs N., Hatzilygeroudis I., “The Decision Making Module”, Deliverables D141 and D142, Project DELTA-CIME, Hellenic ESPRIT III Special Actions (No 7511/C18), CTI, January 1995.

[9] Donciulescou D.A., Filip F.G., “Intelligent DSS in Production Control for Process Industry”, Proceedings of the Advanced Summer Institute'94 (ASI'94) in Computer Integrated Manufacturing & Industrial Automation (CIMIA),1994, 181-187.

[10] Duggan J., Bowden R. and Browne J., “A Simulation Tool to Evaluate Factory Level Schedules”, ESPRIT Workshop, 1989.

[11] GENSYM Corp., “An Introduction to G2”, Cambridge, MA, 1993.

[12] Haddock J., Seshadri N., Srivatsan V.R., “A Decision Support System for Simulation Modelling”, Journal of Manufacturing Systems, Vol.10, No.6, 1991, 484-491.

[13] Hatzilygeroudis I., Sofotassios D., Spirakis P., Triantafyllou V., Tsakalidis A., “A PMS System Integrating Knowledge-Based Technology and Simulation with On-Line Production Control”, Technical Report 94.09.46, Computer Technology Institute, Patras, Greece, 1994 (long version). Also in: Proceedings of the Advanced Summer Institute'94 (ASI'94) in Computer Integrated Manufacturing and Industrial Automation (CIMIA), 1994, 192-197 (short version).
[14] Hatzilygeroudis I., Sofotassios D., Dendirs N., Spirakis P., Tsakalidis A, “Architectural Aspects of an Intelligent DSS for Flow Shop Production Control”, Advanced Manufacturing Forum, Vol.1,1996, 75-84.
[15] Higgins P., Lyons G., Browne J., “Production Management Systems: An Hybrid PMS Architecture”, ESPRIT Workshop, 1989.
[16] Hsu W-L., Prietula M.J., Thomson G.L., "A mixed-initiative scheduling workbench Integrating AI, OR and HCI", Decision Support Systems, Vol.9, 1993, 245-257.
[17] Kaye M., Sun Q., “Data manipulation for the integration of simulation with online production control”, CIM Systems, Vol.3, No.1, 1990, 19-26.
[18] Kerr R.R., Ebsary R.V.,”Implementation of an expert system for production scheduling”, European Journal of Operational Research, Vol.33, 1988, 17-29.
[19] Manheim M.L., "An Architecture for Active DSS", Proceedings of the 21st Hawaii International Conference on System Sciences (HICSS'88), 1988, 356-365.
[20] Mehlhorn K., Tsakalidis A., “Handbook of Theoretical Computer Science. Chapter 9: Data Structures”, North Holland, 1990.
[21] Meyer W., Isenberg R., “Knowledge-based factory supervision: EP932 results”, Int. Journal on Computer Integrated Manufacturing, Vol.3, No.3 & 4, 1990, 206-233.
[22] Orliky J. A., “Material Requirements Planning”,1975, McGraw Hill.

[23] Rao H.R., Sridhar R., Narrain S., "An active intelligent decision support system-Architecture and simulation", Decision Support Systems, Vol.12, 1994, 79-91.
[24] Savell D.V., Perez R.A., Koh S.W., “Scheduling Semiconductor Wafer Production: An Expert System Implementation”, IEEE Expert, Fall 1989, 9-15.
[25] Schallock B., Arlt R., “Interactive Knowledge-Based Shop Floor Control in a Small Manufacturing Enterprise Environment”, Synthesis report, EP1381-5-85, Berlin, July 1992.
[26] Smith S.F., Fox M.S., Ow S.P., "Constructing and Maintaining Detailed Production Plans: Investigations into the Development of Knowledge-Based Factory Scheduling Systems", AI Magazine, Fall 1986, 45-61.
[27] Smith S.F., “Knowledge-based production management: approaches, results and prospects”, Production Planning and Control, Vol.3, No.4, 1992, 350-380.
[28] Teng J.T.C., Mirani R., Sinha A., "A Unified Architecture for Intelligent DSS", Proceedings of the 21st Hawaii International Conference on Systems Sciences (HICSS'88), 1988, 286-294.
[29] Watkins K., “Discrete Event Simulation in C”, McGraw-Hill, 1993.