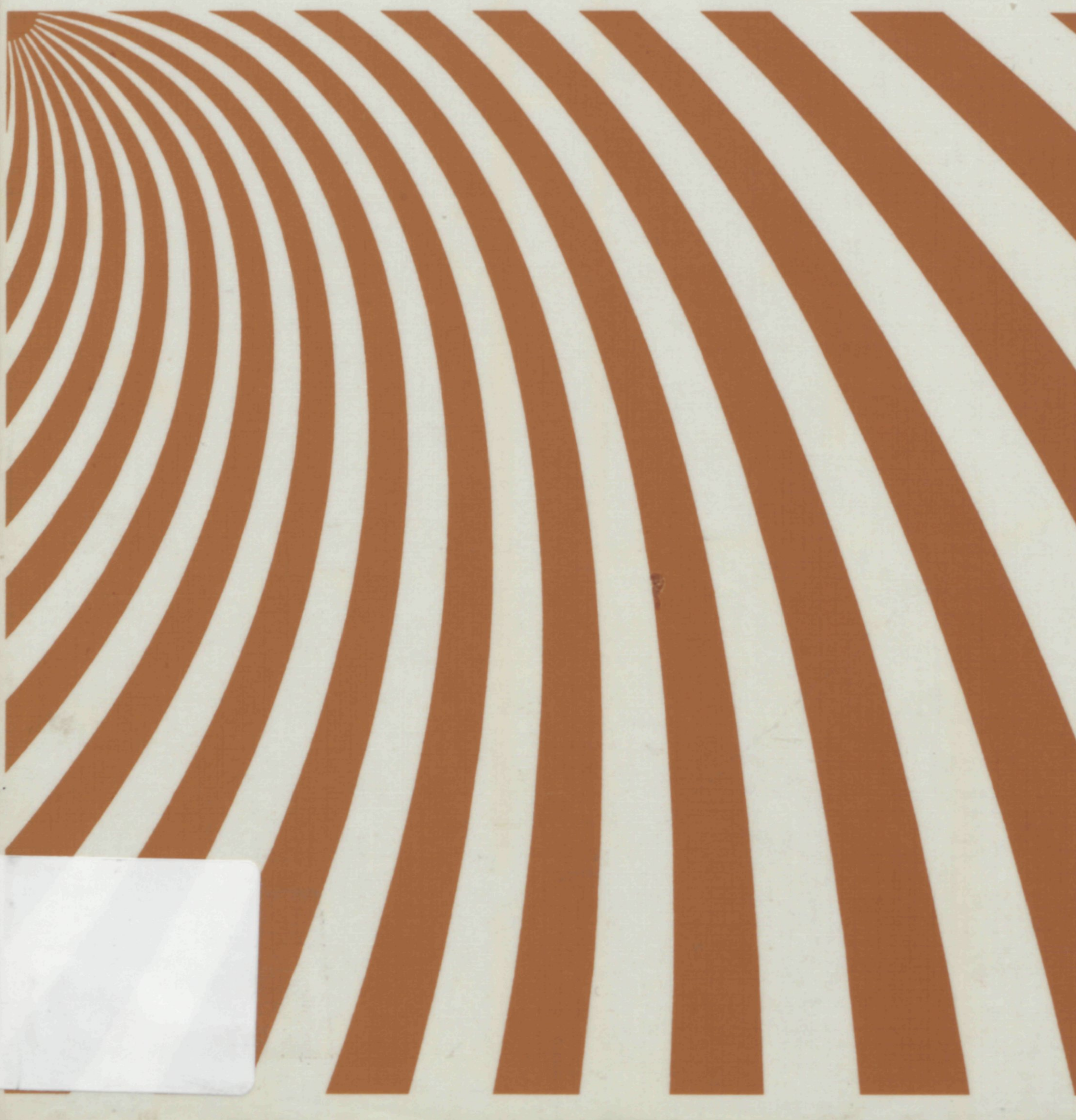
Scheduling farm operations: a simulation model

E. van Elderen



Simulation Monographs

Simulation Monographs is a series on computer simulation in agriculture and its supporting science

Scheduling farm operations: a simulation model

E. van Elderen

Pudoc Wageningen 1987

CIP-gegevens Koninklijke Bibliotheek, Den Haag

ISBN 90-220-0858-4

Printed in the Netherlands

Contents

SUMMARY

The scheduling of operations on a farm is described as a system and the theory and models used are presented. The program which simulates the scheduling system is written in SIMULA. A base model contains the basic components of the system such as men, machines, operations and crops. An experimental frame describes the input and output and defines the simulation. An example is given of the scheduling of operations during wheat harvesting. Verification of the program and validation of the model are discussed. Extensions valid for wheat harvesting are mentioned and suggestions for use in other circumstances and applications are described.

Keywords: Scheduling, Simulation, Farm Operations, Machinery Selection, Workability, Timeliness

1 INTRODUCTION

Every day the farmer faces the problem of scheduling farm operations; 'scheduling' means determining the time when the various operations should be performed. Availability of men and machines, crop requirements, the weather and the timeliness of operations all affect the resulting yield, the quality of the product and the costs of operations. The farmer learns from experience how and when to decide and what to prefer in uncertain weather conditions. Low yields and poor quality due to untimely operations and high costs for men and machines can cause poor results. But how can better results be achieved?

To gain better information in such a situation a farmer can supplement his experience by using a model: it allows experiment in alternative environments or with different management strategies or operational tactics. Models and techniques used to solve scheduling problems are: labour budgeting, linear programming, dynamic programming and simulation. The AIM of the study in this monograph is to develop *a flexible simulation model* of the scheduling problem of farm operations that can be used particularly as a research and advisory tool. The emphasis is on the simulation of realistic sequences of operations and development of crops, products and soil over time and on a realistic use of men and machines during workable intervals. Such a model which is reliable on the micro level (day to day) can be used as an instrument to achieve results that are necessary to make the long term decisions at the macro level of the farm.

The possible **use of a scheduling model** is:

- to test whether the results derived from several seasons are satisfactory for some area of crops and for the available men and machinery;
- to optimize the costs of the schedule by finding a compromise between harvesting as soon as possible and harvesting on a later moment; i.e. a compromise between the avoidable costs of overtime of men and the costs of machine use (such as drying hay or grain in storage instead of in the field) on the one hand and the so-called timeliness losses that result from lower yield or poorer quality when a crop is harvested later on the other hand;
- to show the effect on the results of different areas, different number of men and types of machinery and different tactical rules to schedule operations; machinery selection is based on this information;
- to show the effect on the work organisation, the scheduling of different soil conditions (for instance, with or without drainage; affect on workability) or different techniques (for instance, disease control in crops or the use of chemical additives on silage; affect on workability and quality) or

different crop varieties (for instance, prospect of a high-yield variety versus a local variety with low yield and disease resistance);
- to show the effect of better weather forecasts on the tactical/operational decisions and so on the results.

These practical uses of a scheduling model are supplemented by the use for research purposes such as:
- to show the effect of information collection (aggregation) on the results, for instance, the workability of each crop, product or soil for each hour is collected for a day, a week or even a month and the original chronological sequence is lost;
- to show the effect of relaxation of information on the results , for instance, the simultaneous occurrence of workabilities is relaxed to independent workability constraints of each crop, product or soil;
- to find a heuristic strategy for the simulation model that is efficient, i.e. it does not require too much computer time and takes good decisions or even optimum decisions.

Such a flexible simulation model is described. The theory of scheduling of operations on a farm is described in Chapter 2 'Theory of scheduling'. The conceptual model is described as a system with interrelated components such as crops, men and machinery, and operations. Another view of the system as a sequence of events in time is also described. Finally a review of some models and techniques of solving a scheduling problem is described. Chapter 3 'A simulation tool; SIMULA' introduces SIMULA as a skilful tool to make a simulation program of the scheduling system. Chapter 4 'Base model' describes in detail the general components of the system. This 'base model' is extended in Chapter 5 'Experimental frame' with specific components (weather, scheduling rule and in/output) to form an experiment. The simulation is defined by the base model, the experimental frame and the input. The wheat harvesting is taken as an example to show the input, which defines the men, machines, crops and operations, and the output. The creation of appropriate input is described. Chapter 6 'Verification and validation' describes means to verify the program; the validation includes the behaviour of the model in several circumstances. Chapter 7 'Extensions' describes an extension of the wheat-harvesting model together with suggestions on how to explore the base model in other cases.

Knowledge of mathematics, probability theory or system theory is not assumed. Familiarity with agriculture and simulation may be an advantage.

Likely readers are researchworkers, teachers and students, who are interested in scheduling farm operations and in effects of managerial decisions on such a schedule and who like to use a well-described, accurate simulation tool. Decisions are concerned at the strategic level for instance, machinery selection and crop selection in the stochastic weather and workability environment, and on the tactical level to control a crop disease or to irrigate.

Readers' guide.

Chapter 2 is essential for anyone who wants to study the scheduling problem on a farm. If the base model and the experimental frame are to be used just as they are, the reader can omit Chapter 3 and the details of Chapter 4 and can concentrate on the input and output as described in Chapter 5. Those readers who wants to extend the models for specific situations may benefit from Chapters 3, 4 and 5 (the wheat-harvesting simulation model) and from Chapters 6 and 7 (the verification of new extensions and handling new situations).

2 THEORY OF SCHEDULING

The theory of the scheduling of operations is the background of what happens, and when and how it is performed. For instance wheat harvesting or straw baling occurs on Monday morning and is performed with, for instance, two or one man, a harvester, a baler and a tractor. There are two ways of describing the scheduling problem:

- the real problem is viewed as a system with interrelated components (crops, men and machinery, operations) and an environment (weather), Section 2.1 'Components and relations';
- the real problem is viewed as a sequence of events in time, Section 2.2 'State, event and dynamic aspects'.

The first method emphasizes more the static and the second method more the dynamic aspect of the system. They are complementary descriptions of a model of the scheduling system. Section 2.3 'Models and solution techniques' describes the relationship between some models and techniques such as linear programming, dynamic programming and simulation.

Throughout this chapter the example of wheat harvesting is used. It can be described as follows. The wheat is harvested by a combine-harvester that produces straw in the field and dry or wet grain for storage. Wet grain is dried in a grain drier. The straw is baled by a baler that produces bales in the field and the bales are gathered and stored. A more detailed description of the example is given when necessary in the following sections. 'Harvesting with a combine-harvester' necessitates also men, tractors and trailers to perform the operation (Table 2.1).

It is necessary to keep in mind the aim of the model of the scheduling system. In this monograph **the objective** is to develop models with a correct sequence of operations for crops and products and a realistic use of men and machinery during workable intervals. A correct sequence involves more

Table 2.1 Scheme of products in the wheat harvest and the operations(...).

```
wheat in the field --------------or---------
        :      (harvesting)     :         :
       \:/                     \:/       \:/
        V                       V         V
straw in the field        wet grain --> dry grain in storage
        :      (baling)             (drying)
       \:/
        V
bales in the field ------------
        :      (gathering)      :
       \:/                     \:/
        V                       V
    stubble                bales in storage
```

than meeting constraints placed on men and machines for the period; it presents a feasible and executable schedule. The art of modelling excludes those aspects of the real situation which have little influence on the results and are irrelevant. For example simulation of the actual geographical pattern of men, machines, tractors and trailers in the field or on the road (for instance, Kindler et al., 1981) may be not required at the level of this model; such information can be replaced by defining a rate of operation (including field work, transport and preparations) that may depend on the machines used, the properties of the crop and soil, the position of the field, etc. Thus the description of the system is limited and concentrates on those aspects of the scheduling of operations that may have a significant influence on the results. Other aspects from the real world are not considered. It should be noted that irrelevant aspects of the reality occur at two levels: too much detail and too general. Too general is for instance, the price of land or even the price of machines or crops as far as costs per hour already reflect price, depreciation, etc.

2.1 Components and relationships

A *system* is a limited part of reality and consists of interrelated components (or elements) that are relevant for the behaviour and the results of a system. In an open system the environment has an influence on some components of the system. The irrelevant parts of the real world are disregarded in the system and its environment. The system also contains relationships between the components and between the environment and the components. The components and relationships of the scheduling problem are described in the following sections. The description is general so that the phenomena and the behaviour of the scheduling system may be understood. In addition starting points are created for use in the base model (Chapter 4). A *model* is a simplified, relevant representation of a system and *simulation* is the art of building mathematical models and the study of their behaviour. The components, the environment and the relationships are shown in Table 2.2.

2.1.1 *Materials (crops, products) and weather*

The components 'materials' and 'weather' of the scheduling system can be seen as a separate subsystem: the *biological subsystem*. The general term 'material' will be used for crops, soil, materials (seed, fertilizer, etc.) and

Table 2.2 Scheme of components and relations of the scheduling system.

```
Men &      ----- Operations ----- Materials ----- Weather
machinery            :              (crops)
                     :
                  Decision
```

products (grain, straw, bales), because all behave in the same way in the scheduling system, i.e. are produced by operations or processed by operations. The production and processing of materials will also be described by the general terms delivery/delivering/supply and consumption respectively.

The 'history' of a crop starts with sowing or planting or even with preparation of the soil. The development of a crop is a complex autonomous process influenced by maintenance operations such as fertilizer application, weeding, spraying and irrigation. The harvesting operation is finally followed by operations that prepare the products for storage, consumption or marketing. In the case of a cereal crop, several 'materials' can be distinguished: soil, seed, sown grain, weeded crop, ripe crop harvested grain in storage, straw, bales, bales in storage, stubble field and ploughed field or soil. This is not an exhaustive list of materials and moreover some arise only when an operation is performed, for instance, bales are only produced if baling occurs. Material can refer also to cattle, feed, grass and milk.

All these materials have numerous properties such as development stage, quality, ripeness, moisture content and quantity. Most properties are specific to a material and related to its autonomous development. Some properties, however, are of interest for operations in the scheduling system and belong to each material. Each material has a quantity (for instance, mass, amount, area, number) and variables related to the state of the crop:

- processable; a crop becomes processable for operations after a specific development stage, for instance, grain becomes ripe enough to be harvested on August 5; this state is irreversible for a specific field;
- workable; a material is in the workable state, for instance, if the grain moisture content is lower than 23% and the weather is fair; the moisture content depends on the weather and varies with time, i.e. workable too changes with time;
- ready for processing; ready means processable and workable.

The state 'workable' is related to an interval of the moisture content, for instance, 0-23% m.c. of grain; such an interval may be narrowed to several processing conditions appropriate for specific operations. One condition is for example suited for harvesting dry grain (0-19% m.c.) and another is suited for harvesting wet grain (19-23% m.c.); (Table 2.3). Workable be-

Table 2.3 Relations between material attributes vs. time.

```
processable:                    FFFFFFTT..   ..TTTTTT..   ..TT

processing condition 'dry':     FFFFT..      ..FFFFTF..   ..FF
processing condition 'wet':     FFTTF..      ..FTTTFF..   ..FT
workable:                       FFTTT..      ..FTTTTF..   ..FT

ready:                          FFFFF..      ..FTTTTF..   ..FT

                             ...----------...----------...-------)  time

(*)the sequence of columns reflects possible states of the grain during time.
   (T = true, F = false, . = not considered)
```

comes false after rain and true after drying of grain in the field when moisture evaporates due to the radiation and the vapour pressure deficit. During the state 'workable' the moisture content decreases from wet (19-23% m.c.) to dry (<= 19% m.c.). If wet grain is harvested, it is known in advance that drying is necessary; the drying costs expected can influence the decision to harvest or to wait until a point in time when the expected costs are lower or even zero (dry grain). Therefore cost predicted is another property of materials essential in the scheduling system.

Exercise: Try yourself to describe the 'workable' state and processing conditions of grass for harvesting, taking in account zero grazing, silage and haymaking.

In addition to the above mentioned state variables and costs that control the possibility of an operation, other variables of a material define the so called timeliness function. Figure 2.1 shows the timeliness function as the relationship between the recoverable value (depending on quantity, quality and price) and the time when the operation is performed. An untimely operation of the material affects negatively the results and therefore must be part of the decisions in the scheduling system.

Each material may include several fields, for instance, different wheat varieties, each with its own area, ripeness date (processable) and date when the maximum recoverable value is achieved. Fields are distinguished as a component of the biological subsystem. This subsystem is shown schematically in Table 2.4.

The state variable 'processable' is an irreversible property of a specific field. Thus with some fields of different ripening dates, the variable 'ready' may change over time even if 'workable' does not change.

The above representation of material and weather as components of the

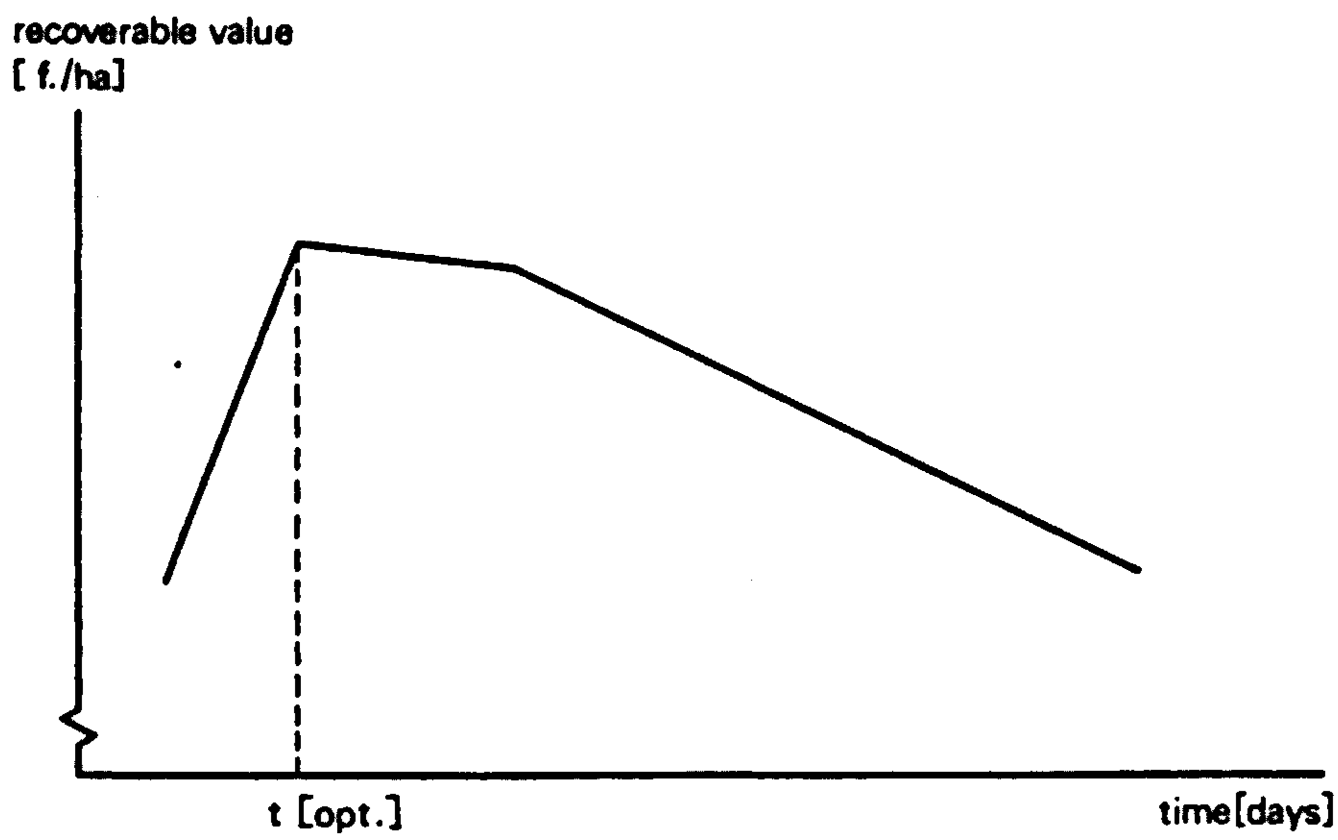


Figure 2.1 Timeliness function: relation between recoverable value and time; t(opt) = moment in time when the value achieves its maximum.

Table 2.4 Scheme of components and relations of the biological subsystem.

```
----- Materials ----- Weather
          :
          :
        Fields
```

biological subsystem shows the essential influence of materials on the possibility and desirability of performing operations and hence on the schedule of operations.

2.1.2 *Men and machinery*

The component 'men & machinery' and the components defined in this section together form a subsystem: the *man-machine subsystem.* Within 'men and machinery' there may be distinguished men (with regular worktime, overtime and no-worktime) and machines able to work or in need of service (lubricating) or repair (failure). Some men are specialised such as cowman, tractor driver or crop production specialist; so a variety of men must be possible in a scheduling system. The variety of 'machines' is even greater: a plough, a planter, a weeder, a harvester, a tractor, a trailer and also draught animals (ox, horse), tools (spade, fork) and equipment or installations (drier, barn). All these elements are resources to perform operations and may be owned or hired.

A set of these elements is needed to allow work, for instance, one man, two oxen and one plough allow ploughing. Some sets only contain one element to allow work: a man weeding by hand; a grain drier. Such a set of elements is called a *gang;* it is distinguished as a component in the man-machine subsystem. A GANG is formally defined as the men and items of machinery required to perform an operation with a specific set of materials according to a method. Thus weeding of beans and weeding of maize need their own gang because the materials processed (beans and maize) are different, although the same elements are used. Also different methods such as selecting and removing diseased plants one row a time or four rows a time need their own gangs according to the definition. Although a gang is an abstract entity it has some essential properties of its own, for instance, the number of required men and items of machinery, a standard rate of operation or capacity [ha/h] and a set-up time. Each time the gang is used set-up time is needed to refuel tractors, to prepare equipment and to drive to the field. After the work is done, some time is needed to return from the field; this period is contained in the set-up time. It is assumed that this simplification of the real situation has no effect on the performance of the scheduling system. The simplification of the reality is extended when the set-up time is contained in the standard rate of operation. The capacity (rate of operation) can be modified by actual, non standard properties of the material processed, for instance, a combine-harvester capacity may depend on the moisture content of the straw, the ripeness and development stage of the cereal and even on the shape of the field, the soil, the slope and the distance to a field.

Scheduling concerns the determination of the time when the various operations should be performed. On a farm it is impossible to work with all the possible gangs at the same time (as sometimes is possible in industry); with two men on a farm it is impossible to harvest the cereal, to bale the straw and to gather the bales simultaneously. It is not necessary to work with one gang at a time, sequentially and not parallel. For this purpose a new component is defined in the man-machine subsystem: a *combination.* The definition of COMBINATION is: a set of gangs that can work simultaneously with the available men and items of machinery. Table 2.5 shows a simple example of some combinations and the required number of men and items of machinery together with the available number. The combinations A1, A2 and A4 include one gang only. Combination A3 includes two gangs: the one man gang for combine-harvesting and the gang for baling. The farmer has to select one and only one combination from the available set of possible combinations. If Set A contains the combinations A1, A2, A3 and A4, then he is restricted to select combine-harvesting (A1 or A2), baling (A4) or both (A3). A3 requires the same number of elements as A2 and A4 together; nevertheless, A2 is used for harvesting when baling is inappropriate and A4 for baling when harvesting is prohibited. Combination B1 is the single item in Set B. The farmer can select this combination independently of set A because the gang for drying wet grain only uses the automatic grain drier and no element used in Set A. Gangs consisting of men and machines belonging to a contract worker or neighbour can be included in a separate set of combinations (C) and treated independently in the selection of work. It is convenient to use two sets in the case of Table 2.5 instead of using one set with the combinations A1, A2, A3 and A4 (without drying) and four similar combinations with drying (A5-A8).

Table 2.5 Some gangs and combinations in the wheat harvest.

Men and machinery	Available number	Required number in gang/combination				
		A1	A2	A3	A4	B1
		Combine harvesting				Drying
		2 men	1 man	1 man + Baling	Baling	
man	2	2	1	2	1	
tractor	2	1	1	2	1	
combine harvester	1	1	1	1		
baler	1			1	1	
grain trailer	2	2	2	2		
grain dryer	1					1

Table 2.6 Scheme of components and relations of the man-machine subsystem.

```
Men &        ----- Gangs -----
machinery          :
                   :
          Combinations
```

Exercise: Try yourself to determine the possible combinations when Table 2.5 is extended with a bale gathering operation (one man, one tractor, one bale loader, four bale trailers); combinations A5, A6 and A7. Try the same exercise when the contract worker uses his men and machinery (C1) for the gathering operation.

The schematic representation of the components of the man-machine subsystem is now as shown in Table 2.6.

The above representation of the man-machine subsystem shows the essential influence of men and machines on the options in performing operations and hence on the schedule.

2.1.3 Operations and decision

The components 'operations' and 'decision' form the *decision subsystem.* An operation can be viewed as the link between one gang with the required men and machines, the materials processed and the materials produced. The combine-harvesting operation links the gang with men, combine harvester, tractors and trailers to the harvested cereal and to the delivered materials (grain and straw). Some operations are used to repair a machine with a failure or to service a machine in need of lubrication; such an operation links the gang to the machine (instead of to materials).

The component 'decision' starts and ends the operation. In order to start two or more operations at the same time, it is necessary that the gangs related to those operations belong to one and only one combination or to a combination from several sets of combinations. Otherwise the decision is not allowed, because more men or machines are required than the available men and machinery to perform the operations at the same moment.

Scheduling is the allocation of operations in time and the schedule may be as in Table 2.7. At 10:00 the grain is dry enough to start the harvesting op-

Table 2.7 Schedule of the wheat harvest (operation scheduled: +).

```
operation:              ---------------------------------------> time

  combine harvesting      ++++++++++

  drying wet grain        ++++++++++++++++++++++++++++++++++++++

  gathering bales                   ++++++++++
                        10:00    15:00    20:00   00:00    clocktime
```

Table 2.8 Scheme of components and relations of the subsystems and the scheduling system.

```
        Man-machine       :   Decision    :          Biological
        subsystem         :   subsystem   :          subsystem

Men &        ----- Gangs ----- Operations ----- Materials ----- Weather
machinery          :               :               :
                   :               :               :
                   :               :             Fields
           Combinations -- Decision
```

eration; the grain is so wet that the drying operation also starts. At 15:00 the harvesting stops for some reason, for instance, rain or store filled with wet grain, and the gathering of bales is selected and starts. At 20:00 the gathering stops (for instance, all bales are gathered) and the drying operation that needs no men (automatic) continues even throughout the night. Such a schedule results positively in a recoverable yield of materials, or negatively in timeliness losses, and in costs of using men and machines. This result is a measure of performance of the schedule and hence of the component 'decision'. The latter component also depends on the method used to solve the scheduling problem and is therefore discussed in Section 2.3 'Models and solution techniques'.

After describing the scheduling problem as a system with components and relationships, the scheme shown in Table 2.8 was obtained. This scheme is general; it does not show the elements of a component (all the materials, operations, men and machines involved in a specific scheduling problem) nor does it show the type of relationship, or even the direction of information.

2.2 State, event and dynamic aspects

The components in a system have numerous properties. The relevant properties are called *state* variables and these constitute the STATE of a system at a specific time. Relevant properties or state variables of a cereal crop include development stage, yield, moisture content and in the scheduling system also processable, workable (Section 2.1.1), processing and delivery. If the state variable changes autonomously such as crop biomass (continuous variable), development stage (discrete variable), moisture content (continuous variable) and workable (discrete variable), then this is called an 'action' of a system. The discrete state variables change their value at discrete moments, for instance, workable becomes true at the moment the moisture content of the grain in the field falls below 23%. Actions are autonomous changes and do not require a decision in the scheduling system. At the moment when the state of the system changes (one or more relevant state variables change) and this change effects the scheduling system (a decision may be needed), then a system *'event'* occurs, for instance, a machine failure occurs, rain starts, the grain becomes unworkable. With these events the reaction of the system to the event is: to stop the operation; such a stop is itself again a change in the state and so an event. An immediate consequence of an event is a 'reaction' that causes another event (both events at the same moment). The 'response' of the system to an event, however, may occur later and cause another event, for instance, machine failure may cause the start of a repair operation. A decided consequence of an event is a response that causes another event. The actions (autonomous changes) and the events (autonomous and decided changes) together make up the *behaviour* of a system. If the behaviour of a system does not depend on decisions, then it is an autonomous system, otherwise it is a decision system. The scheduling system is such a decision system, although some parts, subsystems, are autonomous such as growth and moisture budget of the plant. It is sufficient to speak about 'events' in the scheduling system and to realize that some can be partly described as 'action'. Most of the systems terms are adopted from Ackoff (1971). The environment (the components and their state) is not longer distinguished from the system itself.

In the scheduling system only discrete variables are used to define a change in the state resulting in an event: Even if a continuous variable (moisture content) is the origin of the change, a discrete variable (workable) is related to it for convenience; a change in workable from true to false or reverse may cause an event. Other discrete variables have more than two values, for instance, an element of the scheduling system at any time is in use, waiting and able to be used, or not able to be used. The values of these three states of an element are 1, 2 and 3 or more general run, passive and down. The state 'down' for men occurs out of worktime, for machines after a failure and for materials when workable is false. The exact definition of the states in each component will be described later.

'State' will be used to refer to the set of all state variables and their values and will also be used in a narrow sense to refer to the summarising state variable with the values run, passive or down.

Events are used in the description of the scheduling system as a complementary way of thinking; this stresses the dynamic aspects in addition to the static aspects which are emphasized in the systems method of thinking with components and relations. An event connects different components from cause to result in one and only one correct sequence. Such a sequence has to be reflected in the computational flow of a simulation model. To verify a simulation model events may be used to check if the correct flow of computations is guaranteed (Chapter 6 'Verification and validation').

The following Sections 2.2.1-4 describe a minimum of states necessary to describe the scheduling theory for the three subsystems; more details will be described in Chapter 4 'Base model'.

2.2.1 *Materials and weather*

To discover states and events in materials (crops, soil, products), it is necessary to look at the history of a material and at its environment: the weather. The example is taken from the cereal harvesting. Wheat in the field becomes available in some fields at the beginning of the season. Each field has its own ripening date and a date when the wheat becomes processable. The weather (rain, radiation) influences the moisture content of the grain and of the straw and therefore the workability. The state (narrow sense) of wheat is now defined as:

- run or passive: if the wheat is available and workable and ready (i.e. the first field processable and workable);
- down: otherwise.

The variable available is true as soon as some area of the crop or product arises in the system (initially or from operations); it becomes false as soon as the crop or product is processed. The variable processable becomes true for wheat in a field for instance, two days before the ripeness date; for the wet grain, the straw, the bales and the stubble processable becomes true at the day of delivery. Workable, however, depends on the weather that influences the moisture content of grain, straw, bales and soil. Workability can depend not only on moisture content, it may depend also on rain (no rain for wheat and straw; < 1 mm/h for bales and stubble) or on properties of a field (moisture content, development stage). All these changes in the variables processable, workable and ready modify the state of the material and cause an event.

Exercise: Try to find examples of factors where workability is not only a property of the material (hay in all fields), but depends also on specific properties of hay in a field. Think of conditions for gathering the hay (moisture, quality, development).

Processing of materials causes other events. Combining wheat results in the delivery of (wet) grain and straw; the latter materials become available if they were not already available. The harvesting of grain results at some point in the completion of the harvesting in that field and thus 'processable' or 'workable' of the next field may be false. Completion of the harvesting of all wheat fields results in 'available' of wheat becoming false. These events arising from the material affect the state variables of the material; a change to down or from down may mean that a new decision is required, because the state of the system is changed in such a way that the scheduled operations can or must be revised. The state of the material may now be redefined as:

- run: an operation scheduled to process the material; material available and workable and ready;
- passive: no operation scheduled; material available and workable and ready;

– down: material not available or not workable or not ready.

The change of the state to run or from run depends on the decision whether an operation that processes the material is scheduled or not; these events are caused by a decision and do not require a subsequent decision. To summarise: a change in available, workable or ready changes the state from or to down and may require a decision; other changes in the state (from or to run) are originated by the decision to schedule operations.

Exercise: The change of the state from run to down is one step (can you identify some causes?) and from down to run is possible only via passive (why?; can you distinguish an autonomous part and a decision part?). Because of other preferences in 'decision', the state 'passive' may be continued; do you think that then the change from passive to down requires a decision?

If a material is delivered, for instance, wet grain, then a maximum quantity can be achieved in store or transport. The event that delivery is no longer allowed requires a decision. The maximum quantity is fixed when storage is limited or a variable one when, for instance, the area of bales in the field depends on the predicted weather forecast or on the clocktime to prevent exposure of bales to rain at night. The scheme shown in Table 2.9 shows a possible history of a material with causes and changes of variables. The origin of events is shown in Table 2.10.

Table 2.9 A history of a material with causes and state variables.

Available	Processable	Workable	Ready	Processing	State	Cause of change
false					down	
true	false		false		down	material delivered
	true	false	false		down	field processable
		true	true		passive	material workable
				true	run	operation scheduled
		false	false	false	down	material unworkable
		true	true		passive	again workable
				true	run	operation scheduled
	false		false	false	down	field completed and next not processable
	true		true		passive	field processable
				true	run	operation scheduled
false				false	down	material completed

(*) blank value of variable means the value at the preceding line.

Table 2.10 The transformation of states of material due to events.

Current state	Resulting state of material: run	passive	down
- run		decision	material+decision
- passive	decision		material
- down	x	material	

x= impossible transformation.
Where material is mentioned the transformation is autonomous (action); the weather or a next field may be the very origin.

The above description of states and events in materials shows the influence on operations and decision and hence on the schedule.

2.2.2 *Men and machinery*

The history of men and machines in a scheduling system is considered to discover states and events. With men and machines a variable 'available' can be distinguished. The availability of men and machines may be limited to certain periods or seasons in a year. Seasonal labour, contract work and special equipment such as harvesters are examples of this. The availability of men is restricted also by the usual worktime and overtime with a weekly pattern; a wage allowance during overtime results in extra costs.

Breakdown or failure of an item of equipment (machine, tractor) may occur during work, so 'repair needed' is another relevant variable in the state of an element. The variable 'service needed' occurs when lubrication is necessary for, for instance, a combine-harvester after more than eight hours use. After repair or servicing the variable 'repair needed' or 'service needed' of such an item becomes false.

The state (narrow sense) of a man or an item of equipment can now be defined as:

- run: in use by a gang; available, no repair needed and no service needed;
- passive: not used; available, no repair needed and no service needed;
- down: not available or repair needed or service needed.

A change in the states from or to down causes an event in the system originated by the element (man or machine) itself. The change from or to run is caused by a decision and does not require a subsequent decision. The origin of events is shown in Table 2.11.

The scheme shown in Table 2.12 shows a possible history of a man with causes and changes of variables. The scheme shown in Table 2.13 shows a possible history of a machine with causes and changes of variables.

The remaining components in the man-machine subsystem are gangs and combinations. Both consist of a number of men and items of equipment; it suffices to describe the state of gangs only. A gang can be available in the

Table 2.11 The transformation of states of men or machines due to events.

Current state	Resulting state of man or machine: run	passive	down
- run		decision	man or machine+decision
- passive	decision		man or machine
- down	x	man or machine	

x= impossible transformation.
'Man or machine' indicates an autonomous transformation; start or end of worktime and failure or repair may be origins.

Table 2.12 A history of a man with causes and state variables.

Available	State	Cause of change
false	down	
true	passive	worktime begins and man is available
	run	man is used in a gang
	passive	man not used in any gang
	run	used again
false	down	worktime ends

(blank value of variable means the value at the preceding line)

Table 2.13 A history of a machine with causes and state variables.

Available	Repair needed	Service needed	State	Cause of change
false			down	
true	false	false	passive	machine becomes available in a period or week
			run	machine is used in a gang
	true		down	breakdown occurred, machine failure but no service needed
	false		passive	machine is repaired
			run	machine is used
			passive	machine is not used in a gang
		true	down	machine stopped work and needs service
		false	passive	machine service completed

(blank value of variable means the value at the preceding line)

Table 2.14 The transformation of states of gangs or combinations due to events.

Current state	Resulting state of a gang/combination: run	passive	down
run		decision	gang + decision
passive	decision		gang
down	x	gang	

x= impossible transformation.
'Gang' indicates a transformation that may be originated by men or machines.

same way as men and machines i.e. a periodical and a weekly pattern. In addition to the availability of the gang as such, the availability of sufficient items of the required men and machines is distinguished (for combinations the availability of each gang is involved). Thus the state of a gang or combination is defined as:

- run: in use; available and number of men and machines for use ≥ the required number;
- passive: not used; available and number of men and machines for use ≥ the required number;
- down: not available or number for use less than required number.

An item for use must itself not be in the state 'down'! The change of state from or to down is caused by the gang or one of the elements of men and machines required. The change from or to run, however, is caused by a decision. Table 2.14 shows the origin of such an event for transformations from the current into a resulting state.

The above description of states and events in the man-machine subsystem shows the influence on operations and decisions and hence on the schedule.

2.2.3 *Operations and decision*

The history of a component in the scheduling system is considered to discover states and events related to operations. Two other components are involved: a related gang and the materials processed by the operation. In Section 2.1.1 'Materials and weather' the processing conditions of a material were described, for instance, an interval of grain moisture contents 0-19% (dry), 19-23% (wet). Within an operation reference is made to one or some specific processing conditions of each material processed. It may be desired that wet grain is harvested with the one-man combine-harvesting operation only or in other words the two-men combine-harvesting operation is restricted to the processing condition 'dry'. The materials produced by the operation are not considered in the state of an operation, although, for instance, a maximum quantity of a material produced is involved in the decision whether or not to operate. Because an operation is directly related to one and only one gang, it is not necessary to consider the availability of an operation, for it is already reflected completely in the gang. In every day language operations are the cause of a failure or of completion of harvesting, but in scheduling it is made more specific and a failure is related to a machine, no worktime to men and terminating a field to a material; of course some causes only occur when an operation is involved. Thus the state of an operation is now directly defined as:

- run: in use; related gang not down, processed materials not down and at least one appropriate processing condition of each material processed;
- passive: not used; related gang not down, processed materials not down and at least one appropriate processing condition of each material processed;
- down: gang is down or one of the processed materials is down or has inappropriate processing conditions.

The changes from or to down are originated by the gang or by the materials processed (change in state or processing condition); a decision is involved with the changes from or to run. Table 2.15 shows the origin of such an event for transformations from the current state into a resulting state.

The above description is related to operations which process materials. A similar definition of states is possible for operations repairing or servicing a machine; the definition is:

Table 2.15 The transformation of states of operations due to events.

Current state	Resulting state of an operation: run	passive	down
run		decision	gang or material + decision
passive	decision		gang or material
down	x	gang or material	

x= impossible transformation.
Gang or material have some autonomous effects on operations.

- run: in use; related gang not down, machine down and in need of repair or service;
- passive: not used; related gang not down, machine down and in need of repair or service;
- down: gang is down or no machine in need of repair or service.

The gang for repairing consists of men and tools, and the operation contains a queue of machines in need of repair. Those machines are 'processed' to allow them to work again.

It can be clearly seen that an operation is the entity where the man-machine subsystem meets the biological subsystem (or a machine in the case of service and repair operation) and relevant properties of both sides are incorporated in its own state. The history is looked at more closely and a new variable called a 'phase' is added to the state of an operation. The phase is said to be inactive when the state is passive or down. When the state is run, a number of phases are distinguished. The first phase when the operation starts is 'setup' and lasts for the set-up time of the gang (may be zero). The next phase is 'wait' until all the materials processed are available, for instance, if the grain drying operation starts at the same moment as the combine-harvesting operation, then the drying operation has to wait until the harvesting operation has completed the set-up time, arrived at the wheat field and actually harvested the wheat and delivered the grain to the drier. Once all the materials processed are available, the phase becomes 'busy'. At the moment when the operation stops, as prescribed by a decision, the phase becomes 'inactive'. At the same time the state changes from run to passive or down.

Exercise: How does one interpret a sequence of phases in the operation of gathering bales: inactive, busy, wait, busy, if the area of bales is small and the baling operation and the gathering operation are started at the same time? What happens to the area of bales if the gathering is faster?

The scheme shown in Table 2.16 shows a possible history of an operation with causes and changes of variables.

A phase 'setup' is not considered in service-repair operations, because the set-up duration is contained in the time period needed to repair or service a

Table 2.16 A history of an operation with causes and state variables.

State	Phase	Cause of change
down	inactive	
passive		gang becomes passive, material processed becomes passive or a processing condition of material processed becomes appropriate
run	setup	a decision starts the operation and the duration to set-up the gang begins
	busy	after the completion of the set-up the men and machines actually start work at a field
	wait	this operation completed the processing of the material and another operation scheduled will deliver more material after its set-up duration
	busy	this operation continues the processing of material
passive	inactive	a decision stops the operation for some reason
down		the operation completed the processing of the material or processing conditions become inappropriate or gang becomes down

(blank value of variable means the value at the preceding line)

machine; a phase 'wait' is not applicable either. Why?

So far no event caused by an operation has occurred; this agrees with the preference to be specific in relating changes in the scheduling system to physical objects. However, some events can be listed that are indirectly caused by the fact that an operation is in state 'run':

- field is finished and next field is not available or not yet processable or workable;
- the maximum quantity of a material is achieved;
- a material is delivered and becomes available for processing;
- a machine failure occurs;
- a machine is repaired or service completed (in a service-repair operation).

The other component of the decision subsystem is decision. This component is required in each event; it changes the state of operations, gangs, materials (processed and delivered) and so indirectly may cause another event. When a decision is required, the component knows all the gangs and combinations related to operations that can perform work (i.e. are not down). The man-machine subsystem showed that at most one combination (or gang) should be selected from each set of combinations (Section 2.1.2 'Men and machinery'). The selection is done by assigning urgencies to operations in some way (Elderen, 1977) and by preferring the combination with the maximum urgency. The operations that are scheduled until this point (and one may be involved in the cause that required a decision) can be stopped and the operations just selected are started. The selected operations remain scheduled until the next moment a decision is required. In this way a schedule (Section 2.1.3 'Operations and decision') is made as a sequence of intervals with some operations or without operations along a time axis. The allocation of operations in time, or scheduling, may result in the schedule with causes of events shown in Table 2.17. At 10:00 the grain is dry

Table 2.17 Schedule of operations in the wheat harvest and causes of events

```
                      : worktime of men               : no worktime
Causes:                    : moisture content of grain < 23%
                                 : drier storage filled
                                           : bales are gathered
               time  ------------------------------------------------->
Operations:

  combine harvesting        +++++++++

  drying wet grain          ++++++++++++++++++++++++++++++++++++++

  gathering bales                     +++++++++

                            :         :         :         :
              clocktime   10:00     15:00     20:00     00:00
```

enough to start the harvesting operation; the grain is so wet that the drying operation also starts. At 15:00 the grain drier store is full and harvesting stops. The bale gathering operation is selected and continues until 20:00 when the bales are gathered. The automatic drying operation continues even after the worktime ends. All the different events from men, machines and materials create appropriate states in the decision component such as machine failure, material passive, material down. Within the decision these states are checked to react accordingly. Details are described in Chapter 5 'Experimental frame'.

The above description of states in the decision subsystem and of related events shows how operations and decision act and determine the schedule.

2.2.4 *Miscellaneous*

Some events in the scheduling model are additional, such as the creation of output required by the user. A separate component administration is distinguished that derives the desired information from men, machines, gangs, combinations, operations and materials and composes the output on each day, week, period or season. For this purpose updating of cumulative variables in each component is necessary, for instance, the duration the component was in the state run, passive or down.

Other additional events are not new but withdrawn from men, gangs or materials and concentrated in new components. The weekly pattern of men is such a new component with its own event that affects all the men behaving according to that pattern; this component replaces common parts of each man and is a convenient method of handling identical events in similar components. For the same reason the periodical pattern of gangs and materials is a new component with events controlling the availability in a sequence of periods (seasons, months, weeks). Even the regular pattern of calculation of urgencies can be concentrated in a separate component; the related parts are withdrawn from materials. The system can handle several objects of the same component. Just as it is possible to have some men and items of equipment, for instance, two or more weekly patterns can be created. Of course a specific man has only one pattern and belongs to one of the specified patterns.

2.3 Models and solution techniques

In this section the relationship between a model of the scheduling system and some solution techniques as linear programming (LP), dynamic programming (DP) and simulation is discussed. The techniques differ in the way they derive a solution, i.e. a schedule and costs, and not every aspect of the model can be represented correctly in each technique. Some technique may be more useful than another in certain situations. What are the advantages and disadvantages?

So far in Sections 2.1 'Components and relations' and 2.2 'State, event and dynamic aspects' only one model of the scheduling system has been developed. If irrelevant elements are removed, it is still the same model with the same behaviour. However, more models can be created by introducing simplifications such as:

- remove the set-up duration of a gang and incorporate it in a decreased rate of operation;
- remove the need for service and repair of a machine and incorporate it in the rate of operation of the gangs;
- collect the hourly workability of materials to workability durations within a day and remove the original chronological sequence partly, for instance, two hours (10:00 – 12:00) workable for all the work and two hours (16:00 – 18:00) workable for all the work is in the aggregated form four hours successively on that day;
- relax the workability by giving up the simultaneousness of the workabilility of cereal and of straw and restrict the use of the harvester with the workable duration of the cereal separately from the use of the baler with the workable duration of the straw.

Some simplifications are useful to limit the amount of input data. The relationship between model, simplifications and solution techniques is itself an interesting research topic; very little is known of the quantitative effects on the solutions. Nevertheless some important relations between model and solution technique are discussed.

For each solution technique some advantages and disadvantages are summarized.

Linear programming (LP) is a useful technique with standard formulation (linear equations in matrix form) and standard solution algorithm (e.g. Simplex algorithm). The solution is optimum and derived in one step for the entire model, i.e. all the information from the beginning until the end of a season is available at the start of the iterative algorithm. In the real life situation the farmer does not know the workability at the end of a harvesting season at the beginnning of the season; so the result although optimum for the model, will be too optimistic. Such a matrix without probabilities of workability can be replaced by a so-called chance constrained matrix. It is assumed that the set-up of an operation, the repair and the service are taken into account in the rate of operation. The aggregation of workability to daily intervals or even monthly intervals is possible at the expense of the loss of the sequence of the operations within the interval.

Dynamic programming (DP) has a standard solution pattern (stepwise backward through stages) but no standard formulation or algorithm. The solution is optimum (called a strategy) and derived in many steps for the entire model (network). The probabilities of workability in the next stages can be given as in practice. Aggregation of workability to daily intervals is possible and the sequence of operations within the interval is maintained i.e. the flow of materials is correct, for instance, the straw is not baled before it is delivered by the harvesting operation. An interval between two stages has only one workability without any intermediate changes: the interval is uniform.

Simulation has no standard solution nor a standard formulation or an algorithm. To solve a model with simulation an algorithm must be defined for scheduling operations. Such a heuristic strategy (term from DP, or tactical rule according to planning terminology) is described by Elderen [1977]; it derives urgencies of operations from timeliness functions of crops. Because the strategy selects a path in the network in a myopic (short sighted) way, it cannot achieve the optimum solution. The representation of probabilities of workability is given as in practice by expected durations for the next period. The aggregation of workability to daily intervals is possible and the sequence of operations within the interval is maintained i.e. the flow of materials is correct. In a simulation model a lot of detail can be added (set-up of operation, failure of machine, service, rate of operation related to moisture contents, etc.) with a limited increase in the size of the model.

Practical experience so far is too limited to demonstrate the above advantages and disadvantages of techniques. It is impossible in this publication to fully investigate the differences between the techniques for solving a scheduling model. Nevertheless the scheme is shown in Table 2.18 showing differences and similarities.

The above review shows that for aggregated problems DP creates a realistic, optimum solution of the schedule and is an excellent, objective measure for other techniques. The LP method results in a too optimistic solution and simulation in a sub-optimum solution (Elderen, 1980). Until now no quantative insight was available into the differences with DP. To find the relationship between simplifications, models and techniques, it is necessary to develop useful research tools and practical tools for scheduling problems and farm planning.

So far labour budgeting has not been mentioned. Labour budgeting has

Table 2.18 Some characteristics of scheduling techniques.

	Linear programming	Dynamic programming	Simulation
Solution:			
- procedure	simplex	backward steps	heuristic strategy
- number of steps	one	one at each stage	one at each stage
- optimality	optimum (perfect inform.)	optimum	sub-optimum (myopic search)
Structure of model:			
- workability	duration	chronological sequence	chronological sequence
- aggregated workability	loss of correct flow	arbitrary sequence and uniform interval	arbitrary sequence
Advantages:			
- solution	standard algorithm	complete search	
- model	inequalities	network	network
Disadvantages:			
- solution	too optimistic		sub-optimum
- size of model	limited	limited	
- model of acceptable size	aggregated	aggregated	

aspects similar to simulation with a simple strategy (priorities of operations) and a simplified structure of the model (aggregation and use of man-hours only).

This publication on a simulation model of the scheduling problem describes a research tool which can obtain quantitative data with different models (different levels of simplification). It is a partial answer designed to help obtain quantitative differences in solutions with several techniques and models. These data are needed to select appropriate techniques and models to advise farmers.

3 A SIMULATION TOOL: SIMULA

Three approaches can be used to describe a simulation system (Pidd, 1984). The first one describes the system as the behaviour of each component in the system, of each man, each machine, operation or material; this is called the process-description. The second approach is the activity-description, that describes the circumstances and conditions when it happens and the consequences for each activity (for instance, add men for work at the beginning of worktime). The third approach is the event-description, that describes a change in the system and the consequences under different conditions, for instance, the arrival of men for work happens and the consequences for the gangs and operations are listed. The second and third approach are very similar and both concentrate on changes in the system occurring at a moment in time. The first approach, however, concentrates on the components and their history, their process. This description is closer to the description adopted in system methodology and will be used here. Only some languages are able to describe a system as processes of components. From the available means (GPSS and SIMULA) SIMULA is used , because it is a general purpose language with some discrete dynamic simulation facilities and a language that allows a structured programming approach as in Algol and Pascal. This chapter is not intended to describe the SIMULA language completely. It is assumed that the use of blocks enclosed in a pair of 'begin', 'end', the use of indentation of statements within a block, the declaration of local variables in a block and of procedures is well known (Birtwistle et al., 1973). Those aspects that are typical of SIMULA and useful in simulations are described; one section (3.1 'Components') describes the method distinguishing components and elements, another section (3.2 'Simulation of processes') describes the facilities for simulation and finally (3.3 'Utility programs') some utility programs are mentioned. An excellent and extensive description of these aspects is given by Franta (1977); he explains the process view of simulation by the use of SIMULA. SIMULA has no special facilities to simulate continuous processes; the simulation of continuous parts within a discrete system is possible by developing appropriate procedures or by using external procedures already developed in other languages. SIMULA is available on many main-frame computers and some minicomputers; SIMULA for microcomputers is not yet available but developed partly.

3.1 Components

System methodology distinguishes components and relations between components; a component consists of similar items or elements. The component men and machinery of the scheduling system for example consists of two sub-components men and machines, respectively that each covers some men and several different machines. SIMULA uses the 'class' concept to define a component or a sub-component. One particular item is called an object of the class. For example the information in Table 3.1. was coded in the scheduling model. The men and machinery component is then declared by 'class LABR-EQPMNT' and the two sub-components for men and machines, respectively by 'LABOUR' and 'EQUIPMENT'. Each class or sub-class can have parameters, variables, procedures and statements. The parameters, the variables and the procedures are called the attributes of the class or subclass; so values and instructions (call of functions or procedures) may be attributes. The prefix LABR-EQPMNT before class LABOUR and class EQUIPMENT means that LABOUR and EQUIPMENT are subclasses of LABR-EQPMNT. The latter declares the attributes common to men and to machinery, for instance, time used. The specific attributes of men such as wages belong to subclass LABOUR and those specific for machinery such as storage capacity, power, working width are declared in subclass EQUIPMENT. EQUIPMENT could be subdivided into subclasses for animal power, tractors, self-propelled machines, trailers, installations each with its own attributes. A man is a specific object of class LABOUR and has all the attributes of LABOUR and (by means of the prefix) also of LABR-EQPMNT.

How can one distinguish in the model between men and between different machines? So-called reference variables can be defined; a reference varia-

Table 3.1 Example of code in the scheduling model.

```
class LABR_EQPMNT (formal parameter);
declaration of formal parameters
begin
     declaration of local variables
     declaration of local procedures
     statements
end;

LABR_EQPMNT class LABOUR;
begin
     declaration of local variables
     declaration of local procedures
     statements
end;

LABR_EQPMNT class EQUIPMENT;
begin
     declaration of local variables
     declaration of local procedures
     statements
end;
```

ble MAN[i], for instance, refers to an object of LABOUR and a reference variable MACH[i] refers to an object of EQUIPMENT or refers to no object (called 'none') at all. An object can be created dynamically in the model at appropriate moments in the program by incorporating a statement:

MACH[i]:- new EQUIPMENT (actual parameters);

where ':-' is pronounced as 'denotes' and means that MACH[i] refers to this specific object of subclass EQUIPMENT. Elsewhere in the program outside the class EQUIPMENT (and LABR-EQPMNT) the attributes of this specific object can now be used in a statement by using:

'ref. variable', 'dot', 'attribute name' for instance,
MACH[1] . storage capacity:= 1.5

i.e. machine one's storage capacity becomes 1.5.

In general when a procedure is called or a block of statements is to be executed, then core is dynamically added to the program and after passing the last 'end' the copy in core is lost. In contrast to this: even when all the statements of an object are executed and the last 'end' is passed, the object itself and its attributes remain as long as a reference to the object exists. This creates the option to use an object as a data-block and to refer to each attribute (parameter, variable or procedure) from outside.

Enough is now known about the use of class as a way of defining components and the use of a reference variable to refer to specific objects. The following additional information can be given without explanation:

- a procedure can be declared 'virtual' in a class; if defined in that class it can be redefined in subclasses; the procedure definition at the lowest subclass level will be used; for instance, when the class 'OPERATION' has two subclasses one to operate on materials and one for service and repair of machines a procedure 'START-OPRTN-' can be defined in each subclass with statements specific for each type of operation and still refer to these procedures as an attribute of the superclass 'OPERATION' without knowing which specific type of operation is involved; a convention is adopted to end the names of such virtual procedures with '-', an underscore; note that the underscore in text is printed as '-' and in the tables as an underscore;
- a reference L-E to an object of class LABR-EQPMNT can only refer to attributes of that class; if one wants to refer to attributes of the subclass, a distinction can be made between subclasses by using:

```
inspect L-E when LABOUR do .....
            when EQUIPMENT do .....
            otherwise .....;
```

or when it is known that L-E refers to a machine the following is used:

```
L-E qua EQUIPMENT . attribute;
```

attributes of superclasses can be called directly in a subclass in general;
- a comparison of reference variables uses the symbols '==' and '=/=' to distinguish the fact, that two reference variables may denote the same or different objects, respectively, for instance, if MAN[1] =/= none then...;
- separated compilation of parts of a program is possible and such parts can be used by other parts by using the external declaration and a prefix, for instance:

```
external class SFOBASE-MODEL;
...
SFOBASE-MODEL class SFOEXPERIMENT;
```

if the class SFOBASE-MODEL is already compiled separately.
The options of 'class' are extended in the following section by incorporating dynamic aspects of processes.

3.2 Simulation of processes

Simulation of a system and modelling it according to the process view requires in the language extra facilities. These facilities are available in SIMULA by a system defined class 'simulation' and are fully available by using that class as a prefix of the class 'scheduling farm operations base model':

```
Simulation class SFOBASE-MODEL;
begin
    declaration of global variables
    declaration of global procedures
    declaration of classes
    statements
end;
```

There are three groups of facilities available. The first group concerns the time axis. It consists of the variable 'Time' and a sequencing set. That set has a sequence of event notices; each event notice contains a reference to the scheduled object, the moment that the event will happen, the preceding and the following objects scheduled in the set of events. A new object placed into the sequencing set checks the moments and is positioned accordingly.

The second group offers the queueing facilities. The queue is an object of class 'Head' and a reference to it is declared by:

```
ref (Head) QUEUE1, ...;
```

where attributes are available such as 'First', 'Last' object in the queue. An object that is placed in a queue must be declared with prefix 'link' as:

```
Link class ABC;
```

where 'Link' adds to the class ABC attributes like 'Suc', 'Pred' (to know the successor and the predecessor), 'Out', 'Into' and 'Follow (x)' to move the object out of the queue, into the queue at the end or after object x. With these facilities a queue of men available for work or a queue of fields belonging to a crop can be created.

The third group of facilities concerns the method of handling classes of objects as processes. A class with prefix 'Process' is declared as:

```
Process class LABR-EQPMNT;
```

and now this component of man and machinery is 'living'; each object has its own history and life even in the simulation model. How is that achieved? Usually when a block of statements, a procedure or the statements of a class object are executed, then the statements are handled sequentially (except for choices with 'if ... then ... else ...' and repetitions with 'while ... do ...' or 'for ... do ...') until the very end. The computer can only work sequentially which means that parallel simultaneous processes have to be handled quasi-parallel by inserting activation and deactivation statements in a process to interrupt the sequential execution of statements in the program. The following activation and deactivation procedures are available in superclass 'Process' and can be used in a statement:

```
Activate REF-PROCESS at T;
Reactivate REF-PROCESS at T;
Hold (T1);
Passivate;
Wait (Q1);
Cancel (REF-PROCESS);
```

where REF-PROCESS is a reference to a process, T a variable related to 'Time' and the time axis, T1 a duration and Q1 a queue reference. The (re)activation statement also has other forms to delay the execution for a time period or to place the object at the time-axis after another object scheduled on the time-axis. To stop the execution of the active object at the current 'Time', Passivate, Wait or Cancel is used and the execution continues with the statements of the next, i.e. the first object on the time axis; the former object was 'Current' (a system reference to the active object) and is now removed from the time axis; the system knows where the execution of statements is interrupted for each process object and may be continued later on. By using Hold, the object is replaced on the time axis at a moment T1 from now (now is the current moment in the simulation, known as 'Time'), the execution of statements is interrupted and continued with the next object on the time axis.

Another object can be scheduled within an object by using Activate (if not yet scheduled) or Reactivate with a reference to that other object. If an object is scheduled, it can be removed from the time axis by using Cancel with a reference to the object. Hold, Passivate and Wait remove the 'Current' object from the time-axis. With the use of a reference in Activate, Reactivate and Cancel the object involved is known exactly. The use of Hold, Passivate and Wait, however, is restricted implicitly to the active object referenced by 'Current'; to prevent misunderstanding a convention is adopted: use these procedures only in the statements of a process and not in procedures, because procedures can be called from outside the object.

Exercise: If Passivate was used in a procedure of the class LABOUR and that procedure is called from the current object OPERATION, can you tell which object is removed from the time-axis? Is that the same object where Passivate is mentioned?

An example from wheat harvesting is as follows. The wheat-harvesting operation is scheduled at 8:00, i.e. at 8:00 on the time axis the operation combine-harvesting is 'current'. From that moment, the harvester is being used and a failure may occur; for this reason, the harvester is scheduled on the time axis, for instance, at 10:35. At the moment the operation starts (8:00), the wheat is scheduled at a future moment when the field will be harvested, say at 15:17. At 10:35, the harvester becomes the 'current' object; because of the failure, the harvesting operation is scheduled at 10:35 to finish the operation; the decision process is also scheduled to find the subsequent operation, for instance, the repair of the harvester. The wheat harvesting stops at 10:35 and the event expected at 15:17 will not occur, so this object is cancelled from the time axis.

Insight is now necessary into the possible states of a process object and how these states are achieved. Franta (1977) distinguishes four states:

- active: execution of statements (processor attention); the object is the first on the time axis and referenced as 'Current';
- suspended: an event-notice for the object is on the time axis;
- passivated: no event-notice at any time, but activation can still happen; the object is considered idle, not on the time axis;

– terminated: no event-notice, activation impossible; the object exhausted its action statements and continues to exist as data object as long as a reference to it exists.

The transfer from an existing state of a process object into a following state is achieved by the activation and deactivation statements according to the scheme shown in Table 3.2.

To distinguish between these states 'Process' has some variables:

– Idle: true if not on the time axis (passivated, terminated);
– Terminated: true if statements are exhausted (terminated);
– Evtime: the event time is only given when the object is on the time axis (active, suspended) and equals 'Time' when the object is 'Current'.

To illustrate the use the code is discussed which is shown in Table 3.3 concerning the operations, materials and fields in the base model of the scheduling of farm operations. Within the declaration of a class several sections can be distinguished. The declaration section declares variables and procedures, the initial section reads initial values of variables from input data and creates the initial state of an object, the dynamic section is used as long as the variable END-EXPRMNT, end of experiment, is false ('—' or 'not'; not false means 'true'; so while true do ...) and afterwards the terminal section is executed. In class OPERATION a procedure OPERATE is declared that activates the dynamic section at moment 'Time' and 'prior' to 'Current', this means immediately. The dynamic section shows the use of Hold and Passivate and describes as comment (between ! and ;) where it will be activated at a future moment.

The declaration of class MATERIAL shows the procedure PROCESS-MAT where the procedure OPERATE of an operation (referenced by OPR) is called. In the dynamic section there is the call of PROCESS-MAT and the reactivation of this MATERIAL object at a future moment or Passivate. So if MATERIAL is scheduled and becomes active then PROCESS-MAT is called, that on its turn calls OPR.OPERATE, which activates the

Table 3.2 Transformation of states of a 'Process' object and (de)activation procedures.

Existing state	Following states: active	suspended	passivated	terminated
active	-	Hold Reactivate	Passivate Wait Cancel	B
suspended	A	Reactivate	Cancel	-
passivated	Activate Reactivate	Activate Reactivate	-	-
terminated	-	-	-	-

'A' occurs automaticly by stopping the execution of statements of the preceding active object;
'B' occurs when an active object passes its final end.

Table 3.3 Example of code of operations, materials and fields in the scheduling model.

```
Simulation class SFOBASE_MODEL;
begin
  ...
    Process class OPERATION;
    begin
        ! d e c l a r a t i o n ;
        ...
        procedure OPERATE;
        begin
            ...
            Activate this OPERATION at Time prior;
        end;
        ...
        ! i n i t i a l ;
        ...
        ! d y n a m i c ;
        while not END_EXPRMNT do begin
            ...
            Hold (set-up duration);
            ...
            Passivate;    ! activated by OPERATE;
            ...
            Passivate;    ! activated by START_OPRTN_;
        end;
        ! t e r m i n a l ;
        ...
    end;

    Process class MATERIAL;
    begin
        ! d e c l a r a t i o n ;
        ...
        procedure PROCESS_MAT;
        begin
            ...
            OPR.OPERATE;
            ...
        end;
        ...
        ! i n i t i a l ;
        ...
        ! d y n a m i c ;
        while not END_EXPRMNT do begin
            ...
            PROCESS_MAT;
            ...
            if ... then Reactivate this MATERIAL at ...
            else Passivate;    ! reactivated in ...;
        end;
        ! t e r m i n a l ;
        ...
    end;

    Link class FIELD;
    begin
        ...
    end;
  ...
end;
```

operation. A material is scheduled at the moment when for instance, a field is processed completely or all fields are processed and at such a moment it is appropriate to require from operation to update the state of the materials involved i.e. to consume the quantity of the processed material and to deliver it to other materials (to harvest wheat and to produce grain and straw).

In this way the execution of statements in an operation is controlled by the events occurring to the materials involved.

The declaration of FIELD shows that it is not declared as a living process, but only as an object that can be queued; in this case in a queue belonging to a material.

The example shown does not show the use of activation and deactivation statements in the initial or terminal sections.

The above description of the incorporation of simulation facilities in a SIMULA program will be extended in the following chapters, which describe the base model and the experimental frame.

3.3 Utility programs

SIMULA has several utility programs; some programs are not available on all computers or are available under other names than used within DEC10-SIMULA.

The programs that convert a source program are:

- SIMED, it edits the source program; it indents according to the begin ... end level; it uses capital letters or small letters for system defined words, standard identifiers, own identifiers and comment; in this publication capital letters are preferred for own identifiers and small letters for system defined words, standard identifiers (first letter in capital) and comment;
- SIMIBM, translates a DEC-10 source program to IBM SIMULA.

The program used for conversational input and output of data is SAFEIO; it will be used to prepare or to revise the input files of the scheduling system.

SIMDBM is a Codasyl data base management system, that is not yet used in this scheduling model.

When running an experiment with the simulation model SIMDDT, a debugging system, can be used. When the program detects a run time error, for instance, a reference variable denotes to ‘none’ instead of to an object such as OPR, then SIMDDT is entered and it is possible to see which objects are scheduled, what is the chain of procedures called from the ‘Current’ object, what is the value of variables of each object. The program allows: change of the value of variables, definition of messages at particular lines of the program when executed; the message can contain the value of variables and can be wanted each time the line is executed or only at a certain condition. This program is extremely useful to detect the flow of program execution and of mistakes.

Many utility programs concern the input/output (to prevent the use of the basic primaries to read or write an integer, a real number, a Boolean, a text) and the use of histograms and statistical data such as mean value and standard deviation. All these programs are mentioned before the program declarations by:

```
external integer procedure IMIN, IMAX;
external procedure READ, WRITE, SIGMEAN, HISTP;
```

and can be used in the statements, for instance, C:= IMIN (A, B).

The above sections are sufficient to present some idea of the options of SIMULA as a general purpose language with good simulation facilities. The information descibed is incomplete. For practical use textbooks and, for instance, DECSYSTEM-10 SIMULA Language Handbooks are indispensable. It should be possible, however, to understand the use of the language in the following chapters.

4 BASE MODEL

In Chapter 2 'Theory of scheduling' a general description is given of the scheduling system on a farm with all its components and in Chapter 3 'A simulation tool: SIMULA' the computer language SIMULA is illustrated together with simulation facilities. This chapter integrates the theory and the language and describes the program. The three subsystems (materials, 4.1; men and machinery, 4.2; operations,4.3) and some additional facilities (4.4) are described in general terms; details can be studied by carefully considering the variables, the initialization and the procedures within the classes.

The outline of the base model program is shown in Table 4.1. Classes and references to these classes are shown together with special classes used to place a record, a box in a queue (and so 'link' as prefix), where the record refers to some object. The subclasses are indented to show them clearly. The classes and subclasses prefixed directly or indirectly by Process are entities with a history and are the components of the system as described in Section 2.1 'Components and relations'; they belong to the man-machine subsystem, the biological subsystem, the decision subsystem or are auxiliary components for the system. Other classes are only facilities to create data objects for administrative purposes (COMP-ADMINISTRATION, MATRL-ADMN) or to queue objects (Link class .....). In the base model only those attributes of classes are programmed which do not depend on particular operations or crops (wheat, corn, potatoes), or experimental conditions (weather data). The specific attributes are described in Chapter 5 'Experimental frame' together with extensions of the components DECISION, WEATHER-MATRL and ADMINISTRATOR.

The SFOBASE-MODEL (base model of Scheduling Farm Operations) itself is a class that is compiled seperately and used as a prefix in the experimental frame program (a seperate program). Before the details of the components are described in the next sections a class COMPONENT and the general part of class SFOBASE-MODEL are described.

The general class COMPONENT contains the attributes (parameters, variables and procedures) common to most of the subclasses. Table 4.2 shows the parameters, the virtual procedures and the variables. The parameters are: a name of the component and two references to shifts (Section 4.4 'Miscellaneous' of this chapter). Both references may refer to 'none' (i.e. no shift at all), but if SH-WK =/= none then SH-PRD has also to refer to a shift because the weekly pattern of SH-WK is activated only during relevant periods of SH-PRD. The 'virtual' procedures may be redefined within subclasses as actual procedures and are used:

Table 4.1 General outline of class SFOBASE-MODEL and its attributes.

```
Simulation class SFOBASE_MODEL (  );
begin
    Process class SHIFT_PERD ( );
    Process class SHIFT_WEEK;
    Process class SHIFT_URG;
    class COMP_ADMINISTRATION ( );
    Process class COMPONENT ( );

        COMPONENT class LABR_EQPMNT ( );
            LABR_EQPMNT class LABOUR;
            LABR_EQPMNT class EQUIPMENT;
        COMPONENT class MAN_MACH_SYS ( );
            MAN_MACH_SYS class MAN_MACH_SET;
            MAN_MACH_SYS class GANG_SET;
                GANG_SET class GANG_SET_GNRTD;
        COMPONENT class MM_SYSTMS_SET;

        COMPONENT class OPERATION;
            OPERATION class OPRTN_MATRL;
            OPERATION class SRVC_REPR;

        COMPONENT class MATERIAL ( );
            MATERIAL class PROCESSED_MAT ( );
                PROCESSED_MAT class INITIAL_MAT;
                PROCESSED_MAT class INTERMDT_MAT;
            MATERIAL class FINAL_MAT;
        COMPONENT class ADMINISTRATOR;
    class MATRL_ADMN ( );
    class AREA;
    Link class FIELD;
    Process class UPDT_MAN_MCH;
    Process class DECISION ( );
    Process class WEATHER_MATRL;


    ref (SHIFT_PERD) array SH_PERD [0:SHS_PERD];
    ref (SHIFT_WEEK) array SH_WKLY [0:SHS_WKLY];
    ref (SHIFT_URG) array SH_URG [1:SHS_URG];
    ref (COMPONENT) array COMPNT [1:    ];
    ref (LABOUR) array MAN [1:MANN];
    ref (EQUIPMENT) array MACH [1:MACHNS];
    ref (MAN_MACH_SYS) array MM_S [1:GANGS + COMBS_G*3];
    ref (MAN_MACH_SET) array GANG [1:GANGS];
    ref (GANG_SET) array COMB_G [1:COMBS_G*3];
    ref (OPERATION) array OPRTN [1:OPRTNS_MT + OPRTNS_S_R];
    ref (OPRTN_MATRL) array OPR_MAT [1:OPRTNS_MT];
    ref (SRVC_REPR) array OPR_S_R [0:OPRTNS_S_R];
    ref (MATERIAL) array MATRL [0:MATRLS];
    ref (PROCESSED_MAT) array MATRL_PRC [1:MATRLS_PROC]
    ref (ADMINISTRATOR) ADMN;
    ref (UPDT_MAN_MCH) UPDT_MM_SYS;
    ref (DECISION) DECIDE;
    ref (WEATHER_MATRL) WTH_MAT;

    Link class RECORD_COMP (CMP); ref (COMPONENT) CMP;
    Link class RECORD_LE (LBR_EQP); ref (LABR_EQPMNT) LBR_EQP;
    Link class RECORD_LB (LBR); ref (LABOUR) LBR;
    Link class RECORD_MM_S (MN_MCH_SYS); ref (MAN_MACH_SYS) MN_MCH_SYS;
    Link class RECORD_OPR (OPR_MT); ref (OPRTN_MATRL) OPR_MT;
    Link class RECORD_SR_RP (SR_RP); ref (SRVC_REPR) SR_RP;
    Link class RECORD_MAT (MATRL_RF); ref (MATERIAL) MATRL_RF;
end;
```

Table 4.2 Class COMPONENT; parameters, virtual procedures and declaration of variables.

```
Process class COMPONENT                                  !=======================================;
(NAME_COMP, SH_PRD, SH_WK);
value NAME_COMP;
text NAME_COMP;
ref (SHIFT_PERD) SH_PRD;
ref (SHIFT_WEEK) SH_WK;              !if sh_wk then also sh_prd =/= none!!;
virtual: procedure SHIFT_CHNGE_, RESET_, INIT_INPUT_;
begin
    !       d       e       c       l       a       r       a       t       i       o       n;
    real                                         !------------------------;
    COSTS_MADE,                      !cum costs made, previous time;
    COSTS_H, COSTS_H_FXD, COSTS_D,   !costs/hour(var..,fixed),/day;
    T_RUN_OVERTM,                    !for category of costs ctgr > 1;
    LT_RPD, LT_C;                    !lasttime of update of time used/ costs;
    real array                                   !------------------------;
    TIME_USED [1:3];                 !cumulative time used [d] in state: 1=run, 2=passive;
    !                                3=down;
    boolean                                      !------------------------;
    AVLB_COMP, AVLB_COMP_PR;         !availability is changed by shift_perd or shift_week;
    integer                                      !------------------------;
    CTGR,                            !current category of costs in shift_week;
    STATE, STATE_PREV, STATE_NEXT;   != run, passive or down for current/ previous/ next state;
    text                                         !------------------------;
    NAME12, NAME7;                   !subtexts of name_comp;
    ref (COMP_ADMINISTRATION)
    COMP_ADMIN;                      !used to record details of components if wanted;
```

- to change the state of the component when a shift requires this (SHIFT-CHNGE-),
- to reset the component in a situation that is required as the initial state of a season (RESET-) and
- to read the initial values of variables of the component from input files at the initialization of the experiment (INIT-INPUT-).

Note the convention to use '-' as the last character of a virtual procedure identifier. The variables declared belong to costs, time used, the availability of the component and its state. The comment between '!' and ';' shows further details. The use of the variables in procedures will also contribute to understanding their meaning. Table 4.3 shows two procedures of the component. SHIFT-CHNGE- is called from the shifts that control the availability of the component, AVLB-COMP over periods (Table 4.103) and within a week (Table 4.105); the current category of costs, CTGR is updated. TIME-C-ACCUM accumulates the time durations and the costs with a systen defined procedure 'Accum' that requires the cumulative variable, the previous moment of accumulation (the duration ranges from that moment to the current moment), the level and a change of the level (not used). Later it checks whether more detailed administration of the component is expected by COMP-ADMIN that refers to an object of class COMP-ADMINISTRATION. The latter class has only an empty, virtual procedure TIME-C-ACC- (Table 7.6) that has to be defined in a subclass (outside the base model in the experimental frame such a subclass can define statistics as mean and variance or record costs per category of overtime costs etc.). Table 4.4 shows

Table 4.3 Procedures SHIFT-CHNGE- and TIME-C-ACCUM of class COMPONENT.

```
procedure SHIFT_CHNGE_;                    !----  ----  ----  ----  ----;
begin                                      !called in shift_week and shift_perd;
    TIME_C_ACCUM;
    if SH_WK =/= none then begin
        AVLB_COMP:= SH_WK.AVLB_WK;
        CTGR:= SH_WK.COST_CTGR;
    end else AVLB_COMP:= true;
    AVLB_COMP:= AVLB_COMP and (if SH_PRD =/= none then SH_PRD.AVLB_PRD else true);
end;                                       !redefined in some subclass;

procedure TIME_C_ACCUM;                    !----  ----  ----  ----  ----;
if AVLB_COMP then begin                    !called in state_ch_..., shift_chnge_, administration;
    COSTS_D:= (COSTS_H + COSTS_H_FXD) * 24.0;   !variable + fixed costs per hour --> per day;
    if STATE = RUN then Accum(COSTS_MADE,LT_C, COSTS_D, 0.0) else LT_C:= Time;
    !          !Accum requires: cum.var., prev.moment, level, change of level;
    if STATE = RUN and CTGR > 1 then T_RUN_OVERTM:= T_RUN_OVERTM + Time - LT_RPD;
    Accum (TIME_USED (STATE), LT_RPD, LEVEL1, 0.0);           !in (d);
    !          !time_used (run) is needed in scheduling failures or service;
    if COMP_ADMIN =/= none then COMP_ADMIN.TIME_C_ACC_;
end else
begin
    LT_C:= LT_RPD:= Time;
    if COMP_ADMIN =/= none then COMP_ADMIN.TIME_C_ACC_;
end;
```

Table 4.4 Procedure RESET and the initial section of class COMPONENT.

```
procedure RESET;                           !----  ----  ----  ----  ----;
begin                                      !called in reload;
    COSTS_MADE:= 0.0; LT_C:= LT_RPD:= Time; T_RUN_OVERTM:= 0.0;
    for I1:= RUN,PASSIVE,DOWN do TIME_USED (I1):= 0.0;
    RESET_;
end;

procedure RESET_;;                         !redefined in specific component if wanted;
procedure INIT_INPUT_; ;                   !redefined in specific component;

!     i     n     i     t     i     a     l;

NAME12:- NAME_COMP.Sub(1,12);
NAME7:- NAME_COMP.Sub(1,7);                !name_comp is assumed 24 char. long;
if SH_PRD =/= none then new RECORD_COMP (this COMPONENT). Into (SH_PRD.COMP_Q)
else AVLB_COMP:= true;
if  SH_WK =/= none then new RECORD_COMP (this COMPONENT). Into (SH_WK.CMP_Q);
!else avlb_comp:= avlb_comp and true;
CTGR:= 1;
STATE:= STATE_NEXT:= RUN;                  !initial if not redefined in subclass;

inner;                                     !initial section of subclass executed;
```

the initial section of this class where abbreviated names of twelve and seven characters are denoted to NAME12 and NAME7, the component is queued in the queue of components of the shifts SH-PRD and SH-WK to let those shifts know the components they control or AVLB-COMP is independent of the shifts and set correctly, the cost category and the state are assigned default values and the initial section of a subclass follows (the same as assumed by the language if 'inner' was not programmed). Procedure RESET shows the variables made zero when the system is 'reloaded' for another season. Further resetting of variables in subclasses is requested in RESET- which is defined in the subclass. If the subclass does not require RESET- then the

Table 4.5 Class SFOBASE-MODEL and its parameters.

```
Simulation class SFOBASE_MODEL              !*************************************;
(LE_SQNS, MANN, MACHNS, MM_S_SETS, GANGS, COMBS_G, OPRTNS_MT, OPRTNS_S_R,
MATRLS_INIT, MATRLS_PROC, MATRLS, SHS_PERD, SHS_WKLY, SHS_URG);
integer
LE_SQNS,                  !number of types of labour & equipment;
MANN,                     !number of men;
MACHNS,                   !number of machines as tractors, animals, implements a,b,..,tools;
MM_S_SETS,                !number of sets of combinations, see class definition;
GANGS,                    !number of gangs;
COMBS_G,                  !number of pure combinations: with two or more gangs;
OPRTNS_MT, OPRTNS_S_R,    !number of operations for material, for service & repair;
MATRLS_INIT,              !number of initial materials;
MATRLS_PROC,              !number of mat.processed;
MATRLS,                   !number of materials;
SHS_WKLY, SHS_PERD,       !number of shifttypes for components: week,/period;
SHS_URG;                  !number of shifttypes for urgency calc. of materials;
begin
```

Table 4.6 Declaration of variables of class SFOBASE-MODEL.

```
!   d     e     c     l     a     r     a     t     i     o     n;

integer                          !----------------------;
RUN, PASSIVE, DOWN,              !states of objects;
TOTAL, AVAIL,                    !subscript in le_nmb [le_sqn,..];
MN_MCH_SYSTMS,                   !number of man-machine systems = gangs + combs_g;
OPRTNS,                          !number of operations in total;
U,                               !seed of random number generators;
YR_N,                            !sequence number of current year;
DAY_TP_1JAN, DAY_TYPE_NOW,       !day at 1 jan., current day, 1=monday,etc.;
DAYS_NMB,                        !number of days since jan.1 at 0:00;
LAST_DAY_YEAR,                   !>365 e.g. 1000: to make last three digits equal each year;
I1, J2,                          !for loop indices;                ! for a given date;
MON, TUE, WED, THU, FRI, SAT, SUN;  !sequence number of days: 1, 2, ..., 7;
real                             !----------------------;
DAY_BGN, DAY_END,                !begin, end of work in clocktime hours;
HOUR, DAY_FRAC_NOW,              !fraction of day passed at hour = current time;
TIME_1JAN,                       !time at 1 january 0:00;
LEVEL1;                          !used in accum as a variable = 1.0;
integer array                    !----------------------;
LE_NMB [1:LE_SQNS,1:2];          !number of labour & equipment objects with;
!                                same sequential no., total=1 resp. available=2(not down);
boolean                          !----------------------;
END_SEASON, END_EXPRMNT,         !true if end of season or end of experiment is achieved;
ENDED_PROCS,                     !true if materials are processed;
MM_S_SELECT;                     !true if a man-machine system is selected to work;
text array                       !----------------------;
DAY_TXT3 [0:7], MNTH_TXT3 [1:12];
text                             !----------------------;
MNTH_DT6, YES;
ref (Head) array                 !----------------------;
LE_STATE_Q [1:LE_SQNS,1:3];      !queue for labour & equipment objects of type le_sqn;
!                                states: run, passive or down;
ref (Infile)
PARAMETERS;                      !input file to initialise system and elements;
ref (Printfile)
QNT_TRF_FLD,                     !output file to print quantities from/to fields;
URG_APL_OUTP, TML_OUTP;          !urgencies, timeliness;
```

procedure RESET- ;; with the empty statement ';' is used to prevent errors during running.

The class SFOBASE-MODEL is prefixed by 'simulation' so that 'Process', 'Time', 'Activate', etc. can be used. The parameters of the base model are shown in Table 4.5 and concern the number of objects used such as men, machines. operations, materials and shifts (see comment). These parameters are used later on to declare arrays of references to the different elements that are created in the system. Some general variables, as shown in Table 4.6 are also required; they concern the state of objects, indices for days in the week, type and number of day, text variables for days and months, arrays and queues for men and machine types and references to input and output files. The comment shown between '!' and ';' should give an indication of the meaning for the moment; further explanation is given where necessary with the actual use of the variables. The appropriate initialization is shown in Table 4.7. Some variables are initialized in the experimental frame and others are changed in the model and start with default values. The different components of the scheduling system can now be described.

Table 4.7 Initial section of class SFOBASE-MODEL.

```
! i n i t i a l;

RUN:=1;PASSIVE:=2;DOWN:=3;
TOTAL:=1;AVAIL:=2;
LEVEL1:=1.0;
U   :=907;                    !start value of randomized seeds of random numbers, demos p.49;
MN_MCH_SYSTMS:= GANGS + COMBS_G *3; !number overrated to allow generation of combinations;
OPRTNS:= OPRTNS_MT + OPRTNS_S_R;
for I1:= RUN,PASSIVE,DOWN do for J2:= 1 step 1 until LE_SONS do LE_STATE_Q [J2,I1]:- new Head;

MON:= 1; TUE:= 2; WED:= 3; THU:= 4; FRI:= 5; SAT:= 6; SUN:= 7;
DAY_TXT3 [0]:- Copy ('---'); DAY_TXT3 [1]:- Copy ('Mon'); DAY_TXT3 [2]:- Copy ('Tue');
DAY_TXT3 [3]:- Copy ('Wed'); DAY_TXT3 [4]:- Copy ('Thu'); DAY_TXT3 [5]:- Copy ('Fri');
DAY_TXT3 [6]:- Copy ('Sat'); DAY_TXT3 [7]:- Copy ('Sun');
MNTH_TXT3 [1]:- Copy ('Jan'); MNTH_TXT3 [2]:- Copy ('Feb'); MNTH_TXT3 [3]:- Copy ('Mar');
MNTH_TXT3 [4]:- Copy ('Apr'); MNTH_TXT3 [5]:- Copy ('May'); MNTH_TXT3 [6]:- Copy ('Jun');
MNTH_TXT3 [7]:- Copy ('Jul'); MNTH_TXT3 [8]:- Copy ('Aug'); MNTH_TXT3 [9]:- Copy ('Sep');
MNTH_TXT3 [10]:- Copy ('Okt'); MNTH_TXT3 [11]:- Copy ('Nov'); MNTH_TXT3 [12]:- Copy ('Dec');
MNTH_DT6:- Blanks (6);
YES:- Copy ('yes');                !text in input not in capitals;
```

4.1 Materials and weather

The biological subsystem on an arable farm consists of weather and of crops and soil. To incorporate seed, fertilizer, intermediate products such as straw and final-products, the term material is used, which may refer even to cattle. Within a material different fields with their own attributes of area, ripening date, etc. are distinguished.

In the next sections the components weather (4.1.1), field (4.1.2) and material (4.1.3) are described. Material is divided up in two subclasses, final materials (4.1.7) and processed materials (4.1.4), and the latter is a superclass of initial materials (4.1.5) and intermediate materials (4.1.6).

4.1.1 Weather

The component weather is shown in Table 4.8 as class WEATHER-MATRL prefixed as 'Process'. The base model only assumes that this proc-

Table 4.8 Class WEATHER-MATRL and the declaration of variables.

```
Process class WEATHER_MATRL;                    !***************************************;
begin
   !    d    e    c    l    a    r    a    t    i    o    n;

   real array                                   !-----------------------;
   PROPERTY [1:MATRLS_PROC];          !property of materials controlling workability;
   real                                         !-----------------------;
   DAY_TYPE_WTHR;                     !type of day from input;
end;
```

ess can produce one property of each material processed (a moisture content, for instance, controlling the workability) and a type of the day. Further details such as input of weather data and deriving or reading PROPERTY have to be described in a subclass defined in the experimental frame.

4.1.2 Fields

The field is a name for different things in different materials; it can mean a piece of land with or without a crop, a storage with seed potatoes, fertilizer or harvested grain and even a herd of milking cows on grass (for further information see Section 7.2.4 'Grass and cattle'). Table 4.9 shows the declaration of variables of class FIELD. It has a prefix 'Link', so it is not a process itself (as material) but it is queued in a material. Thus a material such as wheat can contain several fields each with its own attributes. The variables declared concern the actual quantity or area, the cumulative quantities produced and processed, a quantity already processed but not yet consumed, some dates relative to January 1 as day produced, day crop is processable (ripe) and day when the optimum yield is achieved, text variables and a reference to some area. The procedures are shown in Table 4.10. The procedure CONSUMPT-F ( ) is called in material when some area QNT-F is consumed/processed; it decreases quantities and checks if QNT-F does not exceed the current area (otherwise an error message occurs); if the field is processed entirely then the remaining amount processed, QNT-IN-PROCS is transferred to the preceding field or to the successive field or if both do

Table 4.9 Class FIELD and the declaration of variables.

```
Link class FIELD;                               !****************************************;
begin
   !    d    e    c    l    a    r    t    i    o    n;

   real                                         !-----------------------;
   QUANTITY,                          ![ha];
   QUANT_F_PRD, QUANT_F_PRC,          !cum. quantities produced and processed;
   QNT_IN_PROCS,                      !quantity processed but not yet subtracted from quantity;
   DATE_PRODCD,                       !date of delivering field by producing material;
   DATE_PROCSBL,                      !date at which material becomes processable;
   DATE_OPT_YLD;                      !date of processing when max.yield is recovered;
   text                                         !-----------------------;
   NAME_FLD, FLD_SQNO_TXT, MAT_NO_FLD;
   ref (AREA)                                   !-----------------------;
   FLD_AREA;                          !contains attributes common to fields in the same area;
```

Table 4.10 Procedures CONSUMPT-F and MAKE-NAME of class FIELD.

```
procedure CONSUMPT_F (QNT_F);                           !----   ----   ----   ----   ----;
real QNT_F;                                             !called in mat.consumptn_m;
begin
   if QNT_F > QUANTITY + 1.0&-4 then begin
      Outimage;
      Outtext ('Error FLD.CONSUMPT_F n Field, at Time,');
      Outtext (' Amount consumed > Amount in field!'); Outimage;      !may not occur;
      Outtext (NAME_FLD); Outfix (Time,6,12); Outfix (QNT_F,6,12);
      Outfix (QUANTITY,6,12); Outimage;
   end;
   QUANTITY:= QUANTITY - QNT_F;
   QUANT_F_PRC:= QUANT_F_PRC + QNT_F;
   QNT_IN_PROCS:= QNT_IN_PROCS - QNT_F;
   if QUANTITY < 1.0&-4  and QNT_IN_PROCS > 1.0&-4 then begin        !only qnt_in_procs;
      if Pred =/= none then Pred qua FIELD.QNT_IN_PROCS:= QNT_IN_PROCS!reported to;
      else    if Suc =/= none then Suc qua FIELD.QNT_IN_PROCS:= QNT_IN_PROCS!subsequent field;
      !;      else    if QNT_IN_PROCS > 1.0&-3 then begin              !may not occur;
         Outimage;
         Outtext ('Error FLD.CONSUMPT_F in Field, at Time,');
         Outtext (' more Amount processed than Available'); Outimage;
         Outtext (NAME_FLD); Outfix (Time,6,12); Outfix (QNT_IN_PROCS,6,12);
         Outfix (QUANTITY,6,12); Outimage;
      end;
   end;
end;

procedure MAKE_NAME (FLD_SQN, M_NO);
integer FLD_SQN, M_NO;
begin
   FLD_SQNO_TXT.Putint (FLD_SQN);
   MAT_NO_FLD.Putint (M_NO);
end;
```

Table 4.11 Class AREA and the declaration of variables.

```
class AREA;                                  !****************************************;
begin
   !     d     e     c     l     a     r     a     t     i     o     n;

   real                                      !-----------------------;
   ACREAGE,                 ![ha]            !may be used in submodels;
   DISTANCE,                ![km]            !may be used in submodels;
   LATITUDE, LONGITUDE;     ![km]            !may be used in submodels;
   text AREA_NAME;                           !-----------------------;

   !    i    n    i    t    i    a    l;

   AREA_NAME:- Blanks (30);
end;
```

not exist an error message occurs. The quantity in process, QNT-IN-PROCS is relevant for only one field in the queue denoted in material by FLD-C, in general the first field. In CONSUMPT-F a very small quantity 1.0&-4 (= 0.0001) is used to prevent messages caused by rounding errors. Rounding errors can occur because the system defined simulation attribute 'Time' unfortunately is not a long real in DEC10 SIMULA. This means when days are counted in five digits (two for the year and three for the day) that only four decimal digits are significant; 0.0001 day is approx. 0.14 min. With a capacity of 1 ha/h differences of up to 0.0024 ha (24 h/d * 0.0001 d * 1 ha/h) can be expected. The procedure MAKE-NAME positions a field sequence number FLD-SQN (given in material) and a material number M-NO into the subtext of the name. The name is used in the error messages and thus assigns a number and a name, for instance, ' 32e field of mat. 2'.

Table 4.11 shows class AREA with its area, distance, coordinates and a name. Several fields can refer to the same area. Later on it is shown how this reference to an area is used to add a delivered quantity to the already existing field or to create a new field.

Table 4.12 Class MATERIAL; parameter, virtual procedures and declaration of variables.

```
COMPONENT class MATERIAL (MAT_NO);                      !**************************************;
integer MAT_NO;                       !sequence number of material;
virtual: procedure ATTR_UPDT_M_, MAT_DVLPMNT_, DELIVERY_M_,
ATTR_FLD_M_, FLD_EXPCT_M_, ATTR_INTG_F_, DELVR_FLD_M_, ATTR_INTG_M_, MAX_QUANT_M_, CAP_EXC__M_;
begin
   !    d    e    c    l    a    r    a    t    i    o    n;

     integer                                            !----------------------;
     DLR_OPRS, PRC_OPRS,              !number of operations delivering/ processing;
     FLD_Q_SQN, FLD_Q_SQN_PR,         !field sequence numbers;
     LOOP_DYN,                        !counts steps at same time to interrupt a loop;
     M_POS, DATE_NO, J1;              !auxiliary var.;
     boolean                                            !----------------------;
     AVLBL, AVLBL_NEXT,               !true if material is available;
     WORKBL, WORKBL_NEXT,             !true if general material attributes and;
     !                                !weather o.k., same for all fields;
     READY, READY_NEXT,               !true if workable and (first or expected) field processable;
     FLD_AT_TAIL,                     !true if new field is queued at end of fields;
     SUPPLY_NDD,                      !true if no quantity available;
     DLVR_ALLWD,                      !true if available quantity less then actual maximum;
     DLVRY_SETUP, PRCSS_SETUP,        !true after delivery / proressing operation starts setup;
     DELIVERING, PROCESSING;          !true if material delivering/processing occurs;
     real                                               !----------------------;
     QUANT_ARRVD,                     !quantity produced, delivered;
     QUANT_AVLBL,                     !quantity available = arrived - processed;
     QUANT_PROCSD,                    !quantity processed, consumed (incl.losses);
     QUANT_MX,                        !allowed max. of available quantity;
     QUANT_MX_ACT,                    !actual max., defined in subclass as function of ...;
     QNT_MX_FACT,                     !factor used in max_quant_m_ to reduce actual max.quantity;
     QNT_PROC_OPR, QNT_DLVR_OPR,      !quantity processed by operations but not yet transferred;
     QNT_M, QNT_M_D,                  !auxiliary quantities;
     QNT_PRC_DUMM,                    !cum. quantity processed but not available;
     QNT_DLTD,                        !cum. quantity that is deleted if no fields are available;
     CAP_PROC,                        !expected capacity of processing (depends on gangs);
     CAP_OPRS_PRC, CAP_OPRS_DLV,      !sum of capacity of operations processing/ delivering;
     CAP_RATIO;                       !capacity factor due to attributes;
     !long;
     real                                               !----------------------;
     EVTM_FLD, EVTM_MAX,              !event time for field consumed/ maximum quantity delivered;
     EVTM_DLVR_OK,                    !event time to accept delivery again;
     EVTM_DVLPMNT,                    !event time for autonomous development of attributes;
     EVTM,                            !event time;
     LT_DLVR, LT_PROC,                !last times in deliver_mat,process_mat;
     LT_INTEGRT,                      !last time of integration of quantity processed on fields;
     DURTN_CONS, DURTN_DELVR,         !duration of consumption/ delivering;
     DURTN_NO_DLVR;                   !duration delivering not allowed;
     ref (MATRL_ADMN)                                   !----------------------;
     M_ADMN;                          !reference to a class recording more data of material;
     ref (FIELD)
     FLD_C, FLD_D, FLD_E, FLD;        !field consumed, delivered,expected, any;
     ref (Head)
     FLD_Q,                           !queue for fields;
     OPRTN_DLVR_Q, OPRTN_PROC_Q;      !queues for operations delivering,processing material;
```

4.1.3 *Materials*

It is already known that 'material' is a term referring to crops, products in storage, soil and even cattle. By the operation planting potatoes, the materials 'seed potatoes' and 'soil' are consumed and a material 'planted potatoes' is delivered and by harvesting potatoes a material 'potatoes' is consumed and two materials are delivered 'stored potatoes' and 'soil'. Class MATERIAL will be described in sections related to (1) the states of a material, (2) the delivery of material, (3) the processing or consumption of material and (4) the dynamic aspects. Materials are delivered by operations and processed by others, and therefore a chain exists with links between materials and operations; the following scheme shows the part related to one material:

```
---> operation(s) ---> material ---> operation(s) --->
     delivering                      processing
```

Table 4.12 shows the declaration of variables and 'virtual' procedures of a material; the description of variables is shown as comment (!..........;) and in the next sections where necessary.

4.1.3.1 *States of material*

The usual states RUN, PASSIVE and DOWN assigned to the variable STATE from superclass COMPONENT are extended to distinguish more situations in state DOWN. Such an extension is necessary to define STATE appropriately and is useful to analyse the reasons for state DOWN. In Section 2.1.1 'Materials and weather' some relevant attributes have already been described. The Boolean variables AVLBL, WORKBL and READY are used, which are true if the material is available, if the material is workable (the moisture content appropriate) and if the material is ripe (processable) and workable, respectively. Table 4.13 shows the procedures changing these variables. The next situation was defined elsewhere in the calling procedure except for READY-NEXT.

When the existing situation differs from the next one and an object M-ADMN exists for further detailed administration, then 'virtual' procedures are called (the actual version of the procedure will be used). The desired information is defined later in the experimental frame and it is not yet known what will be recorded. The calling of CH-AVLBL, CH-WORKBL and CH-READY from a procedure is followed by calling STATE-CH-MAT to update the variable STATE (for instance, Table 4.17).

Table 4.14 shows the procedure STATE-CH-MAT, which first defines the next state, STATE-NEXT as DOWN if the material is not available (no quantity) or not workable or not ready or not considered as available com-

Table 4.13 Procedures CH-AVLBL, CH-WORKBL and CH-READY of class MATERIAL.

```
procedure CH_AVLBL;                              !----  ----  ----  ----  ----;
!                           !called in supply_expct, dlvry_stop, consumptn_m, reset_, reset_m_;
if not (AVLBL eqv AVLBL_NEXT) then begin
    if M_ADMN =/= none then M_ADMN.ADMN_AVLBL_;
    AVLBL:= AVLBL_NEXT;
end ** of change of availability of material;

procedure CH_WORKBL;                             !----  ----  ----  ----  ----;
!                                                !called in attr_updt_m_;
begin
    if not (WORKBL eqv WORKBL_NEXT) and M_ADMN =/= none then M_ADMN.ADMN_WRKBL_;
    WORKBL:= WORKBL_NEXT;
    CH_READY;
end ** of change of workability of material and weather;

procedure CH_READY;                              !----  ----  ----  ----  ----;
begin                   !called in ch_workbl, supply_expct, consumptn_m, reset_, urg_mat_prc_;
    if FLD_C == none then FLD_C:- FLD_Q.First;  !may be an expected field;
    READY_NEXT:= WORKBL and (if FLD_C =/= none then FLD_C.DATE_PROCSBL <= TIME_YR else false);
    if not (READY eqv READY_NEXT) and M_ADMN =/= none then M_ADMN.ADMN_READY_;
    READY:= READY_NEXT;
end ** of change of readiness of material on first field;
```

Table 4.14 Procedure STATE-CH-MAT of class MATERIAL.

```
procedure STATE_CH_MAT;                          !----  ----  ----  ----  ----;
begin  !called in supply_expct, dlvry_stop, prcsng_expct, consumptn_m, stop_prcsng;
    !                               shift_chnge_,reset_, attr_updt_m, urg_mat_prc;

    if STATE = STATE_NEXT then
    STATE_NEXT:=
    if not (AVLBL and WORKBL and READY and AVLB_COMP) then DOWN
    else if STATE = RUN or PRCSS_SETUP then RUN
    !;  else PASSIVE;
    if STATE <> STATE_NEXT then begin
        ref (RECORD_OPR) REC_OPR_CNS;
        TIME_C_ACCUM;
        if M_ADMN =/= none then M_ADMN.ADMN_M_STATE_;
        STATE_PREV:= STATE;
        STATE:= STATE_NEXT;
        if STATE_PREV = DOWN and STATE = PASSIVE then begin     !cause from material, d ---> p;
            DECIDE.MAT_PASS:=  true;
            activate DECIDE at Time;
        end;
        if STATE_PREV = RUN and STATE = DOWN then begin         !cause from material, r ---> d;
            DECIDE.MAT_DOWN:=  true;
            activate DECIDE at Time;                             !results in stop_oprtn;
        end;
        !       !if state_prev = run and state = passive then caused by decision, r ---> p;
        !       !if state_prev = passive and state= run then caused by decision,  p ---> r;
        !       !if state_prev = passive and state= down then caused by material, p ---> d;
        REC_OPR_CNS:-OPRTN_PROC_Q.First;       !                      !illegal, d ---> r;
        if STATE_PREV = DOWN or STATE = DOWN then
        !           !change in operations  the state of material if d--->r,p or r,p--->d;
        while REC_OPR_CNS =/=none do begin
            REC_OPR_CNS.OPR_MT.STATE_CH_OP_;
            REC_OPR_CNS:- REC_OPR_CNS.Suc;
        end xx of change in operations processing this material;
    end;
end ** change of state of material;
```

ponent in the current period and as RUN or PASSIVE otherwise. The further statements are executed if the state has to change. Some data are recorded in TIME-C-ACCUM and ADMN-M-STATE-. The state is changed. Some statements activate DECIDE and tell the reason (material becomes

Table 4.15 The transformation of states of materials due to events.

STATE_PREV	STATE RUN	PASSIVE	DOWN
RUN		decision	material (*)
PASSIVE	decision		material
DOWN	(not allowed)	material (*)	

(*) In these cases DECIDE is informed and activated to consider the consequences; note that the activation will become effective at Time but after updating the state of operations.

PASSIVE or DOWN, so DECIDE may want to start an operation processing this material or it has to stop an operation processing this material, respectively). When STATE was or becomes DOWN, then the operations that can process this material (found in queue OPRTN-PROC-Q) must be known: STATE-CH-OP- is called to change the state of operations accordingly. The causes of the changes in the state of a material are shown in the scheme of Table 4.15 (almost identical to Table 2.10).

4.1.3.2 Delivery of material

The description of the program related to delivery or supplying a material by an operation is divided up in four parts. The first part is concerned with the beginning of the operation, the second part with the actual use and the third part with stopping the operation. The fourth part deals with some auxiliary procedures.

Table 4.16 shows a declaration of the 'virtual' procedure FLD-EXPCT-M- that creates an expected field with a minimum of data (dates of optimum yield and of material processable equal to the current time since Jan. 1). This field is used only when no fields are available with data to store temporarily expected data valid for a field that will be delivered.

The first part related to the beginning of an operation consists of the procedures SUPPLY-EXPCT and DLVRY-STARTD. Table 4.17 shows SUPPLY-EXPCT which is called from the operation delivering the material. SUPPLY-EXPCT is called just before the operation begins with set-up i.e.

Table 4.16 Procedure FLD-EXPCT-M- of class MATERIAL.

```
procedure FLD_EXPCT_M_;                    !---- ---- ---- ---- ----;
!                               !called in reset_,supply_expct,consumptn_m,urg_mat_prr;
inspect new FIELD do begin
   FLD_D:- this FIELD;
   Into (FLD_Q);
   MAKE_NAME (0, MAT_NO);
   DATE_OPT_YLD:= DATE_PROCSBL:= TIME_YR;
end;
```

Table 4.17 Procedures SUPPLY-EXPCT and DLVRY-STARTD of class MATERIAL.

```
procedure SUPPLY_EXPCT;                        !----  ----  ----  ----  ----;
begin                                          !called in oprtn.go_ahead, .mat_prod_chng;
    DLVRY_SETUP:= true;
    DLR_OPRS:= DLR_OPRS + 1;
    AVLBL_NEXT:= true; CH_AVLBL;        !even if oprtn still in setup, may start processing;
    if SUPPLY_NDD and not PROCESSING and FLD_E =/= none then begin
        FLD_E.Out; FLD_EXPCT_M_;                   !updates expected field;
        FLD_E:- FLD_C:- FLD_D;
        CH_READY;
        ATTR_INTG_F_;                              !updates tml_value;
    end;
    STATE_CH_MAT;
end;

procedure DLVRY_STARTD (OPR_MT2);              !----  ----  ----  ----  ----;
ref(OPRTN_MATRL)OPR_MT2;                       !called in oprtn.go_on, .mat_prod_chng;
begin
    ref (RECORD_OPR) REC_OPR_CNS1;
    CAP_OPRS_DLV:= CAP_OPRS_DLV + OPR_MT2.CAPCTY_ACTL;
    DELIVERING:= true;
    if STATE = RUN and not PROCESSING then begin
        REC_OPR_CNS1:- OPRTN_PROC_Q.First;
        while REC_OPR_CNS1 =/= none do inspect REC_OPR_CNS1.OPR_MT do begin
            if RUN_PHASE = MAT_WAIT then WAIT_MAT_PRC;  !check if processing can start;
            REC_OPR_CNS1:- REC_OPR_CNS1.Suc;
        end;
    end;
end;
```

with refueling, preparing machines and driving to the field. At this stage DLVRY-SETUP and AVLBL-NEXT are made true and the state is changed by calling STATE-CH-MAT. The number of operations DLR-OPRS is increased by one. Due to AVLBL = true the state can become PASSIVE (Table 4.14) and DECIDE may result in starting an operation processing this material. If at this moment no material is available or already delivered, nevertheless AVLBL becomes true even if it was false, so the variable SUPPLY-NDD was used, which remains true when the quantity is zero and supply is needed. In the same situation a so-called expected field is required containing attributes like moisture content, date when the optimum yield is achieved and when it becomes processable if processing is to start. In order to handle the most recent data, the expected field is updated by calling FLD-EXPCT-M-, CH-READY and ATTR-INTG-F- (defined in a subclass).

DLVRY-STARTD (Table 4.17) is called by the operation after its set-up and at this moment the actual capacity of the operation is added to the total capacity delivering this material CAP-OPRS-DLV; DELIVERING now becomes true. If this material expected processing (STATE = RUN) but has no quantity and so could not continue until delivery started, then all the operations processing this material are checked if they are in a phase waiting for materials, MAT-WAIT and can leave this phase (by calling WAIT-MAT-PRC).

Exercise: Do you remember the example of the grain drier which has to wait until the combine-harvester has completed its set-up to begin the delivery of wet grain?

Table 4.18 Procedure DLVRY-CONTND of class MATERIAL.

```
procedure DLVRY_CONTND;                          !----   ----   ----   ----   ----;
if STATE = RUN and PROCESSING and SUPPLY_NDD then begin !called in oprtn.go_on;
    ref (RECORD_OPR) REC_OPR_CNS2;
    REC_OPR_CNS2:- OPRTN_PROC_Q.First;
    while REC_OPR_CNS2 =/= none do inspect REC_OPR_CNS2.OPR_MT do begin
        if RUN_PHASE = GO_ON_WAIT then WAIT_MAT_PRC;     !processing can continue;
        REC_OPR_CNS2:- REC_OPR_CNS2.Suc;
    end;
end;
```

The second part of delivery of material concerns what happens to the material when an operation is actually delivering. Four procedures are involved: DLVRY-CONTND, ACCEPT-UNTIL, CAP-CHNG-DLV and DELIVERY-M-. Table 4.18 shows DLVRY-CONTND that is called in an operation each time it continues its activity of processing and delivery. If processing of material occured and the available quantity is exhausted before delivery starts, then supply is needed before the processing can continue. This procedure now tells that delivery by some operation occurs and it checks the operations processing this material (found in queue OPRTN-PROC-Q) which are in that specific phase of waiting to continue and calls WAIT-MAT-PRC in that operation to continue with processing. The structure of the procedure has some similarity with a part of DLVRY-STARTD (Table 4.17); the differences are PROCESSING and RUN-PHASE. There are two cases when processing is held up; the first one is when a small quantity of material is processed before the set-up of a delivering operation is completed, the second one is when processing is faster than delivery and the delivered material is processed immediately.

Exercise: Can you give an example with wet grain as the material (first case) and an example with bales in the field together with the baling and gathering operations? In this case part of the gathering capacity remains idle (the area that served as dummy is cumulated in QNT-PRC-DUMM, Table 4.30).

ACCEPT-UNTIL (Table 4.19) calculates the moment when the actual

Table 4.19 Procedure ACCEPT-UNTIL of class MATERIAL.

```
procedure ACCEPT_UNTIL;                          !----   ----   ----   ----   ----;
!called in oprtn.go_on, dlvry_stup, start_/stop_prcsng, cap_chng_dlv/_prcsng, max_quant_m_;
if DELIVERING and DLVR_ALLWD then begin          !find point when storage is full;
    QNT_INTGR_M;
    DURTN_DELVR:= if CAP_OPRS_DLV - CAP_OPRS_PRC > 0.0
    then (QUANT_MX_ACT - (QUANT_AVLBL - QNT_PROC_OPR + QNT_DLVR_OPR)) /
    (CAP_OPRS_DLV - CAP_OPRS_PRC) / 24.0
    else -1.0;                                   !evtm expected from other material;
    EVTM_MAX:= Time + DURTN_DELVR;
    if DURTN_DELVR > 0.0 then EVTM_MAX:= RMAX (EVTM_MAX,
    if Time + 1.0&-4 > Time then LT_DLVR + 1.0&-4 else LT_DLVR + 1.0&-3);
    if DURTN_DELVR <= 0.0 then EVTM_MAX := Time - 1.0;  !irrelevant;
    if EVTM_MAX > Time and
    (if not Idle then Evtime > Time and EVTM_MAX < EVTM  else true)
    then reactivate this MATERIAL at Time;
end of accept_until;
```

maximum quantity, QUANT-MX-ACT, is achieved which occurs only when DELIVERING is true and DLVR-ALLWD is true (i.e. delivery is still allowed when the maximum is not yet achieved or exceeded); it considers the quantities processed, QNT-PROC-OPR and produced, QNT-DLVR-OPR, that are already in the pipeline (updated in QNT-INTGR-M Table 4.25) but not yet worked up in the available quantity, QUANT-AVLBL. When the processing is faster than the delivery or the maximum is already achieved then the duration of delivery, DURTN-DELVR is negative and the event time, EVTM-MAX will be earlier than the current 'Time' (no activation). Otherwise a minimum of 0.0001 or 0.001 day (approx. 0.14 or 1.44 min) is added to the last time delivery occurred, LT-DLVR in order to avoid rounding off errors (difference between Time + addition and Time not significant if the Simulation attribute Time is not a long-real; discussion in Section 4.1.2 'Fields' of procedure CONSUMPT-F). The activation of the dynamic section of this material which handles all the event times of a material (Section 4.1.3.4) is achieved by calling 'reactivate'. The activation is not done if EVTM-MAX is not in the future or the dynamic section is already scheduled (not Idle) at Time (Evtime >Time is false) or the current event time, EVTM is earlier than the one just calculated. 'Re'-activate is necessary to ensure rescheduling when the material is already scheduled.

The third procedure, CAP-CHNG-DLV (Table 4.20) is called if the capacity of the operation changes due to attributes of a field; it updates (integrates) the quantities in the pipeline (QNT-INTGR-M), changes the rate or capacity of producing and adjusts the moment a maximum occurs (ACCEPT-UNTIL).

The actual delivery occurs in procedure DELIVERY-M- (Table 4.21), which is called in the operation (ref. OPR-MT1) when the quantity is transferred from the materials processed/consumed to the materials produced/delivered/supplied. The quantity delivered, QNT-M-D is known from the operation and used to update the quantities arrived, available and produced (and in the pipeline). It calls the virtual procedure DELVR-FLD-M-defined in the subclass PROCESSED-MAT (Section 4.1.4), which creates a new field with appropriate attributes or updates attributes of an existing field. If the material was needed to process, it calls immediately to consume the delivered quantity by calling PROCESS-MAT. The maximum quantity is checked by calling the 'virtual' procedure MAX-QUANT-M-. If a check on the flow of quantities is required and QNT-TRF-FLD refers to a printfile

Table 4.20 Procedure CAP-CHNG-DLV of class MATERIAL.

```
procedure CAP_CHNG_DLV (OPR_MT3);                  !----  ----  ----  ----  ----;
ref(OPRTN_MATRL) OPR_MT3;
if DELIVERING then begin                           !called in oprtn.cap_qnt_chng;
    QNT_INTGR_M;
    CAP_OPRS_DLV:= CAP_OPRS_DLV + OPR_MT3.CAP_CHNG;
    ACCEPT_UNTIL;
end;
```

Table 4.21 Procedure DELIVERY-M- of class MATERIAL.

```
procedure DELIVERY_M_ (OPR_MT1);                        !----   ----   ----   ----   ----;
ref (OPRTN_MATRL) OPR_MT1;
begin                                                   !called in oprtn.qnt_transfer;
    QNT_INTGR_M;
    QNT_M_D:= OPR_MT1.QUANT;
    QUANT_ARRVD:= QUANT_ARRVD + QNT_M_D;
    QUANT_AVLBL:= QUANT_AVLBL + QNT_M_D;
    QNT_DLVR_OPR:= QNT_DLVR_OPR - QNT_M_D;
    if QNT_M_D > 0.0 then DELVR_FLD_M_(QNT_M_D, OPR_MT1.MAT_PROC(1).FLD_C);
    !           only properties of first field of first material processed by operation!!;
    if PROCESSING and SUPPLY_NDD then PROCESS_MAT;      !consumes the delivered field;
    MAX_QUANT_M_;
    inspect QNT_TRF_FLD do begin
        Setpos(M_POS);
        Outtext(FLD_D.FLD_SQNO_TXT);
        Outfix(QNT_M_D,2,7);
        if FLD_Q_SQN > FLD_Q_SQN_PR then Outtext("nF");
        FLD_Q_SQN_PR:= FLD_Q_SQN;
    end;
    if QUANT_AVLBL > 1.0&-4 then SUPPLY_NDD:= false;
    if FLD_E =/= none then begin
        if FLD_E.QUANTITY < 1.0&-4 and FLD_E.QNT_IN_PROCS < 1.0&-4
        and FLD_E =/= FLD_C and FLD_E =/= FLD_D then begin
            FLD_E.Out; FLD_E:- none;
        end;
    end;
    if not PROCESSING then FLD_C:- FLD_Q.First;
    if PROCESSING and (FLD_D == FLD_C or EVTM_FLD < Time) then GO_UNTIL_MP;     !adjust evtm_fld;
end;
```

(not to none), then at a position M-POS output starts of a field number and is followed by the quantity delivered to the field and an indication 'nF' if the field is new. SUPPLY-NDD may no longer be true and an expected field, FLD-E is no longer needed and under certain conditions removed out of the queue of fields. The first field in the queue is assigned to FLD-C a reference to the field intended for consumption. If processing occurs and nevertheless the event time is in the past then EVTM-FLD must be updated by calling GO-UNTIL-MP (Table 4.29). This actual version of the 'virtual' declared procedure, DELIVERY-M-, may be redefined in a subclass.

The third part of delivery consists of DLVRY-STOP to stop the delivery of a material by an operation (Table 4.22). The material is processed by calling PROCESS-MAT if the capacity of delivery was smaller than that of processing and as a consequence the processed quantity in the pipeline may be more than is available; these facts must still be known in ADJUST-QUANT (Section 4.1.3.3 'Processing of material', Table 4.30) before DELIVERING becomes false to avoid an error message. It decreases the capacity of producing if it was added in DLVRY-STARTD (Table 4.17). The number of operations delivering this material, DLR-OPRS is decreased by one. If the number of operations is zero, it makes (i) the capacity zero along with the quantity in the pipeline, (ii) DELIVERING and DLVRY-SETUP false, (iii) AVLBL false and changes the state if the available quantity remains or becomes zero and (iv) the time at which the maximum could be achieved, EVTM-MAX equal to the current moment, Time. Further on

Table 4.22 Procedure DLVRY-STOP of class MATERIAL.

```
procedure DLVRY_STOP (OPR_MT4);                         !----  ----  ----  ----  ----;
ref (OPRTN_MATRL) OPR_MT4;
begin                                                   !called in oprtn.termnt,.mat_prod_chng;
    if PROCESSING and QUANT_AVLBL < QNT_PROC_OPR and CAP_OPRS_DLV < CAP_OPRS_PRC
    then PROCESS_MAT;   !otherwise message in dlvr_update when delivering = false;
    inspect OPR_MT4 do
    if RUN_PHASE >= BUSY then                           !go_on was called;
    CAP_OPRS_DLV:= CAP_OPRS_DLV - CAPCTY_ACTL;
    DLR_OPRS:= DLR_OPRS - 1;
    if DLR_OPRS = 0 then begin
        CAP_OPRS_DLV:= 0.0;
        QNT_DLVR_OPR:= 0.0;
        DELIVERING:= DLVRY_SETUP:= false;
        if QUANT_AVLBL < 1.0&-4 then begin              !started without result or processed;
            AVLBL_NEXT:= false; CH_AVLBL;
            STATE_CH_MAT;
        end;
        EVTM_MAX:= Time;
    end;
    if not DLVR_ALLWD then NO_DLVRNG_UNTIL;             !to find time when delivery is accepted;
    ACCEPT_UNTIL;
    if not Idle then reactivate this MATERIAL at Time;
end;
```

NO-DLVRNG-UNTIL is called to find how long delivery remains not allowed (Table 4.24) and ACCEPT-UNTIL is called to adjust event times and to reschedule the material if delivery continues by means of other operations. ACCEPT-UNTIL does not always schedule the material and therefore a call to reactivate is appropriate.

The fourth part of delivery a material concerns some auxiliary procedures. The first one, MAX-QUANT-M- (Table 4.23), is an actual version of a 'virtual' declared procedure calculating the actual maximum quantity and from that whether delivery is allowed or not. If the quantity available and the quantities in the pipeline (QNT-PROC-OPR and QNT-DLVR-OPR; updated in QNT-INTGR-M) do not exceed the current maximum, then the actual maximum becomes equal to QUANT-MX (input value) otherwise it is reduced with a factor, QNT-MX-FACT, to, for instance, 0.5 of QUANT-MX to prevent delivery until the quantity falls below the level of the new actual maximum. The second auxiliary procedure, NO-DLVRNG-UNTIL (Table 4.24) calculates an event time when delivery may start again. If DLVR-ALLWD is true or PROCESSING is not the case, then EVTM-DLVR-OK becomes a moment in the past (i.e. it becomes irrelevant); but

Table 4.23 Procedure MAX-QUANT-M- of class MATERIAL.

```
procedure MAX_QUANT_M_;                                 !----  ----  ----  ----  ----;
begin            !called initial and in delivery_m, consumptn_m, termnt_no_dlvr, reset_;
    QNT_INTGR_M;
    QUANT_MX_ACT:= QUANT_MX
    * (if QUANT_AVLBL - QNT_PROC_OPR + QNT_DLVR_OPR >= QUANT_MX_ACT -1.0&-2
    then QNT_MX_FACT else 1.0);
    DLVR_ALLWD:= QUANT_AVLBL - QNT_PROC_OPR +QNT_DLVR_OPR < QUANT_MX_ACT - 1.0&-2;
    !                                           !accept_until is called later on in oprtn.go_on_;
end;                                            !termnt_no_dlvr is called later on in dynamic;
```

Table 4.24 Procedure NO-DLVRNG-UNTIL of class MATERIAL.

```
procedure NO_DLVRNG_UNTIL;                              !----  ----  ----  ----  ----;
begin                                                   !called in go_until_mp, dlvry_stop, termnt_no_dlvr
   if not DLVR_ALLWD and PROCESSING then begin
      QNT_INTGR_M;
      DURTN_NO_DLVR:=                                   !in days;      !cap_oprs_dlv is 0.0;
      if CAP_OPRS_PRC <= 0.0 then -1.0 else
      (QUANT_AVLBL - QNT_PROC_OPR + QNT_DLVR_OPR - QUANT_MX_ACT) /CAP_OPRS_PRC / 24.0;
      EVTM_DLVR_OK:= Time + DURTN_NO_DLVR +
      (if DURTN_NO_DLVR > -1.0 / 24.0                   !not yet 1 hour from max.;
      then 1.0 / 24.0                                   !add one hour extra;
      else 0.0);
   end
   else EVTM_DLVR_OK:= Time - 1.0;                      !yesterday = irrelevant;
   if EVTM_DLVR_OK > Time and
   (if not Idle then Evtime > Time and EVTM_DLVR_OK < EVTM else true)
   then reactivate this MATERIAL at Time;
end;    !of deriving an event time when delivering will be allowed again;
```

Table 4.25 Procedure QNT-INTGR-M of class MATERIAL.

```
procedure QNT_INTGR_M;                                  !----  ----  ----  ----  ----;
begin                                                   !called in start_prcsng, cap_chng_prcsng/dlv;
   !go_until_mp, accept_until, no_dlvrng_until, delivery_m_, adjust_quant, max_quant_m_;
   ref (RECORD_OPR) REC_OPR_C_D;

   procedure QNT_UPDT_OPR;      !----  ----  ----  ----  ----;
   while REC_OPR_C_D =/= none do inspect REC_OPR_C_D.OPR_MT do begin
      if RUN_PHASE = BUSY then begin
         CAP_QNT_CHNG(1.0);                             !updates quant;
         QNT_M:= QNT_M + QUANT;
      end;
      REC_OPR_C_D:- REC_OPR_C_D.Suc;
   end;

   if LT_INTEGRT < Time then begin
      LT_INTEGRT:= Time;
      if FLD_C =/= none and PROCESSING then begin
         QNT_M:= 0.0;
         REC_OPR_C_D:- OPRTN_PROC_Q.First;
         QNT_UPDT_OPR;
         QNT_PROC_OPR:= QNT_M;
         ATTR_INTG_F_;                                  !updates timeliness losses and costs;
         FLD_C.QNT_IN_PROCS:= QNT_M;
         QNT_M:= 0.0;
      end;
      if FLD_D =/= none  and DELIVERING then begin
         QNT_M:= 0.0;
         REC_OPR_C_D:- OPRTN_DLVR_Q.First;
         QNT_UPDT_OPR;
         QNT_DLVR_OPR:= QNT_M;
         QNT_M:= 0.0;
      end;
   end;
end !! of integration of data for field in process;
```

when processing occurs then a duration is calculated to achieve the exceeded level of the actual maximum by processing the quantity available. However the duration is set to -1.0 if the capacity of processing, CAP-OPRS-PRC is zero or even negative. (This strange situation of zero capacity and PROCESSING true may happen when a started operation processing this material is still in the phase of setting up (travelling to the field) and another operation processing this material stops). The event time is set at a moment at least one hour after the quantity falls below the maximum. The additional one hour is used to prevent a frequent sequence in the program of delivery allowed/not allowed if the actual maximum remains equal to the input value all the time. The scheduling of the material is restricted as in ACCEPT-UNTIL (Table 4.19).

Table 4.25 shows procedure QNT-INTGR-M that integrates quantities processed (or produced) that are in the pipeline but not yet consumed (or delivered) in QNT- PROC-OPR (QNT-DLVR-OPR). Quantities in the pipeline are caused by describing the system as a discrete event system instead of a continuous system! The integration takes place if the latest time of integration is in the past, LT-INTEGRT < Time, and uses procedure QNT-UPDT-OPR to collect QUANT of operations actually processing (or producing). QUANT is updated by calling CAP-QNT-CHNG of an operation. The quantity processed in the pipeline, QNT-PROC-OPR, is used to integrate attributes of a field, for instance, timeliness losses or average moisture content and to assign it to FLD-C's quantity in process.

4.1.3.3 Processing of material

This section on processing of material is divided up in three parts. The first part is related to the start of an operation, the second part to the actual use and the third part to stopping it.

The start of an operation consists of two procedures. PRCSNG-EXPCT (Table 4.26) is called by an operation before its set-up; processing of material is set up by making PRCSS-SETUP true and updating the number of operations, PRC-OPRS; the state of the material changes into RUN. Already at this stage the state is defined as RUN although actual processing has to wait until set-up is completed and may even wait until delivery commences. This however has the advantage that state RUN of a gang, of an operation and of a material starts at the same time. The second procedure, START-PRCSNG (Table 4.27) is called after the set-up of operation is completed; now delivery can occur. If the material was already PROCESSING, then QNT-INTGR-M is called before the capacity of processing, CAP-OPRS-PRC is enlarged and a new event time is calculated when a maximum will be achieved. If processing was not the case, then an expected field (may be the first in the queue) is removed if other fields contain the available quantity (such a removal is not always done in DELIVERY-M-,

Table 4.26 Procedure PRCSNG-EXPCT of class MATERIAL.

```
procedure PRCSNG_EXPCT;                         !----  ----  ----  ----  ----;
begin                                           !called in oprtn.go_ahead;
   PRCSS_SETUP:= true;
   PRC_OPRS:= PRC_OPRS + 1;
   STATE_NEXT:= RUN;
   STATE_CH_MAT;
end;
```

Table 4.27 Procedure START-PRCSNG of class MATERIAL.

```
procedure START_PRCSNG (OPR_MT5);                    !----  ----  ----  ----  ----;
ref (OPRTN_MATRL) OPR_MT5;
begin                                                !called in oprtn.go_on;
    if PROCESSING then QNT_INTGR_M                   !another operation was already started;
    else begin
        PROCESSING:= true;
        if FLD_E =/= none and QUANT_AVLBL > 1.0&-4 then begin
            if FLD_E.QUANTITY < 1.0&-4 then begin
                FLD_E.Out; FLD_E:- none;
            end;
        end;
        FLD_C:- FLD_Q.First;
        while FLD_C.QUANTITY < 1.0&-4 and FLD_C.Suc =/= none do FLD_C:- FLD_C.Suc;
    end;
    CAP_OPRS_PRC:= CAP_OPRS_PRC + OPR_MT5.CAPCTY_ACTL;
    ACCEPT_UNTIL;
end ** of starting an operation processing this material;
```

Table 4.21!); the first field with a quantity is referred to as FLD-C, the field intended for consumption.

The second part of processing is related to the actual consumption of a material. Table 4.28 shows CAP-CHNG-PRCSNG; it integrates the quantities by calling QNT-INTGR-M, updates the total capacity of processing and calls some procedures to update event times. A change of capacity occurs to an operation, for instance, the rate of harvesting occurs when the moisture content of straw changed.

Table 4.28 Procedure CAP-CHNG-PRCSNG of class MATERIAL.

```
procedure CAP_CHNG_PRCSNG (OPR_MT6);                 !----  ----  ----  ----  ----;
ref (OPRTN_MATRL)OPR_MT6;
if PROCESSING then begin                             !called in oprtn.cap_qnt_chng;
    QNT_INTGR_M;
    CAP_OPRS_PRC:= CAP_OPRS_PRC + OPR_MT6.CAP_CHNG;
    GO_UNTIL_MP;
    ACCEPT_UNTIL;
end;
```

Table 4.29 Procedure GO-UNTIL-MP of class MATERIAL.

```
procedure GO_UNTIL_MP;                               !----  ----  ----  ----  ----;
if PROCESSING then
begin                        !called in oprtn.go_on,cap_chng_prcsng, stop_prcsng;
    ref (FIELD) F1;
    QNT_INTGR_M;
    F1:- FLD_C;                                      !skip fld_e:;
    while FLD_AT_TAIL and F1.QUANTITY < 1.0&-4 and F1.Suc =/= none do F1:- F1.Suc;
    while not FLD_AT_TAIL and F1.QUANTITY < 1.0&-4 and F1.Pred =/= none do F1:- F1.Pred;
    DURTN_CONS:= if CAP_OPRS_PRC <> 0.0
    then (F1.QUANTITY - QNT_PROC_OPR) / CAP_OPRS_PRC / 24.0
    else -1.0;                                       !evtm experted from other material;
    EVTM_FLD:= Time + DURTN_CONS;
    if DURTN_CONS > 0.0 then EVTM_FLD:= RMAX (EVTM_FLD,
    if Time + 1.0&-4 > Time then LT_PROC + 1.0&-4 else LT_PROC + 1.0&-3);
    if DURTN_CONS <= 0.0 then EVTM_FLD:= Time - 1.0;    !irrelevant;
    if EVTM_FLD > Time and
    (if not Idle then Evtime > Time and EVTM_FLD < EVTM else true)
    then reactivate this MATERIAL at Time;
    if CAP_OPRS_DLV < CAP_OPRS_PRC and DELIVERING and SUPPLY_NDD
    then DECIDE.DLVRY_SLWR:= true;
    if not DLVR_ALLWD then NO_DLVRNG_UNTIL;           !new event time?;
end ** of go on with processing;
```

GO-UNTIL-MP (Table 4.29) calculates an adjusted event time, the moment when a field is processed completely. Fields without any quantity are omitted. The duration of consumption, DURTN-CONS is calculated for the quantity of the field minus the quantity in the pipeline or is set at a negative irrelevant value if the capacity is still zero (a subsequent recalculation may produce a relevant event time). The event time at which the field is consumed, EVTM-FLD is assigned a moment in the past when the quantity processed already exceeds the quantity at the field (DURTN-CONS <= 0.0) and otherwise a moment in the future with a minimum of 0.0001 or 0.001 day added to the last time processing occurred, LT-PROC in order to avoid rounding off errors (Time is unfortunately not a long real variable and with a day numbering over years Time can exceed 10000 days, so about four significant decimals are left). The material is reactivated to schedule it properly on the time axis with events. The last but one statement let DECIDE know that delivery is slower than processing; this information may be helpful to see from the output when processing capacity is lost due to waiting for delivery. The last statement calls NO-DLVRNG-UNTIL (Table 4.24) to be certain that changes in capacity also affect the event time when delivery may be allowed again.

The remaining of the second part of processing concerns the consumption of material and consists of two procedures both called from an operation when quantities are transferred from one material to another. The first one, ADJUST-QUANT (Table 4.30), integrates quantities and performs four other tasks.

(i) Under the condition that the available quantity is less than the quantity processed (and still in the pipeline), it asks to deliver immediately by calling DELIVER-MAT (Section 4.1.3.4).

(ii) Under the condition that FLD-C refers to a field without a quantity and is not the first or last one in the queue, it transfers the quantity in the pipeline, QNT-IN-PROCS, and removes the empty fields.

(iii) If even after delivery the available quantity remains less than the quantity processed (and in the pipeline), then this difference is added to a cumulative quantity processed in vain (as dummy), QNT-PRC-DUMM, and the quantity processed (FLD-C.QNT-IN-PROCS and QNT-PROC-OPR) is adjusted. If the difference exceeds the quantity processed in 0.001 day and is not due to faster processing than delivering, then a warning message is printed so that the accuracy of the system can be checked.

Exercise: Can you imagine why a message is not send when processing is faster than delivery?

(iv) If the field is consumed (an event occured) and is the only field in the queue and the quantity in the pipeline is not almost the quantity of the field due to rounding off errors, then the quantity in the pipeline is adjusted. The difference is cumulated again in QNT-PRC-DUMM.

Table 4.30 Procedure ADJUST-QUANT of class MATERIAL.

```
procedure ADJUST_QUANT;                                  !----   ----   ----   ----   ----;
begin                                                    !called from oprtn.qnt_transfer;
    QNT_INTGR_M;
    if QUANT_AVLBL < QNT_PROC_OPR - 1.0&-4 and DELIVERING then DELIVER_MAT;

    if PROCESSING and QUANT_AVLBL > 1.0&-4 and FLD_C =/= none then
    !                   !move to field with quantity and remove empty field (fld_e);
    !                   !however no check on readiness of field!!;
    while FLD_C.QUANTITY < 1.0&-4  and
    (if not FLD_AT_TAIL then FLD_C =/= FLD_Q.First else FLD_C =/= FLD_Q.Last) do inspect FLD_C do
    if not FLD_AT_TAIL and Pred=/= none then begin
        Pred qua FIELD.QNT_IN_PROCS:= QNT_IN_PROCS;
        FLD_C:- Pred;
        if FLD_C.Suc =/= FLD_D then FLD_C.Suc.Out;
    end
    else if FLD_AT_TAIL and FLD_C.Suc =/= none then begin
        Suc qua FIELD.QNT_IN_PROCS:= QNT_IN_PROCS;
        FLD_C:- Suc;
        if FLD_C.Pred =/= FLD_D then FLD_C.Pred.Out;
    end;

    !                   !differences due to rounding errors of Time when single precision var.;
    !                   !corrections before delivering + consumption in oprtn.qnt_transfer;
    if QUANT_AVLBL < QNT_PROC_OPR then begin
        QNT_PRC_DUMM:= QNT_PRC_DUMM + QNT_PROC_OPR - QUANT_AVLBL;!cum. differences;
        if QUANT_AVLBL < QNT_PROC_OPR - 24.0&-3 * CAP_OPRS_PRC and
        not (DELIVERING and CAP_OPRS_DLV < CAP_OPRS_PRC          !dlvr slower than prcsng;)
        then begin
            Outimage;
            Outtext ('Warning MATRL.ADJUST_QUANT in Material, at Time:');
            Outtext ('Amount available < Amount processed by operation.'); Outimage;
            Outtext (NAME_COMP); Outfix (Time,6,12); Outfix (QUANT_AVLBL,6,12);
            Outfix (QNT_PROC_OPR,6,12); Outimage;
            Outtext ('Event-time of field passed? Amount in field, Cum. amount not processed.');
            Outimage; Outfix (EVTM_FLD,6,12); Outfix (FLD_C.QUANTITY,6,12);
            Outfix (QNT_PRC_DUMM,6,12); Outimage;
        end;
        if QNT_TRF_FLD =/= none and not (DELIVERING and CAP_OPRS_DLV < CAP_OPRS_PRC     )
        and QUANT_AVLBL < QNT_PROC_OPR - 1.0&-4 then
        inspect QNT_TRF_FLD do begin
            Outimage;
            Outtext ('Warning MATRL.ADJUST_QUANT in Material, at Time:');
            Outtext ('Amount available < Amount processed by operation.'); Outimage;
            Outtext (NAME_COMP); Outfix (Time,6,12); Outfix (QUANT_AVLBL,6,12);
            Outfix (QNT_PROC_OPR,6,12); Outimage;
            Outtext ('Event-time of field passed? Amount in field, Cum. amount not processed.');
            Outimage; Outfix (EVTM_FLD,6,12); Outfix (FLD_C.QUANTITY,6,12);
            Outfix (QNT_PRC_DUMM,6,12); Outimage;
        end;
        FLD_C.QNT_IN_PROCS:= QNT_PROC_OPR:= QUANT_AVLBL;          !adjusted to avlbl quantity;
    end;

    if Abs(EVTM_FLD - Time) < 1.0&-4 and PROCESSING and FLD_Q.First == FLD_Q.Last
    then inspect FLD_C do                                  !length = 1;
    if Abs (QNT_IN_PROCS - QUANTITY) < 24.0&-3 * CAP_OPRS_PRC then begin
        QNT_PRC_DUMM:= QNT_PRC_DUMM + QNT_IN_PROCS - QUANTITY;  !cum. differences;
        QNT_PROC_OPR:= QNT_IN_PROCS:= QUANTITY;          !adjusted to quantity of field;
    end;        !else more fields with quantity or already adjusted;
end ** of quantity adjustment;
```

The second procedure in actual processing is shown in Table 4.31; CONSUMPTN-M is called by the processing operation that requires a transfer of this material to the material(s) produced. The quantity to consume is known by QUANT in the operation. The following statements deal with:

(i) The consumption of quantities on successive fields; it is performed field by field (FLD-C); a 'virtual' procedure ATTR-INTG-M- (still empty in the base model) can integrate costs of material that are independent of field properties varying with time; CONSUMPT-F of FLD-C is called (Table 4.10) to consume the quantity and to transfer a remaining quantity processed (in the pipeline) to another field; some output is given about the field and the quantity on a printfile, QNT-TRF-FLD if it exists; when the current field is exhausted then FLD-C refers to the first field of the queue or if needed to a subsequent field containing a quantity and a pipeline quantity; an empty field may be removed from the queue; if a pipeline quantity is still not met in a field the search is continued; an error message occurs when no field is found.

(ii) The decrease of the quantity available, QUANT-AVLBL, the increase

Table 4.31 Procedure CONSUMPTN-M of class MATERIAL.

```
procedure CONSUMPTN_M (OPR_MT8);                  !----   ----    ----    ----    ----;
ref (OPRTN_MATRL) OPR_MT8;
begin                                             !called in oprtn.qnt_transfer;
    real QNT1,QNT2;
    ref (FIELD) F2;

    QNT1:= QNT_M:= OPR_MT8.QUANT;
    F2:- FLD_C;
    while QNT1 >= 1.0&-4 and
    (if FLD_C =/= none then FLD_C.QUANTITY > 0.0 else false) do
    begin                                         !some quantity processed may be lossed;
        QNT2:= RMIN (QNT1, FLD_C.QUANTITY);       !qnt2 <= fld_c.quantity;
        ATTR_INTG_M_ (QNT2);                      !updates costs of material;
        FLD_C.CONSUMPT_F (QNT2);
        inspect QNT_TRF_FLD do begin
            Setpos(M_POS);
            Outtext(FLD_C.FLD_SQNO_TXT);
            Outtext(' -');
            Outfix(QNT2,2,5);
            if FLD_C.QUANTITY < 1.0&-4 then Outtext('eF');
            if QNT1 > QNT2 + 1.0&-4 then Outimage;
        end;
        QNT1:= QNT1 - QNT2;
        if FLD_C.QUANTITY < 1.0&-4 and FLD_C.QNT_IN_PROCS < 1.0&-4 then FLD_C.Out;
        if FLD_C.QUANTITY < 1.0&-4 then FLD_C:- FLD_Q.First;    !field may be fld_e or none;
        if FLD_C =/= none then                    !adjust to field with quantity;
        while FLD_C.QUANTITY < 1.0&-4 and FLD_C.Suc =/= none do begin
            if FLD_C.QNT_IN_PROCS > 1.0&-4
            then FLD_C.Suc qua FIELD.QNT_IN_PROCS:= FLD_C.QNT_IN_PROCS;
            FLD_C:- FLD_C.Suc;
            FLD_C.Pred.Out;                       !remove empty field;
        end;
        if QNT1 > 1.0&-4 and FLD_C =/= none then!adjust to field with qnt_in_procs;
        while FLD_C.QNT_IN_PROCS < 1.0&-4 and FLD_C.Suc =/= none do FLD_C:- FLD_C.Suc;
        if QNT1 > 1.0&-4 and FLD_C == none then begin
            Outtext ('Error MATRL.CONSUMPTN_M in Material, at Time,');
            Outtext (' Amount processed is not available!'); Outimage;
            Outtext (NAME_COMP); Outfix (Time,6,12); Outfix (QNT1,6,12);       !may not occur;
            Outimage;
            QNT_M:= QNT_M - QNT1; QNT1:= 0.0;
        end;
    end;

    QNT_PROC_OPR:= QNT_PROC_OPR - QNT_M;
    QUANT_AVLBL:= QUANT_AVLBL - QNT_M;
    QUANT_PROCSD:= QUANT_PROCSD + QNT_M;
    QNT_M:= 0.0;
```

Table 4.31 (continued)

```
    if (if FLD_C == none then true
    !;                    else FLD_C == FLD_E and not DLVRY_SETUP
    !;                         and QUANT_AVLBL < 1.0&-3 and FLD_C.QNT_IN_PROCS < 1.0&-4)
    and QUANT_AVLBL > 0.0 then begin
        QNT_DLTD:= QNT_DLTD + QUANT_AVLBL;
        if QUANT_AVLBL > 1.0&-2 then begin
            Outimage;
            Outtext ('Warning MATRL.CONSUMPTN_M in Material, at Time,');
            Outtext (' Available amount deleted, Cum. deleted amount'); Outimage;
            Outtext (NAME_COMP); Outfix (Time,6,12); Outfix (QUANT_AVLBL,6,12);
            Outfix (QNT_DLTD,6,12); Outimage;
        end;
        inspect  QNT_TRF_FLD do begin
            Outimage;
            Outtext ('Warning MATRL.CONSUMPTN_M in Material, at Time,');
            Outtext (' Available amount deleted, Cum. deleted amount'); Outimage;
            Outtext (NAME_COMP); Outfix (Time,6,12); Outfix (QUANT_AVLBL,6,12);
            Outfix (QNT_DLTD,6,12); Outimage;
        end;
        QUANT_AVLBL:= 0.0;
    end;

    if QUANT_AVLBL < 1.0&-4 and DELIVERING then DELIVER_MAT;
    if QUANT_AVLBL < 1.0&-4 then begin
        QUANT_AVLBL:= 0.0;
        SUPPLY_NDD:= true;
        if not DLVRY_SETUP then begin          !else adjusted in dlvry_stop;
            AVLBL_NEXT:= false;
            CH_AVLBL;
            DECIDE.END_FLDS:= true;            !mat_down follows in state_ch_mat;
        end;
        if not AVLBL or (FLD_C == none and not DLVRY_SETUP) then begin
            FLD_Q.Clear;
            FLD_EXPCT_M_; FLD_E:- FLD_C:- FLD_D;!make an experted field;
            ADMN.DISPLAY_DATA_; !updates display data before states are changed;
        end;
    end;
    if FLD_AT_TAIL then FLD_D.Into (FLD_Q) else FLD_D.Follow (FLD_Q);
    !                   !place fld_d in queue when removed with fld_c or not;
    if F2 =/= FLD_C then ATTR_FLD_M_;
    CH_READY;                                  !ready may change with next field;
    STATE_CH_MAT;
    MAX_QUANT_M_;
end;
```

of the quantity processed, QUANT-PROCSD, and the decrease of the quantity in the pipeline, QNT-PROC-OPR (it needs not become zero when another operation has still to transfer processed material).

(iii) If some quantity remains available without a field, then the quantity is deleted (accumulated in QNT-DLTD) and a message is send if it exceeded 0.01 [ha]; the quantity is also deleted if FLD-C refers to the expected field, FLD-E, the quantities are small and no delivery is expected; the latter is performed especially to make QUANT-AVLBL zero regularly to prevent accuracy errors.

(iv) If hardly any quantity is available then delivery is appropriate otherwise supply is needed, AVLBL is adjusted and DECIDE receives a message that the fields are exhausted, END-FLDS, and no delivery is expected (later in STATE-CH-MAT the material becomes DOWN, resulting in activation of DECIDE and finally in stopping of processing of

Table 4.32 Procedure STOP-PRCSNG of class MATERIAL.

```
procedure STOP_PRLSNG (OPR_MT7);                |----  ----  ----  ----  ----;
ref (OPRTN_MATRL) OPR_MT7;                      |called in oprtn.termnt;
begin
    inspect OPR_MT7 do
    if RUN_PHASE >= BUSY then                   |go_on was called;
    CAP_OPRS_PRC:= CAP_OPRS_PRC - CAPCTY_ACTL;
    PRC_OPRS:= PRC_OPRS - 1;
    if PRC_OPRS = 0 then begin
        PROCESSING:= PRCSS_SETUP:= false;
        QNT_PROC_OPR:= 0.0; CAP_OPRS_PRC:= 0.0;
        if STATE = RUN then STATE_NEXT:= PASSIVE;|stop not caused by material;
        STATE_CH_MAT;
        EVTM_FLD:= Time -1.0;
    end;
    GO_UNTIL_MP;                                |continue with remaining operations;
    ACCEPT_UNTIL;
    if not Idle then reactivate this MATERIAL at Time;
end;
```

this material), a new expected field may be required.

(v) FLD-D is placed in the queue when it was removed with the above manipulations with FLD-C; it is not removed, even if it is empty, because delivery may occur or it can be FLD-E;

(vi) If the original field consumed, F2, is not the same as FLD-C then FLD-C may have other properties and procedure ATTR-FLD-M- updates them with an influence on, for instance, rate of processing (the 'virtual' declared procedure ATTR-FLD-M- is not defined in the base model).

(vii) Ready for processing is updated as well as the state of the material and the actual maximum quantity.

The third part of processing concerns the termination of an operation, the stopping of processing this material at least partly is shown in STOP-PRCSNG, Table 4.32. It reduces the capacity of processing if needed (the operation in busy) and the number of operations. If this number is zero PROCESSING becomes false; zeros are assigned, the state is changed, the event time for a field adjusted. Event times are adjusted and the material activated to reschedule itself on the time axis (because a rescheduling by GO-UNTIL-MP or ACCEPT-UNTIL is not certain).

4.1.3.4 *Dynamics*

This section describes the dynamic section and the related procedures. Table 4.33 shows the dynamic section that has to contain a call to one of the activation or deactivation procedures (such as activate, passivate, hold). A permanent cycle is created by using: 'while true do begin ..... end;'. In this case the dynamic section is controlled by END-EXPRMNT; the cycle ends when the experiment is ended. The first part of Table 4.33 shows four 'if-statements' each calling a procedure if the current time is close to (< 0.00001 day or 0.0144 min) an event time defined in the material. The second part calculates the minimum of the event times and schedules the

Table 4.33 Dynamic section of class MATERIAL.

```
! d y n a m i c;

while not END_EXPRMNT do begin
    if Abs (EVTM_MAX - Time) <= 1.0&-5 and DELIVERING then DELIVER_MAT;
    !                                          !due to maximum quantity achieved;
    if Abs (EVTM_FLD - Time) <= 1.0&-5 and PROCESSING then PROCESS_MAT;
    !                                          !due to field completion;
    if Abs (EVTM_DLVR_OK - Time ) <= 1.0&-5 then TERMNT_NO_DLVR;
    !                                          !due to actual quantity <= max;
    if Abs (EVTM_DVLPMNT - Time ) < 1.0&-5 then MAT_DVLPMNT_;
    !                                          !due to autonomous development;
    EVTM:= RMIN (
    (if  PROCESSING and EVTM_FLD > Time + 1.0&-5 then EVTM_FLD else 1.0&+9),
    (if DELIVERING and EVTM_MAX > Time + 1.0&-5 then EVTM_MAX else 1.0&+9)      );
    EVTM:= RMIN (EVTM, RMIN (
    (if EVTM_DLVR_OK > Time + 1.0&-5 then EVTM_DLVR_OK else 1.0&+9),
    (if EVTM_DVLPMNT > Time + 1.0&-5 then EVTM_DVLPMNT else 1.0&+9)      )      );
    LOOP_DYN:= if Time < EVTM then 0 else LOOP_DYN + 1;
    if LOOP_DYN >= 10 then begin
        Outtext ('The system arrived at a loop in the dynamic section of material: ');
        Outtext (NAME_COMP); Outimage;
        Outtext (' at time:'); Outfix (Time,6,12);
        Outtext ('; A  runtime error is forced to enter simddt.'); Outimage;
        Outtext ('You may continue by giving: 'input evtm:= x', where x = time + y');
        Outtext (' and y >0  (e.g. 0.001 day).'); Outimage;
        LOOP_DYN:= LOOP_DYN / 0;                    !causes a runtime error;
        !               !You can continue in simddt by giving: input evtm:= x ,(x > time!);
    end;
    if EVTM > Time + 1.0&-5 and EVTM < 1.0&+8
    then
    reactivate this MATERIAL at EVTM
    else Passivate;     !reactivated in go_until_mp, accept_until, no_dlvrng_until;
end ** of while dynamic;                   ! and in mat_dvlpmnt_, dlvry_stop, stop_prcsng;
```

material at that minimum or removes it from the time axis by calling Passivate. A counter, LOOP-DYN is used to detect an unexpected situation; the debugging system SIMDDT is entered by forcing a division by zero.

The first procedure mentioned in the dynamic section is DELIVER-MAT, Table 4.34. It only works if the last time of delivery, LT-DLVR, is in the past and DELIVERING is true. All the operations delivering this material (in OPRTN-DLVR-Q) are requested to transfer the quantity processed by calling UPDAT-QNT (Table 4.90). If such an operation is busy with processing, it transfers the quantity immediately and that may result for this material in achieving the maximum quantity and no allowance to deliver more i.e. DLVR-ALLWD becomes false. After the requested transfer of material DELIVER-MAT continues with either to call CONTINUE in all the operations delivering (i.e. continue with processing; Table 4.90) or to require a new decision by activating DECIDE after giving it a signal that the maximum quantity is achieved.

The second procedure called in dynamic is PROCESS-MAT, Table 4.34. It has a similar structure as DELIVER-MAT. It requests the transfer of the quantity processed from all the operations processing this material (in OPRTN-PROC-Q) by calling also UPDAT-QNT. It continues with processing by calling CONTINUE or with warning the operations delivering that their material produced is required, MAT-PRD-RQRD: = true.

Table 4.34 Procedures DELIVER-MAT and PROCESS-MAT of class MATERIAL.

```
procedure DELIVER_MAT;                          !---   ----   ----   ----   ----;
!                 !called in dyn., adjust_quant, urg_mat_prc_, consumptn_m;
if LT_DLVR < Time and DELIVERING then begin
    ref (RECORD_OPR) REC_OPR_DLV;
    LT_DLVR:= Time;
    REC_OPR_DLV:- OPRTN_DLVR_Q.First;
    while REC_OPR_DLV =/= none do begin
        REC_OPR_DLV.OPR_MT.UPDAT_QNT;
        REC_OPR_DLV:- REC_OPR_DLV.Suc;
    end;
    if DLVR_ALLWD and DELIVERING then begin
        REC_OPR_DLV:- OPRTN_DLVR_Q.First;
        while REC_OPR_DLV =/= none do begin
            REC_OPR_DLV.OPR_MT.CONTINUE;
            REC_OPR_DLV:- REC_OPR_DLV.Suc;
        end;
    end
    else if not DLVR_ALLWD then begin           !stop producing by activating decide;
        DECIDE.MAT_MAX_QNT:= true;
        activate DECIDE at Time;                 !calls dlvry_stop and no_dlvrng_until;
    end;
end;

procedure PROCESS_MAT;                          !----   ----   ----   ----   ----;
!       !called in dynamic, attr_updt_m_, urg_mat_prc_, delivery_m_, shift_chnge_, dlvry_stop;
if LT_PROC < Time and PROCESSING then begin
    ref (RECORD_OPR) REC_OPR_CNS3, REC_OPR_DLV1;
    LT_PROC:= Time;
    REC_OPR_CNS3:- OPRTN_PROC_Q.First;
    while REC_OPR_CNS3 =/= none do begin
        REC_OPR_CNS3.OPR_MT.UPDAT_QNT;
        REC_OPR_CNS3:- REC_OPR_CNS3.Suc;
    end;
    if (AVLBL or DLVRY_SETUP) and STATE <> DOWN !down may be due to next field not ready;
    and PROCESSING then begin
        REC_OPR_DLV1:- OPRTN_DLVR_Q.First;
        if not SUPPLY_NDD or DELIVERING then begin
            REC_OPR_CNS3:- OPRTN_PROC_Q.First;
            while REC_OPR_CNS3 =/= none do begin
                REC_OPR_CNS3.OPR_MT.CONTINUE;
                REC_OPR_CNS3:- REC_OPR_CNS3.Suc;
            end;
        end
        else                                     !material exhausted;
        while REC_OPR_DLV1 =/= none do inspect REC_OPR_DLV1.OPR_MT do begin
            if STATE = RUN then MAT_PRD_RQRD:= true;    !state of operation!!;
            REC_OPR_DLV1:- REC_OPR_DLV1.Suc;
        end;
    end; !else state changed into \processing or  \ready or (\available and \dlvry_setup);
    !                              ... decide will be activated in state_ch_mat;
end;
```

Exercise: Do you remember the example mentioned in Section 4.1.3.2 (Table 4.18)? Can you give the conditions of the baler operation, bales in the field and the gathering operation to meet a similar situation of waiting for a material?

The scheme in Table 4.35 shows some links in a sequence of operations and materials. The left operation is delivering the material and the right one processes it. DELIVER-MAT calls UPDAT-QNT and CONTINUE from the left operation and PROCESS-MAT from the right one (that also may set MAT-PRD-RQRD in the left one). The right operation on its turn can be referred to as a 'delivering operation' when seen from a subsequent material.

Table 4.35 The relation from a material to operations by procedures.

```
--> operation  --> material  --> operation  -->
    delivering                   processing
    ----------                   ----------

    UPDAT_QNT (called in  )      UPDAT_QNT (called in  )
              (           )                (           )
    CONTINUE  (DELIVER_MAT)      CONTINUE  (PROCESS_MAT)

    MAT_PRD_RQRD (Variable of operation    )
                 (set to true in PROCESS_MAT)
```

Table 4.36 Procedure TERMNT-NO-DLVR of class MATERIAL.

```
procedure TERMNT_NO_DLVR;                   !----  ----  ----  ----  ----;
begin                                       !called in dynamic;
    MAX_QUANT_M_;                           !may change dlvr_allwd;
    if DLVR_ALLWD then begin
        DECIDE.MAT_DLV_OK:= true;
        activate DECIDE at Time;
        EVTM_DLVR_OK:= Time - 1.0;          !yesterday = irrelevant;
    end
    else NO_DLVRNG_UNTIL;                   !revise event time;
end;
```

The third procedure is TERMNT-NO-DLVR, Table 4.36; it terminates the situation of no delivery at the event time the processing is proceded far enough to decrease the available quantity to a level below the maximum; this results in DLVR-ALLWD = true, a signal to DECIDE and an activation to consider a decision.

The fourth procedure MAT-DVLPMNT- is a 'virtual' one and still empty; it is intended to describe the development of material properties over time as caused by autonomous, continuous processes such as growth, disease effects and moisture movements.

The following sections describe the subclasses of material as shown in the following scheme:

```
MATERIAL ---- PROCESSED-MAT ---- INITIAL-MAT (Section 4.1.5)
          |   (Section 4.1.4) |
          |                   └─ INTERMDT-MAT (Section 4.1.6)
          └─FINAL-MAT (Section 4.1.7)
```

The materials in the system that are present at the beginning of a season are called INITIAL-MAT and form a subclass of PROCESSED-MAT, a material processed. An intermediate material has initially no quantity; the quantity is delivered and will be processed later on, so INTERMDT-MAT is also a subclass of PROCESSED-MAT. A final material has initially no quantity; the quantity is delivered but not processed, so FINAL-MAT is a subclass of MATERIAL.

4.1.4 *Materials processed*

Materials that are processed have additional attributes related to the properties controlling the workability and to the urgency of processing. Table 4.37 shows the class declaration with the prefix MATERIAL, some parameters and the virtual procedures. Table 4.38 shows the declaration of the variables; the commentary in the program and the use of the variables in the procedures will clarify their meaning.

Procedure SHIFT-CHNGE- is called in the shifts controlling the availability of objects over periods and within a week. It is a redefinition of the virtual procedure programmed in the superclass COMPONENT (Table 4.3). Table 4.39 shows the procedure: data are updated by calling TIME-C-ACCUM (Table 4.3); PROCESS-MAT (Table 4.34) is called to update the quantities; M-ADMN refers to a class recording data of the material (the base model defines only virtual procedures, Table 7.8); if the availability of the component changes then the state is changed and a shift controlling the calculation of the urgency is activated; at the end of a period some output of the situation occurs and if no shift for the urgency calculation exists, then the procedures URG-MAT-PRC- and URG-MAT-EXT- are called to force at least one calculation per period; finally the current costs category from SH-WK is assigned to CTGR. SH-WK is not used to control AVLB-COMP (as in Table 4.3) but only to record material properties per category of costs, for instance, the number of hours the material was workable during regular time or overtime. SH-PRD is obligatory for a processed material.

The timeliness function is essential for processed materials; it controls the urgency of operations. How this is done will be shown in the following procedures; the theoretical background is described by Elderen (1977). Procedure TML-FNCT-FRC (Table 4.40) calculates the fraction of the timeliness function on a date relative to the date when the optimum yield is achieved. A relative date outside the range of dates results in a fraction valid at the first or last date of the range; a relative date within the range results in a fraction interpolated linearly between the fractions of the adjacent dates. The timeliness function is given in an array TIMELINESS and the related dates in TML-DATE (a series of non decreasing integers; unequal intervals); TML-CNDTN defines the current condition when more timeliness

Table 4.37 Class PROCESSED-MAT; its parameters and virtual procedures.

```
MATERIAL class PROCESSED_MAT                      !****************************************;
(SH_URGN, PROC_CNDTNS, TML_FNCTNS, TML_DATES, CNDTNS_W_T, DAYS_WT_DATA);
ref (SHIFT_URG) SH_URGN;      !object calculating urgency of material;
integer
PROC_CNDTNS,                  !number of (exclusive) conditions in processing;
TML_FNCTNS,                   !number of timeliness functions;
TML_DATES,                    !number of dates with timeliness data given;
CNDTNS_W_T,                   !number of conditions and;
DAYS_WT_DATA;                 !days for which expected workable time is given;
virtual: procedure URG_MAT_PRC_, URG_MAT_EXT_, RESET_M_, DELIVERY_INIT_, DELVR_FLD_INIT_;
begin
```

Table 4.38 Declaration of variables of class PROCESSED-MAT.

```
! d e c l a r a t i o n;

boolean                          !----------------------;
DISURG_USED,              !disurgency to deliver this material;
M_PRC_CND_CH;             !true if a processing condition changed;
real                             !----------------------;
COSTS_TMLNSS,             !cum. costs of timeliness losses (quantity * price);
COSTS_EXC_CP,             !cum. additional costs due to use of excess capacity of gang;
COSTS_EXPCTD,             !due to proc. mat. making avoidable extra costs e.g.drying[fl/ha];
FINSH_MAT,                !expected finishing time of processing;
PRICE_HA,                 !expected price per ha;
TML_VALUE,                !timeliness value of first field (fraction * price/ha);
TML_VL_PREV,              !timeliness value of first field (fraction * price/ha);
TML_LOSS_ALT,             !alternative timeliness;
QNT_ALT_FLD,              !area of field to calculate tml_loss_alt;
QNT_DAY_PR_M,             !area processed until previous day (incl.);
T_WRKBL,                  !long term expectation of workable time in hours per day;
WT_MULT,                  !multiplier to adjust a) an average to a more restrictive workable;
!                         !time (<= 1.0) b) 8-18 to 8-22 o'clock, c) dry to wet grain (>=1.0);
!                         ! d) to tune the urgency to an optimal level;
LOSS_MULT,                !multiplier of timeliness loss:1.0 and 0.0 when
!                         timeliness function is used or not to calculate losses;
CAP_RTIO_PRV,             !previous capacity before change of properties;
URGENCY_PROC, URG_PROC_T,        !urgency of processing or consumption of material;
URG_PROC_ALT,                    !urgency used as minimum;
DISURGNC_DEL, DISURG_DEL_T,      !disurgency of delivering or producing this material;
PROC_PROPRTY;                    !property of material that controls workability (weather);
integer array                    !----------------------;
MAT_DLV_CNDTN [IMIN (1,PROC_CNDTNS):PROC_CNDTNS],
!                                !material delivered if processing condition is valid;
TML_DATE [IMIN(1,TML_DATES):TML_DATES];
!                                !timeliness dates relative to 0.0 , the date with max.yield;
real array                       !----------------------;
TIMELINESS [IMIN(1,TML_FNCTNS):TML_FNCTNS , IMIN(1,TML_DATES):TML_DATES],
!                                !timeliness yield in fraction of max. yield for two conditions;
PRC_CNDTN_LB, PRC_CNDTN_UB [IMIN (1,PROC_CNDTNS):PROC_CNDTNS],
!                                !lower and upper bound of excl.ranges of processing conditions;
T_FRAC_PROC [1:SH_PRD.PERDS + 1],
!                                !fraction of workable time intended for processing;
T_WRKBL_EXP [IMIN(1,CNDTNS_W_T):CNDTNS_W_T , 0:DAYS_WT_DATA];
!                                !time in [h] expected as workable, condition i, days 1 to j;
boolean array                    !----------------------;
M_PRC_CNDTN, M_PRC_CND_PRV [IMIN (1,PROC_CNDTNS):PROC_CNDTNS];
text TAIL;                       !----------------------;
integer                          !----------------------;
PERD_MAT, I_M2,                  !current period number from sh_prd;
TML_CNDTN,                       !condition for one of the  timeliness functions ;
CNDTN_W_T,                       !condition for expected workable time;
MAT_DLVR, MAT_DLVR_PRV;          !materials delivered depending on processing conditions;
ref (RECORD_OPR)                 !----------------------;
REC_OPR_C_D1;
```

functions are used, for instance, one function before a disease, a certain climate, a date and another afterwards. An example of the timeliness functions of wheat is shown in Table 4.41. Fig. 2.1 illustrates such a function.

The calculation of the expected timeliness loss is shown in Table 4.42. The preliminary calculations concern the current type of day (Monday, etc.) DAY-TP--M, an initial loss TML-LOSS of -100, the days needed to finish the consumption of fields, the current period PRD, the expected workable time per day T-WRK-DAY, the duration of the workable time already used at the current moment DUR-WRK-USED and an adjustment of a field reference FLD to the first field with a quantity. The calculations for the exist-

Table 4.39 Procedure SHIFT-CHNGE- of class PROCESSED-MAT.

```
procedure SHIFT_CHNGE_;                           !----  ----  ----  ----  ----;
begin                                             !called in sh_prd, sh_wk;
    TIME_C_ACCUM;
    if PROCESSING then PROCESS_MAT;
    inspect M_ADMN do if AVLB_COMP then ADMN_M_UPDATE_
    else ADMN_M_LT_;                              !updates all lt_... to time;
    AVLB_COMP_PR:= AVLB_COMP;
    AVLB_COMP:= SH_PRD.AVLB_PRD;                  !influence from sh_wk not involved!!;
    if  not (AVLB_COMP_PR eqv AVLB_COMP) then begin
        STATE_CH_MAT;
        if AVLB_COMP and SH_URGN =/= none then activate SH_URGN at Time;
    end;
    PERD_MAT:= SH_PRD.PERD;
    if Abs (TIME_YR - SH_PRD.PERD_END [SH_PRD.PERD - 1] ) < 1.0&-3 then begin
        if not END_SEASON and AVLB_COMP_PR then ADMN.PERD_OUT_M_ (this PROCESSED_MAT);
        if SH_URGN == none then URG_MAT_PRC_;    !not controlled by sh_urgn then calculation;
        if SH_URGN == none then URG_MAT_EXT_;    !       of urgency once per period;
    end;
    if SH_WK =/= none then CTGR:= SH_WK.COST_CTGR;      !used in recording only;
end;
```

Table 4.40 Procedure TML-FNCT-FRC- of class PROCESSED-MAT.

```
real procedure TML_FNCT_FRC_ (DATE_RELTV);       !----  ----  ----  ----  ----;
!                         !called in tml_loss_exp_, urg_mat_prc, attr_intg_f_, delvr_fld_m_;
real DATE_RELTV;          !timeliness fraction at date relative to date with max.yield;
begin
    real        TML,DATE_REM;   !date remainder = date_reltv - timeliness date [i];

    DATE_NO:= 2;
    if DATE_RELTV <= TML_DATE [1]                 !..d...<=..1...2...3..---...tmls...;
    then TML_FNCT_FRC_:= TIMELINESS [TML_CNDTN, 1]
    else if DATE_RELTV >= TML_DATE [TML_DATES]    !...1...2...3...---...tmls...<=d..;
    then TML_FNCT_FRC_:= TIMELINESS [TML_CNDTN, TML_DATES]
    else begin                                    !...1...---...<d<...---...tmls...;
        while DATE_NO < TML_DATES and TML_DATE [DATE_NO] < DATE_RELTV do
        DATE_NO:= DATE_NO + 1;
        DATE_REM:= DATE_RELTV - TML_DATE [DATE_NO-1];
        TML:= TIMELINESS [TML_CNDTN,DATE_NO-1];
        TML_FNCT_FRC_:= TML + (TIMELINESS [TML_CNDTN,DATE_NO] - TML) /
        (TML_DATE [DATE_NO] - TML_DATE [DATE_NO-1]) * DATE_REM;
        !(tml_date[date_no] - tml_date[date_no-1])  >  0.0? also if tml_date[i]= ..[i+1]!;
    end;
end ** of timeliness fraction at a date;
```

Table 4.41 Two timeliness functions of wheat.

-3	-2	0	8	16	40	100	dates relative to optimum (0)
0.15	0.93	1.0	0.98	0.96	0.84	0.0	timeliness fractions without sprouting
0.15	0.93	1.0	0.98	0.56	0.44	0.0	tiemliness fractions with sprouting cereal

ing fields concern: a duration of processing DUR-FLD-PROC is derived; the number of days FLD-FNSH-DAY needed to finish the processing of this and preceding fields in the queue (along with skipping weekend days and adjusting the duration when a next period is entered); the start FLD-START-T and finish FLD-FNSH-T moments of the field; the timeliness loss on that field TML-LOSS-FLD in ha for one hour of delay in processing (calling TML-FNCT-FRC for the start and finish moments); the cumulative

Table 4.42 Procedure TML-LOSS-EXP- of class PROCESSED-MAT.

```
real procedure TML_LOSS_EXP_;                    !----  ----  ----  ----  ----;
begin                                             !called in urg_mat_prc_;
    integer
    PRD,FLD_FNSH_DAY,DAY_TP__M,DAY_TP_FNSH;
    real
    DUR_REMNNG,DUR_WRK_USED,T_WRK_DAY,
    FLD_FINSH_T,FLD_START_T,DUR_FLD_PROC,
    TML_LOSS_FLD,TML_LOSS,TML_LOSS_SUM;

    DAY_TP__M:= Mod (DAY_TP_1JAN + Entier(TIME_YR + 1.0&-4) - 1, 7) + 1;
    TML_LOSS:= -1.0&2;
    FLD_FINSH_T:= RMAX(TIME_YR, Entier(TIME_YR+1.0&-4) + DAY_BGN/24.0);
    FLD_FNSH_DAY:= 1;                                !days needed to finish fields;
    PRD:= PERD_MAT;
    T_WRK_DAY:= WT_MULT *                            !workable time per day;
    (if FLD_FNSH_DAY <= DAYS_WT_DATA then T_WRKBL_EXP [CNDTN_W_T,1] else T_WRKBL);
    DUR_WRK_USED:= DAY_FRAC_NOW * T_WRK_DAY;         !duration used of workable time;
    FLD:- FLD_Q.First;
    if FLD =/= none then begin
        if FLD.QUANTITY < 1.0&-4 then FLD:- FLD.Suc;  !skips expected field without quantity;
    end;
    while FLD =/= none do begin
        DUR_FLD_PROC:=                               !duration of processing;
        FLD.QUANTITY / CAP_PROC / T_FRAC_PROC [PRD];
        DUR_REMNNG:= DUR_WRK_USED:= DUR_WRK_USED + DUR_FLD_PROC;
        while DUR_REMNNG > 0.0 do begin
            DAY_TP_FNSH:= Mod(DAY_TP__M - 1 + FLD_FNSH_DAY - 1,7) + 1;
            if DAY_TP_FNSH = SAT then FLD_FNSH_DAY:= FLD_FNSH_DAY + 2
            else       if DAY_TP_FNSH = SUN then FLD_FNSH_DAY:= FLD_FNSH_DAY + 1;
            !                                        !add 2 or 1 to skip weekend days;
            T_WRK_DAY:= WT_MULT *                    !expected workable time per day;
            (if FLD_FNSH_DAY <= DAYS_WT_DATA then T_WRKBL_EXP [CNDTN_W_T,FLD_FNSH_DAY]
            else T_WRKBL);
            DUR_REMNNG:= DUR_REMNNG - T_WRK_DAY;
            if DUR_REMNNG > 0.0 then begin
                FLD_FNSH_DAY:= FLD_FNSH_DAY + 1;
                if Entier(TIME_YR + 1.0&-4) + FLD_FNSH_DAY > SH_PRD.PERD_END [PRD]
                and PRD < SH_PRD.PERDS + 1
                then begin
                    PRD:= PRD + 1;                   !adjusts to following period;
                    DUR_REMNNG:= DUR_REMNNG *
                    T_FRAC_PROC [PRD - 1] / T_FRAC_PROC [PRD];
                end;
                DUR_WRK_USED:= DUR_REMNNG;
            end ** of adjusting field finish day;
        end ** of while: fld_fnsh_day is found;
        FLD_START_T:= FLD_FINSH_T;                   !are equal if fld.quantity=0.;
        FLD_FINSH_T:= Entier (TIME_YR + 1.0&-4) + DAY_BGN / 24.0 + FLD_FNSH_DAY - 1 +
        (DAY_END - DAY_BGN)/24.0 * DUR_WRK_USED / RMAX (T_WRK_DAY, 0.01);
        !                                            !t_wrk_day>0.0 for mat. processed;
        TML_LOSS_FLD:= (TML_FNCT_FRC_ (FLD_START_T - FLD.DATE_OPT_YLD) -
        !;              TML_FNCT_FRC_ (FLD_FINSH_T - FLD.DATE_OPT_YLD) ) * CAP_PROC;
        !                                            ![ha/h]=fraction*ha/h of delay;
        TML_LOSS_SUM:= TML_LOSS_SUM + TML_LOSS_FLD;
        TML_LOSS:= RMAX (TML_LOSS, TML_LOSS_SUM);    !non decreasing value over all fields;
        FLD:- FLD.Suc;
    end xx of field;
    FINSH_MAT:= FLD_FINSH_T;                         !time last field finished;
    TML_LOSS_ALT:= (TML_FNCT_FRC_ (0.0) -            !lower bound of timeliness loss;
    TML_FNCT_FRC_ (QNT_ALT_FLD / (CAP_PROC * T_WRKBL * WT_MULT * T_FRAC_PROC [PERD_MAT] ) )
    ) * CAP_PROC;
    TML_LOSS_EXP_:= RMAX (TML_LOSS, 0.0);            !loss in ha/h may be more than 1;
end ** of expected timeliness loss of available fields;
```

loss TML-LOSS-SUM over the fields and a non decreasing loss TML-LOSS. Finally the following calculations are executed: the moment of finishing the material FINSH-MAT is recorded; an alternative timeliness loss TML-

LOSS-ALT (used as a lower limit, see URG-MAT-EXT-, Table 4.44) is derived for an imaginary field (QNT-ALT-FLD acreage) that has its optimum yield at the current moment and uses the workable time of the current period; and the resulting expected timeliness loss TML-LOSS-EXP with a minimum of zero.

The above procedure is used to derive an urgency of processing and is called by procedure URG-MAT-PRC-, Table 4.43, that is called in SHIFT-CHNGE- (Table 4.39) at the beginning of a period or in the shift controlling the moments of calculating the urgency (Table 4.108). URG-MAT-PRC-updates the quantities and an expected field before calling TML-LOSS-EXP- to find the urgency of processing, URGENCY-PROC, in f/h. The alternative urgency, URG-PROC-ALT, is derived in the same way. A disurgency of delivery, DISURGNC-DEL, is 0.0 if it is not to be used in the calculation of the urgency of gangs, otherwise it becomes URGENCY-PROC if a delivered field is placed at the head of the queue (such a new field delays the processing of the existing fields and so causes that expected timeliness loss). If however the fields delivered come at the tail of the queue and if the current field for delivery, FLD-D, has an optimum date of processing later than the material is expected to finish the processing of the existing fields or is only an expected field, then the disurgency becomes 0.0; it is otherwise related to the loss due to a delay from the optimum date to the date the material will be finished (a new field delivered at the current moment will be processed after FINSH-MAT although its optimum date is earlier; a delay of delivery is useful and is influenced by a positive disurgency). Finally a 'virtual' procedure, CAP-EXC--M-, not defined in the base model, is called that may derive from the urgency of processing a desire to use a level of excess capacity although it results in extra loss of material, for instance, enlar-

Table 4.43 Procedure URG-MAT-PRC- of class PROCESSED-MAT.

```
procedure URG_MAT_PRC_;                              !----  ----  ----  ----  ----;
begin                                                !called in shift_chnge_, shift_urg.dyn.;
    if DELIVERING then DELIVER_MAT;
    if PROCESSING then PROCESS_MAT;
    if not DELIVERING and not PROCESSING and SUPPLY_NDD and FLD_E =/= none then begin
        FLD_E.Out; FLD_EXPCT_M_; FLD_E:- FLD_C:- FLD_D;
        CH_READY; STATE_CH_MAT;
    end;                                             !updates fld_e.date_opt_yld;
    HOUR_AT_TIME;
    URGENCY_PROC:= TML_LOSS_EXP_ * PRICE_HA;
    !   !fraction of area in ha lossed per h * price per ha, [fl/h:= ha/h*fl/ha];
    URG_PROC_ALT:= TML_LOSS_ALT * PRICE_HA;
    DISURGNC_DEL:=
    if not DISURG_USED then 0.0
    else (if not FLD_AT_TAIL then URGENCY_PROC  !new field delays processing of existing fields;
    !;  else (if FLD_D.DATE_OPT_YLD < FINSH_MAT and FLD_D =/= FLD_E
    !;          then (TML_FNCT_FRC_ (0.0) - TML_FNCT_FRC_ (FINSH_MAT - FLD_D.DATE_OPT_YLD))
    !;          * CAP_PROC * PRICE_HA
    !;          else 0.0));                      !fld_d has optimum after finishing material or
    !                                                is expected field;
    CAP_EXC__M_;
end;                                                 !redefined in some specific materials;
```

Table 4.44 Procedure URG-MAT-EXT- of class PROCESSED-MAT.

```
procedure URG_MAT_EXT_;                              !----  ----   ----   ----   ----;
begin                                               !called in shift_chnge_, sh_urg.dyn., decide.dyn.;
   URG_PROC_T:= if AVLBL then RMAX ( URGENCY_PROC, URG_PROC_ALT)        ![fl/h];
   else 0.01;                                        !small urg to start if delivered;
   DISURG_DEL_T:= DISURGNC_DEL;                      ![fl/h];
end;                                                 !may be redefined in specific material;
```

ging the combine-harvester rate of operation in the wheat harvesting to 1.12 results in 1% additional loss of grain; 1.18 in 2% and 1.22 in 3% additional loss.

Procedure URG-MAT-EXT- extends the urgency calculations by modifying the urgency and disurgency to updated values for the current moment, Table 4.44; it defines URG-PROC-T as the highest of URGENCY-PROC and URG-PROC-ALT (the alternative urgency) if material is available and otherwise as 0.01. The value 0.01 is just sufficient to accept the processing of the material if it is delivered at the same time and is low enough not to make it urgent. The procedure is called from decide as well as from the shifts.

The following procedure is used to update attributes, ATTR-UPDT-M-, Table 4.45; it is called from WEATHER-MATRL where the weather and material data are read as input. A call from MAT-DVLPMNT- is also needed if properties depend on such a development. A processing property, PROC-PROPRTY is derived from the weather object, WTH-MAT. The range of property is subdivided in exclusive intervals of processing condi-

Table 4.45 Procedure ATTR-UPDT-M- of class PROCESSED-MAT.

```
procedure ATTR_UPDT_M_;                              !----  ----   -----  -----  -----;
begin                                                !called in weather, reset_;
   PROC_PROPRTY:= WTH_MAT.PROPERTY [MAT_NO];          !independent of field properties;
   WORKBL_NEXT:= M_PRC_CND_CH:= false;
   MAT_DLVR_PRV:= MAT_DLVR;
   for I_M2:= 1 step 1 until PROC_CNDTNS do begin
      M_PRC_CND_PRV [I_M2]:= M_PRC_CNDTN [I_M2];
      M_PRC_CNDTN [I_M2]:=
      PROC_PROPRTY >= PRC_CNDTN_LB [I_M2] and PROC_PROPRTY <= PRC_CNDTN_UB [I_M2];
      M_PRC_CND_CH:= M_PRC_CND_CH or not (M_PRC_CND_PRV [I_M2] eqv M_PRC_CNDTN [I_M2]);
      !                                               !union of changes in conditions;
      WORKBL_NEXT:= WORKBL_NEXT or M_PRC_CNDTN [I_M2];
      !                                               !union of processing conditions;
      if M_PRC_CNDTN [I_M2] then MAT_DLVR:= MAT_DLV_CNDTN [I_M2];
   end;
   CH_WORKBL;
   STATE_CH_MAT;
   if M_PRC_CND_CH then begin
      if MAT_DLVR_PRV <> MAT_DLVR and PROCESSING  then PROCESS_MAT;
      REC_OPR_C_D1:- OPRTN_PROC_Q.First;
      while REC_OPR_C_D1 =/= none do
      inspect REC_OPR_C_D1.OPR_MT do begin
         if MAT_DLVR_PRV <> MAT_DLVR then MAT_PROD_CHNG (MAT_DLVR_PRV, MAT_DLVR);
         STATE_CH_OP_;
         REC_OPR_C_D1:- REC_OPR_C_D1.Suc;
      end;
      DECIDE.PRC_CND:= true;
      activate DECIDE at Time;
   end;
end;
```

tions each with a lower and an upperlimit and a material delivered. You may think of the moisture content of grain; if it is between 0.0 and 19.0% then the delivered grain is dry, if it is between 19.0 and 23.0% then the delivered grain is wet; outside these ranges the material is not workable. So in the procedure each condition must be checked to see if it is within the limits and if the condition has changed. The workability (WORKBL-NEXT) and the material that can be delivered, MAT-DLVR are derived. The workability and the state are updated by calling CH-WORKBL and STATE-CH-MAT. If some processing condition has changed, then PROCESS-MAT is called when the material delivered also changes; all the operations processing this material (in OPRTN-PROC-Q) are informed by calling MAT-PROD-CHNG (that changes the material produced by the operation; Table 4.91) and STATE-CH-OP- (that updates the state of the operation according to the current processing conditions) and DECIDE receives a signal that some conditions changed and is activated.

A related procedure of a processed material is ATTR-INTG-F-, Table 4.46; it integrates attributes related to a field and is called at the moment of starting an operation (Table 4.86) and integrating the quantities processed (Table 4.25). In this case it integrates the timeliness losses in COSTS-TMLNSS by deriving the timeliness value from the timeliness function for the field consumed, FLD-C. By giving LOSS-MULT in input a zero value (instead of 1.0) the influence of the timeliness function on the timeliness costs is nullified (although the influence on the urgency remains) and no timeliness losses at all are expected or another more appropriate means to calculate the losses (that can be defined by ATTR-INTG-F- in a subclass or in ATTR-INTG-M- as called in CONSUMPTN-M, Table 4.31). An example is the timeliness losses of bales in the field which can be related to the amount of rain on the bales from the moment of baling until the moment of gathering; this requires specific information concerning the material and is only possible in the experimental frame. If such specific information is not available to calculate timeliness losses then one has to use the timeliness function or to give it up.

The procedure DELVR-FLD-M- is shown in Table 4.47 and shows the updating of QUANTITY and of the quantity delivered QUANT-F-PRD to an existing field or to a new field if the existing field is not produced on the

Table 4.46 Procedure ATTR-INTG-F- of class PROCESSED-MAT.

```
procedure ATTR_INTG_F_;                          !----   ----   ----   ----   ----;
begin                                      !called in supply_expct, start_oprtn_, qnt_intgr_m;
    TML_VL_PREV:= TML_VALUE;
    TML_VALUE:= PRICE_HA * TML_FNCT_FRC_ (TIME_YR - FLD_C.DATE_OPT_YLD);
    if QNT_PROC_OPR > 0.0 then begin
        COSTS_TMLNSS:=
        COSTS_TMLNSS + (PRICE_HA - 0.5 * (TML_VL_PREV + TML_VALUE)) * LOSS_MULT *
        (QNT_PROC_OPR - FLD_C.QNT_IN_PROCS);    !loss at time now may differ from luss at start;
    end;
end;
```

Table 4.47 Procedure DELVR-FLD-M- of class PROCESSED-MAT.

```
procedure DELVR_FLD_M_
(QNT_PROD__M_, FLD_ORIGINAL_);                   !----  ----  ----  ----  ----;
real   QNT_PROD__M_;
ref (FIELD) FLD_ORIGINAL_;
begin                                            !called in delivery_m_;
    if FLD_D.DATE_PRODCD < Entier (TIME_YR)      !same day and same name then same field;
    or (if FLD_D.FLD_AREA == none then true      !fld_e;
    !; else FLD_D.FLD_AREA.AREA_NAME <> FLD_ORIGINAL_.FLD_AREA.AREA_NAME) then
    inspect new FIELD do begin
        FLD_Q_SQN:= FLD_Q_SQN + 1;
        FLD_D:- this FIELD;
        if FLD_AT_TAIL then Into (FLD_Q) else Follow (FLD_Q);
        DATE_PRODCD:= DATE_PROCSBL:= DATE_OPT_YLD:= TIME_YR;
        MAKE_NAME (FLD_Q_SQN, MAT_NO);
        FLD_AREA:- FLD_ORIGINAL_.FLD_AREA;
    end;
    inspect FLD_D do begin
        QUANTITY:= QUANTITY + QNT_PROD__M_;
        QUANT_F_PRD:= QUANT_F_PRD + QNT_PROD__M_;
    end;
end !! of delivering intermediate material;
```

current day or its area name is not the same as the area name of the field consumed. When a field of wheat belonging to another area is harvested, a new field of straw and a new field of grain is created. When several fields with straw and different dates of production are baled on the same day and refer to the same area name, then only one field of bales is made. The new field is placed in the queue of fields at tail or head, sets its dates of production, of processable state and of the optimum yield equal to the current moment, TIME-YR; a name is made (Table 4.10), and a reference is made to the area of the field from which the material originated.

The remaining procedures of a processed material are RESET- (that resets the initial situation of a material for a following season, Table 4.48) and INIT-INPUT- that reads the input data (Section 5.2.2 'Initialization of objects', Table 5.57).

Table 4.48 Procedure RESET- of class PROCESSED-MAT.

```
procedure RESET_;                                !----  ----  ----  ----  ----;
begin                                            !called in reset;
   QUANT_ARRVD:= QUANT_AVLBL:= QUANT_PROCSD:= QNT_DLTD:= QNT_PRC_DUMM:= 0.0;
   LT_INTEGRT:= Time;
   AVLBL_NEXT:= false;
   CH_AVLBL;
   SUPPLY_NDD:= true;
   FLD_C:- FLD_D:- FLD:- none; FLD_Q.Clear;
   FLD_Q_SQN:= FLD_Q_SQN_PR:= 0;
   COSTS_TMLNSS:= COSTS_EXC_CP:= 0.0;
   PERD_MAT:= 1;
   RESET_M_;                                     !reads initial fields;
   if FLD_C == none then begin
       FLD_EXPCT_M_; FLD_E:- FLD_C:- FLD_D;
   end;
   CAP_RATIO:= 1.0;
   ATTR_UPDT_M_;
   CH_READY;
   STATE_CH_MAT;
   DLVR_ALLWD:= true;
   MAX_QUANT_M_;
end;
```

4.1.5 *Initial material*

Initial materials start with fields at the beginning of a season and are processed later in the season. So INITIAL-MAT is a subclass of PROCESSED-MAT. In addition to this prefixed class it contains the appropriate initialization (Tables 4.49-4.51). Table 4.49 RESET-M- is called from RESET- and resets the material by calling DELIVERY-INIT for each initial field required, it updates the availability of the material, the field for consumption and sets the end of process ENDED-PROCS to false. DELIVERY-INIT, Table 4.50 updates material attributes and calls for a new field. This procedure has a similar function as to DELIVERY-M- (Table 4.21). Table 4.51 shows the procedure DELVR-FLD-INIT, that creates a new field, a new area for each field and reads data from PARAMETERS (a reference to an

Table 4.49 Procedure RESET-M- of class INITIAL-MAT.

```
procedure RESET_M_;                                 !----  ----  ----  ----  ----;
inspect PARAMETERS do begin                         !called in reset_;
    FIND_DATA_AT ('DATA LIST of FLD.');
    N_INIT_FLDS:= Inint;
    for I_M2:= 1 step 1 until N_INIT_FLDS do DELIVERY_INIT_;
    if QUANT_AVLBL > 0.0 then AVLBL_NEXT:= true; CH_AVLBL;
    if AVLBL and FLD_E =/= none then begin
        FLD_E.Out; FLD_E:- none;                    !deletes expected field;
    end;
    FLD_C:- FLD_Q.First;
    FLD_D:- if FLD_AT_TAIL then FLD_Q.Last else FLD_Q.First;
    if AVLBL then ENDED_PROCS:= false;
end;              !note that delivering of initial material by processing remains possible;
```

Table 4.50 Procedure DELIVERY-INIT of class INITIAL-MAT.

```
procedure DELIVERY_INIT_;                           !----  ----  ----  ----  ----;
inspect PARAMETERS do begin                         !called in reset_m_;
    Inimage;                                        !each field on new line;
    QNT_M_D:= Inreal;
    QUANT_ARRVD:= QUANT_ARRVD + QNT_M_D;
    QUANT_AVLBL:= QUANT_AVLBL + QNT_M_D;
    FLD_Q_SQN_PR:= FLD_Q_SQN;
    DELVR_FLD_INIT_ (QNT_M_D);
    if QUANT_AVLBL > 0.0 then SUPPLY_NOD:= false;
end of inspect parameters;
```

Table 4.51 Procedure DELVR-FLD-INIT of class INITIAL-MAT.

```
procedure DELVR_FLD_INIT_ (QNT_PROD__M_);          !----  ----  ----  ----  ----;
real QNT_PROD__M_;                                  !called in delivery_init_;
inspect new FIELD do inspect PARAMETERS do begin
    FLD_Q_SQN:= FLD_Q_SQN + 1;
    if FLD_AT_TAIL then Into (FLD_Q) else Follow (FLD_Q);
    QUANTITY:= QNT_PROD__M_;
    QUANT_F_PRD:= QNT_PROD__M_;
    DATE_PRODCD:= TIME_YR;
    DATE_PROCSBL:= DATE_TO_DAYNO (Inint, Inint);
    DATE_OPT_YLD:= DATE_TO_DAYNO (Inint, Inint);
    MAKE_NAME (FLD_Q_SQN, MAT_NO);
    FLD_AREA:- new AREA;                            !each init. field has its own areal;
    FLD_AREA.ACREAGE:= QNT_PROD__M_;
    Lastitem;
    FLD_AREA.AREA_NAME:= Intext (IMIN(30, Length - Pos + 1));
end ** of delivering initial material;
```

input file). This procedure is comparable with DELVR-FLD-M- (Table 4.47).

Each initial material will be processed; it may be delivered also by calling from an operation DELIVERY-M- in class MATERIAL (Table 4.21) and DELVR-FLD-M- in class PROCESSED-MAT (Table 4.47). So an initial material can also act as an intermediate material that is delivered and processed. For example if one want to plough other fields than the original fields with wheat then it is possible.

4.1.6 Intermediate material

An intermediate material is delivered during the season and will be processed. Straw is such an intermediate material that is delivered by harvesting wheat and is processed by baling. This subclass is empty but may be specified by creating specific subclasses, for instance, a subclass STRAW or BALES to define specific virtual procedures as DELVR-FLD-M-, ATTR-UPDT-M-, ATTR-INTG-F-.

4.1.7 Final material

A final material will not be processed, so the attributes of a material processed are not needed and it is a direct subclass of MATERIAL. It redefines the procedures MAX-QUANT-M- (no maximum) and DELVR-FLD-M-, Table 4.52. The last one considers only a new field when the area name to which it belongs differs from the current area name.

Table 4.52 Procedure DELVR-FLD-M- of class FINAL-MAT.

```
proredure MAX_QUANT_M_; ;                               !no maximum considered;

procedure DELVR_FLD_M_
(QNT_PROD__M_,FLD_ORIGINAL_);                           !----   ----   ----   ----   ----;
real    QNT_PROD__M_;
ref (FIELD) FLD_ORIGINAL_;
begin                                                   !called in delivery_m_;
    !                                                   !same name then same field;
    if (if FLD_D.FLD_AREA == none then true else
    FLD_D.FLD_AREA.AREA_NAME <> FLD_ORIGINAL_.FLD_AREA.AREA_NAME) then begin
        FLD_Q_SQN:= FLD_Q_SQN + 1;
        FLD_D:- new FIELD;
        FLD_D.Into (FLD_Q);
        FLD_D.MAKE_NAME (FLD_Q_SQN, MAT_NO);
        FLD_D.FLD_AREA:- FLD_ORIGINAL_.FLD_AREA;
    end;
    inspect FLD_D do begin
        QUANTITY:= QUANTITY + QNT_PROD__M_;
        QUANT_F_PRD:= QUANT_F_PRD + QNT_PROD__M_;
    end;
end ** of delivering final material;
```

4.2 Men and machinery

Sections 2.1.2 'Men and machinery' and 2.2.2 'Operations and decision' have already shown that men and machines constitute man-machine systems. They are called gangs when they are related to an operation and are called combinations when they reflect that two or more gangs can work simultaneously as far as men and items of equipment on the farm are concerned.

4.2.1 Men and machines

The common part of men and machines is programmed in class LABR-EQPMNT, Table 4.53. It has four parameters; three belong to the prefixed class COMPONENT (Table 4.2); one parameter LE-NO-LE is the category number of the type of labour & equipment to which this object belongs. All men can have the same category number, all horses or type of tractors or trailers can have their category number. In other words the same category number can be used by several objects which are considered identical i.e. it does not matter which one is used in a gang. The reference variable G-ASSEMBL refers to the gang using this man or machine object. Procedure SHIFT-CHNGE-, Table 4.54, is redefined and differs from the one shown in Table 4.3; it adds costs per hour, COSTS-H, and a next state depending on its availability, AVLB-COMP. The procedures starting, START-ACTVTY, and stopping, STOP-ACTVTY, the activity of the object, Table

Table 4.53 Class LABR-EQPMNT; parameter, virtual procedure and declaration of variables.

```
COMPONENT class LABR_EQPMNT (LE_NO_LE);             !========================================;
integer LE_NO_LE;                                   !category of men or machines;
virtual :procedure STATE_CH_LE_;
begin
   !    d    e    c    l    a    r    a    t    i    o    n;

   ref (MAN_MACH_SET)                                !----------------------;
   G_ASSEMBL;                                       !gang assembled this object for use;
```

Table 4.54 Procedure SHIFT-CHNGE- of class LABR-EQPMNT.

```
procedure SHIFT_CHNGE_;                             !----  ----  ----  ----  ----;
begin                                               !called in shift_week, shift_perd;
    TIME_C_ACCUM;
    if SH_WK =/= none then begin
        AVLB_COMP:= SH_WK.AVLB_WK;
        CTGR:= SH_WK.COST_CTGR;
        COSTS_H:= SH_WK.COSTS_NOW;
    end else AVLB_COMP:= true;
    AVLB_COMP:= AVLB_COMP and (if SH_PRD =/= none then  SH_PRD.AVLB_PRD else true);
    STATE_NEXT:= if not AVLB_COMP then DOWN
    else if STATE = RUN then RUN
    !;                  else PASSIVE;
    STATE_CH_LE_;                               !with additional conditions for state_next;
end;
```

4.55, derive the next state of man or machine and call the 'virtual' procedure STATE-CH-LE-, which changes the state (Table 4.57 and 4.59). The initial section, Table 4.56, shows that one object is added to the total number of this category.

The subclass LABOUR is shown in Table 4.57. It contains only the procedure STATE-CH-LE- to change the state if the next state does not equals the current state; such a change is prescribed by (i) gangs requiring a man or releasing a man (by calling START- /STOP-ACTVTY) and by (ii) SHIFT-CHNGE- when the costs of men changes or regular worktime/overtime starts or finishes. The latter situation results in updating the number of man available, LE-NMB [.,AVAIL]. At the moment the last statement is achieved, the object becomes a terminated process, a data block. Men are not considered as 'living' objects in the simulation because no event can happen due to a man; the occurrence of worktime and no-worktime is handled for all men together in the 'living' process SHIFT-WEEK (Section 4.4.2 'Shifts within a week') for convenience reasons only; that process is referred to by SH-WK, a parameter in the superclass COMPONENT (Table 4.2).

The other subclass EQUIPMENT is shown partly in Table 4.58. It has

Table 4.55 Procedures START-ACTVTY and STOP-ACTVTY of class LABR-EQPMNT.

```
procedure START_ACTVTY;                       !----  ----  ----  ----  ----;
if STATE = PASSIVE then begin                 !called in gang.assemble;
    STATE_NEXT:= RUN;
    STATE_CH_LE_;
end;

procedure STOP_ACTVTY;                        !----  ----  ----  ----  ----;
begin                                         !called in gang.disassemble;
    STATE_NEXT:= if AVLB_COMP then PASSIVE else DOWN;
    STATE_CH_LE_;
end;
```

Table 4.56 Initial section of class LABR-EQPMNT.

```
I    i    n    i    t    i    a    l;

LE_NMB [LE_NO_LE,TOTAL]:= LE_NMB [LE_NO_LE,TOTAL] + 1;  !total number of category;
STATE_NEXT:= STATE:= DOWN;
Into (LE_STATE_Q [LE_NO_LE, STATE]);      !used in gang.store_costs!!;
```

Table 4.57 Class LABOUR and its procedure STATE-CH-LE-

```
LABR_EQPMNT class LABOUR;                       !**************************************;
begin
    procedure STATE_CH_LE_;                     !----  ----  ----  ----  ----;
    if STATE <> STATE_NEXT then begin           !called in shift_chnge_,start/stop_actvty;
        STATE_PREV:= STATE;
        TIME_C_ACCUM;
        if STATE = DOWN then LE_NMB [LE_NO_LE,AVAIL]:= LE_NMB [LE_NO_LE,AVAIL] + 1;
        if STATE_NEXT = DOWN then LE_NMB [LE_NO_LE,AVAIL]:= LE_NMB [LE_NO_LE,AVAIL] - 1;
        STATE:= STATE_NEXT;
        Into (LE_STATE_Q [LE_NO_LE,STATE]);
    end ** of state change;
end ** of class labour;
```

Table 4.58 Class EQUIPMENT and declaration of variables.

```
LABR_EQPMNT class EQUIPMENT;                  !*************************************;
begin
   !     d     e     c     l     a     r     a     t     i     o     n;

   real                                        !---------------------;
   REPAIRTIME, SERVICETIME,                    !part of downtime, rum. in hours;
   RPR_LB, RPR_UB, RPR_DUR,                    !lower/ upper bound/ duration of repair in [h];
   SRVC_DUR_LB, SRVC_DUR_UB, SRVC_DUR,         !lower/ upper bound/ duration of service in [h];
   SRVCFR_RUNT,                                !minimum run time free of service in [h];
   FLRFR_LB, FLRFR_UB,                         !lower/ upper bound of failure free run time in [h];
   FLRFR_DUR,                                  !duration of failure free run time in [h];
   FLR_AT_RUNT,                                !failure will occur at runtime, in [d];
   SRVC_TRMNTD,                                !service terminated at runtime, in [d];
   REPT_VAR, REPT_MEAN,                        !variance and mean of repair duration;
   SERT_VAR, SERT_MEAN,                        !variance and mean of service duration;
   STORECAP_E;                                 !storage capacity of trailers [ha];
   boolean                                     !---------------------;
   RPR_ND, SRVC_ND;                            !true if repair, service needed;
   integer                                     !---------------------;
   S_R_OPR_NO,                                 !type of operation for repair or service;
   U_FL_RP, U_SRVC,                            !random numbers;
   REPR_OBS, SERV_OBS;                         !number of repair/service observations;
```

three sections: declaration, initial and dynamic; it may be an object to which events can happen such as failure, service needed and service or repair is done. The variables declared will become clear when they are needed in the procedures or in the dynamic section. Uniform distributions of the duration to repair a failure, the duration to service the machine and the duration of the time the machine is used and is failure free are used; at the end of such a duration the machine is repaired, serviced or fails (only during STATE = RUN). The lower- and upper-bounds (...-LB, ...-UB) or limits of these distributions are read from the input file, Section 5.2.2 'Initialization of objects', Table 5.44.

The first procedure STATE-CH-LE-, Table 4.59, is larger than the one for man (Table 4.57) and adds (i) an activation of a class referenced by UPDT-MM-SYS (Section 4.2.3) that updates the man-machine systems when the available number of this category of equipment changed and some man-machine systems cannot (or can again) be used; (ii) a signal to the gang using this equipment that a failure has occurred, G-ASSEMBL.FAILURE-E becomes true; (iii) a reactivation under certain conditions of this item of equipment to schedule it at the moment a failure is expected during its use (or when not used but scheduled it is canceled from the time axis of future events); (iv) a call to SERVICE-NEED after the use of the machine. The latter procedure, SERVICE-NEED, Table 4.60, checks if the service free runtime (a minimum) is already exceeded. Service is performed only after use of the machine! It updates the duration SRVC-DUR, places this machine in a queue, MCHN-S-R-Q, of the operation, OPR-S-R [S-R-OPR-NO], caring for service or repair of this machine, it updates its own state and that of the operation and signals DECIDE that something happened to a machine. Procedure SRVC-RPR-DON, Table 4.60, is called from the operation when the service or the repair is done. If repair was needed it records

Table 4.59 Procedure STATE-CH-LE- of class EQUIPMENT.

```
procedure STATE_CH_LE_;                         !----   ----    ----    ----    ----;
begin           !called in shift_chnge_,start/stop_actvty,service_need,srvc_rpr_don,dyn.;
    if RPR_ND or SRVC_ND then STATE_NEXT:= DOWN;        !was not yet included in shift_chnge_;
    if STATE <> STATE_NEXT then begin
        STATE_PREV:= STATE;
        TIME_C_ACCUM;
        if STATE = DOWN then LE_NMB [LE_NO_LE,AVAIL]:= LE_NMB [LE_NO_LE,AVAIL] + 1;
        if STATE_NEXT = DOWN then LE_NMB [LE_NO_LE,AVAIL]:= LE_NMB [LE_NO_LE,AVAIL] - 1;
        if (STATE = DOWN or STATE_NEXT = DOWN) then begin
            !                           !change in available number;
            UPDT_MM_SYS.TEST_LE [LE_NO_LE]:= true;
            activate UPDT_MM_SYS at Time;
            if RPR_ND or SRVC_ND then reactivate UPDT_MM_SYS at Time prior;
            !           !prevents immediately a start of operations involved;
        end;
        STATE:= STATE_NEXT;
        Into (LE_STATE_Q [LE_NO_LE,STATE]);
        if STATE = DOWN and STATE_PREV = RUN and RPR_ND
        then G_ASSEMBL.FAILURE_E:= true;        !signal to gang that a failure occurred;
        if STATE = RUN and FLR_AT_RUNT < 1.0&9/24.0 and S_R_OPR_NO > 0 and not Terminated
        then reactivate this EQUIPMENT at Time + FLR_AT_RUNT - TIME_USED [RUN]
        else    if (if not Idle then Evtime > Time else false) then Cancel (this EQUIPMENT);
        !                               !cancels a reactivation when operation stops;
        if STATE = PASSIVE and not SRVC_ND then SERVICE_NEED;   !checks need after work;
    end;
end ** of state change;
```

Table 4.60 Procedures SERVICE-NEED and SRVC-RPR-DON of class EQUIPMENT.

```
procedure SERVICE_NEED;                         !----   ----    ----    ----    ----;
!                                               !called in dyn., state_ch_le_;
if TIME_USED [RUN] - SRVC_TRMNTD > SRVCFR_RUNT and S_R_OPR_NO > 0 then begin
    SRVC_ND:= true;
    SRVC_DUR:= Uniform (SRVC_DUR_LB,SRVC_DUR_UB,U_SRVC);
    new RECORD_LE (this EQUIPMENT).Into (OPR_S_R [S_R_OPR_NO].MCHN_S_R_Q);
    STATE_NEXT:= DOWN;
    STATE_CH_LE_;
    OPR_S_R [S_R_OPR_NO].STATE_CH_OP_;          !will not be called in updt_mm_s;
    DECIDE.MACHN_FLR:= true;                    !signal to decide;
end;

procedure SRVC_RPR_DON;                         !----   ----    ----    ----    ----;
begin                                           !called in reset_, oprtn.transfer;
    if RPR_ND then begin
        REPAIRTIME:= REPAIRTIME + RPR_DUR;
        SIGMEAN (REPT_VAR,REPT_MEAN, REPR_OBS, RPR_DUR);
        RPR_ND:=  false;
        FLRFR_DUR:= Uniform (FLRFR_LB,FLRFR_UB,U_FL_RP);
        FLR_AT_RUNT:= TIME_USED [RUN] + FLRFR_DUR / 24.0;
    end
    else if SRVC_ND then                        !required already in oprtn.transfer;
    begin
        SERVICETIME:= SERVICETIME + SRVC_DUR;
        SIGMEAN (SERT_VAR, SERT_MEAN, SERV_OBS, SRVC_DUR);
        SRVC_ND:= false;
        SRVC_TRMNTD:= TIME_USED [RUN];          !updated at failure or service needed;
    end;
    if not (RPR_ND or SRVC_ND) then begin
        STATE_NEXT:= if AVLB_COMP then PASSIVE else DOWN;
        STATE_CH_LE_;
    end;
    if STATE = PASSIVE then begin
        DECIDE.MACHN_OK:= true;
        activate DECIDE at Time;
    end;
end;
```

the total repair time, REPAIRTIME, the variation, the mean and the number of observations by calling the system procedure SIGMEAN, it calculates a new failure free duration FLRFR-DUR and a future time FLR-AT-RUNT when a failure occurs; it is measured in days of the time the machine is used in state RUN, TIME-USED [RUN]. Otherwise service was needed and similar statements are executed. If neither repair nor service is needed any longer, then the next state of the machine is PASSIVE or DOWN depending on its availability. In the first case DECIDE is reactivated after informing it of the reason that the machine is in order again, MACH-OK becomes true.

The dynamic section, Table 4.61 is entered if the experiment is not ended and failure or service is expected (durations of free time less than infinity, 1.0 & 9 (=1.0 * 10**9)) and an appropriate operation for service or repair, S-R-OPR-NO is available, otherwise the last end is passed, the object is teminated and remains a data block. The dynamic block contains two statements:

if STATE = RUN and .... then begin ..... end;
Passivate;

These statements are executed in sequence each time the object is activated at failure time. The need for repair, RPR-ND becomes true, a repair duration RPR-DUR is updated, a call to SERVICE-NEED to learn if service also is needed, queueing the machine in the repair operation, changing the state of the machine and of the operation and an activation of DECIDE after making known that a machine failure occurred, MACHN-FLR becomes true.

Table 4.61 Dynamic section of class EQUIPMENT.

```
!       d       y       n       a       m       i       c;

if (FLRFR_LB < 1.0&9 or SRVCFR_RUNT < 1.0&9)
and S_R_OPR_NO > 0 and S_R_OPR_NO <= OPRTNS_S_R
then
!               r e p e a t;
while not END_EXPRMNT do begin
    if STATE = RUN and Abs (FLR_AT_RUNT - TIME_USED (RUN)) < 1.0 / 24.0 / 60.0  !1 min.;
    then begin
        TIME_C_ACCUM;
        RPR_ND:= true ;
        RPR_DUR:= Uniform (RPR_LB,RPR_UB,U_FL_RP);
        SERVICE_NEED;                          !if repair then try service;
        STATE_NEXT:= DOWN;
        STATE_CH_LE_;
        !               !calls updt_mm_s and gang.avlblty_test and so oprtn__g.state_ch_op_;
        new RECORD_LE (this EQUIPMENT).Into (OPR_S_R (S_R_OPR_NO).MCHN_S_R_Q);
        OPR_S_R (S_R_OPR_NO).STATE_CH_OP_;     !not called in updt_mm_s;
        DECIDE.MACHN_FLR:= true;                       !initiative from equipment;
        activate DECIDE at Time;
    end;
    Passivate;                                 !activated in state_ch_le;
```

This class EQUIPMENT is a description of different objects such as a planting or sowing machine, a harvester, a plough, trailers, tools, installations (a drier), power (oxen, horses, tractors). Subclasses were not necessary, for instance to handle the capacity of power units [kW] for this is implicitly defined by giving the identical items the same category number LE-NO-LE (Table 4.53) and different types different category numbers. It is even possible to consider men as an object of class EQUIPMENT. They are referred to as MACH[i] and have the same features to become ill (comparable with failure) or to need holidays (comparable with service). Also the reverse is possible to consider machines as an object of class LABOUR if no failure and no service is needed and a storage capacity is not relevant. In general it is sufficient to define for each object of class LABOUR a SHIFT-PERD and a SHIFT-WEEK to control the availability during periods and within a week (Section 4.4). In objects of class EQUIPMENT SH-PRD and SH-WK may both refer to none and in that case the machine is considered available all the time. It is easy however to use objects of shift classes, for instance, to introduce machines of a contract-worker or neighbour for restricted time periods or to restrict the availability of milking machines to two intervals a day.

4.2.2 *Man-machine systems*

A man-machine system is a set consisting of men and of items of equipment. There are two forms (Section 2.1.2 'Men and machinery'): (i) a gang for performing an operation, (ii) a combination that consists of two or more gangs working simultaneously. Table 4.62 shows the class MAN-MACH-

Table 4.62 Class MAN-MACH-SYS; parameters, virtual procedures and declaration of variables.

```
COMPONENT class MAN_MACH_SYS (SET_NO, GNGS);          !****************************************;
integer SET_NO,                              !type number of set of man-machine systems;
GNGS;                                        !number of gangs, !actual parameter has to be 1 for a gang;
virtual: procedure START_WORK_C_, STOP_WORK_C_;
begin
    !      d      e      c      l      a      r      a      t      i      o      n;

    integer                                          !----------------------;
    LE_SQN, GN,                              !auxiliary variables;
    MM_S_SQN;                                !sequence number of type of man machine system;
    integer array                                    !----------------------;
    RQRD_GNG [1:GNGS],                       !required type of gang;
    RQRD_NMB_LE [1:LE_SQNS];                 !required number of elements of labour & equipment;
    boolean array                                    !----------------------;
    AVL [1:LE_SQNS];                         !true if available number >= required number;
    boolean                                          !----------------------;
    AVL_LE,AVL_PREV,                         !true if all categories of man/mach.i: avl[i]=true;
    AVL_GNGS,                                !true if avlb_comp for all gangs;
    APPLICABLE, APPL_M_AVLBL;                !not down & mat.produced delivery allowed;
    !                                   (material processed available or delivered)/ resp. true;
    real                                             !----------------------;
    URGENCY, URGENCY_CORR;                   !urgency without/ with corrections;
    ref (OPERATION)                                  !----------------------;
    OPRTN__G;                                !none for combinations;
```

SYS with the prefix COMPONENT and the declaration of the parameters and variables common for both forms. Besides the usual parameters of COMPONENT (Table 4.2) two other parameters are needed. The first one, SET-NO gives the set of man-machine systems (Section 4.2.3) to which this system belongs and the second parameter GNGS gives the number of gangs in a system (1 for a gang, > 1 for a combination). The variables concern the sequence number of the gang MM-S-SQN, the gangs that are required RQRD-GNG, the number of each category of men and machines required RQRD-NMB-LE and the availability of the required number AVL, availability of all categories AVL-LE, of all gangs AVL-GNGS, the applicability of the man-machine system APPLICABLE and APPL-M-AVLBL (independent of materials processed), the urgency of using URGENCY and URGENCY-CORR (incl. costs, Table 4.71) and a reference to an operation.

The procedure SHIFT-CHNGE-, Table 4.63, is redefined and contains the usual statements; it adds AVL-LE (true if all categories of men and machines are available in the required number) and AVL-GNGS (true if all gangs are available) to define the next state of the system. Table 4.64 shows STATE-CH-MMS that is as usual and adds the call to update the state of the related operation STATE-CH-OP-.

The remaining procedures are called from a class updating man-machine systems (Section 4.2.3) when, for instance, the worktime of men is over, a machine is repaired or costs changed due to overtime. AVL-TEST sets AVL[i] of category i of men or machines to true if the available number on the farm is not less than the required number for the system; Table 4.65. AVLBLTY-TEST establishes AVL-LE (reflects all categories of men and machines) and AVL-GNGS (reflects all the gangs required) and changes

Table 4.63 Procedure SHIFT-CHNGE- of class MAN-MACH-SYS.

```
procedure SHIFT_CHNGE_;                          !----  ----  ----  ----  ----;
begin                                            !called in shift_week, shift_perd;
    TIME_C_ACCUM;
    if SH_WK =/= none then begin
        AVLB_COMP:= SH_WK.AVLB_WK;
        CTGR:= SH_WK.COST_CTGR;
    end else AVLB_COMP:= true;
    AVLB_COMP:= AVLB_COMP and (if SH_PRD =/= none then  SH_PRD.AVLB_PRD else true);
    STATE_NEXT:= if not (AVL_LE and AVL_GNGS and AVLB_COMP) then DOWN
    else         if STATE = RUN then RUN
    !;                              else PASSIVE;
    STATE_CH_MMS;
end;
```

Table 4.64 Procedure STATE-CH-MMS of class MAN-MACH-SYS.

```
procedure STATE_CH_MMS;                       !----  ----  ----  ----  ----;
if STATE <> STATE_NEXT then      !called in shift_chnge_,avlblty_test, start/stop_work_g/_c_;
begin
    STATE_PREV:= STATE;
    TIME_C_ACCUM;
    STATE:= STATE_NEXT;
    if OPRTN__G =/= none then OPRTN__G .STATE_CH_OP_;
end ** change of state of man_machine system;
```

Table 4.65 Procedures AVL-TEST and AVLBLTY-TEST of class MAN-MACH-SYS.

```
procedure AVL_TEST (LE_NO);                           !----  ----   ----   ----   ----;
integer LE_NO;                                        !called init., in updt_mm_sys.state_costs;
AVL [LE_NO]:= LE_NMB [LE_NO, AVAIL] >= RQRD_NMB_LE [LE_NO];     !required number available;

procedure AVLBLTY_TEST;                               !----  ----   ----   ----   ----;
begin                                                 !called init., in updt_mm_sys.state_costs;
    AVL_PREV:= AVL_LE and AVL_GNGS;
    AVL_LE:= true;
    for LE_SQN:= 1 step 1 until LE_SQNS do
    AVL_LE:= AVL_LE and AVL [LE_SQN];                 !required number, not specific elements;
    AVL_GNGS:= true;
    for GN:= 1 step 1 until GNGS do
    AVL_GNGS:= AVL_GNGS and GANG [RQRD_GNG [GN]].AVLB_COMP;     !gangs available;
    if not ( (AVL_LE and AVL_GNGS) eqv AVL_PREV) then begin
        STATE_NEXT:= if not (AVL_LE and AVL_GNGS and AVLB_COMP) then DOWN
        else    if STATE = RUN then RUN
        !;                      else PASSIVE;
        STATE_CH_MMS;
    end;
end xx of availability test of man_machine system based on its elements;
```

the state accordingly. AVL-LE, AVL-GNGS and AVLB-COMP discriminates for STATE-NEXT only between DOWN and not DOWN (i.e. RUN or PASSIVE); further discrimination with these variables is not possible; the same holds for SHIFT-CHNGE- (Table 4.63) and a similar case for men and machines (Table 4.54). The third procedure called from the class updating man-machine systems is COST-CHANGE, Table 4.66, that is called when men changes regular worktime into overtime or their costs; it updates the current costs per hour of the man-machine system.

The subclass MAN-MACH-SET defines gangs, which are sets of men and machines. Table 4.67 shows the declaration of this class. The variables declared concern the set-up time SETUP... (the time needed from the start of the gang at the farm until it is at the field ready to operate; it includes also the time from the field to the farm), the storage capacity STORECAP-G of trailers or driers, the expected capacity of processing CAPACITY with the gang in [ha/h], a signal FAILURE-E when a failure occurred to one of the machines involved and some references (their use will become clear in the procedures). The procedure STORE-COSTS, Table 4.68, is called in the initial section and derives from the first element of each category the storage capacity, the costs per hour and the fixed costs per hour (for instance, fuel costs or contract work costs) from the machines belonging to the gang; men are assumed to have no costs at this moment, these costs are incorpo-

Table 4.66 Procedure COST-CHANGE of class MAN-MACH-SYS.

```
procedure COST_CHANGE (CHNG_UNIT,LE_NO1);             !----  ----   ----   ----   ----;
!                                                     !called in updt_mm_sys.state_costs;
real    CHNG_UNIT;
integer LE_NO1;
begin
    TIME_C_ACCUM;
    COSTS_H:= COSTS_H + CHNG_UNIT * RQRD_NMB_LE [LE_NO1];
end ** change of costs per hour due to man shift;
```

Table 4.67 Class MAN-MACH-SET and its declaration of variables.

```
MAN_MACH_SYS class MAN_MACH_SET;                      !**************************************;
virtual:procedure URGENCY_GANG_;
begin
   !    d    e    c    l    a    r    a    t    i    o    n;
   real
   SETUPTIME,                                 !belongs to runtime, cum. in hours;
   SETUP1 ,SETUP2_N, SETUP_GNG,               !duration for first/ next times in a day/ actual [h];
   STORECAP_G,                                !storage capacity e.g. trailers [ha];
   CAPACITY;                                  !capacity of processing [ha/h], updated in operation;
   boolean
   FAILURE_E;                                 !true if an item of equipment gets a failure;
   ref (OPRTN_MATRL) OPR_MT__G;
   ref (SRVC_REPR) OPR_SR__G;
   ref (LABR_EQPMNT) LBEQP;
   ref (Head) ASSMBL_Q;
   ref (RECORD_LE) REC_LE;
```

Table 4.68 Procedure STORE-COSTS of class MAN-MACH-SET.

```
procedure STORE_COSTS;                                       !----  ----  ----  ----  ----;
begin                                                        !called in initial;
   real        STR,CST,CSF;
   for LE_SON:= 1 step 1 until LE_SONS do inspect LE_STATE_Q [LE_SON, DOWN].First
   when EQUIPMENT do begin     !labour or none inappropriate;  !state=down initially;
       STR:= STR + STORECAP_E * RQRD_NMB_LE [LE_SON];
       CST:= CST + COSTS_H * RQRD_NMB_LE [LE_SON];
       CSF:= CSF + COSTS_H_FXD * RQRD_NMB_LE [LE_SON];
   end;
   !  !elements are assumed to have identical storage capacity and costs per hour;
   STORECAP_G:= STR; COSTS_H:= CST; COSTS_H_FXD:= CSF;
end ** storage capacity and equipment costs of a man_machine system;
```

rated later on by COST-CHANGE (Table 4.66). START-WORK-G defines the next state if the gang was able to work (PASSIVE) and assembles the required number of elements of men and machines by calling ASSEMBLE; Table 4.69. This procedure assembles only passive elements from the required categories and writes an error message if not enough elements in LE-STATE-Q [category,PASSIVE] are found. The elements start their activity (Table 4.55) and are queued in ASSMBL-Q. This queue is used in DISASSEMBLE (called in STOP-WORK-G) that stops the activity of elements and clears the contents of the queue (Table 4.70). START-WORK-G and STOP-WORK-G are called from the operation.

Table 4.71 shows the procedure that calculates the urgency of operating. URGENCY is the sum of the urgencies of processing of the materials processed and of the negative disurgencies, if used, of the materials produced, unless final materials. Urgencies and disurgencies are transformed from f/h (Table 4.44) into f/ha to add them to other costs in f/ha. The urgency is again transformed to [f/h] by multiplying by the rate of operation, CAPACITY. The corrected urgency, URGENCY-CORR is corrected for (i) the costs expected from the processed materials (for instance, drying costs when wet grain is harvested), (ii) the actual capacity fraction of the operation (depends, for instance, on the moisture content of straw) and (iii) the costs per hour of the gang. Drying costs of wet grain are involved in the expected

Table 4.69 Procedures START-WORK-G and ASSEMBLE of class MAN-MACH-SET.

```
procedure START_WORK_G;                              !----  ----  ----  ----  ----;
if STATE = PASSIVE then begin                        !called in start_oprtn_;
   ASSEMBLE;
   STATE_NEXT:= RUN;
   STATE_CH_MMS;
end ** of start of operating with a gang;

procedure ASSEMBLE;                                  !----  ----  ----  ----  ----;
begin                                                !called in start_work_g;
   integer RQ, RQLE;
   for LE_SQN:= 1 step 1 until LE_SQNS do begin
       RQLE:= RQRD_NMB_LE (LE_SQN);
       for RQ:= 1 step 1 until RQLE do begin
           LBEQP:- LE_STATE_Q (LE_SQN, PASSIVE).First;
           if LBEQP == none
           then begin
               Outtext ('Error GANG.ASSEMBLE in Gang, at Time: no equipment of Category');
               Outimage; Outtext (NAME_COMP); Outfix (Time,6,12); Outint (LE_SQN,6); Outimage;
           end
           else begin
               LBEQP.START_ACTVTY;
               new RECORD_LE (LBEQP).Into (ASSMBL_Q);
               LBEQP.G_ASSEMBL:- this MAN_MACH_SET;
           end;
       end;
   end;
end ** of assemble;
```

Table 4.70 Procedures STOP-WORK-G and DISASSEMBLE of class MAN-MACH-SET.

```
procedure STOP_WORK_G;                               !----  ----  ----  ----  ----;
begin                                                !called in stop_oprtn_;
   DISASSEMBLE;
   FAILURE_E:= false;                  !is already used in decision if a failure occurred;
   STATE_NEXT:= if AVL_LE and AVL_GNGS and AVLB_COMP then PASSIVE else DOWN;
   STATE_CH_MMS;
end ** of stop of operating with a gang;

procedure DISASSEMBLE;                               !----  ----  ----  ----  ----;
begin                                                !called in stop_work_g;
   REC_LE:- ASSMBL_Q.First;
   while REC_LE =/= none do begin
       LBEQP:- REC_LE.LBR_EQP;
       LBEQP.STOP_ACTVTY;
       REC_LE:- REC_LE.Suc;
   end;
   ASSMBL_Q.Clear;
end;
```

costs COSTS-EXPCTD of grain processed by a combine-harvesting gang and later as actual costs COSTS-H-FXD of the drying gang; the first one helps to decide whether or not to harvest wet grain and the second one to decide between several drying installations (owned, cooperative or contract work). The urgency of a gang repairing a machine is derived from the urgency of the gang to which the machine belonged at the moment the failure occurred. In order to have the latter urgency calculated in time, the gangs used for service and repair are assigned higher numbers in the array of reference variables GANG[i] than the gangs for operating on materials.

The other subclass GANG-SET defines pure combinations, which are sets of gangs (at least two gangs). Pure combinations and gangs together are called man-machine systems and sometimes also combinations; see Table

Table 4.71 Procedure URGENCY-GANG- of class MAN-MACH-SET.

```
procedure URGENCY_GANG_;                              !----   ----   ----   ----   ----;
begin                                                 !called in decision;
    URGENCY:= URGENCY_CORR:= 0.0;
    inspect OPR_MT__G do begin
        for I2:= 1 step 1 until MATRLS_PRCSD do begin
            URGENCY:= URGENCY + MAT_PROC [I2].URG_PROC_T / MAT_PROC [I2].CAP_PROC;![Dfl/ha];
            URGENCY_CORR:= URGENCY_CORR - MAT_PROC [I2].COSTS_EXPCTD;          ![Dfl/ha];
        end;
        for I2:= 1 step 1 until MATRLS_PROCD do
        URGENCY:= URGENCY -
        (if MAT_PROD [I2].MAT_NO > MATRLS_PROC then 0.0       !final materials;
        else    if MAT_PROD [I2] qua PROCESSED_MAT.DISURG_USED
        !;      then MAT_PROD [I2] qua PROCESSED_MAT.DISURG_DEL_T / MAT_PROD [I2].CAP_PROC
        !;      else 0.0);
        URGENCY:= URGENCY * CAPACITY;                          ![Dfl/h];
        URGENCY_CORR:= URGENCY_CORR * CAPACITY;                ![Dfl/h];
        URGENCY_CORR:= (URGENCY_CORR + URGENCY) * CAP_FRAC;    !actual fraction from operation;
    end;
    if OPR_SR__G =/= none then begin
        URGENCY:= if OPR_SR__G.MACHN == none then 0.0
        else RMAX (1.0, OPR_SR__G.MACHN.G_ASSEMBL.URGENCY);
        URGENCY_CORR:= URGENCY;
    end;
    URGENCY_CORR:= URGENCY_CORR - COSTS_H - COSTS_H_FXD;       !costs of gang;
end;
```

Table 4.72 Class GANG-SET and its procedures START-WORK-C and STOP-WORK-C.

```
MAN_MACH_SYS class GANG_SET;                       !**************************************;
begin
    !       gangs in rqrd_gng[.] ordered in sequence of operations as performed on a field;

    !    d    e    c    l    a    r    a    t    i    o    n;

    procedure START_WORK_C_;                       !----   ----   ----   ----   ----;
    if STATE = PASSIVE then begin                  !called in decision;
        STATE_NEXT:= RUN;
        STATE_CH_MMS;
    end ** of start with combination;

    procedure STOP_WORK_C_;                        !----   ----   ----   ----   ----;
    begin                                          !called in decision;
        STATE_NEXT:= if AVL_LE and AVL_GNGS and AVLB_COMP then PASSIVE else DOWN;
        STATE_CH_MMS;
    end ** of stop with combination;
```

4.1 for the references MM-S [.], GANG [.] and COMB-G [.], respectively. Table 4.72 shows the declarations of the class. Gangs within a combination are assumed to be ordered in the same sequence as operations on a field are performed, for instance, combine-harvesting, baling, gathering bales, ploughing; such a fixed sequence is convenient in decision. The virtual procedures START-WORK-C- and STOP-WORK-C- change the state of a combination and are called from a subclass of DECISION (Table 5.12).

Both gangs and combinations have no dynamic section; the objects exist after initialization as data blocks.

4.2.3 *Miscellaneous*

In this section the following are described: (i) the class updating the man-machine systems when costs or availability of men or machines changed; (ii) the sets containing man-machine systems; and (iii) a procedure creating combinations from the gangs instead of deriving them from the input.

Class UPDT-MAN-MCH, Table 4.73, has arrays of Booleans to know which category of men or machine changed their availability due to work-time or failure/repair, TEST-LE[.], or changed their costs, COSTS-CHNG[.] and an array of costs changes, COSTS-INCRSE[.]. Table 4.74 shows the procedure STATE-COSTS that updates the state and costs of man-machine systems. The categories of men and machines, requiring a test, TEST-LE[.] = true, all the man-machine systems are considered if sufficient elements of that category, LE-SQN2 are still available by calling AVL-TEST (.), Table 4.65, that sets AVL [category]. Afterwards this va-

Table 4.73 Class UPDT-MAN-MCH and its declaration of variables.

```
Process class UPDT_MAN_MCH;                        !**************************************;
begin
   !     d     e     c     l     a     r     a     t     i     o     n;

   boolean array                                   !----------------------;
   TEST_LE [1:LE_SQNS],                            !true if test is required;
   COSTS_CHNG [1:LE_SQNS];                         !true if change of costs occur for man;
   real array                                      !----------------------;
   COSTS_INCRSE [1:LE_SQNS];
```

Table 4.74 Procedure STATE-COSTS of class UPDT-MAN-MCH.

```
procedure STATE_COSTS_;                            !----   ----   ----   ----   ----;
begin                                              !called in dynamic;
   integer
   LE_SQN2, MM_S1;
   ADMN.DISPLAY_DATA_;                             !update before states are changed;
   for LE_SQN2:= 1 step 1 until LE_SQNS do         !only if test_le is set to true in shift;
   if TEST_LE [LE_SQN2] then begin
      for MM_S1:= 1 step 1 until MN_MCH_SYSTMS do
      if MM_S [MM_S1] =/= none then MM_S [MM_S1].AVL_TEST (LE_SQN2);
      TEST_LE [LE_SQN2]:= false;
   end;
   for MM_S1:= 1 step 1 until MN_MCH_SYSTMS do
   if MM_S [MM_S1] =/= none then MM_S [MM_S1].AVLBLTY_TEST;
   for LE_SQN2:= 1 step 1 until LE_SQNS do
   if COSTS_CHNG [LE_SQN2] then begin
      for MM_S1:= 1 step 1 until MN_MCH_SYSTMS do if MM_S [MM_S1] =/= none then
      MM_S [MM_S1].COST_CHANGE (COSTS_INCRSE [LE_SQN2],LE_SQN2);
      COSTS_INCRSE [LE_SQN2]:= 0.0;
      COSTS_CHNG [LE_SQN2]:= false;
   end;
end;
```

Table 4.75 Dynamic section of class UPDT-MAN-MCH.

```
!     d     y     n     a     m     i     c;
!           r e p e a t;
while not END_EXPRMNT do begin
   integer M1;

   STATE_COSTS_;
   for M1:= 1 step 1 until MATRLS_PROC do
   if MATRL [M1].M_ADMN =/= none then MATRL [M1].M_ADMN.ADMN_M_STATE_;
   !                          for correct administration of passive & no_gangs;
   Passivate;                 !activated in sh_wk, sh_prd, equipment;
end xx of while;
```

riable is used in AVLBLTY-TEST, Table 4.65, called subsequently for each man-machine system. Finally the costs are changed in each system if the costs are changed for a category of men or machines, LE-SQN2. The dynamic section is shown in Table 4.75; it is repeated as long as the experiment is not ended. It calls STATE-COSTS, an update of administration of materials (belongs to the experimental frame) and Passivate. This section is activated by scheduling this process when a failure occurred to a machine or a machine is repaired or in shifts of men (Section 4.4 Tables 4.103 and 4.105). Table 4.76 shows the changes in the state of men, machines and man-machine systems and their causes; see also Section 2.2.2 'Men and machinery'.

The sets containing man-machine systems, MM-SYSTMS-SET, Table 4.77, only contain a queue MM-S-SET-Q. Such a set is used during decision to select one if any of the man-machine systems in the queue to perform operations. So at most one system can be selected from each set. The use of sets is already described in Section 2.1.2 'Men and machinery' with the discussion of Table 2.5.

In order to create combinations from gangs the class of combinations is extended by defining a subclass GANG-SET-GNRTD for those combinations generated in a procedure. Table 4.78 shows the subclass with one va-

Table 4.76 Transformation of states in men, machines and man-machine systems and their causes.

Current state	Resulting state		
	RUN	PASSIVE	DOWN
RUN		decision	man (worktime ends) machine (failure orcurs) system (availability ends)
PASSIVE	decision		man (worktime ends) machine (service needed) system (availability ends)
DOWN	x	man (worktime begins) machine (service or repair done) system (availability begins)	

Note that system means man-machine system; their availability is originated by men and machines. The change from RUN to DOWN also requires a decision to stop operations.

Table 4.77 Class MM-SYSTMS-SET.

```
COMPONENT class MM_SYSTMS_SET;                   !***************************************;
!                                                !a gang or combination belongs to one set;
!                                                !in decision the sets are handled;
!                  !independently with respect to the needed categories of men and machines;
begin
   ref (Head) MM_S_SET_Q;

   MM_S_SET_Q:- new Head;
end ** of class set of man-machine systems;
```

riable ACCEPTABLE and two procedures; INIT-INPUT- is an empty version of the 'virtual' declared procedure to prevent the use of previous definitions in superclasses; REQUIRED-LE adds from two man-machine systems the required number of elements of men and machine and checks if that number is on the farm and the new generated system is acceptable. If so it calculates costs, assigns the required gangs and activates further initialization. This procedure is called in a procedure generating combinations. Table 4.79 shows the generating procedure COMBS-G-GENERATION. It

Table 4.78 Class GANG-SET-GNRTD.

```
GANG_SET class GANG_SET_GNRTD;                       !*************************************;
begin
    boolean ACCEPTABLE;                              !----------------------;

    procedure INIT_INPUT_;;

    procedure REQUIRED_LE (MMS1, MMS2);              !---  ----  ----  -----  -----;
    ref (MAN_MACH_SYS) MMS1, MMS2;                   !called in combs_g_generation;
    begin
        ACCEPTABLE:= true;
        for LE_SON:= 1 step 1 until LE_SONS do begin
            RQRD_NMB_LE [LE_SON]:= MMS1.RQRD_NMB_LE [LE_SON] + MMS2.RQRD_NMB_LE [LE_SON];
            ACCEPTABLE:= ACCEPTABLE and RQRD_NMB_LE [LE_SON] <= LE_NMB [LE_SON, TOTAL];
        end;
        if ACCEPTABLE then begin
            COSTS_H:= MMS1.COSTS_H + MMS2.COSTS_H;
            COSTS_H_FXD:= MMS1.COSTS_H_FXD + MMS2.COSTS_H_FXD;
            RQRD_GNG [1]:= MMS1.RQRD_GNG [1];
            for GN:= 2 step 1 until GNGS do RQRD_GNG [GN]:= MMS2.RQRD_GNG [GN -1];
            activate this GANG_SET;                  !initial statements executed;
        end;
    end;
end of gang set generated;
```

Table 4.79 Procedure COMBS-C-GENERATION of class SFOBASE-MODEL.

```
procedure  COMBS_G_GENERATION;                       !----  ----  ----  ----  ----;
begin                                                !called in experimental model if wanted;
    boolean                                          !----------------------;
    G_AGAIN;                                         !true if gang is accepted in loop;
    integer                                          !----------------------;
    SET, GNRTD;                                      !set of mm_systems, generated combinations;
    text                                             !----------------------;
    T_C, T_C1, T_C2;
    ref (Head)                                       !----------------------;
    GANGS_CONSDR_Q, GANGS_AGAIN_Q, MMS_CONSDR_Q, COMBS_GENRTD_Q;!queues of man-machine systems;
    ref (RECORD_MM_S) R_MMS, R_G;
    ref (MAN_MACH_SYS) MMS, G_R;
    ref (GANG_SET_GNRTD) COMB;

    for SET := 1 step 1 until MM_S_SETS do begin
        Outtext ('* * *  Set of gangs and combinations:'); Outint (SET,6);
        Outtext ('  contains:');Outimage;
        GANGS_CONSDR_Q:- new Head;                   !queue of gangs considered;
        MMS_CONSDR_Q:- new Head;                     !queue of man-machine systems considered;
        R_MMS:- MM_S_SET [SET].MM_S_SET_Q.First;
        while R_MMS =/= none do begin
            MMS:- R_MMS.MN_MCH_SYS;
            new RECORD_MM_S (MMS).Into (GANGS_CONSDR_Q);    !existing gangs in queue;
            new RECORD_MM_S (MMS).Into (MMS_CONSDR_Q);
            Outtext ('Existing gang is:  '); Outtext (MMS.NAME_COMP);
            Outtext ('  and has number Mm_s and G :'); Outint (MMS.MM_S_SON,6); Outimage;
            R_MMS:- R_MMS.Suc;
        end;
```

Table 4.79 (continued)

```
while not GANGS_CONSDR_Q.Empty and not MMS_CONSDR_Q.Empty and GNRTD < COMBS_G*3
do begin
    T_C:- Copy ('G    &Mm_s    ');
    T_C1:- T_C.Sub (2,2);
    T_C2:- T_C.Sub (10,3);
    GANGS_AGAIN_Q:- new Head;
    COMBS_GENRTD_Q:- new Head;
    R_G:- GANGS_CONSDR_Q.First;
    while R_G =/= none do begin
        G_AGAIN:= false;
        G_R:- R_G.MN_MCH_SYS;                          !gang;
        R_MMS:- MMS_CONSDR_Q.First;
        while R_MMS =/= none do begin
            MMS:- R_MMS.MN_MCH_SYS;                    !man-machine system;
            if G_R.MM_S_SQN < GANG [MMS.RQRD_GNG [1]].MM_S_SQN then begin
                !                              !only gangs with mm_s_sqn < are considered;
                !because 2 gangs with same mm_s_sqn (=) refer both to one operation!;
                T_C1.Putint (G_R.MM_S_SQN);
                T_C2.Putint (MMS.MM_S_SQN);
                COMB:- new GANG_SET_GNRTD
                (T_C,SH_PERD[0],SH_WKLY[0],SET,G_R.GNGS + MMS.GNGS);
                !                                      !new combination generated;
                COMB.REQUIRED_LE (G_R, MMS);
                if COMB.ACCEPTABLE then begin
                    G_AGAIN:= true;                    !gang used at least once;
                    GNRTD:= GNRTD + 1;
                    COMB.MM_S_SQN:= GANGS + GNRTD;
                    Outtext ('Acceptable new combination consists of:  ');
                    Outtext (COMB.NAME_COMP); Outtext ('  and has number Mm_s:');
                    Outint (COMB.MM_S_SQN,6); Outimage;
                    new RECORD_MM_S (COMB).Into (COMBS_GENRTD_Q);
                    if GNRTD <= COMBS_G *3 then
                    MM_S [GANGS + GNRTD]:- COMB_G [GNRTD]:- COMB
                    else begin
                        Outtext
                        ('You exceed the available number of references to '
                        'combinations:');
                        Outint (COMBS_G *3,6); Outimage;
                        Outtext ('  by generating new ones upto'); Outint (GNRTD,6);
                        Outtext ('  Reference in system is impossible.'); Outimage;
                    end;
                end;
            end;
            R_MMS:- R_MMS.Suc;
        end;
        if G_AGAIN then new RECORD_MM_S (G_R).Into (GANGS_AGAIN_Q);
        R_G:- R_G.Suc;
    end;
    GANGS_CONSDR_Q:- GANGS_AGAIN_Q;              !queue of gangs used at least once;
    MMS_CONSDR_Q:- COMBS_GENRTD_Q;               !queue of combinations generated that;
  end;                                           !can be considered for adding a gang;
end of sets;
COMBS_G:= IMIN (COMBS_G *3, GNRTD);              !adjust overrated range of references;
end;
```

generates combinations for each set of man-machine system seperately. The set contains the gangs considered; these gangs are queued in two queues; one queue contains the gangs considered GANGS-CONSDR-Q and the other queue contains the man-machine systems considered MMS-CONSDR-Q for generating new combinations. Initially both queues are identical. A gang and a man-machine system are taken (the sequential number MM-S-SQN of the man-machine system is higher than that of the gang); a new combination referred by COMB is generated, REQUIRED-LE (Table 4.78) is called and if acceptable some information is written and the combination is placed in a queue of generated combinations COMBS-GENRTD-Q and references MM-S[.] and COMB-G[.] are assigned to it. In REQUIRED-LE the new combination is activated which results in its incorporation in the set of man-machine systems (Table 5.49). A message is send when the number of available references (an overestimated number of three times those defined in input; COMBS-G*3) is exceeded by the number of combinations generated. If the gang was used at least once in generating a new combination, then it is placed in a queue of gangs that have to be considered again GANGS-AGAIN-Q otherwise it need not be considered any more. After considering all the gangs from the queue GANGS-CONSDR-Q, an attempt is made again to generate combinations from the gangs in queue GANGS-AGAIN-Q and the combinations in COMBS-GENRTD-Q and so on. This is allowed by giving those queues the references GANGS-CONSDR-Q and MMS-CONSDR-Q, respectively. Finally the number of combinations is adjusted. It is sufficient to generate a new combination only if the added gang has a lower number MM-S-SQN than those in the man-machine system.

Exercise: What is the reason that only gangs with a lower number MM-S-SQN have to be considered? Can you explain why a gang not placed in GANGS-AGAIN-Q need no further consideration with man-machine systems from COMBS-GENRTD-Q? The reference will become MMS-CONSDR-Q.

It is not allowed to create a combination with one gang included twice (or more) because such a combination will try to use the same gang (and operation) twice and that is impossible after the state changed into RUN (Table 4.69). The only solution is to take care of sufficient identical gangs (and operations) in input with different MM-S-SQN. If gangs A, B and C are identical, it is suggested that gang A and gang BC are introduced; the latter incorporates gangs B and C, has twice the capacity of A and uses two times the number of men and machines of A. The only generated combination will be A & BC.

Exercise: Can you find the four generated combinations if gangs A, B and C were introduced? The following 'multiple' gangs consist of four, eight, etc. times gang A; do you agree?

The difference between generated combinations and combinations defined in input is that in input unrealistic combinations can be prevented, for instance, when there are two gangs with different capacity but performing the same type of operation, then it is unrealistic to select a combination including the gang with the lower capacity instead of the other combination involving the gang with the higher capacity; therefore the former combination is not needed.

Exercise: Do you see the relation between the number of references and combinations requested from input and the reason for overestimating the number of references available for generating combinations (Table 4.1) and adjusting it afterwards (Table 4.79)?

4.3 Operations and decision

The following sections describe the decision subsystem: two types of operations (one for materials and one for service and repair) and a preliminary design of decision are described.

4.3.1 Operations processing material

To describe the base model of scheduling farm operations, the theory given in the Sections 2.1.3 and 2.2.3 'Operations and decision' is used. Before the details of the program are described in the following sections the relationships between the procedures and the use of the procedures over time are considered. These relationships between procedures of an operation are essential for such a dynamic counterpart of a gang (the static part of physical entities such as man and machines). The scheme in Table 2.16 of Section 2.2.3 illustrates the changes over time for two variables. The pro-

Table 4.80 A history of an operation; state, causes and procedures.

State variables of operation STATE	 RUN_PHASE	Procedures used due to a cause
DOWN	INACTIVE	Gang becomes passive; Material processed becomes passive or processing condition o.k.; STATE_CH_OP_
PASSIVE		A decision is made to start the operation; START_OPRTN_
RUN		Materials processed and produced are informed; GO_AHEAD
	SETUP_WAIT	The setup of the operation needs time; Hold (.)
	MAT_WAIT	Check if all materials processed are there; WAIT_MAT_PRC
	PREPARED	The operation starts with work on a field; GO_ON_
	BUSY	When a field is processed; QNT_TRANSFER
	GO_ON_WAIT	When another field can be processed; GO_ON_
	BUSY	When a storage (e.g. a drier) becomes full; QNT_TRANSFER
	GO_ON_WAIT	etc.
		A decision is made to stop the operation; STOP_OPRTN_ (if the operation could continue)
PASSIVE		(if the operation is prohibited due to the gang or the materials processed; STATE_CH_OP_)
DOWN		FINISH
	INACTIVE	

(blank value of variable means the value at the preceding line) gram names are now used in a similar scheme as shown in Table 4.80.

The scheme stresses the relationships; the following discussion will help to illustrate what may happen in time. An operation is DOWN if the appropriate gang is not available or the material processed is not available or workable or the required conditions for processing are not fulfilled. STATE-CH-OP- is called when one of these situations changes with the result that STATE may change to PASSIVE. A decision is now meaningful. As a result of a decision START-OPRTN- is called which starts the related gang, assembles and starts the men and machines required and activates the operation itself. The variable STATE now is RUN and GO-AHEAD is called which tells the materials processed by this operation that processing is expected and the materials produced that supply of material is expected. The variable RUN-PHASE then becomes SETUP-WAIT. After the use of the set-up duration by calling Hold (.) RUN-PHASE becomes MAT-WAIT. WAIT-MAT-PRC is then called which checks if the material(s) processed are available (or delivered by another operation). If this is true, then the state variable RUN-PHASE becomes PREPARED for operating, but otherwise no further action takes place until WAIT-MAT-PRC results in the situation where waiting is no longer needed (i.e. the material processed will be delivered), for instance, the grain drying operation has to wait until the combine-harvesting operation has completed its set-up and wet grain is actually delivered although the grain drying gang itself has no set-up time. The operation continues by calling GO-ON-, which calculates the actual capacity for processing (rate of operation), tells the material(s) processed by the operation that processing is started and the material(s) produced that the delivery is started. It also calls procedures in the materials processed and produced which calculate the moments when the field is finished and the maximum quantity is achieved, respectively. The variable RUN-PHASE becomes BUSY. At the event moment (finish of field or maximum achieved the procedure QNT-TRANSFER is called which delivers the processed quantity to the materials produced and consumes the quantity from the material processed, for instance, at the moment a wheat field is finished or the drier capacity is achieved the operation is requested to transfer the harvested acreage, i.e. to subtract the acreage from wheat and to add it to straw and to wet grain. The variable RUN-PHASE becomes GO-ON-WAIT. The operation continues with a following field if possible. This sequence of GO-ON and QNT-TRANSFER is repeated until a decision is made to stop the operation and STOP-OPRTN- is called, which stops the related gang and the men and machines involved. The variable STATE becomes PASSIVE if the operation can be used (gang available and processed material(s) available and workable) and DOWN otherwise. Procedure FINISH is then called, which tells the material(s) processed that processing stops and the materials produced that delivery finishes. The variable RUN-PHASE becomes INACTIVE. The above description of the story of an operation stresses the relationships between an operation and the related components of the system such as men, machines and materials. It is however impossible to illustrate in one scheme each possible story (= sequence of procedures). The decision to stop the operation is, for instance, also possible when RUN-PHASE still equals SETUP-WAIT. The description is given in full detail in the following sections with the discussion of the program. In Chapter 6 'Verification and validation' the events are described and a technique is demonstrated which verifies the correctness of the procedures and of the relationships between components (men, machines, gangs, operations, materials and decision).

So far the operation has been described as a link between a material processed and a material produced:

Material processed -- Operation -- Material produced.

There is also another relationship that needs attention; the material as a link between an operation producing it and an operation processing it:

Operation producing -- Material -- Operation processing.

A combine-narvester operation, for instance, produces wet grain that is processed by the grain drier. The scheme in Table 4.81 illustrates the connections between the producing operation A and the processing operation B as caused by material M over time (starting from a decision until the moment the operations are working). Operation A can work (STATE = PASSIVE) but operation B has no material for processing and cannot work (STATE = DOWN). Because of a decision to start operation A (by calling its START-OPRTN-), the state changes to RUN and GO-AHEAD warns material M (wet grain) that is produced by A (harvesting). The state of M changes and accordingly the state of operation B (drying) by calling its STATE-CH-OP-(resulting in PASSIVE) and activating decision. Operation A becomes now in RUN-PHASE = SETUP-WAIT and starts the set-up duration by calling Hold (.). Now the decision is made and results in the calling of START-OPRTN- of B and in the appropriate state (= RUN). Subsequently GO-AHEAD, Hold (.) and WAIT-MAT-PRC are called and RUN-PHASE is updated to SETUP-WAIT and MAT-WAIT. The set-up duration of B is assumed to be smaller than that of A. Operation B is, for instance, the wet grain drying operation and its set-up duration is zero; it is now necessary to wait until the set-up duration of the combine-harvesting operation A is completed, WAIT-MAT-PRC of A is called and the operation can start its work on the field and the wet grain M can be delivered. GO-ON- of operation A is called and via material M WAIT-MAT-PRC of operation B is called that now activates B; this results in calling GO-ON- of B. Both operations are now in RUN-PHASE = BUSY. When only a small quantity of

Table 4.81 A history of two operations producing and processing one material.

Operation producing M State variables of operation A		Comment, Procedures in operation		Operation processing M State variables of operation B	
STATE	RUN_PHASE	A	B	STATE	RUN_PHASE
PASSIVE	INACTIVE			DOWN	INACTIVE
		A decision starts A;			
		START_OPRTN_			
RUN					
		GO_AHEAD			
		The material M produced			
		by A changes its state			
		and that of B;			
			STATE_CH_OP_		
				PASSIVE	
		and activates decision.			
	SETUP_WAIT				
		Hold ( )			
		A decision starts B;			
			START_OPRTN_		
				RUN	
			GO_AHEAD		
					SETUP_WAIT
			Hold ( )		
					MAT_WAIT
			WAIT_MAT_PRC		
		Because material M processed			
		by B is not yet available			
		(due to setup of A) B has to wait.			
		After setup of A			
	MAT_WAIT				
		WAIT_MAT_PRC			
	PREPARED				
		GO_ON_			
		Material M will be delivered, so			
		again	WAIT_MAT_PRC		
	BUSY				PREPARED
			GO_ON_		
					BUSY

(blank value of variable means the value at the preceding line)

material M was available, operation B had processed M first before the set-up of A was finished and operation A will be warned that M is required; also in that case WAIT-MAT-PRC of B is called after the set-up of A (Section 4.1.3.2 'Delivery of material'). So WAIT-MAT-PRC of B is called after set-up of A either by DLVRY-STARTD of M (Table 4.17) or by DLVRY-CONTND of M if M was already processed by B (Table 4.18). Operation B could have a similar effect on operation C via material N; so a chain of connected operations and materials exist. Think about combine-harvesting, straw baling, bale gathering and ploughing.

Exercise: Do you think that the state of straw baling becomes PASSIVE when the straw is delivered by the combine-harvesting but the moisture content of the straw is still too high (straw remains unworkable)?

The subsequent phases of an operation and the consequences on other operations will now be considered. These phases are in general independent of the phases of other operations, except in cases where operation A delivers material M and operation B processes M at the same time and either M starts without any acreage available for processing or processing by operation B is faster than delivering by A. This situation results in idle time of operation B due to shortage of M. Because such a combination of operations was accepted with capacity of A less than capacity of B in the scheduling system (formulated in input) and decision selected this combination as the most urgent one this situation must be accepted as legal; Table 4.30 shows procedure ADJUST-QUANT where the quantity processed, QNT-PROC-OPR, is adjusted to the quantity available, QUANT-AVLBL, and where a warning message is not send if CAP-OPRS-DLV < CAP-OPRS-PRC (Section 4.1.3.3 'Processing of material' the description of Table 4.30, task iii). Further direct relationships between operations A and B are not prescribed in the program of the scheduling system; the indirect relationships are described in the following sections where the relationship between operation and material plays a central role.

After the above general explanation of the relationships between materials due to an operation and of the relations between operations due to a material, a more detailed description is given in the following sections.

A reader not interested in all the programming details can omit the Sections 4.3.1.1-5 and continue with the service and repair operations (Section 4.3.2).

In the following sections distinction is made between (1) a general part of an operation (the variables, the state), the parts concerned with (2) starting an operation, (3) using and (4) stopping it, and finally (5) the dynamic part of the 'living' component.

4.3.1.1 State of operation

In Table 4.82 class OPERATION is shown, which consists only of declaring the common attributes of the two types of operation such as 'virtual' procedures, possible situation of a phase within the dynamics of operation (INACTIVE, etc.) and a reference to the counterpart, the gang. The phases are initialized to specific integer values.

Table 4.83 shows the variables of class OPRTN-MATRL, an operation for materials; most of the variables will become clear by describing their function in the following sections. At most four materials can be processed or can be produced and are referred to as MAT-PROC [i] and MAT-PROD [i]. The actual number is prescribed by materials processed MATRLS-PRCSD and materials produced MATRLS-PRDCD, where the last one is a subset of the materials produceable, MATRLS-PRDCBL. An example is an operation harvesting wheat, which processes wheat and produces straw and dry or wet grain. So from the three produceable materials only two are produced at a specific moment; it is also allowed to use one gang and one operation to produce dry grain and others to produce wet grain.

Table 4.82 Class OPERATION; virtual procedures, declaration of variables and initial section.

```
COMPONENT class OPERATION;                             !*************************************;
virtual: procedure START_OPRTN_, STOP_OPRTN_, STATE_CH_OP_,GO_ON_;
begin
   !     d     e     c     l     a     r     a     t     i     o     n;

   integer                                              !---------------------;
   RUN_PHASE,                                           !phase defined in dynamic section:;
   INACTIVE, SETUP_WAIT, MAT_WAIT, PREPARED, BUSY, GO_ON_WAIT;
   real                                                 !---------------------;
   COSTS_GNG, START_TIME;
   ref (MAN_MACH_SET) GANG_G;                           !---------------------;

   !     i     n     i     t     i     a     l;

   STATE:= STATE_NEXT:= DOWN;
   RUN_PHASE:= INACTIVE:=0; SETUP_WAIT:=1; MAT_WAIT:= 2; PREPARED:= 3; BUSY:=4; GO_ON_WAIT:=5;
end ** of class operation;
```

Table 4.83 Class OPRTN-MATRL; declaration of variables.

```
OPERATION class OPRTN_MATRL;                           !*************************************;
begin
   !     d     e     c     l     a     r     a     t     i     o     n;

   integer                                              !---------------------;
   I2, MATRLS_PRCSD, MATRLS_PROCD, MATRLS_PROCBL, CNDTN_MT_PRC;
   integer array                                        !---------------------;
   M_PRC, M_PRD [1:4],                     !auxiliary variables, max. 4 materials;
   PRCSNG_CNDTN [1:3,1:4];                 !possible conditions (max.3) of materials processed;
   real                                                 !---------------------;
   SETUP_OPR,                              !setup duration from gang;
   QUANT, QUANT_PRC_OPR,                   !actual/ cum.quantity processed;
   QNT_DAY_PR_O,                           !quantity processed until previous day (incl.);
   CAPCTY_ACTL, CAP_FRAC, CAP_CHNG,!capacity [ha/h]/ -fraction/ -change of capacity;
   QNT_NOT_PROC;                           !due to delivering at a slower rate than capcty_actl;
   !long; real                                          !---------------------;
   LT_MAT_TRNSF, LT_PROC_UPDT;             !last time of transfer/ update of processed quantity;
   boolean                                              !---------------------;
   MAT_PRD_RQRD;                           !true if material produced is required in subsequent oper.;
   boolean array                                        !---------------------;
   PRCSNG_MT_OK [1:4];                     !processing condition of materials ok;
   ref (MATERIAL) MP;                                   !---------------------;
   ref (PROCESSED_MAT) array MAT_PROC [1:4];            !max. 4 materials processed;
   ref (MATERIAL) array MAT_PROD [1:4];                 !max. 4 materials produced;
```

Each material processed can define processing conditions, for instance, wheat-harvesting conditions may be a moisture content of grain less than 19% (dry grain), a moisture content 19-23% (wet grain); an operation can refer to one, two or three such conditions as acceptable for processing that material. A combine-harvesting operation with two men may accept, for instance, the dry grain only and the same operation with one man may accept both dry and wet grain. Another example is that a cultivating operation and a ploughing operation require their own range of moisture content of the soil; so a number of processing conditions have to be defined for the soil and each operation processing it refers to at most three appropriate ranges.

The change of the state of the operation is shown in Table 4.84. Procedure STATE-CH-OP- consists of two parts, one finding the next state and the other changing the state; the last part is as usual. The finding of a next

Table 4.84 Procedure STATE-CH-OP- of class OPRTN-MATRL.

```
procedure STATE_CH_OP_;                              !----  ----  ----  ----  ----;
!                                     !called in gang.state_ch_mms, state_ch_mat,attr_updt_m_;
begin
    if STATE = STATE_NEXT then begin
        for I2:= 1 step 1 until MATRLS_PRCSD do PRCSNG_MT_OK [I2]:=
        MAT_PROC [I2].M_PRC_CNDTN [PRCSNG_CNDTN [1, I2]] or
        MAT_PROC [I2].M_PRC_CNDTN [PRCSNG_CNDTN [2, I2]] or
        MAT_PROC [I2].M_PRC_CNDTN [PRCSNG_CNDTN [3, I2]];
        CNDTN_MT_PRC:= 0;                      !use of run = 1 < passive = 2 < down = 3;
        for I2:= 1 step 1 until MATRLS_PRCSD do CNDTN_MT_PRC:= IMAX (CNDTN_MT_PRC,
        if MAT_PROC [I2].STATE = DOWN or not PRCSNG_MT_OK [I2] then DOWN
        else MAT_PROC [I2].STATE);
        STATE_NEXT:= if GANG_G.STATE = DOWN or CNDTN_MT_PRC = DOWN
        then DOWN else GANG_G.STATE;
    end;
    if STATE <> STATE_NEXT then begin
        STATE_PREV:= STATE;
        TIME_C_ACCUM;                          !costs from gang, costs_h are 0.0;
        if STATE = RUN  !stop needed;  or STATE = DOWN  !consider start;
        then activate DECIDE at Time;          !due to change of material or gang;
        STATE:= STATE_NEXT;
    end;
end ** of change of state of operation;
```

state is necessary for it is not prescribed before calling this procedure. Two other states are considered: the state of the gang and the state of the materials processed along with their processing conditions. If the state of the gang, GANG-G.STATE equals DOWN or the state of the materials processed or the processing conditions do not fit, CNDTN-MT-PRC is DOWN then STATE-NEXT becomes DOWN; otherwise the materials can be processed and the next state is that of the gang. All the materials processed are checked to show that all relevant processing conditions, M-PRC-CNDTN[.,.] are in order and the condition PRCSNG-MT-OK[.] is set accordingly. The state of the material processed and PRCSNG-MT-OK both influence CNDTN-MT-PRC by taking the maximum of STATE or DOWN. Use is made here of the fact that STATE is assigned with variables initialized to integer values: RUN (= 1) or PASSIVE (= 2) or DOWN (= 3). This procedure STATE-CH-OP- is called when the state or conditions of a processed material are changed (Table 4.14 and 4.45) or when the state changed in the related gang (Table 4.64); the latter change can be caused by the start of worktime of men. The resulting changes in STATE are shown in the scheme of Table 4.85 with the causes (also Table 2.15 in Section 2.2.3

Table 4.85 The transformation of states of operations due to events.

Current STATE	Resulting STATE of an operation: RUN	PASSIVE	DOWN
RUN		decision	gang, material
PASSIVE	decision		gang, material
DOWN	(not allowed)	gang, material	

'Operations and decision').

Another state variable of an operation is RUN-PHASE, which is changed in the dynamic part and can be:

- INACTIVE, before start and after stop of operation,
- SETUP-WAIT, MAT-WAIT, just before and after set-up,
- PREPARED, everything in order to continue with processing,
- BUSY while processing,
- GO-ON-WAIT, just after transfer of quantity.

This state variable is necessary during several procedures in the following sections to check if the procedure can be executed properly.

4.3.1.2 Set-up of operation

In this section three procedures are described which are related to the set-up of an operation from the beginning until it begins with processing. Set-up is for example to gather the machines, to refuel tractors and to go to the field (it here also includes the teardown of an operation after the work is done).

Table 4.86 shows procedure START-OPRTN-. The start of an operation is requested by decision and is only possible if the operation is in STATE = PASSIVE and RUN-PHASE = INACTIVE. It starts the work of a gang GANG-G.START-WORK-G (Table 4.69; that calls to change the state of the operation Table 4.84) and activates its own dynamic part. In between some properties of the materials processed such as capacity ratio and timeliness value are updated only to be in time for some output requested just after deciding. The dynamic part (Table 4.93) of operation calls GO-AHEAD, Table 4.86, that assigns appropriate values to the set-up duration, SETUP-OPR, the current costs of the gang, COSTS-GNG, the quantity

Table 4.86 Procedures START-OPRTN- and GO-AHEAD of class OPRTN-MATRL.

```
procedure START_OPRTN_;                                !----   ----   ----   ----   ----;
if STATE = PASSIVE and RUN_PHASE = INACTIVE then begin          !called in decision;
    GANG_G.START_WORK_G;                               !calls also state_ch_op_;
    for I2:= 1 step 1 until MATRLS_PRCSD do inspect MAT_PROC (I2) do
    if not  PROCESSING then begin
        ATTR_FLD_M_;                                   !updates cap_ratio;
        ATTR_INTG_F_;                                  !updates tml_value;
    end;                                               !update before decision report;
    activate this OPRTN_MATRL at Time;
end;

procedure GO_AHEAD;                                    !----   ----   ----   ----   ----;
begin                                                  !called in dynamic;
    SETUP_OPR:= GANG_G.SETUP_GNG;
    COSTS_GNG:= GANG_G.COSTS_MADE;
    QUANT:= 0.0;
    START_TIME:= Time;
    for I2:= 1 step 1 until MATRLS_PRCSD do MAT_PROC (I2).PRCSNG_EXPCT;
    for I2:= 1 step 1 until MATRLS_PRDCD do MAT_PROD (I2).SUPPLY_EXPCT;
end ** of go_ahead with setup of operation;
```

Table 4.87 Procedure WAIT-MAT-PRC of class OPRTN-MATRL.

```
procedure WAIT_MAT_PRC;                              !----   ----   ----   ----   ----;
begin                                    !called in dynamic, mat.dlvry_startd, .dlvry_contnd;
   boolean NO_WAIT_NDD;
   if RUN_PHASE = MAT_WAIT or RUN_PHASE = GO_ON_WAIT then begin
      NO_WAIT_NDD:= true;
      for I2:= 1 step 1 until MATRLS_PRCSD do inspect MAT_PROC (I2) do begin
         NO_WAIT_NDD:= NO_WAIT_NDD and (QUANT_AVLBL > 1.0&-4 or DELIVERING);
      end;
      if NO_WAIT_NDD then reactivate this OPRTN_MATRL at Time
      else Cancel (this OPRTN_MATRL);
   end;
end;
```

processed, QUANT, and the time at start, START-TIME. Furtheron it calls PRCSNG-EXPCT to let the materials processed know that processing is expected (Table 4.26) and SUPPLY-EXPCT to let the materials produced know that supply can be expected (Table 4.17). The latter may result in starting processing of this delivered material at the same moment. At the same moment STATE becomes RUN for the operation, the gang, the men and machines involved and the material processed!

The third procedure, WAIT-MAT-PRC, is shown in Table 4.87. It is called in dynamic after set-up is passed and now checks if there is any quantity available for processing or delivery occurs for the materials processed. If no waiting is needed, NO-WAIT-NDD = true, dynamic is reactivated, otherwise the process is cancelled from the event list and remains waiting until this procedure is called to check again. Such a call occurs from DLVRY-STARTD or DLVRY- CONTND in material (Tables 4.17 and 4.18; Section 4.1.3.2 'Delivery of material').

Exercise: Can you remember the different situations of these two procedures? In the first case the operation has just finished set-up (RUN-PHASE = MAT-WAIT; waiting for material) and in the second case the operation is in phase GO-ON-WAIT because it processed all the available material before delivery occurred. Can you remember examples of both cases (Section 4.1.3.2)?

4.3.1.3 Actual use of operation

This section consists of two main parts: the first one handles the regular procedures and the second part the procedures needed when intermediate changes occur.

The first main part contains two procedures, one called when the process starts, GO-ON- and the other when some quantity has to be transferred, QNT-TRANSFER. These two are repeatedly called in dynamic. GO-ON- is shown in Table 4.88 and shows three if-statements mainly based on the phase of dynamic (Section 4.3.1.5 'Dynamic aspects of operation'). If RUN-PHASE = PREPARED the set-up duration is just passed. The first if-statement updates a capacity fraction, CAP-FRAC, as a multiplication (an

Table 4.88 Procedure GO-ON- of class OPRTN-MATRL.

```
procedure GO_ON_;                                      !----   ----   ----   ----   ----;
begin                                                  !called in dynamic;
    if RUN_PHASE = PREPARED then begin
        CAP_FRAC:= 1.0;
        for I2:= 1 step 1 until MATRLS_PRCSD do CAP_FRAC:= MAT_PROC [I2].CAP_RATIO * CAP_FRAC;
        CAPCTY_ACTL:= GANG_G.CAPACITY * CAP_FRAC;
        for I2:= 1 step 1 until MATRLS_PRCSD do MAT_PROC [I2].START_PRCSNG (this OPRTN_MATRL);
        for I2:= 1 step 1 until MATRLS_PRDCD do  MAT_PROD [I2].DLVRY_STARTD(this OPRTN_MATRL);
    end;
    if MAT_PRD_RQRD then
    for I2:= 1 step 1 until MATRLS_PRDCD do MAT_PROD [I2].DLVRY_CONTND;
    MAT_PRD_RQRD:= false;
    if RUN_PHASE = PREPARED or RUN_PHASE = GO_ON_WAIT then begin
        LT_PROC_UPDT:= LT_MAT_TRNSF:= Time; QUANT:= 0.0;
        for I2:= 1 step 1 until MATRLS_PRCSD do MAT_PROC [I2].GO_UNTIL_MP;
        for I2:= 1 step 1 until MATRLS_PRDCD do  MAT_PROD[I2].ACCEPT_UNTIL;
    end;
end ** of go_on with processing;
```

assumption!) of all the capacity ratios, CAP-RATIO, of the materials processed; such a CAP-RATIO may depend on moisture content or other properties of material or even fields (shape, location etc.) of material. Furtheron it calls in the materials processed START-PRCSNG (Table 4.27) and in the materials produced DLVRY-STARTD (Table 4.17). The second if-statement is used when this operation is warned that the material produced is required for further processing, Table 4.35; MAT-PRD-RQRD is true and DLVRY-CONTND (Table 4.18) of the materials produced is called to activate an operation waiting for processing the material. The third if-statement updates the latest times of processing and material transfer to the current time, makes the quantity processed zero and calls procedures in materials processed to update the event time the field is consumed (Table 4.29) and in materials produced to update the event time a maximum is achieved (Table 4.19).

Procedure QNT-TRANSFER is shown in Table 4.89 and is executed only if the operation is in phase BUSY and the latest time not at the current moment i.e. the transfer is not yet done at Time. The first task is to check in the materials processed if enough quantity is available by calling ADJUST-QUANT (Table 4.30) that adjusts the quantity processed by one or more operations, QNT-PROC-OPR, to the quantity of the material, QUANT-AVLBL, and to that of the quantity, QUANTITY, at a field. If that quantity, QNT-PROC-OPR, processed by several operations, is less than the quantity processed by this operation, QUANT, then delivering was slower than processing and the excess quantity could not actually be processed (although time and capacity were there) and is cumulated in a quantity not processed, QNT-NOT-PROC. No warning is sent to the terminal for this situation can be desired with the given men and machinery. The total amount processed by this operation is accumulated in QUANT-PRC-OPR. The next task is to perform delivery of materials produced, DELIVERY-M- (Table 4.21), that can still use the attributes of the field consumed and to perform

Table 4.89 Procedure QNT-TRANSFER of class OPRTN-MATRL.

```
procedure QNT_TRANSFER;                                  !----  ----  ----  ----  ----;
if RUN_PHASE = BUSY and LT_MAT_TRNSF < Time then begin          !called in dynamic;
    LT_MAT_TRNSF:= Time;
    CAP_QNT_CHNG (1.0);                                   !updates quant;
    if QUANT > 1.0&-4 then begin                          !prevents too small fields;
        for I2:= 1 step 1 until MATRLS_PRCSD do begin
            MP:- MAT_PROC [I2];
            MP.ADJUST_QUANT;                              !sufficient quantity?;
            if MP.QNT_PROC_OPR < QUANT + 24.0&-3 then            !lower or little higher;
            QNT_NOT_PROC:= QNT_NOT_PROC + (QUANT - MP.QNT_PROC_OPR);
            if QNT_TRF_FLD =/= none and MP.QNT_PROC_OPR < QUANT - 24.0&-3       !much lower;
            then inspect QNT_TRF_FLD do begin
                Outimage;
                Outtext ('Warning OPR_MAT.QNT_TRANSFER: Some quantity cannnt be processed');
                Outimage; Outtext ('in Operation,          at Time, Amount processed,');
                Outtext ('Amount in mat., Cum. amount not processed'); Outimage;
                Outtext (NAME_COMP); Outfix (Time,6,12); Outfix (QUANT,6,12);
                Outfix (MP.QNT_PROC_OPR,6,12);Outfix (QNT_NOT_PROC,6,12); Outimage;
            end;
            if MP.QNT_PROC_OPR < QUANT + 24.0&-3 then            !lower or little higher;
            QUANT:= MP.QNT_PROC_OPR;    !adjusts quant to qnt_proc_opr;
        end;
        QUANT_PRC_OPR:= QUANT_PRC_OPR + QUANT;
        inspect QNT_TRF_FLD do begin
            Setpos(1); Outfix(Time,3,10);
        end;
        for I2:= 1 step 1 until MATRLS_PRDCD do MAT_PROD [I2].DELIVERY_M_ (this OPRTN_MATRL);
        for I2:= 1 step 1 until MATRLS_PRCSD do MAT_PROC [I2].CONSUMPTN_M (this OPRTN_MATRL);
        if QNT_TRF_FLD =/= none then QNT_TRF_FLD.Outimage;
        QUANT:= 0.0;
    end;
end ** of mat_transfer of processed quantity;
```

consumption of materials processed, CONSUMPTN-M (Table 4.31). This task is surrounded by output on a printfile referred to by QNT-TRF-FLD (may be none). The quantity is transferred and QUANT becomes zero.

Table 4.90 shows two procedures UPDAT-QNT and CONTINUE; both activate this operation at the current time when a material requests to process or to deliver a material (Table 4.34). UPDAT-QNT activates the dynamic part of the operation only if the phase is BUSY and CONTINUE only if the phase is GO-ON-WAIT. The activation in UPDAT-QNT is 'at Time prior', that means it is scheduled before the current object (a material); so it acts immediately with the result that QNT-TRANSFER is called, the phase updated to GO-ON-WAIT and the operation is passivated by calling Passivate (Table 4.93 in Section 4.3.1.5 'Dynamic aspects of operation'). Now the interrupted execution of statements in material is continued. CON-

Table 4.90 Procedures UPDAT-QNT and CONTINUE of class OPRTN-MATRL.

```
procedure UPDAT_QNT;                                    !----  ----  ----  ----  ----;
!                                                       !called  in mat.process_mat,.deliver_mat;
if RUN_PHASE = BUSY then activate this OPRTN_MATRL at Time prior;
!                                                       !continues with qnt_transfer;

procedure CONTINUE;                                     !----  ----  ----  ----  ----;
!                                                       !called in mat.process_mat,.deliver_mat;
if RUN_PHASE = GO_ON_WAIT then activate this OPRTN_MATRL at Time;
!                                                       !continues with go_on;
```

TINUE schedules 'at Time', that means at the current moment but after all objects scheduled at this moment, so material can execute its own statements until it is canceled or reactivated (Table 4.33 in Section 4.1.3.4 'Dynamics'). These activations could be placed directly in the calling procedures of material, but it is preferred to have the activation and deactivation within the class itself if possible.

The second main part in the use of operations concerns some intermediate changes of capacity or even a change of materials produced. Table 4.91 shows procedure CAP-QNT-CHNG, that is executed only if not yet updated (LT-PROC-UPDT) or the capacity fraction, FRAC, is not equal to one. The quantity processed, QUANT, is updated with the actual capacity, CAPCTY-ACTL times a duration, if the phase is BUSY. If the capacity changes due to properties of material and field, then it calls in the materials processed and delivered CAP-CHNG-PRCSNG (Table 4.28) and CAP-CHNG-DLV (Table 4.20) respectively where the quantity is integrated, the capacity of processing and delivery changes and new event times for finishing a field and achieving a maximum quantity are derived. The following procedure MAT-PROD-CHNG, Table 4.91, changes the materials pro-

Table 4.91 Procedures CAP-QNT-CHNG and MAT-PROD-CHNG of class OPRTN-MATRL.

```
procedure CAP_QNT_CHNG (FRAC);                    !----   ----   ----   ----   ----;
real FRAC;                   !called in m.qnt_intgr_m, (.attr_fld_m_), this qnt_transfer;
if (LT_PROC_UPDT < Time or FRAC <> 1.0) then begin
    if RUN_PHASE = BUSY then QUANT:= QUANT + CAPCTY_ACTL * (Time - LT_PROC_UPDT) * 24.0;
    LT_PROC_UPDT:= Time;                          !capcty_actl = 0.0 if state <> run;
    if FRAC <> 1.0 then begin
        CAP_CHNG:= CAPCTY_ACTL * (FRAC - 1.0);
        CAPCTY_ACTL:= CAPCTY_ACTL * FRAC;
        CAP_FRAC:= CAP_FRAC * FRAC;
        if RUN_PHASE = BUSY  or  RUN_PHASE = GO_ON_WAIT then
        for I2:= 1 step 1 until MATRLS_PRCSD do MAT_PROC[I2].CAP_CHNG_PRCSNG(this OPRTN_MATRL);
        if RUN_PHASE = BUSY  or  RUN_PHASE = GO_ON_WAIT then
        for I2:= 1 step 1 until MATRLS_PRDCD do MAT_PROD[I2].CAP_CHNG_DLV (this OPRTN_MATRL);
    end;
end;

procedure MAT_PROD_CHNG (MD_PRV, MD_NOW);         !----   ----   ----   ----   ----;
integer MD_PRV, MD_NOW;                           !called in matrl.attr_updt_m_;
begin
    integer MD_POS;
    for I2:= 1 step 1 until MATRLS_PRDCD do
    if MAT_PROD [I2] == MATRL [MD_PRV] then MD_POS:= I2;
    if MD_POS <> 0 then begin                     !=0 means md_prv was not producable;
        MAT_PROD [MD_POS]:- MATRL [MD_NOW];
        if STATE = RUN then begin
            MATRL [MD_PRV]. DLVRY_STOP (this OPRTN_MATRL);!process_mat called in attr_updt_m_;
            MATRL [MD_NOW].SUPPLY_EXPCT;
            !                            !md_now can become passive and decide will be informed;
            if RUN_PHASE >= BUSY
            then MATRL [MD_NOW].DLVRY_STARTD (this OPRTN_MATRL);
            if MATRL [MD_NOW].DLVR_ALLWD then begin
                DECIDE.MAT_MAX_QNT:= true;        !operation has to stop;
                activate DECIDE at Time;
            end;
        end;
    end;
end;
```

duced. For example a combine harvesting operation processes wheat and produces straw and dry or wet grain. At the moment the moisture content in the wet range becomes in the dry range, the grain produced must no longer be transferred to the wet grain storage but to the dry grain storage. Wet grain is indexed as MD-PRV, the previous material delivered and dry grain, the material to be delivered from now on is MD-NOW. In array MAT-PROD [ ] there are references to the materials produceable, where the materials produced at a moment are those from 1 to MATRLS-PRDCD. So first the position MD-POS is found (not found assumes MD-NOW already in appropriate position). The main result is that MAT-PROD [MD-POS] refers to MATRL [MD-NOW], and if the operation is used it calls DLVRY-STOP to tell wet grain that delivery stops and secondly starts delivery of dry grain by calling SUPPLY-EXPCT and if set-up is passed also DLVRY-STARTD. If delivery is not allowed (due to maximum storage in use) then DECIDE receives a signal, MAT-MAX-QNT:= true, and is reactivated because the operation cannot continue.

4.3.1.4 Stopping of operation

There are two procedures to stop an operation in use. The first procedure, STOP-OPRTN-, Table 4.92, stops the work of a gang, that disables the men and machines, changes the state of the gang and of the operation (from RUN into PASSIVE or DOWN, Table 4.70) and reactivates its own dynamic part immediately. Reactivate means schedule again even if in set-up (it is interrupted) and 'at Time prior' means that QNT-TRANSFER transfers quantities immediately. The second procedure, TERMNT, Table 4.92, calculates the next set-up duration of the gang as the duration known for a second or following time this day or as the remaining duration if set-up was not completed. The set-up duration is cumulated in SETUPTIME of the gang, the costs of a gang are updated before they are used to accumulate the costs of the operation in COSTS-MADE. The materials processed and

Table 4.92 Procedures STOP-OPRTN- and TERMNT of class OPRTN-MATRL.

```
procedure STOP_OPRTN_;                         !----  ----  ----  ----  ----;
begin                                          !called in decision;
    GANG_G.STOP_WORK_G;                        !calls also state_ch_op_;
    reactivate this OPRTN_MATRL at Time prior; !even if in setup;
end;

procedure TERMNT;                              !----  ----  ----  ----  ----;
if RUN_PHASE <> INACTIVE then begin            !called in dynamic;
    GANG_G.SETUP_GNG:= if START_TIME + SETUP_OPR / 24.0 < Time + 1.0&-4 then GANG_G.SETUP2_N
    else SETUP_OPR - 24.0 * (Time - START_TIME);!remaining setup time;
    GANG_G.SETUPTIME:= GANG_G.SETUPTIME + RMIN (SETUP_OPR, 24.0 * (Time - START_TIME));
    GANG_G.TIME_C_ACCUM;                       !to update costs;
    COSTS_MADE:= COSTS_MADE + (GANG_G.COSTS_MADE - COSTS_GNG);
    for I2:= 1 step 1 until MATRLS_PRCSD do MAT_PROC [I2].STOP_PRCSNG (this OPRTN_MATRL);
    for I2:= 1 step 1 until MATRLS_PRDCD do MAT_PROD [I2].DLVRY_STOP(this OPRTN_MATRL);
    CAPCTY_ACTL:= 0.0;
end ** of terminate operation;
```

produced have to stop processing (STOP-PRCSNG, Table 4.32) and to stop delivery (DLVRY-STOP, Table 4.22), respectively. The actual capacity, CAPCTY-ACTL is made zero. No duration is incorporated for teardown i.e. driving from the field to the barn and putting the machines away; here this duration is already contained in the set-up for reasons of simplicity.

4.3.1.5 Dynamic aspects of operation

The dynamic part of the operation is shown in Table 4.93. This part can be executed time after time until end of experiment, END-EXPRMNT, becomes true. If STATE equals RUN as a result of START-OPRTN- then GO-AHEAD (Table 4.86) is called, the RUN-PHASE is set to SETUP-WAIT and the execution of statements is interrupted and will continue later on; it is rescheduled by Hold( ) to represent the set-up of the operation. After the set-up RUN-PHASE is updated to MAT-WAIT. If set-up was not interrupted by stopping the operation then STATE still is RUN and WAIT-MAT-PRC is called to show whether the operation has to wait for delivery of materials needed to start with processing; if it has to wait it is canceled (Table 4.87) otherwise it continues its execution of the dynamic section and the phase becomes PREPARED. Continuation results in calling GO-ON- (Table 4.88), updating the phase to BUSY and calling Passivate that interrupts the excecution of dynamic for an unknown duration until activated in UPDAT-QNT (Table 4.90) that is called by an event in one of the materials processed or produced. Such an activation at Time prior results immediately in calling QNT-TRANSFER (Table 4.89), an updating of the phase to GO-ON-WAIT and calling Passivate if the state still is RUN. This interrupts the execution until activated in CONTINUE (Table 4.90) that may be called from material (if such a call is not appropriate then DECIDE is warned by

Table 4.93 Dynamic section of class OPRTN-MATRL.

```
!    d    y    n    a    m    i    c;
!         s t a r t;
while not END_EXPRMNT do begin
    if STATE = RUN then begin
        GO_AHEAD;
        RUN_PHASE:= SETUP_WAIT;
        Hold (SETUP_OPR / 24.0);
        RUN_PHASE:= MAT_WAIT;
    end;
    if STATE = RUN then WAIT_MAT_PRC;          !cancel/ activate depending on no_wait_ndd;
    if STATE = RUN then RUN_PHASE:= PREPARED;  !setup completed, all materials available;
    !          r e p e a t   p r o c e s s i n g;
    while STATE = RUN do begin
        GO_ON_; RUN_PHASE:= BUSY; Passivate;    !actv. by updat_qnt;
        QNT_TRANSFER; RUN_PHASE:= GO_ON_WAIT;
        if STATE = RUN then Passivate;          !actv. by continue;
    end of while processing;
    if STATE <> RUN then TERMNT;
    RUN_PHASE:= INACTIVE;
    while STATE <> RUN do Passivate;            !actv. by start_oprtn;
end xx of while not end_of_experiment at start;
```

DELIVER-MAT (Table 4.34) or this operation waits until delivery (PROCESS-MAT, Table 4.34) with a signal, MAT-PRD-RQRD). If the state is not RUN due to STOP-OPRTN- then TERMNT (Table 4.92)is called, the phase is updated to INACTIVE and Passivate is called (repeatedly if activated wrongly outside START-OPRTN-). This interrupts the execution until activated again by START-OPRTN-.

The verification of this part of the program is described in Chapter 6 'Verification and validation' and especially Section 6.1.4 'Events related to operations'. This section completes the description of operations processing materials.

4.3.2 Operations for service and repair

The structure of this class is to some extent comparable to that of operations processing materials. The content of the procedures is however different, so it must be described completely. Table 4.94 contains the declaration of variables. The reference variable MACHN refers to a piece of equipment which will be serviced or repaired in this operation.

STATE-CH-OP-, Table 4.95, first derives a next state of the operation, STATE-NEXT, that becomes the state of the gang (its counterpart) if the

Table 4.94 Class SRVC-REPR; declaration of variables.

```
OPERATION class SRVC_REPR;                        !=====================================;
begin
   !     d     e     c     l     a     r     a     t     i     o     n;

   ref (EQUIPMENT)                                !----------------------;
   MACHN;                                      !refers to machine fur repair or servire;
   ref (Head)                                     !----------------------;
   MCHN_S_R_Q;                                 !queue of machines with a failure;
   real    LT_TRANS, T_USED;                      !----------------------;
```

Table 4.95 Procedure STATE-CH-OP- of class SRVC-REPR.

```
procedure STATE_CH_OP_;                        !----   ----   ----   ----   ----;
begin                  !called in gang.state_ch_mms, transfer, mach[].dyn., .service_need;
   if STATE = STATE_NEXT  then begin
      if MACHN =/= none then begin
         if not  (MACHN.RPR_ND or MACHN.SRVC_ND) then MACHN:- none;
      end;
      while MACHN == none and MCHN_S_R_Q.First =/= none do begin
         MACHN:- MCHN_S_R_Q.First qua RECORD_LE.LBR_EQP qua EQUIPMENT;
         MCHN_S_R_Q.First.Out;
         if not (MACHN.RPR_ND or MACHN.SRVC_ND) then MACHN:- none;
      end;
      STATE_NEXT:= if MACHN == none then DOWN
      else    if MACHN.STATE = DOWN and (MACHN.RPR_ND or MACHN.SRVC_ND)
      !;      then GANG_G.STATE
      !;      else DOWN;
   end;
   if STATE <> STATE_NEXT then begin
      STATE_PREV:= STATE;
      TIME_C_ACCUM;
      STATE:= STATE_NEXT;
   end;
end ** of change of state of service and repair operation;
```

machine is in need of service or repair, otherwise it becomes DOWN. Previously the reference MACHN is updated to the first one in the queue in need of service or repair MCHN-S-R-Q. The change of state is as usual. The start of the operation by START-OPRTN-, shown in Table 4.96, is called from decision and executed if the state is PASSIVE. It starts the gang (that updates the state of the operation to RUN) and activates the dynamic part. GO-ON- is shown in Table 4.97 and called from dynamic; it only sets the start time and the current costs of the gang. Procedure TRANSFER is shown in Table 4.98; it is called from dynamic when the repair or service is done. It calculates the time used T-USED and the cumulative costs and it calls in MACHN the procedure SRVC-RPR-DON (Table 4.60) if there was a need for repair (or service) and the time used is almost the duration required (less than 0.144 min difference or 0.0001 day). The reference to MACHN is updated by calling STATE-CH-OP-. Table 4.99 shows STOP-OPRTN-, the procedure is called from decision to stop the operation; it stops the gang and activates the dynamic part immediately.

Table 4.96 Procedure START-OPRTN- of class SRVC-REPR.

```
procedure START_OPRTN_;                               !----  ----  ----  ----  ----;
if STATE = PASSIVE and RUN_PHASE = INACTIVE then begin        !called in decision;
    GANG_6.START_WORK_6;
    activate this SRVC_REPR at Time;
end;
```

Table 4.97 Procedure GO-ON- of class SRVC-REPR.

```
procedure GO_ON_;                                     !----  ----  ----  ----  ----;
if RUN_PHASE = PREPARED or RUN_PHASE = GO_ON_WAIT then begin   !called in dyn.;
    START_TIME:= Time;
    COSTS_GNG:= GANG_6.COSTS_MADE;
end;
```

Table 4.98 Procedure TRANSFER of class SRVC-REPR.

```
procedure TRANSFER;                                   !----  ----  ----  ----  ----;
if RUN_PHASE = BUSY then begin                        !called in dyn.;
    GANG_6.TIME_C_ACCUM;                              !updates costs of gang;
    T_USED:= T_USED + 24.0 * (Time - START_TIME);!in [h];
    COSTS_MADE:= COSTS_MADE + (GANG_6.COSTS_MADE - COSTS_GNG);
    inspect MACHN do if RPR_ND and RPR_DUR < T_USED + 24.0&-4
    or SRVC_ND and SRVC_DUR < T_USED + 24.0&-4 then begin
        SRVC_RPR_DON;                                 !transfers state of machine;
        T_USED:= 0.0;
    end;
    ADMN.DISPLAY_DATA_;
    STATE_CH_OP_;                                     !updates machn and the state if no machines;
end;
```

Table 4.99 Procedure STOP-OPRTN- of class SRVC-REPR.

```
procedure STOP_OPRTN_;                                !----  ----  ----  ----  ----;
begin                                                 !called in decision to interrupt;
    GANG_6.STOP_WORK_6;
    reactivate this SRVC_REPR at Time prior;
end;
```

Table 4.100 Dynamic section of class SRVC-REPR.

```
!       d     y     n     a     m     i     c;
!             r e p e a t;
while not END_EXPRMNT do begin
    while STATE = RUN do begin
        RUN_PHASE:= PREPARED;
        GO_ON_;
        RUN_PHASE:= BUSY;
        reactivate this SRVC_REPR at          !can be interrupted by stop_oprtn_;
        Time + ( - T_USED +
        (if MACHN.RPR_ND then MACHN.RPR_DUR
        else   if MACHN.SRVC_ND then MACHN.SRVC_DUR
        !;      else 0.0)       ) / 24.0;
        TRANSFER;
        RUN_PHASE:= GO_ON_WAIT;
    end;
    RUN_PHASE:= INACTIVE;
    while STATE <> RUN do Passivate;          !activated in start_oprtn_;
end xx of while;
```

Table 4.100 shows the dynamic part that updates RUN-PHASE and starts with GO-ON- after its activation by START-OPRTN-. It schedules itself according to the durations of repair or service of the machine and continues at that moment with calling TRANSFER. If another machine waited for repair or service it goes on with GO-ON-, etc.; if no machine requires repair or service then the RUN-PHASE becomes INACTIVE and the dynamic part calls Passivate. After scheduling this operation for the repair or service duration of the machine it can also be interrupted by calling STOP-OPRTN- due to a decision (for instance, a pause); in that case TRANSFER calculates the time used already, T-USED, MACHN remains in need of repair or service and Passivate is called; the operation waits for the activation from START-OPRTN-.

4.3.3 Decision

Class DECISION is itself a process and declares only some variables at the level of the base model, Table 4.101. Most variables are used to send a signal to this class when it has to be activated from shifts, machines or materials for some reason; see the description in those classes for their meaning.

This class is extended in the experimental frame in Chapter 5, because decision strategies do not belong to the system but to the 'environment'.

Table 4.101 Class DECISION; virtual procedure and declaration of variables.

```
Process class DECISION;                    !****************************************;
virtual: procedure URG_GANGS_;
begin
   !    d    e    c    l    a    r    a    t    i    o    n;

   boolean                                 !-----------------------;
   SHIFT_CH, MACHN_FLR, MACHN_OK,
   MAT_MAX_QNT, MAT_DLV_OK, MAT_PASS, MAT_DOWN, PRC_END, END_FLDS, DLVRY_SLWR;
end;
```

4.4 Miscellaneous

In this section some additional helpful classes and procedures are described. The procedures concern the transformation of dates to day numbers or the reverse and of moments derived from the system variable 'Time'. The classes concern the controlling of the moments something has to happen to some components, for instance, the moments when worktime of men starts or ends. They are called 'shifts'. There is a shift defining periods in a year; another defines the worktime within a week and the third one defines the moments of calculation the urgency of materials. All this is concentrated in 'shifts' for convenience only; it could be formulated within each component but it is more efficient to use a shift with a queue containing the components behaving in the same way. Because several objects of such shifts can be created, man a-d can be queued in object A and man e-k in object B with a different behaviour.

4.4.1 Shifts of periods in a year

Table 4.102 shows the declaration of SHIFT-PERD as a Process; it has parameter PERDS as the number of periods within a year. The end of a period is defined as a day number relative to January 1. The end of period n is the beginning of period n+1. Period 1 begins at Jan. 1 and ends at PERD-END[1]; period PERDS ends at PERD-END[PERDS] and the rest of the year is period PERDS+1. A period i is defined 'available' or not by AVLB-PERD[i] that means that each component in queue COMP-Q is available or not, respectively, during period i. The dynamic section is shown in Table 4.103; each component in the queue calls its own SHIFT-CHNGE- (Table 4.3, 4.39, 4.54 and 4.63) and activates a weekly shift if needed. When the

Table 4.102 Class SHIFT-PERD; parameter and declaration of variables.

```
Process class SHIFT_PERD (PERDS);                 !****************************************;
integer PERDS;                                    !number of periods in a year or season;
begin
   !     d     e     c     l                      a     r     a     t     i     o     n;
   integer                                              !-----------------------;
   PERD,                                          !current period number;
   LE_NO_CTG;                                     !category of men/machines;
   integer array                                        !-----------------------;
   PERD_END [0:PERDS +1];                         !last date period is valid;
   boolean                                              !-----------------------;
   AVLB_PRD, AVLB_PRD_PREV;                       !true if available in current/ previous period;
   boolean array                                        !-----------------------;
   AVLB_PERD [1:PERDS + 1];                       !false after perds;
   text                                                 !-----------------------;
   PLUS;
   ref (COMPONENT)                                      !-----------------------;
   CMPNT;
   ref (Head)
   COMP_Q;                                        !queue for components;
   ref (RECORD_COMP)                              !ref. to 'link'object (to queue components);
   REC_COMP;                                      !operations not wanted (covered by gangs);
```

Table 4.103 Dynamic section of class SHIFT-PERD.

```
!   d   y   n   a   m   i   c;
!       r e p e a t ;
while not END_EXPRMNT do begin
    PERD:= PERD + 1;
    AVLB_PRD_PREV:= AVLB_PRD;
    AVLB_PRD:= AVLB_PERD [PERD];
    ADMN.DISPLAY_DATA_;          !update before states are changed;
    LE_NO_CTG:= 0;
    REC_COMP:- COMP_Q.First;
    while REC_COMP =/= none do begin
        CMPNT:- REC_COMP.CMP;
        inspect CMPNT do begin
            SHIFT_CHNGE_;
            if AVLB_PRD and SH_WK =/= none then activate SH_WK at Time;
        end;
        inspect CMPNT when LABR_EQPMNT do if LE_NO_CTG <> LE_NO_LE then begin
            LE_NO_CTG:= LE_NO_LE;
            UPDT_MM_SYS.TEST_LE [LE_NO_CTG]:= true;
            activate UPDT_MM_SYS at Time;
        end;
        REC_COMP:- REC_COMP.Suc;
    end;
    DECIDE.SHIFT_CH:= true;
    activate DECIDE at Time;
    reactivate this SHIFT_PERD at (YR_N - 1) * LAST_DAY_YEAR + PERD_END [PERD];
    if PERD > PERDS then PERD:= 0;
end ** of while;
```

component is a man or machine then the updating of man-machine systems is needed (a signal TEST-LE[ ] is set to true and UPDT-MM-SYS is activated, Section 4.2.3). Finally a decision is required and this object is rescheduled at the end of the current period; the preceding years of the simulation are considered by YR-N, the year number, times the number of days in a year (for convenience only it is preferred that LAST-DAY-YEAR is 500.0 or 1000.0 instead of 365.0). The control of the state of an operation (Table 4.84) is related to the state of the gang; it is therefore unnecessary to include operations in a shift (if done however it has no effect on the state of the operation). This instrument can be used to define the availability of machines from a neighbour or contract worker during some periods in a year.

4.4.2 *Shifts within a week*

This shift is developed to control the availability of men for work, but it can be used also for machines or other subclasses of COMPONENT. Table 4.104 shows the class SHIFT-WEEK. The seven days of a week (Monday = 1, .. etc.) belong to one of the categories considered 1, ..., DAY-CTGRS, such as workdays, weekend days, Saturday or Sunday; the catagory of a day is given in DAY-CATAGORY [DAY-TYPE], where DAY- TYPE is 1, ..., 7. Each day has a number of shifts SHIFTS and SHIFT-END [0:SHIFTS] contains the clocktime when availability or costs change. A number of costs categories, COST-CTGRS, are distinguished and category 0 is used to denote that AVLB-WK is false (men are not available for work) and 1:COST-CTGRS to assign costs per hour in COST-H-CTGR [0:COST-CTGRS], for

Table 4.104 Class SHIFT-WEEK; parameters and declaration of variables.

```
Process class SHIFT_WEEK                           !**************************************;
(DAY_CTGRS, COST_CTGRS, SHIFTS, PERIODS);
integer
DAY_CTGRS,          !number of daytypes per week;
COST_CTGRS,         !number of cost categories considered ;
SHIFTS,             !shifts per day for a man or a machine;
PERIODS;            !number of periods in a year with different work-time and overtime moments;

begin
   !    d    e    c    l    a    r    a    t    i    o    n;

    real array                                    !----------------------;
    COST_H_CTGR [0:COST_CTGRS],                   !costs/hour of man per category of costs;
    SHIFT_END [0:SHIFTS,1:PERIODS];               !time in hours at end of shift, same all
    !         [0       ,j] = 0.0                  !days, shift_end [shifts, j]= 24.0 * n;
    integer array                                 !----------------------;
    COSTS_SH_CTG [1:DAY_CTGRS, 1:SHIFTS, 1:PERIODS],!cost category per day,shift,period;
    PERIOD_END [1:PERIODS + 1],                   !last date period is valid;
    DAY_CATAGORY [1:7];                           !subscript 1....7 for monday,.....,sunday;
    integer                                       !----------------------;
    COST_CTGR,                                    !current category of variable costs (overtime);
    SHIFT, LE_NO_CTGR,                            !current shift, category of men or mach.;
    PERIOD, DAY_TYPE;                             !current period, type of day (1,...,7);
    real                                          !----------------------;
    COSTS_NOW,                                    !current variable costs;
    COSTS_PREV, COSTS_INCR, R;
    boolean                                       !----------------------;
    AVLB_WK, AVLB_WK_PREV,                        !true if available in current/previous interval;
    AVLB_COMP_WK;                                 !true if any component avlb_comp;
    ref (Head)                                    !----------------------;
    CMP_Q;                                        !queue for components;
    ref (RECORD_COMP)
    REC_CMP;                                      !reference to a record containing a component;
```

instance, 0.0 f/h for regular time and 15.0 f/h for overtime of permanent labour. Some periods, PERIODS, are considered in a year; they end at PERIOD-END [ ]. In COSTS-SH-CTG [i,j,k] the costs category is given for each day-category i, each shift j and each period k. The dynamic section is shown in Table 4.105. It derives DAY-TYPE, PERIOD, SHIFT, the previous costs, COSTS-PREV, the current costs category, COST-CTGR and the current costs, COSTS-NOW, the increase of costs, COSTS-INCR, and the previous and current availability, AVLB-WK-PREV, AVLB-WK. If the availability or the costs changed or no component was available thus far, then each component, CMP, in CMP-Q is inspected. SHIFT-CHNGE- is called for each component and AVLB-COMP-WK is set to true if a component is available (AVLB-COMP is true) in the current period and shift. When the component is an object of class LABR-EQPMNT and the sequence number, LE-NO-CTGR is not yet considered then some signals are set in UPDT-MM-SYS by calling UPDT-LE (Table 4.106). UPDT-MM-SYS is activated (Section 4.2.3) to update the availability of the man-machine systems immediately (before decisions are required). Under conditions when work can start or has to stop or only costs change, then DECIDE is activated after setting SHIFT-CH to true. A reasonable selection of conditions has been made here, but it was also possible to be less restrictive and

Table 4.105 Dynamic section of class SHIFT-WEEK.

```
!       d       y       n       a       m       i       c;
!               r e p e a t;
while not END_EXPRMNT do begin
    DAY_TYPE:= WTH_MAT.DAY_TYPE_WTHR;   !wthr is activated prior to shifts;
    if SHIFT = SHIFTS then              !update of period only after complete cycle of shifts;
    while PERIOD_END [PERIOD] < DAYS_NMB do PERIOD:= PERIOD + 1;
    !                                   !days_nmb is updated in admnstr;
    SHIFT:= Mod (SHIFT,SHIFTS) + 1;
    COSTS_PREV:= COSTS_NOW;
    COST_CTGR:=                         !0 means men are not available in current shift;
    if PERIOD > PERIODS then 0          !men not available after periods;
    else COSTS_SH_CTG [DAY_CATAGORY [DAY_TYPE],SHIFT,PERIOD];
    COSTS_NOW:= COST_H_CTGR [COST_CTGR];
    COSTS_INCR:= COSTS_NOW - COSTS_PREV;
    AVLB_WK_PREV:= AVLB_WK;
    AVLB_WK:= COST_CTGR <> 0;           !false for category 0;
    LE_NO_CTGR:= 0;
    if not (AVLB_WK_PREV eqv AVLB_WK) or COSTS_NOW <> COSTS_PREV or not AVLB_COMP_WK then begin
        AVLB_COMP_WK:= false;
        ADMN.DISPLAY_DATA_;             !update before states are changed;
        REC_CMP:- CMP_Q.First;
        while REC_CMP =/= none do begin
            inspect REC_CMP.CMP do begin
                SHIFT_CHNGE_;
                AVLB_COMP_WK:= AVLB_COMP_WK or AVLB_COMP;
                !       true if any component exists that is available in current shift;
            end;
            inspect REC_CMP.CMP when LABR_EQPMNT do if LE_NO_CTGR <> LE_NO_LE then begin
                LE_NO_CTGR:= LE_NO_LE;
                UPDT_LE;
            end;
            REC_CMP:- REC_CMP.Suc;
        end;
        reactivate UPDT_MM_SYS at Time prior;           !update before decision;
    end ** then branch of change;
    if (not AVLB_WK_PREV and AVLB_WK)                   !start or select work;
    or (AVLB_WK_PREV and not AVLB_WK and MM_S_SELECT)   !stop non-autom. work;
    or (COSTS_INCR <> 0.0)                              !start?, finish?;
    then begin
        DECIDE.SHIFT_CH:= true;
        activate DECIDE at Time;
    end;
    if END_SEASON or PERIOD > PERIODS then begin
        Passivate;                                      !activated by sh_prd;
        SHIFT:= SHIFTS; PERIOD:= 1;
    end
    else begin
        R:= Time + (SHIFT_END [SHIFT, PERIOD] - SHIFT_END [SHIFT-1, PERIOD]) / 24.0;
        if Abs(R - Entier(R+1.0&-3)) < 1.0&-3 then R:= Entier(R+1.0&-3);
        reactivate this SHIFT_WEEK at R;
    end;
end xx of while;
```

Table 4.106 Procedure UPDT-LE of class SHIFT-WEEK.

```
procedure UPDT_LE;                      !----   ----   ----   ----   ----;
begin                                   !called in dynamic;
    if not (AVLB_WK_PREV eqv AVLB_WK) then      !set test_avl in update_mms;
    UPDT_MM_SYS.TEST_LE [LE_NO_CTGR]:= true;
    if COSTS_INCR <> 0.0 then begin
        UPDT_MM_SYS.COSTS_CHNG [LE_NO_CTGR]:= true;
        UPDT_MM_SYS.COSTS_INCRSE [LE_NO_CTGR]:= COSTS_INCR;
    end;
end;
```

to move in each case to DECIDE. Finally this class is passivated if the season is over or no component is available (activation occurs in the shift of periods) otherwise it is reactivated at the moment when the current shift ends.

This shift (of men) can, for example, handle the regular worktime on workdays from 07:00-12:00 and 13:00-18:00 the overtime from 12:00-13:00 and 18:00-22:00 and on Saturday from 07:00-18:00 and the no worktime outside these ranges. The overtime during pauses, during the evening and during Saturday can each have their own category of costs per hour. So there is a flexible instrument to control the availability of men. This method can also be used to define the availability of machines during a week, for example, milking machines each day, for instance, from 06:00-10:00 and 16:00-20:00. It is sufficient in most cases to involve only men in the weekly shifts; the availability of men controls the state of all the man-machine systems except those operating without men such as a grain drier (an automatic system).

4.4.3 Shifts for materials

Such a shift is used to control the clocktimes when the calculation of urgency and disurgency of materials processed is desired. It has a structure (Table 4.107) similar to that of the above shifts. In its dynamic section, Table 4.108, it calls procedures to calculate the urgency of each processed material contained in MAT-Q by calling URG-MAT-PRC- and URG-MAT-EXT- (Tables 4.43 and 4.44), writes the derived values if desired on a file URG-APL-OUTP, calls the 'virtual' procedure URG-GANGS- in DECIDE (Table 4.101) to calculate the urgency of the gangs and passivates or reactivates itself at the next clocktime.

The cycle handled is implicitly defined by the last hour URG-CALC-HR-[POINTS] and may be 24 hours from the starting time; 168 hours or one week or any positive number can be used. The above shifts have a shortest cycle of a year, a week and a day for SHIFT-PERD, SHIFT-WEEK and SHIFT-URG, respectively.

Table 4.107 Class SHIFT-URG; parameter and declaration of variables.

```
Process class SHIFT_URG (POINTS);                          !::::::::::::::::::::::::::::::::::::::::;
integer POINTS;                                            !number of calculation points in time cycle;
begin
   !    d    e    c    l    a    r    a    t    i    o    n;
   integer                                                 !-----------------------;
   POINT_NO, I9;
   real                                                    !-----------------------;
   R1;
   real array                                              !-----------------------;
   URG_CALC_HR [0:POINTS];             !hours (may be >24.0) are handled relative to preceding;
   boolean                                                 !-----------------------;
   AVLB_COMP_UG;                       !true if any material available;
   ref (Head)                                              !-----------------------;
   MAT_Q;
   ref (RECORD_MAT)
   REC_MAT;
```

Table 4.108 Dynamic section of class SHIFT-URG.

```
!       d       y       n       a       m       i       c;
!               r e p e a t;
while not END_EXPRMNT do begin
    AVLB_COMP_UG:= false;
    HOUR_AT_TIME;
    REC_MAT:- MAT_Q.First;
    while REC_MAT =/= none do begin
        if REC_MAT.MATRL_RF.AVLB_COMP then inspect REC_MAT.MATRL_RF qua PROCESSED_MAT do begin
            URG_MAT_PRC_;
            URG_MAT_EXT_;
            AVLB_COMP_UG:= AVLB_COMP_UG or AVLB_COMP;
            !   true if a component exists that is available in the current period;
        end;
        REC_MAT:- REC_MAT.Suc;
    end;
    if AVLB_COMP_UG then inspect URG_APL_OUTP do begin
        Outfix( TIME_YR,2,6); Outfix( HOUR,1,5);
        for I9:= 1 step 1 until MATRLS_PROC do inspect MATRL_PRC [I9] do
        Outfix( RMAX( URGENCY_PROC, URG_PROC_ALT) / CAP_PROC, 0,8); Setpos (Pos + 10);!fl/ha;
        for I9:= 1 step 1 until MATRLS_PROC do inspect MATRL_PRC [I9] do
        if DISURG_USED then Outfix ( DISURGNC_DEL / CAP_PROC, 0,8) else Setpos(Pos+8);
        Outtext('    [fl/ha]'); Outimage;
    end;
    if AVLB_COMP_UG then DECIDE.URG_GANGS_;     !activates decide if necessary;
    POINT_NO:= if POINT_NO = POINTS then 1 else POINT_NO + 1;
    if END_SEASON or not AVLB_COMP_UG then begin
        Passivate;                              !activated in shift_chnge_ of material;
        POINT_NO:= 0;
    end
    else begin
        R1:= Time + (URG_CALC_HR [POINT_NO] - URG_CALC_HR [POINT_NO - 1]) / 24.0;
        if Abs (R1 - Entier(R1+1.0&-3)) < 1.0&-3 then R1:= Entier (R1 + 1.0&-3);
        reactivate this SHIFT_URG at R1;
    end;
end of while;
```

4.4.4 *Dates and time*

Some procedures are necessary to transform data related to date and time. Table 4.109 shows the transformation from month and date to day number relative to Jan. 1 as the first day and without taking into account leap years (Febr. 29 not counted). The formulae are derived from Stuff & Dale (1973) who counted days relative to March 1. The opposite to derive a month and date from a day number is shown in Table 4.110; the result is a text variable

Table 4.109 Procedure DATE-TO-DAYNO of class SFOBASE-MODEL.

```
integer procedure DATE_TO_DAYNO
(MONTH_NO, DATE_MN_NO);                          !----   ----   ----   ----   ----;
integer MONTH_NO, DATE_MN_NO;
!                                                !called in shift_perd/ _week, delvr_fld_init;
begin               !dayno will become relative to 1st jan = 1, no leap years considered;
   if MONTH_NO > 12 or DATE_MN_NO > 31 then begin
       Outimage;
       Outtext ('Warning:  You try to find a dayno with Month');
       Outtext (' and Date, at Time'); Outimage;
       Outint (MONTH_NO,6); Outint (DATE_MN_NO,6); Outfix (Time,6,12); Outimage;
   end;
   if MONTH_NO < 3 then MONTH_NO:= MONTH_NO + 12;
   DATE_TO_DAYNO:= Mod (Entier (MONTH_NO * 30.6 + DATE_MN_NO - 32.3) -1, 365) + 1;
end ** developed from stuff & dale, agric. meteor. 12 [1973] 441-442;
```

Table 4.110 Procedure DATE-FROM DAYNO of class SFOBASE-MODEL.

```
procedure DATE_FROM_DAYNO (DAYNO);                  !----  ----  ----  ----  ----;
integer DAYNO;                                       !dayno relative to 1st jan. = 1, no leap years;
begin
    integer MN, DT;
    text DATE_TXT;
    if DAYNO > 365 then begin
        Outimage;
        Outtext ('Warning:  You try to find a date for dayno ='); Outint (DAYNO,6);
        Outtext (', it is reduced mod( ,365).'); Outimage;
    end;
    DAYNO:= Mod (DAYNO -1, 365) + 1;
    DAYNO:= if DAYNO < 60 then DAYNO + 365 else DAYNO;
    MN:= Entier ( (DAYNO + 32.3) / 30.6);
    DT:= Entier (DAYNO - MN * 30.6 + 33.3);
    MN:= Mod (MN - 1, 12) + 1;
    MNTH_DT6:= MNTH_TXT3 (MN);
    DATE_TXT:- MNTH_DT6.Sub (4,3);
    DATE_TXT.Putint (DT);
end ** developed from stuff & dale, agric. meteor.12 '1973', 441-442;
```

Table 4.111 Procedure HOUR-AT-TIME of class SFOBASE-MODEL.

```
procedure HOUR_AT_TIME;                              !----  ----  ----  ----  ----;
begin                                                !called in decision, shift_urg;
    HOUR:= (Time - Entier(Time + 1.0&-4)) * 24.0;             !+-0.0 at 24.0h;
    DAY_FRAC_NOW:= if HOUR <= DAY_BGN then 0.0
    else    if DAY_END <= HOUR then 1.0
    !;      else (HOUR - DAY_BGN) / (DAY_END - DAY_BGN);
end;
```

Table 4.112 Procedure TIME-YR of class SFOBASE-MODEL.

```
real procedure TIME_YR;                              !----  ----  ----  ----  ----;
TIME_YR:= Time - TIME_1JAN;                          !time in year relative to 1 januari 0.0 o'clock;
```

of six characters, MNTH-DT6. The procedure HOUR-AT-TIME, Table 4.111, calculates the current clocktime from the system variable Time and a fraction of the day already used at this clocktime when it is between the beginning and the end of the workday (depends on input data of shift of week; Section 4.4.2). When the simulation covers some years then 'Time' continues over the years; to show the duration since 1 Jan. 0:00 procedure TIME-YR is used, Table 4.112.

This section completes the description of the base model.

5 EXPERIMENTAL FRAME

This chapter extends the base model to the experimental frame and describes the related input and output. The experimental frame consists of a subprogram with some general specifications and a main program that initializes the system, sets up one or more experiments and simulates the scheduling of operations over several seasons for each experiment.

5.1 General specifications

The general specifications concern the weather data (and material properties), the decision making and the administration. They are general in the sense that they can be used for each scheduling system defined in the input (number of men and machines, type of operations and materials, etc.); they are specifications in the sense that they expect in input specific materials and properties. In the base model and in the experimental frame a balance between generality of use and flexibility of use was invested; therefore because of the assumptions made, there is limited flexibility. In Chapter 7 relevant specifications for the wheat harvesting and other extensions of an experimental frame are described. Table 5.1 shows the structure of the subprogram with the classes and references to objects.

5.1.1 Weather and material data

In the base model class WEATHER-MATRL, Table 4.8, contains the variables of the materials processed to control their workability, PROPERTY. Now this class is extended by defining a subclass WEATHER that mainly reads chronological data from a file. Table 5.2 shows the variables so that the moment up to which the properties are valid is known (MONTH, DATE, CLOCK and TIME-CL), the rain, a reference to an inputfile, etc. The meaning of the variables will soon become clear.

Table 5.1 General outline of class SFOEXPERIMENT and its attributes.

```
external class SFOBASE_MODEL;
SFOBASE_MODEL class SFOEXPERIMENT;
begin
   ref (WEATHER) WTHR;
   WEATHER_MATRL class WEATHER;

   ref (URGENCY_DCSN) DECIDE_URG;
   DECISION class URGENCY_DCSN;

   ref (ADMINISTRATION) ADMNSTR;
   ADMINISTRATOR class ADMINISTRATION;
end;
```

Table 5.2 Class WEATHER; declaration of variables.

```
WEATHER_MATRL class WEATHER;                    !****************************************;
begin
   !   d   e   c   l   a   r   a   t   i   o   n;

   boolean                                      !----------------------;
   ENDED_DATA;                    !true if end of weather datafile occurs;
   integer                                      !----------------------;
   I3, YEAR, MONTH, DATE, CLOCK;  !integer clocktimes in inputfile!!;
   real                                         !----------------------;
   RAIN_SUM, RAIN,                !(mm);
   TIME_NEXT,                     !time when new weather data are required;
   TIME_CL_PRV, TIME_CL,          !time at clock since 1 Jan.;
   DURTN_DATA;                    !from current moment until data are valid at time_next;
   ref (Infile)WTHR_MT_DATA;                    !----------------------;
   text DATAFILE, YR;                           !----------------------;
```

Table 5.3 Procedure NEW-FILE of class WEATHER.

```
procedure NEW_FILE;           !----  ----  ----  ----  ----;
begin                         !called in main;
   DATAFILE:- PARAMETERS.Intext (12);
   YR:- DATAFILE.Sub (5,2);
   YEAR:= 1900 + YR.Getint;
   ENDED_DATA:= false;
   activate this WEATHER at Time prior;
end;
```

Table 5.3 shows procedure NEW-FILE that is called from the main program when a new season of weather and material data is required in the experiment. It reads the name of the file from a file referenced by PARAMETERS; the name consists of 'file name.extension' where character 5 and 6 contains the year index. The dynamic section is activated after ENDED-DATA becomes false. The dynamic section is shown in Table 5.4 and consists of three groups of statements that are entered if the end of experiment, END-EXPRMNT is not yet achieved. The first group initializes the file reference and opens the file WTHR-MT-DATA. The third group is used when the end of that file is achieved: the file is closed, ENDED-DATA becomes true, the state of materials is changed, DECIDE is activated and Passivate is called (a reactivation can occur when another season is requested in the experiment). The second group reads the data; derives the type of day at Jan. 1; calculates the next moment of reading TIME-CL and TIME-NEXT (properties are valid from now on until that moment) and the duration, DURTN-DATA; calls ATTR-UPDT-M- (Table 4.45) of materials to update the material attributes according to PROPERTY; DECIDE is activated after giving it a signal when nothing is selected to work and men are available, the object is rescheduled at the next time and after that moment the sum of rain is calculated and finally Lastitem is called to know in time if the end of file is achieved or more data are available. It is assumed that the data file contains MONTH, DATE and CLOCK, DAY-TYPE-WTHR giving the type of the day (1 = Monday, ...), RAIN in [mm] and PROPERTY, an array with one element for each material processed (Table 4.8). The first three are integers (even CLOCK i.e. data on full hours only!) and the last

Table 5.4 Dynamic section of class WAETHER.

```
!        d      y      n      a      m      i      c;

while not END_EXPRMNT do begin
    WTHR_MT_DATA:- new Infile (DATAFILE);
    WTHR_MT_DATA.Open(Blanks(150));
    RAIN_SUM:= 0.0;
    DAY_TP_1JAN:= 0;
    WTHR_MT_DATA.Inimage;

    while not WTHR_MT_DATA.Endfile do begin
        !                   r e a d;
        inspect WTHR_MT_DATA do begin   !data for spell from now until month,date,clock;
            MONTH:= Inint; DATE:= Inint; CLOCK:= Inint; DAY_TYPE_WTHR:= Inreal;
            RAIN:= Inreal;                                  !in mm;
            for I3:= 1 step 1 until MATRLS_PROC do PROPERTY [I3]:= Inreal;
            Inimage;
        end;
        if DAY_TP_1JAN = 0 then begin
            I3:= DATE_TO_DAYNO (MONTH, DATE);
            DAY_TP_1JAN:= Mod (DAY_TYPE_WTHR - 1 - (I3 -1) + 7000, 7) + 1;
            Outimage;
            Outtext ('Day type at 1 Jan.'); Outint (YEAR,6); Outint (DAY_TP_1JAN,6);
            Outtext ('  (1= Monday ,etc.); Data start at  ');
            Outtext (DAY_TXT3 [DAY_TYPE_WTHR]); Outtext ('   ');
            Outtext (MNTH_TXT3 [MONTH]); Outint (Entier (DATE),6); Outimage;
            TIME_CL_PRV:= I3 - 1;
        end;
        TIME_CL:= DATE_TO_DAYNO (MONTH, DATE)- 1.0 + CLOCK / 24.0;
        TIME_NEXT:= TIME_1JAN + TIME_CL;
        DURTN_DATA:= RMAX(1.0&-4,(TIME_CL - TIME_CL_PRV) * 24.0);
        ADMNSTR.DISPLAY_DATA_;
        if not END_SEASON then for I3:= 1 step 1 until MATRLS_PROC do MATRL [I3].ATTR_UPDT_M_;
        if not MM_S_SELECT and MAN [1].STATE <> DOWN and not ENDED_PROCS then
        DECIDE_URG.WTHR_UPDATE:= true;
        activate DECIDE at Time;
        reactivate this WEATHER at TIME_NEXT prior;     !prior to other processes;
        RAIN_SUM:= RAIN_SUM + RAIN;                     !cum.rainfall upto the current time;
        WTHR_MT_DATA.Lastitem;
        TIME_CL_PRV:= TIME_CL;
    end xx while loop;
    WTHR_MT_DATA.Close;
    ENDED_DATA:= true;
    for I3:= 1 step 1 until MATRLS_PROC do inspect MATRL [I3] do begin
        WORKBL_NEXT:= false; CH_WORKBL; STATE_CH_MAT;
    end;
    if not DECIDE.Terminated then reactivate DECIDE at Time;!finishes operations immediately;
    Passivate;                                          !wait until reactivated by new_file;
end ** of data & while  not end_of_experiment;
```

three are declared as reals but may be integers in the file without causing errors during reading. An example is shown in Table 5.5 that shows the data of Wednesday 1 August 1962 related to six materials belonging to the wheat harvesting (wheat, straw, bales on the field, bales loaded, stubble and wet grain). Even if the workability of bales loaded is independent of the weather, the file has to contain a PROPERTY[4] due to the generality of this experimental frame; in this case 0.00 as for wet grain. PROPERTY[ ] is used to control the workability of a material processed as if only one property is relevant; if more properties are relevant, for instance, the moisture content of grain and the occurrence of condensation moisture on the wheat plant, then the moisture content was used to define PROPERTY and multi-

Table 5.5 Example of the data file read in class WEATHER.

DATE Aug.1		CLOCK	DAY	RAIN [mm]	PROPERTY [1:6] [1] cereal field	[2] straw	[3] bales field	[4] bales loaded	[5] soil	[6] wet grain
8	1	1	3	0.00	-21.52	-20.37	0.00	0.00	43.09	0.00
8	1	2	3	0.00	-21.76	-21.91	0.00	0.00	43.12	0.00
8	1	3	3	0.00	-22.00	-23.37	0.00	0.00	43.15	0.00
8	1	4	3	0.00	-22.23	-24.75	0.00	0.00	43.17	0.00
8	1	5	3	0.00	-22.46	-26.05	0.00	0.00	43.20	0.00
8	1	6	3	0.00	-22.69	-27.29	0.00	0.00	43.22	0.00
8	1	7	3	0.00	-22.91	-28.46	0.00	0.00	43.21	0.00
8	1	8	3	0.00	-23.13	-29.57	0.00	0.00	43.14	0.00
8	1	9	3	0.00	-23.35	-29.06	0.00	0.00	43.03	0.00
8	1	10	3	0.00	-23.56	-21.51	0.00	0.00	42.83	0.00
8	1	11	3	0.00	22.83	18.45	0.00	0.00	42.62	0.00
8	1	12	3	0.00	22.09	16.89	0.00	0.00	42.45	0.00
8	1	13	3	0.00	21.28	15.52	0.00	0.00	42.24	0.00
8	1	14	3	0.00	20.56	14.71	0.00	0.00	42.05	0.00
8	1	15	3	0.00	19.85	14.01	0.00	0.00	41.80	0.00
8	1	16	3	0.00	19.36	13.40	0.00	0.00	41.56	0.00
8	1	17	3	0.00	18.98	12.88	0.00	0.00	41.32	0.00
8	1	18	3	0.00	18.71	12.52	0.00	0.00	41.16	0.00
8	1	19	3	0.00	18.48	12.32	0.00	0.00	41.07	0.00
8	1	20	3	0.00	18.31	12.21	0.00	0.00	41.03	0.00
8	1	21	3	0.00	-18.37	-12.20	0.00	0.00	41.03	0.00
8	1	22	3	0.00	-18.66	-14.25	0.00	0.00	41.03	0.00
8	1	23	3	0.00	-18.95	-16.17	0.00	0.00	41.03	0.00
8	1	24	3	0.40	-20.89	-36.88	0.40	0.00	41.82	0.00

(Data from Aug.1, 1962, 1:00 - 24:00, Wednesday = day 3)

plied by 1 if condensation did not occur and by -1 when condensation prevents the combine harvesting operation; PROPERTY is therefore moved out of the range of moisture contents that are acceptable as workable, for instance, 0% – 23%. The same is true for the moisture content of straw and soil. The property of the bales on the field is the rain intensity in [mm/h], 0.4 mm at 24:00. A data file, consisting of lines only at clocktimes when a property changes significantly, suffices; removing the first nine lines is possible. Properties can therefore be given at irregular intervals.

5.1.2 Decision

Class DECISION in the base model, Table 4.101, mainly declares Booleans as signals to know the reason why it was activated. The definition is extended for deciding according to the heuristic urgency strategy (Elderen, 1977, p. 13-33). If one want to select man-machine systems according to solutions of linear programming models or to the strategy of a dynamic programming model of the scheduling problem, then an appropriate subclass of DECISION must be defined. If one wants to use some procedures of the urgency strategy, a subclass of URGENCY-DCSN is useful.

The main idea of the urgency strategy is the use of timeliness data of a material to find an urgency, to transform this urgency of a material to an urgency of a gang or a combination (set of gangs) able to process the material by an operation and to select gangs by using their urgency. To select efficiently sets of combinations were distinguished (Section 4.2.3 MM-SYSTMS-SET), where within each set a selection takes place independent of other sets as far as the use of men & machinery is concerned. An example is to put all the gangs with men in one set and all gangs without men such as drying installations, working automatically, in another set (Table 2.5). Gangs from a contract worker can be put in a third set. It is also possible to handle sets for different groups of men and machines, which do not interfere; a set related to arable work, a set for poultry or cattle work. For this reason a queue is used in a subclass of COMPONENT (Table 4.77); it has the advantage of making a set available by AVLB-COMP equal to true during some periods as defined in SH-PRD and during certain hours in a week as defined in SH-WK (Sections 4.4.1 and 4.4.2). It is elegant to work with different sets for different seasons. At the moment gangs or combinations are created in input they are put into the queue of the prescribed set. This short description of man-machine system sets suffices to start with the description of the decision making itself.

The declaration of variables of subclass URGENCY-DCSN is shown in Table 5.6. The function of these variables and those of the superclass DECISION (Table 4.101) will become clear on the following pages. The dynamic section is shown in Table 5.7 and shows that all processes scheduled at the same time precede DECISION except administration; this is achieved by reactivating the DECISION process. So if attributes are updated by WEATHER, then the consequences in other processes are considered be fore a decision is made. A series of Boolean variables is used as signals when a decision was required for some reason. Shifts changing the state of the system within a week and per period set SHIFT-CH. A machine failure, MACHN-FLR, an achievement of a maximum quantity, MAT-MAX-QNT and a material no longer processable, MAT-DOWN can have the consequence of stopping some operations. When a machine is repaired,

Table 5.6 Class URGENCY-DCSN; declaration of variables.

```
DECISION class URGENCY_DCSN;                          !**************************************;
begin
   !    d    e    c    l    a    r    a    t    i    o    n;

   boolean                                            !-----------------------;
   LOOP_BREAK,                                        !interrupts too many decisions at the same time;
   URGNC_CALC, WTHR_UPDATE, MM_S_SELCT_P, MCH_FLR_STOP;   !reasons of calling dyn.;
   integer                                            !-----------------------;
   G, I4, I6, K, C_S, STOP, START, LOOP_CNT;
   real                                               !-----------------------;
   TIME_PREV, URG_MAX;
   ref (MAN_MACH_SYS) array                           !-----------------------;
   MM_S_OLD, MM_S_NEW [1:MM_S_SETS];
   ref (MAN_MACH_SYS) MM_S3, MM_S2;
   ref (MAN_MACH_SET) GNG1;
   ref (RECORD_MM_S) REC_MM_S;
   ref (RECORD_MAT) REC_M;
   ref (Head) MAT_PROD_Q;
```

Table 5.7 Dynamic section of class URGENCY-DCSN.

```
! d y n a m i c;

while not END_EXPRMNT do
if Nextev.Evtime = Time and Nextev =/= ADMNSTR
then reactivate this DECISION at Time
!                                         !almost all other processes are done earlier;
else begin
    HOUR_AT_TIME;
    if not ADMN.Idle then ADMNSTR.DISPLAY_DATA_;
    if LOOP_BREAK and TIME_PREV < Time then LOOP_BREAK:= false;
    while (SHIFT_CH or                                !shift change or costs increased;
    (MACHN_FLR or MAT_MAX_QNT or MAT_DOWN)            !combination has to stop;
    or (MACHN_OK or MAT_PASS or MAT_DLV_OK or WTHR_UPDATE or PRC_END
    or URGNC_CALC))                                   !new combination may be considered;
    and not LOOP_BREAK
    do begin

        LOOP_CNT:= if TIME_PREV < Time then 0 else LOOP_CNT + 1;
        if LOOP_CNT >= 10 then LOOP_BREAK := true;    !arbitrary stop after 10 repetitions;
        if LOOP_BREAK then begin
            Outimage;
            Outtext ('The 10th decision was made at the same time:'); Outfix (Time,6,12);
            Outtext (', this sequence is interrupted at the 11th decision!'); Outimage;
            Outtext ('The machine systems selected at the 9th, 10th and 11th step are:');
            Outimage;
            for I4:= 1 step 1 until MM_S_SETS do if MM_S_OLD[I4] =/= none then
            Outtext (MM_S_OLD [I4].NAME_COMP);
            Outimage;
            for I4:= 1 step 1 until MM_S_SETS do if MM_S_NEW[I4] =/= none then
            Outtext (MM_S_NEW [I4].NAME_COMP);
            Outimage;
        end;
        TIME_PREV:= Time;
        MCH_FLR_STOP:= MACHN_FLR;
        SHIFT_CH:= MACHN_FLR:= MAT_MAX_QNT:= MAT_DOWN:= false;
        MACHN_OK:= MAT_PASS:= MAT_DLV_OK:= WTHR_UPDATE:= PRC_END:= false;
        URGNC_CALC:= false;
        for I4:= 1 step 1 until MATRLS_PROC do inspect MATRL_PRC [I4] do begin
            if DELIVERING and SUPPLY_NOD then DELIVER_MAT;! quant_avlbl updated;
            URG_MAT_EXT_;                             !updates urgency at time;
        end;

        URG_GANGS_;
        MM_S_SELCT_P:= MM_S_SELECT;
        for I4:= 1 step 1 until MM_S_SETS do MM_S_OLD [I4]:- MM_S_NEW [I4];
        for C_S:= 1 step 1 until MM_S_SETS do
        if END_SEASON or WTHR.ENDED_DATA or not MM_S_SET [C_S].AVLB_COMP
        then MM_S_NEW [C_S]:- none
        else begin
            APPLCB_GANGS;
            APPLCB_MM_SYSTMS;
            SELECT_MM_S;
        end;
        STOP_START_OPRTNS;
        if LOOP_BREAK then begin
            for I4:= 1 step 1 until MM_S_SETS do if MM_S_NEW[I4] =/= none then
            Outtext (MM_S_NEW [I4].NAME_COMP);
            Outimage;
        end;
    end of while causes to change combinations;

    Passivate;
end of else_branch and of while, so repeat;
```

MACHN-OK, a material became processable, MAT-PASS, the delivery became true, MAT-DLV-OK, the weather data are updated, WTHR-UPDATE, processing conditions of a material changed, PRC-CND or the urgency of passive gangs was calculated, URGNC-CALC then other combinations are possible and may be preferrred. These variables are made false and can become true again due to the decisions made. The urgency of materials at the current time is updated by calling URG-MAT-EXT- (Table 4.44) and sometimes the available quantity by calling DELIVER-MAT. The urgency of the gangs is derived in URG-GANGS-. For each set i the selected man-machine system is recorded in MM-S-OLD[i]; a new system MM-S-NEW[i] is required if the set i is available (AVLB-COMP true). The selection is performed after deriving the applicability of gangs and man-machine systems by calling APPLCB-GANGS, APPLCB-MM-SYSTMS and SELECT-MM-S. These procedures will be described on the following pages along with STOP-START-OPRTNS, which performs the stopping of operations in MM-S-OLD and starting of operations in MM-S-NEW. The execution continues (due to while ...) with considering if some new signals are set which require a new decision otherwise it calls PASSIVATE for waiting until the next activation time. Such a 'while' statement can cause an unexpected loop so a LOOP-BREAK was inroduced that forces the acceptance of the eleventh decision at one moment; a message is sent to indicate the moment and the selected man-machine systems.

Now the procedures called in the dynamic section are considered. The first one is URG-GANGS- (Table 5.8), which calculates the urgency of gangs and is called from the dynamic section of decision or from the shift controlling the urgency calculation of materials (Table 4.108 in Section 4.4.3 'Shifts for materials'). It calls URGENCY-GANG- (Table 4.71) to calculate the urgency of a gang and may activate this class. The applicability of gangs within a particular set of man-machine systems C-S is handled in APPLCB-GANGS (Table 5.9). Only gangs are handled by using the condition:

inspect REC-MM-S.MN-MCH-SYS when MAN-MACH-SET do

which excludes GANG-SET. Now an applicability APPL-M-AVLBL is defined that requires that processed materials are available. This depends on:

Table 5.8 Procedure URG-GANGS- of class URGENCY-DCSN.

```
procedure URG_GANGS_;                                 !----  ----  ----  ----  ----;
begin                                                 !called in shift_urg, this dynamic;
    for K:= 1 step 1 until GANGS do begin
        GANG (K).URGENCY_GANG_;
        if GANG (K).STATE = PASSIVE and GANG(K).URGENCY_CORR > 0.0 then URGNC_CALC:= true;
    end;
    if URGNC_CALC then activate DECIDE at Time;
    if Current == this DECISION then URGNC_CALC:= false;        !no extra decision is wanted;
end;
```

Table 5.9 Procedure APPLCB-GANGS of cLass URGENCY-DCSN.

```
procedure APPLCB_GANGS;                            !----  ----  ----  ----  ----;
begin                                              !called in dynamic;
   boolean APPL;
   REC_MM_S:- MM_S_SET [C_S].MM_S_SET_Q.First;
   while REC_MM_S =/= none do begin                !man_mach_set objects are gangs;
      inspect REC_MM_S.MN_MCH_SYS when MAN_MACH_SET do begin
         APPL_M_AVLBL:= STATE <> DOWN              !state of gang / operation resp. ;
         and (if OPRTN__G =/= none then OPRTN__G.STATE <> DOWN else false)
         and (URGENCY_CORR >= 0.0);                !and urgency - costs >= 0.0;
         if APPL_M_AVLBL then inspect OPR_MT__G do begin
            for I6:= 1 step 1 until MATRLS_PRDCD do
            APPL_M_AVLBL:= APPL_M_AVLBL and MAT_PROD [I6].DLVR_ALLWD;
            !      !n.k. if ...&mat_proc[] available (or mach. for repair);
            APPL:= true;
            for I6:= 1 step 1 until MATRLS_PRCSD do
            APPL:= APPL  and  not MAT_PROC [I6].SUPPLY_NDD;
            !appl, applicable can change if mat. is delivered in a combination;
         end;
         if OPR_SR__G =/= none then APPL:= OPR_SR__G.MACHN =/= none;
         APPLICABLE:= APPL and APPL_M_AVLBL;
      end of man_mach_set and inspect;
      REC_MM_S:- REC_MM_S.Suc;
   end of while;
end ** of deriving applicability of gangs;
```

- all the necessary elements from men & machinery are available, STATE <> DOWN;
- the related operation (if any) is not DOWN, which guarantees that the attributes of the processed materials are appropriate for processing (incl. AVLBL);
- the urgency corrected for costs is positive;
- the materials produced can be delivered, DLVR-ALLWD.

An overall applicability, APPLICABLE, was derived which becomes true when all materials processed do not need supply (have some acreage: APPL is true) and APPL-M-AVLBL is true. If the acreage of a processed material is positive then APPLICABLE equals APPL-M-AVLBL, but if the acreage is zero (supply needed) then APPLICABLE is false and APPL-M-AVLBL may be true as if the material will become available by delivery. For example first there is combine-harvesting, wet grain will be delivered, availability of the wet grain (Table 4.17) changes AVLB to true and delivery is anticipated; the state of wet grain is changed and another decision is required that will select also grain drying. For gangs related to service and repair operations, APPL is true when a machine needs repair or service.

It is rather complicated to derive an applicability of man-machine systems because one gang, which is applicable can deliver the material needed by another gang in the combination, which was not yet applicable only for the need of supply of material. The procedure APPLCB-MM-SYSTMS is shown in Table 5.10 and contains procedure SAVE-MAT-PRD, that is used later on. The execution starts with the gang or combination in the current set MM-S-SET [C-S]. Attention is payed first to the case with two or more gangs in the combination. The urgencies are cumulated, the intersection of

Table 5.10 Procedure APPLCB-MM-SYSTMS of class URGENCY-DCSN.

```
procedure APPLCB_MM_SYSTMS;                                  !----  ----  ----  ----  ----;
begin                                                        !called in dynamic;

    procedure SAVE_MAT_PRD (OPRTN_MT); ref (OPRTN_MATRL) OPRTN_MT;
    begin
        integer I8;
        inspect OPRTN_MT do
        for I8:= 1 step 1 until MATRLS_PRDCD do
        new RECORD_MAT (MAT_PROD (I8)).Into  (MAT_PROD_Q);
    end;

    REC_MM_S:- MM_S_SET (C_S).MM_S_SET_Q.First;
    while REC_MM_S =/= none do
    inspect REC_MM_S.MN_MCH_SYS do begin
        boolean APPLY_M_DLVR, APPLCBL, APPLCBL_MM_S;
        real URG1, URG2;
        APPLY_M_DLVR:= APPLCBL:= APPLCBL_MM_S:= true;
        for G:= 1 step 1 until GNGS do begin    !g in sequence of processing materials;
            GNG1:- GANG (RORD_GNG (G));
            APPLY_M_DLVR:= APPLY_M_DLVR  and  GNG1.APPL_M_AVLBL;
            URG1:= URG1 + GNG1.URGENCY;
            URG2:= URG2 + GNG1.URGENCY_CORR;
            APPLCBL:= APPLCBL  and  GNG1.APPLICABLE;
        end;
        URGENCY:= URG1;
        URGENCY_CORR:= URG2;
        APPLY_M_DLVR:= APPLY_M_DLVR and URGENCY_CORR >= 0.0;!urgency - costs > 0.0;
        APPLCBL:= APPLCBL and URGENCY_CORR >= 0.0;      !urgency - costs > 0.0;

        if not APPLY_M_DLVR then APPLICABLE:= false     !state,urgency or dlvr not ok;
        else    if APPLCBL then APPLICABLE:= true       !ok;
        !;      else if C_S > 1 or GNGS > 1 then begin  !material needed may be produced;
            MAT_PROD_Q:- new Head;
            !fill mat_prod_q with materials produced with man-mach.systems already selected;
            I6:= C_S -1;                                !all mm_s_new from preceding mm_s_sets;
            for I4:= 1 step 1 until I6 do inspect MM_S_NEW (I4)
            when MAN_MACH_SET do if OPR_MT__G =/= none then SAVE_MAT_PRD (OPR_MT__G)
            when GANG_SET do for G:= 1 step 1 until GNGS do
            if GANG (RORD_GNG (G)).OPR_MT__G =/= none then
            SAVE_MAT_PRD (GANG (RORD_GNG (G)).OPR_MT__G);

            for G:= 1 step 1 until GNGS do if APPLCBL_MM_S then begin
                !g in sequence of processing mat.!! -> saved before requested as mat_proc;
                boolean PROCSNG_G_OK;
                PROCSNG_G_OK:= true;
                GNG1:- GANG (RORD_GNG (G));
                if GNG1.APPLICABLE then SAVE_MAT_PRD (GNG1.OPR_MT__G)
                else inspect GNG1.OPR_MT__G do
                for I6:= 1 step 1 until MATRLS_PRCSD do
                if MAT_PROC (I6). SUPPLY_NDD and PROCSNG_G_OK then begin
                    boolean PROCSNG_M_OK;
                    REC_M:- MAT_PROD_Q.First;
                    while not PROCSNG_M_OK  and  REC_M =/= none do begin
                        PROCSNG_M_OK:= MAT_PROC (I6) == REC_M.MATRL_RF;
                        REC_M:- REC_M.Suc;
                    end;
                    PROCSNG_G_OK:= PROCSNG_G_OK  and  PROCSNG_M_OK;
                end;
                APPLCBL_MM_S:= APPLCBL_MM_S  and  PROCSNG_G_OK;
            end of testing gangs in combination;
            APPLICABLE:= APPLCBL_MM_S;
        end of try delivering m;
        REC_MM_S:- REC_MM_S.Suc;
    end of inspect and mm_s_set;
    end ** of deriving applicability of man-machine systems;
```

all APPL-M-AVLBL of the gangs involved is assigned to APPL-M-DLVR; if true then the applicability of the combination guarantees the availability or delivery of materials processed, otherwise the application of the gang is not allowed for some reason or another. The intersection of APPLICABLE of the gangs involved is assigned to APPLCBL; if true then the availability of the processed materials is guaranteed, otherwise some material for processing has to be delivered (supply needed) and it is uncertain if that is possible. A non-negative urgency is required for both variables. Three situations now occur:

- APPL-M-DLVR is false then the combination cannot be used due to men & machinery, to urgency of at least one gang or to delivery is not allowed; APPLICABLE becomes false;
- APPLCBL is true (implies APPL-M-DLVR is true) so all the materials processed are available; APPLICABLE becomes true;
- APPL-M-DLVR is true but APPLCBL is false then one of the gangs cannot perform work until material is delivered so an attempt is made to see if another gang in the combination delivers the required material.

For the last situation the procedure SAVE-MAT-PRD is used to save in a new queue MAT-PROD-Q the materials produced. It is assumed that gangs in a combination are ordered in the sequence of processing so a gang producing a material precedes a gang processing that material. Now if the first gang is applicable, then the materials produced are saved in MAT-PROD-Q and APPLCBL-MM-S remains true. If the second gang needs supply of a material, then the queue is checked until that material is found; this results in PROCSNG-M-OK becoming true, PROCSNG-G-OK and APPLCBL-MM-S remaining true and APPLICABLE becoming true. If such a material is not found, however, then PROCSNG-M-OK remains false, PROCSNG-G-OK becomes false along with APPLCBL-MM-S and APPLICABLE. An example is useful; if there is straw in the field but no bales the following can happen. When the selection of straw baling is started, then the combination baling and gathering still cannot be used because the bales are not yet available (supply needed). Selection of baling however results in START-OPRTN- (Table 4.86) of the baling operation and in SUPPLY-EXPCT (Table 4.17) of the material produced i.e. bales in the field, that makes AVLBL true, updates the state of that material and requires another decision. The gathering operation is then in the situation of APPL-M-AVLBL true and APPLICABLE false. If baling and gathering can be performed simultaneously (not obvious), i.e. occur in one combination, then APPL-M-DLVR is true and APPLCBL false; so an attempt is made to find if the other gang (baling) produces the bales in the field, if so APPLICABLE of the combination becomes true. Finally it is possible to explain how this procedure is used for a similar check for gangs in succeeding sets, for instance, wet grain delivered in the first set by harvesting and needed in the second set by the drying operation. For this purpose the queue is filled beforehand with materials

Table 5.11 Procedure SELECT-MM-S of class URGENCY-DCSN.

```
procedure SELECT_MM_S;                          !----  ----  ----  ----  ----;
begin                                           !called in dynamic;
    URG_MAX:= -0.01;
    MM_S2:- none;
    REC_MM_S:- MM_S_SET [C_S].MM_S_SET_Q.First;
    while REC_MM_S =/= none do begin
        MM_S3:- REC_MM_S.MN_MCH_SYS;
        if URG_MAX < MM_S3.URGENCY_CORR and MM_S3.APPLICABLE then begin
            MM_S2:- MM_S3;                      !only one mm_s is chosen from a set;
            URG_MAX:= MM_S3.URGENCY_CORR;
        end;
        REC_MM_S:- REC_MM_S.Suc;
    end;
    MM_S_NEW [C_S]:- MM_S2;
end !! of selecting combinations;
```

Table 5.12 Procedure STOP-START-OPRTNS of class URGENCY-DCSN.

```
procedure STOP_START_OPRTNS;                    !----  ----  ----  ----  ----;
begin                                           !called in dynamic;
   integer array GANG_DECSN [1:GANGS];
   MM_S_SELECT:= false;
   for I4:= 1 step 1 until MM_S_SETS do begin
       if MM_S_OLD [I4] =/= MM_S_NEW [I4] or MCH_FLR_STOP then begin
           if MM_S_OLD [I4] =/= none then MM_S_OLD [I4].STOP_WORK_C_;  !gangs have a dummy..;
           if MM_S_NEW [I4] =/= none then MM_S_NEW [I4].START_WORK_C_; !..virtual procedure;
           J:= if MM_S_OLD [I4] =/= none then  MM_S_OLD [I4].GNGS else 0;
           for G:= 1 step 1 until J do
           GANG_DECSN [MM_S_OLD [I4].RORD_GNG [G]]:= STOP;          !stop;
           J:= if MM_S_NEW [I4] =/= none then MM_S_NEW [I4].GNGS else 0;
           for G:= 1 step 1 until J do
           GANG_DECSN [MM_S_NEW [I4].RORD_GNG [G]]:=
           GANG_DECSN [MM_S_NEW [I4].RORD_GNG [G]] + START;         !(stop)+start;
           J:= if MM_S_OLD [I4] =/= none then MM_S_OLD [I4].GNGS else 0;
           for G:= 1 step 1 until J do
           if GANG_DECSN [MM_S_OLD [I4].RORD_GNG [G]] = STOP
           or GANG [MM_S_OLD [I4].RORD_GNG [G]].FAILURE_E then
           GANG [MM_S_OLD [I4].RORD_GNG [G]].OPRTN__G.STOP_OPRTN_;    !-> stop_work_g;
       end;
       J:= if MM_S_NEW [I4] =/= none then MM_S_NEW [I4].GNGS else 0;
       for G:= 1 step 1 until J do
       GANG [MM_S_NEW [I4].RORD_GNG [G]].OPRTN__G.START_OPRTN_;       !-> start_work_g;
       !                                              !no effect if already started;
       MM_S_SELECT:= MM_S_SELECT or MM_S_NEW [I4] =/= none;
   end;
end;
```

produced by the man-machine systems MM-S-NEW selected in the preceding sets.

After deriving the corrected urgency and the applicablility of gangs and combinations within a set, the man machine system is selected with the maximum urgency from the applicable systems. The selected one is referred to by MM-S-NEW [ ] (Table 5.11).

The selection results in stopping and starting of operations, Table 5.12. For each set the previous selected system, MM-S-OLD, and the selected one MM-S-NEW are compared. If they are different or a failure occurred then STOP-WORK-C- and START-WORK-C- are called in combinations (in gangs they are virtual dummy procedures) to change the state (Table 4.72). Later it checks which gangs have to be stopped, or started or remain unchanged (i.e. STOP and START). The last situation avoids an unnecessary stop and start and the related set-up of a gang. The operations are stopped by calling STOP-OPRTN-, that in its turn calls STOP-WORK-G (Tables 4.92 and 4.70) and started by START-OPRTN-, that calls START-WORK-G (Tables 4.86 and 4.69) to change the state of the gang. START-OPRTN- is called for each operation selected and has no effect if it was already started (STATE not PASSIVE). MM-S-SELECT becomes true if any man-machine system is selected.

5.1.3 Administration and messages: output

Class ADMINISTRATOR of the base model, Table 5.13, is not described there because it only contains two 'virtual' procedures to request at appropriate moments and points in the program the updating of output data. Subclass ADMINISTRATION extends this definition by defining procedures to record data and to write data. Table 5.14 shows the declaration of variables.

Table 5.13 Class ADMINISTRATOR and its virtual procedures.

```
COMPONENT class ADMINISTRATOR;                      !**************************************;
virtual: procedure PERD_OUT_M_, DISPLAY_DATA_;
begin
   l     d     e     c     l     a     r     a     t     i     o     n;

   procedure PERD_OUT_M_ (P_M);     ref (PROCESSED_MAT) P_M;;
   procedure DISPLAY_DATA_;;
end;
```

Table 5.14 Class ADMINISTRATION and its declaration of variables.

```
ADMINISTRATOR class ADMINISTRATION;                 !**************************************;
begin
   l     d     e     c     l     a     r     a     t     i     o     n;

   integer                                          !----------------------;
   IS, ST, M, O, HR, HR_PREV, LT_HR;
   real array                                       !----------------------;
   COSTS_CATGRY [1:5];      !categories of men, mach., gangs, operations, materials resp.;
   real                                             !----------------------;
   QNT, LT_DISPLAY;
   boolean                                          !----------------------;
   EACH_PERIOD, PERD_OUT, DISPLAY;
   text array                                       !----------------------;
   STATE_TEXT_M [1:MATRLS_PROC], STATE_TEXT_O [1:OPRTNS];
   ref (Printfile) REPORT_PERD;                     !----------------------;
   ref (Head) PLOT_Q;
   ref (PLOT_ADMIN) PLOT;
```

Table 5.15 Procedure SHIFT-CHNGE- of class ADMINISTRATION.

```
procedure SHIFT_CHNGE_;                          !----  ----  ----  ----  ----;
begin                                            !called in sh_prd;
   if AVLB_COMP then PERD_OUT:= true;
   AVLB_COMP_PR:= AVLB_COMP;
   if SH_PRD =/= none then AVLB_COMP:= SH_PRD.AVLB_PRD;
   if not END_SEASON then reactivate this ADMINISTRATION at Time;
end;
```

Table 5.15 shows SHIFT-CHNGE- that controls the availability when a period begins; note that SH-WK, shifts in a week, has no influence on the availability of this component. It activates the dynamic part that will use PERD-OUT to write the data of the preceding period.

The recording of data is performed (i) frequently (required by decision) to have each half hour of the day the state of materials and operations and (ii) daily to produce daily and periodical output on costs and use of components. Table 5.16 shows procedure DISPLAY-DATA- that is called in many cases to record for each half hour the state of materials and operations by '+', '-' and ' ' when STATE = RUN, PASSIVE and DOWN respectively (Table 5.19). Procedure DAILY-DATA, Table 5.17, is used to update the accumulation of time and costs of each component and to record the costs per category of components: (1) men, (2) machines, (3) gangs, (4) operations and (5) materials. The sum of the costs of men and machines are equal to the costs of the gangs and to the costs of operations. The additional daily calculations ('tasks') performed are: (i) reset the set-up duration of a gang at the duration for the first time on a day; (ii) update the quantity processed of materials (by calling PROCESS-MAT); (iii) add to the timeliness costs after the season the maximum value of the timeliness loss of the remaining area; (iv) request the display output or write the current day number on a terminal; (v) write that the end of a period has been reached.

Table 5.16 Procedure DISPLAY-DATA of class ADMINISTRATION.

```
procedure DISPLAY_DATA_;                               !----   ----    ----    ----    ----;
if DISPLAY then begin                                  !called in weather, decision, daily_data;
    integer H;          !shift_perd, shift_week, updt_man_mch,serv_repr, mat.consumptn_m;
    HOUR_AT_TIME;                                      !gives huur;
    HR:= 2.0 * HOUR;
    if HR = 0 and LT_HR < 48 then HR:= 48;
    if HR > HR_PREV and LT_DISPLAY < Time - 1.0&-4 then begin
        for M:= 1 step 1 until  MATRLS_PROC do
        if MATRL [M].STATE = PASSIVE
        then begin
            for H:= HR_PREV+1 step 1 until HR do STATE_TEXT_M [M].Putchar ('-');
        end
        else    if MATRL [M].STATE = RUN
        then begin
            for H:= HR_PREV+1 step 1 until HR do STATE_TEXT_M [M].Putchar ('+');
        end
        else STATE_TEXT_M [M].Setpos (HR + 1);
        for O:= 1 step 1 until OPRTNS do
        if OPRTN [O].STATE = PASSIVE
        then begin
            for H:= HR_PREV+1 step 1 until HR do STATE_TEXT_O [O].Putchar ('-');
        end
        else    if OPRTN [O].STATE = RUN
        then begin
            for H:= HR_PREV+1 step 1 until HR do STATE_TEXT_O [O].Putchar('+');
        end
        else STATE_TEXT_O [O].Setpos (HR + 1);
    end;
    LT_DISPLAY:= Time;
    LT_HR:= HR;
    if HR = 48 then HR:= 0;
    HR_PREV:= HR;
end;
```

Table 5.17 Procedure DAILY-DATA of class ADMINISTRATION.

```
procedure DAILY_DATA;                              !----  ----  ----  ----  ----;
if AVLB_COMP or AVLB_COMP_PR then begin            !called in dynamic;
    for IS:= 1 step 1 until 5 do COSTS_CATGRY [IS]:= 0.0;
    for IS:= 1 step 1 until MANN do begin
        MAN [IS].TIME_C_ACCUM;
        COSTS_CATGRY [1]:= COSTS_CATGRY [1] + MAN [IS].COSTS_MADE;
    end;
    for IS:= 1 step 1 until MACHNS do begin
        MACH [IS].TIME_C_ACCUM;
        COSTS_CATGRY [2]:= COSTS_CATGRY [2] + MACH [IS].COSTS_MADE;
    end;
    for IS:= 1 step 1 until MN_MCH_SYSTMS do
    inspect MM_S [IS] do begin
        TIME_C_ACCUM;
        if IS <= GANGS then COSTS_CATGRY [3]:= COSTS_CATGRY [3] + COSTS_MADE;
        if IS <= GANGS then GANG[IS].SETUP_GNG:= GANG[IS].SETUP1;
    end;
    for IS:= 1 step 1 until OPRTNS do begin
        OPRTN [IS].TIME_C_ACCUM;
        COSTS_CATGRY [4]:= COSTS_CATGRY [4] + OPRTN [IS].COSTS_MADE;
    end;
    ENDED_PROCS:= true;
    for IS:= 1 step 1 until MATRLS_PROC do
    inspect MATRL_PRC [IS] do begin
        if PROCESSING then PROCESS_MAT;
        if WTHR.ENDED_DATA then
        COSTS_TMLNSS:= COSTS_TMLNSS + QUANT_AVLBL * PRICE_HA * LOSS_MULT;
        COSTS_MADE:= COSTS_TMLNSS;
        ENDED_PROCS:= ENDED_PROCS and not AVLBL;
        TIME_C_ACCUM;
        COSTS_CATGRY [5]:= COSTS_CATGRY [5] + COSTS_MADE;
    end;
    inspect REPORT_PERD do begin
        Outimage; Outint (DAYS_NMB, 4); Outtext (' Quant_avl'); Setpos(14);
        for IS:= 1 step 1 until MATRLS do
        Outfix(MATRL [IS].QUANT_AVLBL,2,13);Outimage;
        Outtext (' Quant_procsd'); Setpos (14);
        for IS:= 1 step 1 until MATRLS do Outfix (MATRL [IS].QUANT_PROCSD,2,13);
        Outimage;
    end;
    PLOT:- new PLOT_ADMIN; if AVLB_COMP then PLOT.Into (PLOT_Q);
    inspect PLOT do begin
        for IS:= 1 step 1 until MATRLS_PROC do QUANT_M [IS]:= MATRL [IS].QUANT_PROCSD;
        for IS:= 1,2,3,4,5 do COSTS_CUM [IS]:= COSTS_CATGRY [IS];
        COSTS_CUM [6]:= COSTS_CATGRY [3] + COSTS_CATGRY [5];
        DAY_NMB:= DAYS_NMB;
        DAY_TP:= DAY_TYPE_NOW;
    end;
    if DISPLAY then begin
        DISPLAY_DATA_;
        DISPLAY_OUTPUT;
    end else
    begin
        Outtext (DAY_TXT3 [DAY_TYPE_NOW]); Outtext ('   ');
        DATE_FROM_DAYNO (DAYS_NMB); Outtext (MNTH_DTG);
        Outint (DAYS_NMB,4);  Outtext (' :'); Breakoutimage;
    end;
    if PERD_OUT and SH_PRD =/= none then begin
        Outimage; Setpos (119); Outtext ('end period'); Outint (SH_PRD.PERD - 1, 2); Outimage;
    end;
end;
```

Writing data is the purpose of the following procedures. Table 5.18, DISPLAY-OUTPUT, writes daily (if required: DISPLAY = true) a heading, one line for each material and operation with its name, the area processed

Table 5.18 Procedure DISPLAY-OUTPUT of class ADMINISTRATION.

```
procedure DISPLAY_OUTPUT;                                   !----  ----  ----  ----  ----;
begin                                                       !called in daily_data;
    Outtext (DAY_TXT3 [DAY_TYPE_NOW]); Setpos (5);
    DATE_FROM_DAYNO (DAYS_NMB); Outtext (MNTH_DT6); Outint (WTHR.YEAR, 6);
    Setpos (Pos + 10); Outtext (' Time = '); Outint (DAYS_NMB, 6);
    Outtext ('  Clock vs state: +=run, -=passive'); Outimage;
    Outtext
    ('Names       Area [ha]cumCosts: 1 2 3 4 5 6 7:8 910  12: 14  16  18: 20  22: 24:');
    Outimage;
    for M:= 1 step 1 until MATRLS_PROC do inspect MATRL_PRC [M] do begin
        Outtext (NAME12); Outfix (QUANT_PROCSD - QNT_DAY_PR_M, 2,6);
        Outfix (QUANT_PROCSD, 1,6); Outfix (COSTS_MADE, 0,6);
        Outtext (STATE_TEXT_M [M]); Outimage;
        STATE_TEXT_M [M]:- Copy ('               :          : :          :       :   :');
        QNT_DAY_PR_M:= QUANT_PROCSD;
    end; Outimage;
    for O:= 1 step 1 until OPRTNS do inspect OPRTN [O] do begin
        Outtext (NAME12);
        if O<= OPRTNS_MT then begin
            inspect OPR_MAT[O] do begin
                Outfix (QUANT_PRC_OPR - QNT_DAY_PR_O, 2,6); Outfix (QUANT_PRC_OPR, 1,6);
                QNT_DAY_PR_O:= QUANT_PRC_OPR;
            end;
        end else Setpos (Pos+12);
        Outfix (GANG_G.COSTS_MADE, 0,6); Outtext (STATE_TEXT_O [O]); Outimage;
        STATE_TEXT_O [O]:- Copy (STATE_TEXT_M[1]);
    end; Outimage;
    Outtext ('Cum.costs:  (men + machines = operations)  +  materials -->TOTAL costs');
    Outimage;
    Outtext (DAY_TXT3 [DAY_TYPE_NOW]); Setpos (5); Outtext (MNTH_DT6);
    Outfix (COSTS_CATGRY [1], 0,6); Outfix (COSTS_CATGRY [2], 0,11);
    Outfix (COSTS_CATGRY [3], 0,13); Outfix (COSTS_CATGRY [5], 0,15);
    Outfix (COSTS_CATGRY [3] + COSTS_CATGRY [5], 0,15); Outimage;
    Outtext ('= = = = = = = = = = = = = = ='); Outimage;
end;
```

(this day and cumulative), the costs made and the state at each half hour of the day and finally the costs of the men, machines, gangs, materials and in total. An example of the output is shown in Table 5.19 for the wheat har-

Table 5.19 Output for a display terminal as given by procedure DISPLAY-OUTPUT.

```
Mon Aug 13  1962            Time =    225  Clock vs state: +=run, -=passive
Names        Area [ha]cumCosts: 1 2 3 4 5 6 7:8 910  12: 14  16  18: 20  22: 24:
Wheat (winte 11.31  21.7   713-------------++++++++++++++++++++-++++++++++---:
Straw swath   4.00  12.6    93----------------------++++++-------------------:
Bales in fie  3.48  11.8    72--------------------+-------+++----------++----:
Bales on tra  0.00   8.3     0              :       -------------------------:
Stubble fiel  0.00   0.6     0              ---------------------------------:
Wetgrain in   2.46  12.2     0              :       +++++++++++++++++++++++++:

2 men Combin  6.80   7.4   102              ++++++++  : :        +++++++ :   :
1 man Combin  4.51  14.3   135              --------++++++++++++ ------++:   :
Baling......  4.00  12.6    43              ---------++++++--------------:   :
Bale loading  3.48  11.8    47              --------+------+++---------++:   :
Bale gatheri  0.00   0.0     0              -----------------------------:   :
Bale unloadi  0.00   8.3    19              :       ---------------------:   :
1 Ploughing.  0.00   0.6     0              -----------------------------:   :
2 Ploughing.  0.00   0.0     0              -----------------------------:   :
Grain drying  2.46  12.2   379              :       +++++++++++++++++++++++++:
2 men Servic                36              :         : :       + :      :   :
1 man Servic                 0              :         : :         :      :   :

Cum.costs:  (men + machines = operations)  +  materials -->TOTAL costs
Mon Aug 13   383          379           762           878           1640
= = = = = = = = = = = = = =
```

vesting on August 13, 1962. This output is printed on a terminal or on a screen but does not move the cursor on a screen to update data during the simulation.

The periodical output is required at the end of each period (if available AVLB-COMP; Table 5.15) and at the end of the season. Table 5.20 shows that it calls only one procedure, CUM-USE-COSTS, Table 5.21, to write the cumulative use and costs of components. That procedure calls on its turn for each component USE-COSTS, Table 5.22 that writes the name, the time used during the states RUN, PASSIVE and DOWN, the costs and the overtime. Additional output concerns: (i) the heading; (ii) the costs per category of components, such as men, machines, gangs, operations and materials; (iii) the service and repair time of machines; (iv) the set-up time of gangs; (v) the quantity not processed by operations due to material shortage and

Table 5.20 Procedure PERD-OUTPUT of class ADMINISTRATION.

```
procedure PERD_OUTPUT;                              !----  ----  ----  ----  ----;
if (PERD_OUT  and  EACH_PERIOD)   or                !called in dynamic;
END_SEASON then begin
    CUM_USE_COSTS;
    PERD_OUT:= false;                               !set to true in shift_chnge_ of material;
end else PERD_OUT:= false;
```

Table 5.21 Procedure CUM-USE-COSTS of class ADMINISTRATION.

```
procedure CUM_USE_COSTS;                            !----  ----  ----  ----  ----;
inspect REPORT_PERD do begin                        !called in perd_output;
    Eject(1); Outtext (EXPER_IDF); Outimage; Outint (@THR.YEAR,4); Outint (YR_N,4); Setpos(13);
    Outtext('title   Time  (h]:run,passive,down,        Csts:cum/catg');
    Outtext('      Runtm[h] '); Outtext(' until');
    Outfix(TIME_YR,1,6); Outtext(',at date'); Outint(DAYS_NMB,4); Outimage;
    for IS:= 1 step 1 until MANN do USE_COSTS (MAN [IS]);
    Outfix (COSTS_CATGRY [1],0,6); Outimage; Setpos(105); Outtext ('Service,Repair_t[h]');
    for IS:= 1 step 1 until MACHNS do begin
        USE_COSTS (MACH [IS]);
        Setpos (105);
        inspect MACH [IS] do begin
            Outfix (SERVICETIME,2,9); Outfix (REPAIRTIME,2,9);
        end;
    end;
    Setpos(65); Outfix (COSTS_CATGRY [2],0,6); Outimage;
    Setpos(105); Outtext('setup-time (h]');
    for IS:= 1 step 1 until MN_MCH_SYSTMS do if MM_S [IS] =/= none then begin
        USE_COSTS (MM_S [IS]); if IS = GANGS then Outfix (COSTS_CATGRY [3],0,6);
        Setpos(105); if IS <= GANGS then   Outfix(GANG [IS].SETUPTIME,2,9);
    end; Outimage; Setpos(105); Outtext('q_not_/&q_processed[ha]');
    for IS:= 1 step 1 until OPRTNS do begin
        USE_COSTS (OPRTN [IS]);
        Setpos(105); if IS <= OPRTNS_MT then Outfix (OPR_MAT [IS].QNT_NOT_PROC,4,8);
        if IS <= OPRTNS_MT then Outfix (OPR_MAT [IS].QUANT_PRC_OPR,2,10);
    end;
    Setpos (65); Outfix (COSTS_CATGRY [4],0,6); Outimage;
    Setpos (90); Outtext ('Processed,Available,Qnt_dumm,Qnt_dltd[ha]');
    for IS:= 1 step 1 until MATRLS do begin
        USE_COSTS (MATRL [IS]);
        if IS = MATRLS_PROC then Outfix (COSTS_CATGRY [5],0,6);
        Setpos (90); Outfix (MATRL [IS].QUANT_PROCSD,2,9); Outfix (MATRL [IS].QUANT_AVLBL,2,9);
        Outfix (MATRL[IS].QNT_PRC_DUMM,4,9); Outfix (MATRL[IS].QNT_DLTD,4,9);
    end;
    Outimage;
end inspect report;
```

the actual quantity processed; (vi) quantities (= area) of material processed, still available, in shortage (Table 4.30) and deleted if no fields are available (Table 4.31). Table 5.23 shows an example from the wheat harvesting at the end of day number 252 (harvesting completed for 60 ha). An annual summary of costs is also presented together with a list of values of the random number generator drivers used in the machines for failure, repair and service (such a list is also presented at the beginning of an experiment). Figure 5.1 shows the files used to read the input data and a print of the cumulative quantities that are processed and the costs made until a given date. The costs of gangs and of operations are not always equal because the costs of the drying operation may not yet be updated at 24:00.

The dynamic section of the administration object is shown in Table 5.24. It consists of a part repeated daily until the end of the season and a part at the end of the season (writing costs per category of components); both are repeated until all the seasons are passed and the end of the experiment is achieved. When more processes are scheduled at the same moment, then this object is placed at the end (reactivate ... at Time) otherwise it calls DAILY-DATA, PERD-OUTPUT and Hold for one day. At the end of the season an annual summary is produced (Tables 5.23 and 5.27) and a plot like Figures 5.1 and 6.1 is produced by procedure YRLY-PLOT, Table 5.25.

In the initial section of the execution of statements a conversation is used to show if the use of components is reported each period or only at the end of the season: EACH-PERIOD := true or false when the answer is 'yes' or 'no'. Each answer not equal to 'yes' means 'no', even 'YES'. Table 5.26 shows the statements. The use of the display is controlled by DISPLAY (true or false). Table 5.27 shows the conversation on the terminal embedded in the output during the execution of the program for the wheat harvesting of 1962. This output concerns the messages from the program; those from the DEC-10 computer system are not shown. The system messages occur when the file defining the experiment (Section 5.2 'Set-up of experiments: input') could not be found (a new file is requested). The program messages inform the user about the stage of the execution such as Setup ...; Setup of experiment completed ...; Experiment continues ...; Simulation

Table 5.22 Procedure USE-COSTS of class ADMINISTRATION.

```
procedure USE_COSTS (COMP2);                          !----  ----  ----  ----  ----;
ref (COMPONENT) COMP2;                                !called in cum_use_costs;
inspect COMP2 do
inspect REPORT_PERD do begin
    Outimage;
    Outtext (NAME_COMP); Setpos (29);
    for ST:= RUN, PASSIVE, DOWN do Outfix (TIME_USED [ST]*24.0,0,6);
    Setpos (Pos + 8);
    Outfix (COSTS_MADE,2,9);
    if SH_WK =/= none then begin
        Setpos(70); Outfix (T_RUN_OVERTM*24.0,1,10); Outtext (' overtime');
    end; Setpos(65);
end of output of component;
```

Figure 5.1 Quantities processed and costs vs. time.

```
        EXPERIMENT  with aggregated DAILY data, 'Simulation Monograph'                                  EXAMPLE       1986-02-24  09:36:19
Files used in input:      experi.mnt       ENVRMT.XPL       MMSSSF.XPL        MATWWT.XPL        WHTFLD.XPL        GDLR62.XPL

QUANTITIES and COSTS vs. TIME in year:  1962

Date  Day    QUANTITY processed (ha)  /  Quantity expected  60 (ha)             COSTS of Operations,          Crops,  Total DAY
      number                                                                  ( Men + Mach. = Gangs = Oper. ) + Mat. =  sum number


             246802468 246802468 246802468 246802468 246802468 (%)
                     20        40        60        80       100
            :---------:---------:---------:---------:---------:
Aug  5   217:         :         :         :         :         :  Sun        0     0     0     0     0     0  217
         218:5 6 1    :         :         :         :         :  Mon       15    96   111    15    42   153  218
         219:5 461    :         :         :         :         :  Tue       15   162   177   177    42   219  219
Aug  8   220:5-461----:---------:---------:---------:---------:  Wed       15   162   177   177    42   219  220
         221:5  64    :         :         :         :         :  Thu       15   162   177   177   102   279  221
         222:5  64    :         :         :         :         :  Fri       15   162   177   177   102   279  222
         223:5     46 :         :         :         :         :  Sat      233   264   496   394   271   768  223
         224:5     46 :         :         :         :         :  Sun      233   301   534   534   271   805  224
         225:5     4  62      1 :         :         :         :  Mon      383   379   762   684   878  1640  225
         226:5     4  :  6      :    1    :         :         :  Tue      488   460   948   948  1351  2298  226
         227:5      4 :    6 3 2:        1:         :         :  Wed      620   562  1182  1080  1793  2975  227
         228:5      4 :        6:  32     :   1     :         :  Thu      725   706  1431  1185  2398  3830  228
         229:5        4         : 6 3     :   1     :         :  Fri      725   801  1526  1526  2465  3990  229
Aug 18   230:5--------:--4------:---6----3:-------1-:---------:  Sat     1007   903  1910  1808  2941  4851  230
         231:5        :  4      :     6  3:       1 :         :  Sun     1007   940  1947  1947  2941  4888  231
         232:  5      :    . 4  :       6 :   32    :  1      :  Mon     1127  1030  2157  2067  3732  5889  232
         233:  5      :       4 :         6         3  2  1   :  Tue     1232  1093  2325  2283  4225  6550  233
         234:  5      :         :  4      :  6      :      32 :  Wed     1307  1217  2524  2524  4706  7230  234
         235:      5  :         :     4   :  6      :       3 :  Thu     1307  1217  2524  2524  4742  7267  235
         236:      5  :         :     4   :  6      :       3 :  Fri     1307  1217  2524  2524  4742  7267  236
         237:      5  :         :       4 :   6     :         3  Sat     1434  1282  2717  2717  5040  7757  237
         238:      5  :         :       4 :   6     :         3  Sun     1434  1282  2717  2717  5040  7757  238
         239:         :    5    :       4 :   6     :         3  Mon     1434  1282  2717  2717  5040  7757  239
```

Figure 5.1 (continued)

```
Aug 28  240:---------:-----5--:-------4-:----6----:---------3 Tue      1434   1282   2717   2717   5040   7757   240
        241:         :      5 :       4 :   6     :         3 Wed      1434   1282   2717   2717   5040   7757   241
        242:         :        :  5    4 :   6     :         3 Thu      1434   1282   2717   2717   5040   7757   242
        243:         :        :       4 : 5 6     :         3 Fri      1434   1282   2717   2717   5040   7757   243
        244:         :        :       4 : 5 6     :         3 Sat      1434   1282   2717   2717   5040   7757   244
        245:         :        :       4 : 5 6     :         3 Sun      1434   1282   2717   2717   5040   7757   245
        246:         :        :       4 :   6     :  5      3 Mon      1434   1282   2717   2717   5040   7757   246
        247:         :        :       4 :   6     :         5 Tue      1434   1282   2717   2717   5040   7757   247
        248:         :        :       4 :   6     :         5 Wed      1434   1282   2717   2717   5040   7757   248
        249:         :        :       4 :   6     :         5 Thu      1434   1282   2717   2717   5040   7757   249
Sep  7  250:---------:--------:-------4-:----6----:---------5 Fri      1434   1282   2717   2717   5040   7757   250
        251:         :        :       4 :   6     :         5 Sat      1434   1282   2717   2717   5040   7757   251
Sep  9  252:         :        :       4 :   6     :         5 Sun      1434   1282   2717   2717   5040   7757   252

            1 = Wheat (winter)..........
            2 = Straw swath in field....
            3 = Bales in field..........
            4 = Bales on trailers.......
            5 = Stubble field...........
            6 = Wetgrain in dryer store.
```

Table 5.23 Output on a diskfile (PERIOD.LPT) showing the use and costs of components.

```
       EXPERIMENT  with aggregated DAILY data, 'Simulation Monograph'                      EXAMPLE       1986-02-24  09:36:19
1962   1     title   Time  [h]:run,passive,down,        Costs:cum/catg       Runtm[h]  until 252.0,at date 252
```

title	run	passive	down	Costs: cum	catg	Runtm[h]		
Man 1..................	185	265	0	688.06		45.9 overtime		
Man 2..................	187	263	0	746.43	1434	49.8 overtime		
							Service,Repair_t[h]	
Tractor 1..............	174	5874	0	0.00	.		0.00	0.00
Tractor 2..............	174	5874	0	0.00			0.00	0.00
Combine harvester 4.5m..	88	5931	29	0.00			4.80	0.00
Baler..................	56	5992	0	0.00			0.00	0.00
Bale (un-)luader........	86	5962	0	0.00			0.00	0.00
Grain trailer 1, 1ha....	88	5960	0	0.00			0.00	0.00
Grain trailer 2, 1ha....	88	5960	0	0.00			0.00	0.00
Bale trailers, 4*0.5ha..	86	5962	0	0.00			0.00	0.00
Plough 1, 1.5m.........	64	5984	0	0.00			0.00	0.00
Plough 2, 1.0m.........	55	5993	0	0.00			0.00	0.00
Grain dryer, 2ha........	214	5834	0	1282.12	1282		0.00	0.00
							setup-time [h]	
2 men Combine harvesting	15	415	5618	132.00			1.80	
1 man Combine harvesting	73	357	5618	496.18			9.30	
Baling..................	56	394	5598	337.84			10.90	
Bale loading............	32	418	5598	185.52			3.40	
Bale gathering..........	37	413	5598	168.85			2.60	
Bale unloading..........	18	432	5598	67.66			3.00	
1 Ploughing.............	64	386	5598	0.00			6.40	
2 Ploughing.............	55	395	5598	0.00			4.46	
Grain drying............	214	5834	0	1282.12			0.00	
2 men Service & repair..	5	445	5598	46.44			0.00	
1 man Service & repair..	0	450	5598	0.00	2717		0.00	
Cmb+Baling..............	33	397	5618	374.03				
Cmb+Loading.............	10	420	5618	64.61				
Cmb+Gathering...........	14	417	5618	109.55				
Cmb+Unloading...........	3	427	5618	38.13				
Cmb+1Ploughing..........	0	430	5618	0.00				
Bal+Loading.............	8	442	5598	0.00				
Bal+Gathering...........	5	445	5598	30.00				
Bal+Unloading...........	0	450	5598	0.00				
Bal+1Ploughing..........	0	450	5598	0.00				
Load+1Ploughing.........	3	447	5598	0.00				
Gath+1Ploughing.........	1	449	5598	0.00				
Unl+1Ploughing..........	4	446	5598	0.00				
1Plg+2Ploughing.........	55	395	5598	0.00				

Table 5.23 (continued)

```
                                                                                             q_not_/&q_processed(ha)
2 men Combine harvesting     15    10  6024        132.00                                       0.0000     12.96
1 man Combine harvesting     73    26  5949        496.18                                       0.0000     47.04
Baling..................     56    64  5928        337.84                                       1.4790     60.00
Bale loading............     32   106  5910        185.52                                      -0.0001     33.51
Bale gathering..........     37   102  5910        168.85                                      -0.0000     26.49
Bale unloading..........     18   105  5925         67.66                                      -0.0002     33.51
1 Ploughing.............     64   117  5867          0.00                                       0.0000     37.29
2 Ploughing.............     55   126  5867          0.00                                       0.0000     22.71
Grain drying............    214     0  5834       1282.12                                      -0.0000     41.72
2 men Service & repair..      5    15  6028         46.44
1 man Service & repair..      0     0  6048          0.00   2717
                                                                          Processed,Available,Qnt_dumm,Qnt_dltd(ha)
Wheat (winter)..........     88    74   679       3695.40         37.5 overtime   60.00    0.00   0.0000   0.0000
Straw swath in field....     56   115   669        745.81         22.5 overtime   60.00    0.00   1.4790   0.0000
Bales in field..........     68   166   606        598.77         23.6 overtime   60.00    0.00  -0.0001   0.0000
Bales on trailers.......     18   168   654          0.00          4.5 overtime   33.51    0.00  -0.0002   0.0000
Stubble field...........     64   255   521          0.00          0.0 overtime   60.00    0.00   0.0000   0.0000
Wetgrain in dryer store.    214     0   626          0.00   5040  77.7 overtime   41.72    0.00  -0.0000   0.0000
........................      0     0     0          0.00                          0.00   60.00   0.0000   0.0000
........................      0     0     0          0.00                          0.00   60.00   0.0000   0.0000
........................      0     0     0          0.00                          0.00   60.00   0.0000   0.0000

Year  1962  No.    1  Annual Summary of Results.

Cum.costs:  (men + machines = operations)  +  materials -->TOTAL costs
Sun Sep  9  1434      1282        2717           5040            7757

Wheat (winter)..........  Timeliness losses are: Dfl. 3695.400482
Straw swath in field....  Timeliness losses are: Dfl.  745.813995
Bales in field..........  Timeliness losses are: Dfl.  598.768356
        EXPERIMENT  with aggregated DAILY data, 'Simulation Monograph'                    EXAMPLE        1986-02-24  09:36:19
Machine Random number driver U_FL_RP      U_SRVC and driver U of drivers at time:  500.000000
Tractor 1............... 25889954655     7456260      6203608
Tractor 2............... 28213752259     5005808      6203608
Combine harvester 4.5m.. 14462391249 11741273487      6203608
Baler................... 13036873933     8233927      6203608
Bale (un-)loader........ 32645902289     6344958      6203608
Grain trailer 1, 1ha.... 25213121347     4056762      6203608
Grain trailer 2, 1ha.... 16923066691     8245150      6203608
Bale trailers, 4*0.5ha.. 23725407807      598195      6203608
Plough 1, 1.5m.......... 29445642297     7379012      6203608
Plough 2, 1.0m.......... 25669295745     7785705      6203608
Grain dryer, 2ha........  9003842577     6203608      6203608
```

Table 5.24 Dynamic section of class ADMINISTRATION.

```
!     d     y     n     a     m     i     c;

while not END_EXPRMNT do begin
    !           s t a r t;
    while not END_SEASON do
    if Nextev.Evtime = Time then reactivate this ADMINISTRATION at Time
    !           !delayed until other processes at same moment are executed;
    else begin
        DAYS_NMB:= Entier (TIME_YR + 1.0&-4);    !days_nmb at 24h00 = past day;
        DAY_TYPE_NOW:= Mod (DAY_TP_1JAN + DAYS_NMB - 2, 7) + 1;
        DAILY_DATA;
        END_SEASON:=
        (ENDED_PROCS  and  SH_PRD.PERD_END (SH_PRD.PERD - 1) = DAYS_NMB) or WTHR.ENDED_DATA;
        PERD_OUTPUT;
        AVLB_COMP_PR:= false;
        if not END_SEASON then begin
            if AVLB_COMP then Hold(1.0)
            else Passivate;                     !activated in sh_prd;
        end;
    end of while not end_of_season;

    !           s e a s o n   f i n i s h e d;

    if END_SEASON then begin
        ref (Printfile) SUMMARY_TXT;

        procedure YRLY_SUMMARY;                         !----   ----   ----   ----   ----;
        inspect SUMMARY_TXT do begin
            Outimage;
            Outtext ('Year'); Outint (WTHR.YEAR,6); Outtext ('  No.'); Outint (YR_N,6);
            Outtext ('  Annual Summary of Results.'); Outimage;
            for IS:= 1 step 1 until MATRLS_PROC do
            if MATRL[IS].QUANT_AVLBL > 1.0&-4 then begin
                Outtext (MATRL[IS].NAME_COMP); Outtext ('  still available');
                Outfix (MATRL[IS].QUANT_AVLBL,6,12); Outtext (' (ha)'); Outimage;
            end; Outimage;
            Outtext ('Cum.costs:  (men + machines = operations)  +  materials -->TOTAL costs');
            Outimage;
            Outtext (DAY_TXT3 [DAY_TYPE_NOW]); Setpos (5); Outtext (MNTH_DT6);
            Outfix (COSTS_CATGRY [1], 0,6); Outfix (COSTS_CATGRY [2], 0,11);
            Outfix (COSTS_CATGRY [3], 0,13); Outfix (COSTS_CATGRY [5], 0,15);
            Outfix (COSTS_CATGRY [3] + COSTS_CATGRY [5], 0,15); Outimage; Outimage;
            for IS:= 1 step 1 until MATRLS_PROC do
            if MATRL [IS].COSTS_MADE > 0.0 then begin
                Outtext (MATRL[IS].NAME_COMP); Outtext ('  Timeliness losses are: Dfl.');
                Outfix (MATRL[IS].COSTS_MADE,6,12); Outimage;
            end;
        end;

        for SUMMARY_TXT:- REPORT_PERD, Sysout do YRLY_SUMMARY;
        REPORT_PERD.Eject(1);
        YRLY_PLOT;

        Passivate;                      !wait until new season, then activated by sh_prd;
    end;
end ** of season and of experiment;
```

completed (Table 5.27) or are related to the generation of combinations (Table 4.79). For reference purposes a code 'EXAMPLE' is used with additional information about the experiment, the date of running and the clock-time; this line is written on the terminal as well as on the output files. A user can meet messages that are sent as warnings or errors in the program, Tables 4.30 and 4.31 (consumption of material), 4.69 (assembling men and

Table 5.25 Procedure YRLY-PLOT of class ADMINISTRATION and Class PLOT-ADMIN.

```
procedure YRLY_PLOT;                              !----  ----   ----   ----   ----;
inspect REPORT_PERD do begin                      !called in dynamic if end_season;
   Outtext (EXPER_IDF); Outimage;
   Outtext ('Files used in input:  '); for I5:= 0,1,2,3,4 do begin
   Setpos (Pos + 5); Outtext (INFILE_TXT10 [I5]); end;
   Setpos (Pos + 5); Outtext (WTHR.DATAFILE); Outimage; Outimage;
   Outtext ('QUANTITIES and COSTS vs. TIME in year:'); Outint (WTHR.YEAR,6);
   Outimage; Outimage;
   Outtext ('Date  Day    QUANTITY processed [ha]  /');
   QNT:= MATRL_PRC [1].QUANT_ARRVD;    !assumed that matrl[1] starts with max.;
   Setpos (43); Outtext ('Quantity expected');
   Outfix (QNT, 0,4); Outtext (' [ha]'); Setpos (80);
   Outtext ('  COSTS of Operations,          Crops,  Total DAY'); Outimage;
   Outtext ('       number'); Setpos (80);
   Outtext ('( Men + Mach. = Gangs = Oper. ) + Mat. =  sum number'); Outimage;
   Outimage; Outimage;
   Setpos(13); for M:= 1,2,3,4,5 do Outtext(' 246802468'); Outtext (' [%]'); Outimage;
   Setpos(21); for M:= 1,2,3,4,5 do begin
       Outint(M * 20,3); Setpos(Pos+7);
   end; Outimage;
   Setpos (13); Outtext(':'); for M:= 1,2,3,4,5 do Outtext ('---------:'); Outimage;
   PLOT:- PLOT_Q.First; if PLOT =/= none then DATE_FROM_DAYNO (PLOT.DAY_NMB);
   Outtext (MNTH_DT6); Setpos (1);
   while PLOT =/= none do inspect PLOT do begin
       if DAY_NMB = Entier (DAY_NMB/10) * 10 or Suc == none then begin
           DATE_FROM_DAYNO (DAY_NMB); Outtext (MNTH_DT6);
       end else Setpos (7);
       Outint (DAY_NMB,6); Setpos (13); Outtext (':');
       if DAY_NMB = Entier (DAY_NMB/10) * 10 then begin
           for M:= 1,2,3,4,5 do Outtext ('---------:')
       end else for M:= 1,2,3,4,5 do Outtext ('         :');
       for M:= 1 step 1 until MATRLS_PROC do begin
           Setpos (13 + QUANT_M [M] * 50.0 / QNT);
           if Pos > 13 and M <= 9 then Outint (M,1) else if Pos > 13 then Outchar(Char(M+55));
       end; Setpos(66); Outtext( DAY_TXT3 [DAY_TP]); Setpos (77);
       for M:= 1 step 1 until 6 do Outfix (COSTS_CUM [M],0,8); Outint (DAY_NMB,6); Outimage;
       PLOT:- PLOT.Suc;
   end; Outimage; Outimage;
   PLOT_Q.Clear;
   for M:= 1 step 1 until MATRLS_PROC do begin
       Setpos (14); if M <= 9 then Outint (M,4) else Outchar ( Char (M+55)); Outtext (' = ');
       Outtext (MATRL [M].NAME_COMP); Outimage;
   end; Eject(1);
end inspect;

Link class PLOT_ADMIN;                            !**************************************;
begin
   integer DAY_NMB, DAY_TP;                       !----------------------;
   real array COSTS_CUM [1:6];                    !----------------------;
   integer array QUANT_M [1:MATRLS_PROC];         !----------------------;
end;
```

Table 5.26 Initial section of class ADMINISTRATION.

```
!    i    n    i    t    i    a    l;

Outimage;
Outtext ('Do you want disk files to record: ...'); Outimage;
Outtext ('- use of components each period? (yes or no= at end of season only):');
Outimage; Lastitem;
EACH_PERIOD:= YES = Intext (3);
Outtext ('  You said:  '); if EACH_PERIOD then Outtext ('yes') else Outtext ('no'); Outimage;
Outtext ('Do you want display output of materials and operations? (yes or no):');
Outimage; Lastitem;
DISPLAY:= Intext (3) = YES;
Outtext ('  You said:  '); if DISPLAY then Outtext ('yes') else Outtext ('no'); Outimage;
REPORT_PERD:- new Printfile ('period.lpt /A:append');
REPORT_PERD.Open (Blanks (132));
PLOT_Q:- new Head;

Passivate;                                   !wait until materials exist;

for M:= 1 step 1 until MATRLS_PROC do STATE_TEXT_M [M]:- Copy (Blanks(48));
for O:= 1 step.1 until OPRTNS do STATE_TEXT_O [O]:- Copy (Blanks(48));
```

Table 5.27 Conversation and output on the terminal.

```
**>Setup of simulation; run at  1986-02-24  09:36:19

Which file contains the names of files defining the experiments?
EXPERI.MNT

**-->Experiment starts
Give code of experiment in <= 14 char., please:
EXAMPLE
        EXPERIMENT  with aggregated DAILY data, 'Simulation Monograph'                          EXAMPLE        1986-02-24  09:36:19

Current value of driver U of random number drivers is:        907  at time:    0.000000

Do you want disk files to record: ...
- use of components each period? (yes or no= at end of season only):no
  You said:  no
Do you want display output of materials and operations? (yes or no):yes
  You said:  yes
Do you want to generate combinations of gangs? (yes or no=read from file):no

A blank line is met as end of list of    13 combinations;    30 were requested (now reduced!).

**-->Setup of experiment completed, simulation starts

Day type at 1 Jan.  1962     1  (1= Monday ,etc.); Data start at  Wed   Aug     1

**--..>Experiment continues for year  1962

                              ---------(output as in Table 5.19)---------

Year  1962  No.     1  Annual Summary of Results.

Cum.costs:  (men + machines = operations)  +  materials -->TOTAL costs
Sun Sep  9  1434       1282         2717            5040          7757

Wheat (winter)..........  Timeliness losses are: Dfl. 3695.400482
Straw swath in field....  Timeliness losses are: Dfl.  745.813995
Bales in field..........  Timeliness losses are: Dfl.  598.768356

Current value of driver U of random number drivers is:    6203608  at time:  500.000000

**-->Experiment completed

**>Simulation completed at1986-02-24  09:37:11
```

machines), 4.79 (generating combinations), 4.109 and 4.110 (transformation of dates) and 5.28 (search of keyword in file). The program execution was not interrupted by causing a runtime error; afterwards SIMDDT can be used to find more about the situation resulting in such a message (Section 7.3.2).

5.2 Set-up of experiments: input

The second part of the experimental frame is the main program that uses files to set up a number of experiments, to create the experiment and all the objects required with their initial values and to simulate a number of seasons per experiment.

The declaration of the main program is shown in Table 5.28 and concerns references to inputfiles, text variables to identify an experiment in terminal and line printer output and a procedure KEYWORD (T) that reads line after line in an inputfile PAR to find text T as the first text on that line (skips blanks and tabs). The dynamic section of the main program is preceded by asking the name of the input file (Table 5.35) with reference EXP-FILES that contains for each experiment the names of the four files with data related to objects and a number (1,...) of files with seasonal weather data and material properties. Table 5.29 shows that this file is used as long as Endfile is false to find :

Table 5.28 Main program; declaration of variables and procedure KEYWORD.

```
begin   !main program;                          !**************************************;

   !  d     e     c     l     a     r     a     t     i     o     n;
   !                                                  !  of simulation;

   integer                                      !----------------------;
   I, C_N;
   ref (Infile)                                 !----------------------;
   EXP_FILES, ENVR_EXP, M_M_SYS, MATRL_DATA, INIT_FIELDS, PAR;
   text array                                   !----------------------;
   INF_TXT [0:4];
   text                                         !----------------------;
   EXP_IDENTIF, EXP_I_TODAY, EXP_I_DAYT, EXP_CODE, EXP_F_TXT;


   procedure KEYWORD (T);                             !----  ----  ----  ----  ----;
   value T; text T;                                   !called in init_exper, dynamic;
   inspect PAR do begin
       text JT;
       JT:- Blanks (T.Length);
       while JT <> T  and  not Endfile do begin
           Inimage; Lastitem;          !keyword expected as first word on a line;
           if Image.Length >= T.Length  then JT:= Intext (T.Length);
       end;
       if Endfile then begin
           Outimage;
           Outtext ('Error : End of file found while scanning for  '); Outtext (T);
           Outtext ('  as first word on a line in an input file.'); Outimage;
       end;
       !       otherwise keyword found;
   end;
```

Table 5.29 Dynamic section of main program; simulation of experiments.

```
Outtext ('**>Setup of simulation; run at  '); Outtext (TODAY); Outtext ('  ');
Outtext (DAYTIME); Outimage;
Outimage;
Outtext ('Which file contains the names of files defining the experiments?'); Outimage;
Inimage; EXP_F_TXT:- Intext (20);
INF_TXT [0]:- EXP_F_TXT.Sub (1, IMIN(12, EXP_F_TXT.Length));
EXP_FILES:- new Infile (EXP_F_TXT);
EXP_FILES.Open (Blanks (132));

! d     y     n     a     m     i     c     of simulation of experiments;

inspect EXP_FILES do
while not Endfile do begin  !file names expected as first name on line (exactly 10 char.);
    PAR:- EXP_FILES;
    KEYWORD ('EXPERIMENT');
    Outimage;
    Outtext ('**-->Experiment starts'); Outimage;
    Setpos (1); EXP_IDENTIF:= Intext (90);
    Outtext ('Give code of experiment in <= 14 char., please:'); Outimage;
    Sysin.Inimage;
    EXP_CODE:= Sysin.Intext (20);
    EXP_I_TODAY:= TODAY; EXP_I_DAYT:= DAYTIME;
    Outtext (EXP_IDENTIF); Outimage;
    Inimage; Lastitem; INF_TXT [1]:- Intext (12);
    ENVR_EXP:- new Infile (INF_TXT [1]);              !environment of experiment;
    Inimage; Lastitem; INF_TXT [2]:- Intext (12);
    M_M_SYS:- new Infile (INF_TXT [2]);               !man _machine system;
    Inimage; Lastitem; INF_TXT [3]:- Intext (12);
    MATRL_DATA:- new Infile (INF_TXT [3]);            !materials;
    Inimage; Lastitem; INF_TXT [4]:- Intext (12);
    INIT_FIELDS:- new Infile (INF_TXT [4]);           !initial fields at start of season;
```

(i) the keyword 'EXPERIMENT' (capital letters!); this line from Table 5.35 is used to identify the experiment, along with a date, clocktime and a code of about 14 characters (Tables 5.23 and 5.27);

(ii) four names of files (in exactly 12 characters giving a filename and the extension; after a name with less than 12 significant characters blanks (no tabs) are used) referenced by:

- ENVR-EXP the environment of the experiment;
- M-M-SYS the men, machines, gangs, combinations and operations;
- MATRL-DATA the materials;
- INIT-FIELDS the fields of the initial materials.

The description in the following sections concerns (1) the creation of objects, (2) the initialization of the base model and the objects, (3) the use of several seasons and (4) the restraints on input data.

5.2.1 *Creation of objects*

The very first object created is the experiment itself. Table 5.30 shows that SFOEXPERIMENT and its fourteen parameters are prefixes to a block of statements beginning with 'begin'. The prefix is the external class SFOEXPERIMENT (Table 5.1); SFOBASE-MODEL and SIMULATION are prefixes to class SFOEXPERIMENT. The parameters are read from file

Table 5.30 Creation of object of class SFOEXPERIMENT with actual parameters.

```
inspect ENVR_EXP do begin
   external class SFOEXPERIMENT;                        !used as prefix for block level 4;

   integer array
   INPT [1:14];
   integer I10;

   Open (Blanks (132));
   PAR:- ENVR_EXP;
   KEYWORD ('GENERAL DATA');
   KEYWORD ('PARAMETERS EXP.');
   SCANTO (Image, ':');
   for I10:= 1 step 1 until 14 do INPT [I10]:= Inint;

   SFOEXPERIMENT                       !*************************************;
   (INPT [1],           !types of man & machines;
   INPT [2],            !total number of men (usually belonging to one type);
   INPT [3],            !total number of machines (of all types );
   INPT [4],            !number of sets for combinations and gangs;
   INPT [5],            !number of gangs;
   INPT [6],            !number of combinations consisting of two or more gangs;
   INPT [7],            !number of operations to process materials;
   INPT [8],            !number of operations to service or repair a machine;
   INPT [9],            !number of material initially available fields;
   INPT [10],           !number of materials processed;
   INPT [11],           !number of materials;
   INPT [12],           !types of shifts defining availability in a period;
   INPT [13],           !types of shifts defining presence in a week;
   INPT [14])           !types of shift defining moments of calculation of urgency of material;

   begin
      !d    e    c    l    a    r    a    t    i    o    n;
      !                                         !  of experiment;
```

ENVR-EXP. Because of prefix SIMULATION this block of the main program is automatically a process itself and referred to by the reference variable 'MAIN'; so use can be made of, for instance, 'Hold(.)' in this block or 'Activate MAIN' elsewhere. This block consists of the declaration section (procedure creating objects), the initial statements and the dynamic section controlling the seasons with weather data involved in an experiment (Section 5.2.3.).

The creation of general objects (by: 'reference':- new 'class-name') concerns object(s) of class (Table 5.31) :

- SHIFT-PERD to control the availability of men, machines and materials over periods;
- SHIFT-WEEK to control the availability of men during a week;
- SHIFT-URG to control the urgency calculations of materials;
- WEATHER to control the input of weather data and related material properties;
- URGENCY-DCSN to control the decision making; and
- ADMINISTRATION to control the output.

Each object is activated just after its creation to allow the initialization (Section 5.2.2). Such an immediate activation of an object can be achieved by, for instance:

- Activate X;

Table 5.31 Creation of general objects of an experiment; part of procedure INIT-EXPER.

```
procedure INIT_EXPER;                                   !----  ----  ----  ----  ----;
begin                                                   !called in this dynamic;
   !                                                    !initialisation of system for an experiment;
   PAR:- PARAMETERS:- ENVR_EXP;
   inspect ENVR_EXP do begin
       KEYWORD ('SHIFT PERIOD');
       for I:= 1 step 1 until SHS_PERD do begin
           FIND_DATA_AT ('PARAMETERS PRD.');
           SH_PERD [I]:- new SHIFT_PERD (Inint);        !number of periods;
           activate SH_PERD [I];
       end;
       KEYWORD ('SHIFT WEEK');
       for I:= 1 step 1 until SHS_WKLY do begin
           FIND_DATA_AT ('PARAMETERS WK.');
           SH_WKLY [I]:- new SHIFT_WEEK (Inint,Inint,Inint,Inint);
           !                    !daytypes, cost categories, shifts, periods;
           activate SH_WKLY [I];
       end;
       SH_WK_MAN:- SH_WKLY [1];
       KEYWORD ('SHIFT URGENCY');
       for I:= 1 step 1 until SHS_URG do begin
           FIND_DATA_AT ('PARAMETERS URG.');
           SH_URG [I]:- new SHIFT_URG (Inint); !number of calculation points;
           activate SH_URG [I];
       end;
       WTH_MAT:- WTHR:- new WEATHER;
       activate WTHR;
       DECIDE:- DECIDE_URG:- new URGENCY_DCSN;
       activate DECIDE;
       ADMN:- ADMNSTR:-
       new ADMINISTRATION ('Administration',
       SH_PERD [if SHS_PERD > 0 then 1 else 0], SH_WKLY [0]);
       activate ADMNSTR;                                !not counted in c-n as a component;
       Close;
   end ** of inspect environment;
```

– Activate X at Time prior;.

The input needed as actual parameters of an object and the input needed to initialize an object are read subsequently from the file referenced by ENVR-EXP. The parameters are requested by the class and all its prefix classes in the order of their definition: superclass parameters before class parameters. Keywords are used to control appropriate input lines in the file. The requested keyword is looked for in the file until a match or the end of file is found.

The creation of objects related to the man-machine subsystem (file M-M-SYS) concerns objects of class (Table 5.32):

- UPDT-MAN-MCH to update the state of gangs and combinations according to the availability of men and machines;
- MM-SYSTMS-SET a set to contain gangs and combinations from which only one can be selected in decision;
- LABOUR the men;
- EQUIPMENT the machines, tractors, trailers, tools, etc.;
- MAN-MACH-SET the gangs;
- GANG-SET the combinations;
- OPRTN-MATRL the operations processing materials; and
- SRVC-REPR the operations servicing and repairing machines.

Table 5.32 Creation of objects of the man-machine subsystem; part of procedure INIT-EXPER.

```
PAR:- PARAMETERS:- M_M_SYS;
inspect M_M_SYS do begin
    Open (Blanks (132));
    UPDT_MM_SYS:- new  UPDT_MAN_MCH;
    activate UPDT_MM_SYS;
    KEYWORD('SETS OF MAN-MACHN.SYSTEMS');
    FIND_DATA_AT ('PAR. LIST of SETS'); Lastitem;
    for I:= 1 step 1 until MM_S_SETS do begin
        if Image.Strip <> notext then begin
            MM_S_SET [I]:-
            new MM_SYSTMS_SET (Intext (24), SH_PERD [Inint], SH_WKLY [Inint]);
            activate MM_S_SET [I];
            Inimage;
        end
        else begin
            Outimage;
            Outtext ('A blank line is met as end of list of'); Outint (I-1,6);
            Outtext (' sets;'); Outint (MM_S_SETS,6);
            Outtext (' were requested (now reduced!).'); Outimage;
            MM_S_SETS:= I - 1;
        end;
    end;
    KEYWORD ('LABOUR');
    FIND_DATA_AT ('PAR. LIST of MEN'); Lastitem;
    for I:= 1 step 1 until MANN do begin
        if Image.Strip <> notext then begin
            COMPNT [I]:- MAN [I]:-
            new LABOUR (Intext(24),SH_PERD [Inint],SH_WKLY [Inint],Inint);
            !       !name,shift period.type, shift week type, category number of man;
            activate MAN [I];                   !execute initial section;
            Inimage;
        end
        else begin
            Outimage;
            Outtext ('A blank line is met as end of list of'); Outint (I-1,6);
            Outtext (' men;'); Outint (MANN,6);
            Outtext (' were requested (now reduced!).'); Outimage;
            MANN:= I - 1;
        end;
    end;
    C_N:= MANN;
    KEYWORD ('EQUIPMENT');
    FIND_DATA_AT ('PAR./DATA LIST of MACHN'); Lastitem;
    for I:= 1 step 1 until MACHNS do begin
        if Image.Strip <> notext then begin
            COMPNT [C_N + I]:- MACH [I]:-
            new EQUIPMENT (Intext(24), SH_PERD [Inint], SH_WKLY [Inint], Inint);
            !                                                  !category of machine;
            activate MACH [I];                !execute initial section (input, defaults);
            Inimage;
        end
        else begin
            Outimage;
            Outtext ('A blank line is met as end of list of'); Outint (I-1,6);
            Outtext (' machines;'); Outint (MACHNS,6);
            Outtext (' were requested (now reduced!).'); Outimage;
            MACHNS:= I - 1;
        end;
    end;
    C_N:= C_N + MACHNS;
```

Table 5.32 (continued)

```
KEYWORD ('GANGS');
FIND_DATA_AT ('PAR./DATA LIST of GANGS'); Lastitem;
for I:= 1 step 1 until GANGS do begin
    if Image.Strip <> notext then begin
        COMPNT [C_N + I]:- MM_S [I]:- GANG [I]:-       !set of mm_sys:  v;
        new MAN_MACH_SET (Intext(24), SH_PERD [Inint], SH_WKLY [Inint], Inint, 1);
        !                                               !number of gangs     =1;
        GANG [I].RORD_GNG [1]:= GANG [I].MM_S_SQN:= I;
        activate GANG [I];
        Inimage;
    end
    else begin
        Outimage;
        Outtext ('A blank line is met as end of list of'); Outint (I-1,6);
        Outtext (' gangs;'); Outint (GANGS,6);
        Outtext (' were requested (now reduced!).'); Outimage;
        GANGS:= I - 1;
    end;
end;
C_N:= C_N + GANGS;
Outtext ('Do you want to generate combinations of gangs?');
Outtext (' (yes or no=read from file):');
Outimage; Sysin.Lastitem;
if YES = Sysin.Intext (3) then begin
    Outimage;
    Outint (COMBS_G,6); Outtext (' *3 references are reserved!'); Outimage;
    Outtext (' If this is insufficient then revise input value of number');
    Outtext (' of combinations.'); Outimage;
    COMBS_G_GENERATION;
    for I:= 1 step 1 until  COMBS_G do COMPNT [C_N + I]:- COMB_G [I];
end
else begin
    KEYWORD ('COMBINATIONS_G');
    FIND_DATA_AT ('PAR./DATA LIST of COMB_G'); Lastitem;
    for I:= 1 step 1 until COMBS_G do begin
        if Image.Strip <> notext then begin
            COMPNT [C_N + I]:- MM_S [GANGS + I]:- COMB_G [I]:-
            new GANG_SET (Intext(24), SH_PERD[Inint], SH_WKLY[Inint], Inint, Inint);
            !           !name,shift,shift, set of mm_sys, number of gangs in comb.;
            activate COMB_G [I];
            Inimage;
        end
        else begin
            Outimage;
            Outtext ('A blank line is met as end of list of'); Outint (I-1,6);
            Outtext (' combinations;'); Outint (COMBS_G,6);
            Outtext (' were requested (now reduced!).'); Outimage;
            COMBS_G:= I - 1;
        end;
    end;
end;
C_N:= C_N + COMBS_G;
MN_MCH_SYSTMS:= GANGS + COMBS_G;                  !updated to prevent ref == none;
```

Just after creating an object each object is activated to read from the same file M-M-SYS the initial data. Keywords are used in the file and the program to mark the start of specific data; blank lines are used to mark the end of a list of data of objects ('Image.Strip = notext' is true for a blank line). This allows use of more or less input than requested; this feature is useful in use of the same file in another environment.

The creation of objects belonging to the biological subsystem (file MATRL-DATA) concerns objects of class (Table 5.33):

Table 5.32 (continued, 2nd)

```
    KEYWORD ('OPERATIONS MATERIAL');
    FIND_DATA_AT ('DATA LIST of OPR_MAT'); Lastitem;
    for I:= 1 step 1 until OPRTNS_MT do begin
        if Image.Strip <> notext then begin
            COMPNT [C_N + I]:- OPRTN [I]:- OPR_MAT [I]:-
            new OPRTN_MATRL ('........................', SH_PERD [0], SH_WKLY [0]);
            !                    !default values;
            activate OPR_MAT [I];
            Inimage;
        end
        else begin
            Outimage;
            Outtext ('A blank line is met as end of list of'); Outint (I-1,6);
            Outtext (' operations mat.;'); Outint (OPRTNS_MT,6);
            Outtext (' were requested (now reduced!).'); Outimage;
            OPRTNS_MT:= I - 1;
        end;
    end;
    C_N:= C_N + OPRTNS_MT;
    if OPRTNS_S_R > 0 then begin
        KEYWORD ('OPERATIONS SERVICE_REPAIR');
        FIND_DATA_AT ('DATA LIST of OPR_S_R'); Lastitem;
    end;
    for I:= 1 step 1 until OPRTNS_S_R do begin
        if Image.Strip <> notext then begin
            COMPNT [C_N + I]:- OPRTN [I + OPRTNS_MT]:- OPR_S_R [I]:-
            new SRVC_REPR ('........................', SH_PERD [0], SH_WKLY [0]);
            !                    !default values;
            activate OPR_S_R [I];
            Inimage;
        end
        else begin
            Outimage;
            Outtext ('A blank line is met as end of list of'); Outint (I-1,6);
            Outtext (' operatinns serv.&repair;'); Outint (OPRTNS_S_R,6);
            Outtext (' were requested (now reduced!).'); Outimage;
            OPRTNS_S_R:= I - 1;
        end;
    end;
    C_N:= C_N + OPRTNS_S_R;
    OPRTNS:= OPRTNS_MT + OPRTNS_S_R;
    Close;
end ** of inspect man machine systems;
```

- INITIAL-MAT the materials that start each season with fields;
- INTERMDT-MAT the materials that are delivered in the scheduling system and are processed;
- FINAL-MAT the materials that are delivered but not processed.

These objects (and also most objects related to the man-machine subsystem) are referenced by COMPNT[.]; this will be used to initialize the scheduling system each season appropriately (Section 5.2.3).

Table 5.33 Creation of objects of the biological subsystem; part of procedure INIT-EXPER.

```
PAR:- PARAMETERS:- MATRL_DATA;
inspect MATRL_DATA do begin
    Open (Blanks (132));
    KEYWORD ('INITIAL MATERIALS');
    for I:= 1 step 1 until MATRLS_INIT do begin
        FIND_DATA_AT ('PARAMETERS MAT.'); Lastitem;
        COMPNT [C_N + I]:- MATRL [I]:- MATRL_PRC [I]:-
        new INITIAL_MAT (
        Intext (24),            !name;
        SH_PERD [Inint],        !shift changes availability over periods;
        SH_WKLY [Inint],        !shift changes presence in a week, sh_wkly [0] ** none;
        I,                      !sequence number of material (not given in input);
        SH_URG [Inint],         !at shift time urgency calculation is performed;
        Inint,                  !number of processing conditions for operations;
        Inint,                  !number of points given of the timeliness function;
        Inint,                  !number of timeliness functions;
        Inint, Inint);          !number of days, conditions of workable time expectations;
        activate MATRL [I];             !execute initial section (input, defaults);
    end;
    KEYWORD ('INTERMEDIATE MATERIALS');
    for I:= MATRLS_INIT + 1 step 1 until MATRLS_PROC do begin
        FIND_DATA_AT ('PARAMETERS MAT.'); Lastitem;
        COMPNT [C_N + I]:- MATRL [I]:- MATRL_PRC [I]:-
        new INTERMDT_MAT (Intext (24), SH_PERD [Inint], SH_WKLY [Inint], I,
        SH_URG [Inint], Inint, Inint, Inint, Inint, Inint);
        activate MATRL [I];
    end;
    for I:= MATRLS_PROC + 1 step 1 until MATRLS do begin
        COMPNT [C_N + I]:- MATRL [I]:-
        new FINAL_MAT ('........................', SH_PERD [0], SH_WKLY [0], I);
        activate MATRL [I];
    end;
    C_N:= C_N + MATRLS;
    Close;
end ** of inspect materials data;
```

5.2.2 *Initialization of objects and input data*

To create and initialize objects three inputfiles related to general objects, objects of the man-machine subsystem and of the biological subsystem, respectively were used. Keywords are used to find the parameters for a specific class. To achieve a comparable security for the initial data of each object keywords are again used; procedure FIND-DATA-AT (Table 5.34) is used to find the keyword at the beginning of a line and to skip to ':'. It is now possible to use as much text (table headings) as useful before a keyword; after ':' the related set of data are available. For each class the program that initialized an object and the related input example are presented in tables.

Table 5.35 shows the contents of a file referenced by EXP-FILES; it contains the keywords 'EXPERIMENT' and 'WEATHER DATA FILES', the names of four files as read in the program shown in Table 5.29 and one file with weather data (Table 5.5). After the keywords and after the file names (of 12 characters), comment is possible on the same line.

The first object created was the block referenced by MAIN (Section 5.2.1 and Table 5.30) and the necessary parameters are read from file ENVR-

Table 5.34 Procedure FIND-DATA-AT of class SFOBASE-MODEL.

```
procedure FIND_DATA_AT (FRONTTEXT);          !----  ----  ----  ----  ----;
value FRONTTEXT;
text FRONTTEXT;
inspect PARAMETERS do begin         !called in initial sections to find appropriate data;
   while not Endfile and not FRONTCOMPARE (Image, FRONTTEXT) do Inimage;
   if not Endfile then SCANTO (Image, ':')
   else begin
      Outimage;
      Outtext ('Error: End of file found while scanning for ''');
      Outtext (FRONTTEXT); Outtext ('' at begin of line in an input file.'); Outimage;
   end;
end;
```

Table 5.35 Input of the files for an experiment and the files of weather data.

```
       EXPERIMENT  with aggregated DAILY data, 'Simulation Monograph'

ENVRMT.XPL       7 perds starting at 5 Aug., 4 shifts at 7.00, 17.00, 22.00 and 24.00h
MMSSSF.XPL      SSF = setup, service, failure
MATWWT.XPL      WWT = m.c.grain (= 23%
WHTFLD.XPL      4 wheat fields ripe at 5 Aug. 24.00h; processable at 5 Aug. 24.00h


       WEATHER DATA FILES        (each in first 12 positions of a line)

GDLR62.XPL                       (char. 5 and 6 refer to year 1962)
```

EXP. This file contains all the information needed to declare arrays of references (Table 5.36) and to create and initialize the general objects. The fourteen actual parameters of SFOBASE-MODEL concern eleven data elements needed for:

- the categories of men and machines distinguished LE-SQNS; each category may contain more than one item and each item can replace any other item of the same category;
- the number of men , MANN, each man referenced by MAN [.];
- the number of machines, MACHNS, each machine referenced by MACH [.];
- the number of sets of man-machine systems distinguished, MM-S-SETS; those sets are handled independently to select one man-machine system or none from each set; each set referenced by MM-S-SET [.];
- the number of gangs, GANGS, each gang referenced by GANG [.] and MM-S [.];
- the number of combinations, COMBS-G, each combination (set of two or more gangs) referenced by COMB-G [.] and MM-S [.]; the maximum number of references is increased three times to allow the generation of an unknown number of combinations from the existing gangs and the available men and machines; the number of references is adjusted afterwards;
- the number of operations for processing materials, OPRTNS-MT, each operation referenced by OPR-MAT [.] and OPRTN [.];
- the number of operations to service or repair machines, OPRTNS-S-R,

Table 5.36 Input related to the parameters of the experiment.

GENERAL DATA of experiment

	Number of Categories, of men, machines	Men	Machines	Number of Sets of man-mach systems	Gangs	Combinations	Number of operations for Material process.	Service -repair	Number of materials Initial	+interm.= Processed	+final =Total	Number of shifts to control	availability in Periods	calculate Urgency Week
PARAMETERS EXP.:	10	2	11	2	11	30	9	2	1	6	9	1	1	1

each operation referenced by OPR-S-R [.] and OPRTN [.];
- the number of materials initially available with some fields, MATRLS-INIT, each initial material referenced by MATRL [.] and MATRL-PRC [.] (belonging to the materials processed);
- the number of materials processed, MATRLS-PROC, each material referenced by MATRL [.] and MATRL-PRC [.];
- the number of materials, MATRLS; each material is referenced by MATRL [.] and is a processed material (initial material or intermediate material) or a final material.

and three data elements for:
- the number of shifts controlling the availability over periods, SHS-PERD, each shift referenced by SH-PERD [.];
- the number of shifts controlling the availability within a week, SHS-WKLY, each shift referenced by SH-WKLY [.];
- the number of shifts controlling the calculation of urgency of processed materials, SHS-URG; each shift referenced by SH-URG [.].

Table 5.37 shows the initialization of an object of class SHIFT-PERD; it includes the creation of a queue referred by COMP-Q to contain components behaving according to this shift of periods. After the data are read as in Table 5.39 a default value is assigned to the end of a last period of a year (defined in the main program, for instance, at day 500); the current period PERD is set to zero and the process is passivated until also other objects exist and the execution of the dynamic part (Table 4.103) is meaningful. The input data of Table 5.39 describe that there are eight periods considered; the first ends on 5 August and '-' means that each component contained in COMP-Q is not available; the following six periods are weekly periods when the components are available (the date is preceded by '+'); the last period ends at 31 Dec. The availability of components can be used for each subclass of class COMPONENT (i.e. for each process) such as LABOUR, EQUIPMENT or MATERIAL.

The initialization of an object of class SHIFT-WEEK is shown in Table 5.38 and the input in Table 5.39. The data start with the actual parameters

Table 5.37 Initial section of class SHIFT-PERD.

```
!       i       n       i       t       i       a       l;
PLUS:- Copy ('+');
COMP_Q:- new Head;
inspect PARAMETERS do begin
    FIND_DATA_AT ('DATA PRD.');
    for PERD:= 1 step 1 until PERDS do begin
        Lastitem;
        AVLB_PERD [PERD]:= Intext (1) = PLUS;
        PERD_END [PERD]:= DATE_TO_DAYNO (Inint, Inint);         !last date period is valid;
    end;                                     !month, calender date;
end;                                         !dayno is relative to jan. 1;
PERD_END [PERDS + 1]:= LAST_DAY_YEAR;
PERD:= 0;
Passivate;                                   !wait until decide and components exist;
```

Table 5.38 Initial section of class SHIFT-WEEK.

```
I    i    n    i    t    i    a    l;

inspect PARAMETERS do begin
    FIND_DATA_AT ('DATA WK.CTG.');
    for DAY_TYPE:= 1 step 1 until 7 do DAY_CATAGORY [DAY_TYPE]:= Inint;
    FIND_DATA_AT ('DATA WK.COSTS');
    for I1:= 1 step 1 until COST_CTGRS do COST_H_CTGR [I1]:= Inreal;    !   [0] = 0.0;
    for PERIOD:= 1 step 1 until PERIODS do begin
        FIND_DATA_AT ('DATA WK.SHIFTS');
        PERIOD_END [PERIOD]:= DATE_TO_DAYNO (Inint, Inint);
        for SHIFT:= 1 step 1 until SHIFTS do begin
            SHIFT_END [SHIFT, PERIOD]:= Inreal;
        end;
        FIND_DATA_AT ('DATA WK.4');
        for DAY_TYPE:= 1 step 1 until DAY_CTGRS do
        for SHIFT:= 1 step 1 until SHIFTS do COSTS_SH_CTG [DAY_TYPE,SHIFT, PERIOD]:= Inint;
    end;
    if this SHIFT_WEEK == SH_WKLY [1] then begin
        FIND_DATA_AT ('DATA WK.5');
        DAY_BGN:= Inreal;
        DAY_END:= Inreal;
    end;
end of inspect parameters;

PERIOD_END [PERIODS + 1]:= LAST_DAY_YEAR;
CMP_Q:- new Head;
SHIFT:= SHIFTS; PERIOD:= 1;
COSTS_NOW:= COST_H_CTGR [0];
Passivate;               !wait until day_tp_1jan is initialized by weather's dynamic;
```

and tells that three categories of days are distinguished (workdays, Saturday, Sunday), three categories of costs are considered (one for regular time and two for overtime), a day is built up of four shifts (from 00:00 – 07:00 07:00 – 17:00, 17:00 – 22:00 and 22:00 – 24:00) and only one pattern is handled. The category of day (1, 2 or 3) is assigned to Monday – Sunday; the costs per hour are assigned for each of the three categories (0.00 f/h for regular time and 15.00 or 20.00 for overtime; these costs influence the urgency of a gang and may prohibit the use of the gang). The next data form the pattern valid until 15 Sept. and concern the clocktime at which the four shifts end (07:00, 17:00, 22:00 and 24:00). For each category of days, there is a line denoting the category of costs valid during a shift: Category '0' means no work; '1' regular time; '2' overtime of 15.00 f/h; '3' overtime of 20.00 f/h, but it is not used. The first line describes that on Monday – Friday (belonging to Category 1) the pattern is no work from 00:00 to 07:00 and from 22:00 to 24:00, regular time from 07:00 to 17:00 and overtime from 17:00 to 22:00. The example in Table 5.39 shows only one period that ends on 15 Sept., but several periods with weekly patterns are possible, for instance, one from 1 Jan. to 30 June and another afterwards with different moments of worktime, pauses or different costs. After 15 Sept. men are not available in this case. DAY-BGN and DAY-END are read only in the first object because Table 5.38 reads these data only for SH-WKLY [1] (their use is shown in Tables 4.42 and 4.111); they are assumed to be valid for each pattern of shifts considered.

Table 5.39 Input related to an object of class SHIFT-PERD and to an object of class SHIFT-WEEK.

```
        SHIFT PERIOD
        ************


              Number of periods       (first period starts at Jan.1,incl.; 0.00h is the end of period 0 )
                                      (period after last one ends at end of year)
PARAMETERS PRD.:       8
              ------------------------------------------------------------------------------------------

              End of period:  1      2      3      4      5      6      7      8      9      10 (+ n*10)
                                                          (Available: + else - / Month / Date of end of a period)
DATA PRD.:                   -08 05 +08 12 +08 19 +08 26 +09 02 +09 09 +09 15 -12 31

****************************************************************************************************************

        SHIFT WEEK
        **********


              Number of       Number of       Number of
              categories of   Shifts per day  Periods with
              Days   Costs                    different pattern of shifts
PARAMETERS WK.: 3      3              4             1
              ------------------------------------------------------------------------------------------

              Category of day to which        Monday Tuesday Wednesd.Thursd. Friday Saturd. Sunday belongs
DATA WK.CTG.:                                 1      1      1      1      1      2      3
                                              ------------------------------------------------------------

              Costs per hour in category      1      2      3      4      5      6      7      8      9      10 (+n*10)
                          (of costs)          regular overtime ........
                                              time   [Dfl/h]
DATA WK.COSTS:                                0.00   15.00  20.00
                                              ------------------------------------------------------------
```

Table 5.39 (continued)

```
                End.of period   End of shift:   1       2       3       4       5       6       7       8       9       10 (+n*10)
                Month   Date                    at clocktime

DATA WK.SHIFTS: 09      15                      7.00    17.00   22.00   24.00
                                                -------------------------------------------------------------------------------

                                                Category of costs during a shift        (1 - n lines per category of day)
                                                                                        (0 means not available during shift)

DATA WK.4:                                      0       1       2       0
                                                0       2       2       0
                                                0       0       0       0
                                                -------------------------------------------------------------------------------

                Clocktime of worktime on workdays
                Begin   End     (only given in first shift of week)

DATA WK.5:      7.00    22.00

***********************************************************************************************************************************
```

Table 5.41 Input related to an object of class SHIFT-URG.

```
        SHIFT URGENCY
        *************

                Number of points        (last point defines length of cycle in hours e.g. 168 means a cycle of one week)
                                        (point 0 is at 0.00h)

PARAMETERS URG.:        6
                -----------------------------------------------------------------------------------------------------------

                Clocktime at point      1       2       3       4       5       6       7       8       9       10 (+n*10)

DATA URG.:                              6.00    12.00   16.00   18.00   20.00   24.00

***********************************************************************************************************************************
```

The object controlling the urgency calculations of materials originates from class SHIFT-URG. The initialization of such an object is shown in Table 5.40; a queue for materials is created, MAT-Q, six moments are read (Table 5.41) to calculate the urgency (06:00, 12:00, ... , 24:00) and the execution is interrupted by calling Passivate. At each moment assigned, a calculation of urgency takes place for all the materials contained in MAT-Q. In general, urgency depends on the amount of material and thus changes during processing and delivery; a recalculation of the urgency during a day so influences the processing. The objects of the classes WEATHER, URGENCY-DCSN and ADMINISTRATION do not require input data; the parameters are given default values in the program (Table 5.31). The initialization of the first two objects concerns hardly more than calling Passivate and the initialization of the ADMINISTRATION object, Table 5.26, includes (i) a terminal conversation about the wanted output (Table 5.27), (ii) a call to Passivate and (iii) initial values to text variables.

The initial data of objects related to the man-machine subsystem (Table 5.32) are read from a file referenced by M-M-SYS; it starts with the object of class UPDT-MAN-MCH which only calls Passivate. The initialization of objects of class MM-SYSTMS-SET includes only the creation of a queue (Table 4.77) to contain man-machine systems. These objects pass their final 'end' and act as a terminated process that cannot be activated any more; it is a data block. The input shown in Table 5.42 shows the parameters: name of component and references to the shifts of periods and within a week. No reference is made to shifts within a week (parameter '0' refers to SH-WKLY[0] == none), for such a weekly pattern will be assigned to men (with effect on most man-machine systems) and is not wanted for the grain drier. With the initialization of the prefixed superclass COMPONENT (Table 4.4), this object is placed in a queue, COMP-Q, in SH-PRD (a reference to SH-PERD[1], an object of SHIFT-PERD). The remaining text in Table 5.42 concerns the keywords and some comment (headings before the input lines and reference indices ...[.] on lines after the input data of an object). The blank line after the list is used to detect the end of the list if the number of requested sets (Table 5.36, Parameter 4) were more than two. The same technique is used for the following lists of objects.

Table 5.40 Initial section of class SHIFT-URG.

```
!     i     n     i     t     i     a     l;

MAT_Q:- new Head;
inspect PARAMETERS do begin
    FIND_DATA_AT ('DATA URG.');
    for POINT_NO:= 1 step 1 until POINTS do URG_CALC_HR [POINT_NO]:= Inreal;
end;
POINT_NO:= 0;
Passivate;          !wait until materials are known in operations, needed in urg_gangs;
!                   !activated in shift_change_ of material;
```

Table 5.42 Input related to objects of class MM-SYSTMS-SET.

```
       SETS OF MAN-MACHN.SYSTEMS
       ************************

              Parameters                                                     (reference type
              Name            Type of shift used to                          MM_S_SET [.] )
              (24char)        control availability in
                              Periods Week    (if week then also periodl)

PAR. LIST of SETS:
Set of man-machn.sys...1      1       0                                             [1]
Set of man-machn.sys...2      1       0                                             [2]

****************************************************************************************************************************
```

Table 5.43 Input related to objects of class LABOUR.

```
       LABOUR
       ******

              Parameters
              Name            Type of shift used tu  Category of     (reference type
              (24char)        control availability in man            MAN [.]         )
                              Periods Week

PAR. LIST of MEN:
Man 1...................      1       1              1               [1]
Man 2...................      1       1              1               [2]

****************************************************************************************************************************
```

The input for objects of class LABOUR is shown in Table 5.43 and concerns only parameters; the initialization is shown in Table 4.56. The first parameter is the name of the object; two parameters are used to control the availability of this object of class LABOUR over periods and within a week; the parameter LE-NO-LE is a category of labour and equipment. Objects of the same category are considered identical and queued in LE-STATE-Q[i,j] of category i and the prevailing STATE j. So both men belong to category 1 and are identical. The men are referenced as MAN [1] and MAN [2]; [1] and [2] is only additional information and not input data.

Table 4.4 shows the initialization of all objects of classes with prefix COMPONENT and involves making shorter names (7 and 12 characters long instead of 24), positioning the object in appropriate queues of shifts over periods and per week and initial default values of the category of costs and the state.

The initialization of EQUIPMENT objects is shown in Table 5.44 and concerns mainly input as shown in Table 5.45. After the name, the two types of shift and the category of man or machines there are: fixed costs per hour; a storage capacity for trailers and driers; lower and upper limits of a duration without failures; a minimum duration needed to justify a service, lower and upper limits of a repair duration and a service duration, respectively and the sequence number of the service-repair operation involved

Table 5.44 Procedure RESET- and initial section of class EQUIPMENT.

```
procedure RESET_;                                 !----  ----  ----  ----  ----;
begin                                             !called in reset, init.;
   RPR_ND:= true;                                 !to force first calculation of flrfr_dur;
   SRVC_RPR_DON;                                  !also sets state_next to passive;
   SERV_OBS:= REPR_OBS:= 0;
   SERT_VAR:= SERT_MEAN:= REPT_VAR:= REPT_MEAN:= 0.0;
   REPAIRTIME:= SERVICETIME:= 0.0;
end;

!    i    n    i    t    i    a    l;

procedure INIT_INPUT_;                            !----  ----  ----  ----  ----;
inspect PARAMETERS do begin
   COSTS_H_FXD:= Inreal;
   STORECAP_E:= Inreal;
   FLRFR_LB:= Inreal; FLRFR_UB:= Inreal;
   SRVCFR_RUNT:= Inreal / 24.0;
   RPR_LB:= Inreal; RPR_UB:= Inreal;
   SRVC_DUR_LB:= Inreal; SRVC_DUR_UB:= Inreal;
   S_R_OPR_NO:= Inint;
   if FLRFR_LB < 1.0&-4 or FLRFR_UB < 1.0&-4 then FLRFR_LB:= FLRFR_UB:= 1.0&10;
   if SRVCFR_RUNT < 1.0&-4 then SRVCFR_RUNT:= 1.0&10;
   U_FL_RP:= U:= Mod (3125 * U, 8388593);
   U_SRVC:= U:= Mod (3125 * U, 8388593);
   !see a.thesen computer methods in o.r. p.196, random seeds for random number generators;
   SRVC_TRMNTD:= 0.0;
end of inspect parameters;

INIT_INPUT_;
Passivate;                                        !wait until decide is created;
RESET_;

inner;                                            !initialisation of subclasses, if any;
```

Table 5.45 Input related to objects of class EQUIPMENT.

EQUIPMENT
****

Parameters Name (24char)	Type of shift Periods	availability in Week	Data Category of man, machine	Costs [Dfl/h]	Storage capacity [ha]	Runtime duration free of: Failures lower bound [h]	upper bound [h]	Service minimum [h]	Uniform distribution of Repair duration lower bound [h]	upper bound [h]	Service dur. lower bound [h]	upper bound [h]	Type of service repair operation	(reference type MACH [.])
.	.	.	.	.	.			.						
PAR./DATA LIST of MACHN.:														
Tractor 1...............	0	0	2	.0	.0	.0	.0	.0	.0	.0	.0	.0	0	[1]
Tractor 2...............	0	0	2	.0	.0	.0	.0	.0	.0	.0	.0	.0	0	[2]
Combine harvester 4.5m..	0	0	3	.0	.0	0.0	0.0	8.0	1.0	1.0	0.6	0.6	1	[3]
Baler...................	0	0	4	.0	.0	.0	.0	.0	.0	.0	.0	.0	0	[4]
Bale (un-)loader........	0	0	5	.0	.0	5.0	15.0	.0	0.25	0.25	.0	.0	1	[5]
Grain trailer 1, 1ha....	0	0	6	.0	1.0	.0	.0	.0	.0	.0	.0	.0	0	[6]
Grain trailer 2, 1ha....	0	0	6	.0	1.0	.0	.0	.0	.0	.0	.0	.0	0	[7]
Bale trailers, 4*0.5ha..	0	0	7	.0	2.0	.0	.0	.0	.0	.0	.0	.0	0	[8]
Plough 1, 1.5m..........	0	0	8	.0	.0	.0	.0	.0	.0	.0	.0	.0	0	[9]
Plough 2, 1.0m..........	0	0	9	.0	.0	.0	.0	.0	.0	.0	.0	.0	0	[10]
Grain dryer, 2ha........	0	0	10	6.0	2.0	.0	.0	.0	.0	.0	.0	.0	0	[11]

****

(zero refers to a not existing object). Initialization includes the change of zero values of the failure-free and service-free intervals to infinity (equivalent to no failures and no need for service) and the creation of random seeds for random number generators used to find a failure moment and a repair duration, U-FL-RP, or a service duration, U-SRVC. RESET- is called to make the first failure free duration and initial values for the number of observations during a season, the mean and the variation of the observations. In the example of Table 5.45 it can be seen that shifts are not used (parameter '0'); it suffices that men are controlled by shifts over periods and within a week; machines are assumed to be available all the time. The two tractors belong to the same category (2) and two grain trailers belong to category 6. When trailers are always used together, then it is sufficient to create one object, as in category 7, that represents, for instance, four trailers of 0.5 ha storage capacity.

The initialization of gangs as objects of class MAN-MACH-SET is shown in Table 5.46. It concerns the input of parameters (name, type of shift over periods and within a week and set of man-machine systems to which this gang belongs) and of data such as the rate of operation or working capacity, the first and subsequent set-up durations on a day (includes refuelling, machine preparation and travelling time), the total number of categories of men and machines involved and for each category the category number and number of items required. The gang is queued in one of the sets of man-machine systems, the storage capacity and the costs are derived from the required men and machines (Table 4.68) and a queue is created to assemble the required men and machines when the gang starts work (Table 4.69). Table 5.47 shows an example of the input. GANG[9] is the only gang belonging to set 2; indeed the grain drier as an automatic installation can be selected for work independent of the other gangs because men and the other machines are not involved.

Table 5.46 Initial section of class MAN-MACH-SET.

```
!   i   n   i   t   i   a   l;

procedure INIT_INPUT_;                      !----  ----  ----  ----  ----;
inspect PARAMETERS do begin
    integer
    LE_S, LE_I;
    CAPACITY:= Inreal;
    SETUP1:= Inreal; SETUP2_N:= Inreal;
    LE_S:= Inint;
    for LE_I:= 1 step 1 until LE_S do begin
        LE_SQN:= Inint;
        RQRD_NMB_LE [LE_SQN]:= Inint;       !required number of elements;
    end;
end ** of inspect parameters;

new RECORD_MM_S (this MAN_MACH_SET).Into (MM_S_SET [SET_NO].MM_S_SET_Q);
INIT_INPUT_;
SETUP_GNG:=SETUP1;
STORE_COSTS;
ASSMBL_Q:- new Head;
```

Table 5.47 Input related to objects of class MAN-MACH-SET.

GANGS
*****

Parameters Name (24char)	Type of shift availability in Periods	Week	Data Type of Set of man-mach systems	Working capacity (ha/h)	Setup duration First time in a day (h)	Next (h)	Categories of men and machines involved Number of cate-gories	Category / number needed (men, tractors, equipment,..)=1 line)								(reference type MM_S [.] GANG [.])
.	.		systems .				gories .	.	.	.	.	.	.	.	.	
PAR./DATA LIST of GANGS:																
2 men Combine harvesting	0	0	1	1.0	0.4	0.2	4	1	2	2	1	3	1	6	2	[1]
1 man Combine harvesting	0	0	1	0.74	0.7	0.2	4	1	1	2	1	3	1	6	2	[2]
Baling..................	0	0	1	1.37	0.7	0.2	3	1	1	2	1	4	1			[3]
Bale loading............	0	0	1	1.25	0.2	0.2	4	1	1	2	1	5	1	7	1	[4]
Bale gathering..........	0	0	1	0.78	0.2	0.2	4	1	1	2	1	5	1	7	1	[5]
Bale unloading..........	0	0	1	2.28	0.2	0.2	4	1	1	2	1	5	1	7	1	[6]
1 Ploughing.............	0	0	1	0.65	0.5	0.2	3	1	1	2	1			8	1	[7]
2 Ploughing.............	0	0	1	0.45	0.5	0.2	3	1	1	2	1			9	1	[8]
Grain drying............	0	0	2	0.2	.0	.0	1							10	1	[9]
2 men Service & repair..	0	0	1	.0	.0	.0	1	1	2							[10]
1 man Service & repair..	0	0	1	.0	.0	.0	1	1	1							[11]

****

Table 5.48 Input related to objects of class GANG-SET.

```
COMBINATIONS_6
**************

          Parameters                                        Data                                  (reference type
          Name    Type of shift  Type of     Number          Types of gang included                 MM_5 [gangs + .]
          (24char)availability in Set of     of gangs        in combination                         COMB_6 [.]       )
                  Periods Week   man-mach    in com-
                  .       .      systems     bination        .     .     .     .     .

PAR./DATA LIST of COMB_6:
Cmb+Baling.............. 0       0      1           2               2     3                                 [1]
Cmb+Loading............. 0       0      1           2               2     4                                 [2]
Cmb+Gathering........... 0       0      1           2               2     5                                 [3]
Cmb+Unloading........... 0       0      1           2               2     6                                 [4]
Cmb+1Ploughing.......... 0       0      1           2               2     7                                 [5]
Bal+Loading............. 0       0      1           2               3     4                                 [6]
Bal+Gathering........... 0       0      1           2               3     5                                 [7]
Bal+Unloading........... 0       0      1           2               3     6                                 [8]
Bal+1Ploughing.......... 0       0      1           2               3     7                                 [9]
Load+1Ploughing......... 0       0      1           2               4     7                                 [10]
Gath+1Ploughing......... 0       0      1           2               5     7                                 [11]
Unl+1Ploughing.......... 0       0      1           2               6     7                                 [12]
1Plg+2Ploughing......... 0       0      1           2               7     8                                 [13]

****************************************************************************************************
```

The initialization of objects of class GANG-SET concerns the input shown in Table 5.48 consisting of the usual parameters, the number of gangs belonging to this combination and the sequential number of those gangs, for instance, COMB-G[1] consists of GANG[2] and GANG[3]. Table 5.49 shows the program that queues the combination in a set of man-machine systems, reads the data and calculates costs and the number of required men and machines from those of the gangs. Table 5.50 shows the further initialization of gangs and of combinations; 'inner; 'means that statements of subclasses are inserted at that place in the program. The test if all the required elements of men and machines are available is performed by calling AVLBLTY-TEST that checks if the available number in a category is sufficient to meet the required number (Table 4.65); if you require more items of a category than are available then the combination exists but cannot be used. Table 5.48 shows only the relevant combinations; combinations with GANG[8] are irrelevant in this particular case because GANG[7] ploughs with a higher capacity (Table 5.47; 0.65 > 0.45 ha/h). Such less relevant combinations are not prevented when the combinations are generated (Table 4.79) instead of read from an input file.

Exercise: Can you think of a situation where combinations with GANG[8] are relevant (a plough that may replace another almost identical machine in some cases; look at Table 5.45)?

The initialization of objects of OPRTN-MATRL is shown in Table 5.51; it concerns the input and after Passivate (necessary to wait until materials are created) this object is placed in queues of the materials processed and the

Table 5.49 Initial section of class GANG-SET.

```
!i     n     i     t     i  _  a     l;

procedure INIT_INPUT_;                    !----  ----  ----  ----  ----;
inspect PARAMETERS do                     !calculation of required number and costs per hour;
for I1:= 1 step 1 until GNGS do begin
    integer GNG;
    GNG:= RQRD_GNG [I1]:= Inint;
    for LE_SON:=1 step 1 until LE_SONS do RQRD_NMB_LE [LE_SON]:=
    RQRD_NMB_LE [LE_SON] + GANG [GNG].RQRD_NMB_LE [LE_SON];
    COSTS_H:= COSTS_H + GANG [GNG].COSTS_H;
    COSTS_H_FXD:= COSTS_H_FXD + GANG[GNG].COSTS_H_FXD;
end of inspect parameters;

new RECORD_MM_S (this GANG_SET).Into (MM_S_SET [SET_NO].MM_S_SET_Q);
INIT_INPUT_;
```

Table 5.50 Initial section of class MAN-MACH-SYS.

```
!    i    n    i    t    i    a    l;

inner;                                    !initial of subclasses first;

STATE_NEXT:= STATE:= DOWN;
for LE_SON:= 1 step 1 until LE_SONS do AVL_TEST (LE_SON);
AVLBLTY_TEST;                             !sets also state_next;
```

Table 5.51 Initial section of class OPRTN-MATRL.

```
!   i   n   i   t   i   a   l;

procedure INIT_INPUT_;                          !----  ----  ----  ----  ----;
inspect PARAMETERS do begin                     !gang exists already!!!;
   GANG_G:-GANG [Inint]; GANG_G.OPRTN__G:- GANG_G.OPR_MT__G:- this OPRTN_MATRL;
   MATRLS_PRCSD:= Inint;
   for I2:= 1 step 1 until MATRLS_PRCSD do begin
      M_PRC [I2]:= Inint;
      PRCSNG_CNDTN [1,I2]:= Inint; PRCSNG_CNDTN [2,I2]:= Inint;
      PRCSNG_CNDTN [3,I2]:= Inint;
   end;
   MATRLS_PRDCD:= Inint; MATRLS_PRDCBL:= Inint;
   for I2:= 1 step 1 until MATRLS_PRDCBL do M_PRD [I2]:= Inint;
   !  1:matrls_prdcd have to refer to materials delivered;
   !  matrls_prdcd+1:matrls_prdcbl refer to materials replacing one of the previous mat.;
end of inspect parameters;

INIT_INPUT_;
NAME_COMP:= GANG_G.NAME_COMP;
CAP_FRAC:= 1.0;
Passivate;                                      !wait until materials are created;
for I2:= 1 step 1 until MATRLS_PRCSD do begin
   MAT_PROC [I2]:-MATRL_PRC [M_PRC [I2]];
   new RECORD_OPR (this OPRTN_MATRL).Into (MAT_PROC [I2].OPRTN_PROC_Q);
end;
for I2:= 1 step 1 until MATRLS_PRDCBL do begin  !mat_prod[] may change due to mat_proc[];
   MAT_PROD [I2]:- MATRL [M_PRD [I2]];           ! matrl [0] == none;
   new RECORD_OPR (this OPRTN_MATRL). Into (MAT_PROD [I2]. OPRTN_DLVR_Q);
end;

inner;                                          !initialisation of subclasses, if any;
```

materials delivered. The input is shown in Table 5.52 and consists of a reference to a gang, the number of materials processed, the type of that material with three processing conditions that are sufficient for processing and defined in that material (repetition of the same condition is used to achieve three data elements), the number of materials produced and produceable with the type of those materials. When three types are produceable (2, 6 and 7 in OPR-MAT[1]) only the first two are produced simultaneously, for instance, the combine-harvesting operation produces straw and wet grain or straw and dry grain although straw, wet grain and dry grain are produceable materials. The input has to present first the materials that are produced, then the materials selectable in increasing sequence (the same sequence of materials delivered (wet or dry grain) will be used for the processing conditions in the material processed (wheat), Table 5.56). The initialization of objects of class SRVC-REPR is shown in Table 5.53 and concerns the reference to a gang, a reference in the gang to this service-repair object and a queue to contain machines in need of service or repair. The input is shown in Table 5.54. For operations, a shift over periods or within a week is not required, because the related gang controls these facts already; so the program forces references to non-existent shifts (Table 5.32) by giving actual parameters such as SH-PERD[0] and SH-WKLY[0], which refer to 'none'.

The initialization of objects of subclasses of class MATERIALS starts with (Table 5.55) the creation of queues for fields, FLD-Q, for operations

Table 5.52 Input related to objects of class OPRTN-MATRL.

```
       OPERATIONS MATERIAL
       *******************

              Data                                                                                        (reference
              Type of        Materials processed (<= 4)           Materials produced                      type
              gang needed    Number  Type of  Processing          Number  Number, Types of materials      OPRTN [.]
              (=GANG [.])            material conditions          produced   producable (<= 4)            OPR_MAT [.])
                                     (1/line) 1  2  3 of material simultan.(=..   .

DATA LIST of OPR_MAT:
              1              1       1        2  2  2             2       3       2       6       7       [1]
              2              1       1        1  2  2             2       3       2       6       7       [2]
              3              1       2        1  1  1             1       1       3                       [3]
              4              1       3        1  1  1             2       2       4       5               [4]
              5              1       3        1  1  1             2       2       5       8               [5]
              6              1       4        1  1  1             1       1       8                       [6]
              7              1       5        1  1  1             1       1       9                       [7]
              8              1       5        1  1  1             1       1       9                       [8]
              9              1       6        1  1  1             1       1       7                       [9]

***************************************************************************************************************
```

Table 5.54 Input related to objects of class SRVC-REPR.

```
       OPERATIONS SERVICE_REPAIR
       *************************

              Data                (reference type
              Type of             OPRTN [oprtns_mt + .]
              gany needed         OPR_S_R [.]      )

DATA LIST of OPR_S_R:
              10                        [1]
              11                        [2]

***************************************************************************************************************
```

Table 5.53 Initial section of class SRVC-REPR.

```
I     i     n     i     t     i     a     l;

procedure INIT_INPUT_;                         !----  ----  ----  ----  ----;
inspect PARAMETERS do begin                    !gang and mach exist already!!!;
   GANG_G:- GANG (Inint);
   GANG_G.OPRTN__G:- GANG_G.OPR_SR__G:- this SRVC_REPR;
end of inspect parameters;

INIT_INPUT_;
NAME_COMP:= GANG_G.NAME_COMP;
MCHN_S_R_Q:- new Head;

inner;                                         !initialisation of subclasses, if any;
```

processing, OPRTN-PROC-Q, or delivering the material, OPRTN-DLVR-Q, the state of the object and a position that is used for output (Tables 4.21 and 4.31). The objects of the classes INITIAL-MAT and INTERMDT-MAT are subclasses of class PROCESSED-MAT; they have the same input and initialization. An example of the input is shown in Table 5.56. The program is shown in Table 5.57; it shows the input, the default values of processing conditions, material delivered, timeliness condition and workable time condition and the queueing of this material in the object controlling the urgency calculations. The input (Table 5.56) consists of one line with parameters: a name, three types of shift (the shift over periods is necessary, the shift within a week is optional and can be used to record data per shift, the shift for the urgency is useful), the number of processing conditions considered, the number of timeliness functions used and the number of dates in each function, the number of workable time conditions and the days for which a duration will be given (the default condition is 1 or 0). The next line shows the following 'general data': an expected capacity (rate) of processing this material that is used only in the calculation of the urgency, an expected price of the material (to transform the fractional data of the timeliness function to money value), a text that tells that material delivered will be queued at the end of the existing fields ('tail') or in front of the fields ('head'; even 'TAIL' in capital letters is interpreted as 'head'), a maximum quantity is given and delivery stops when it is achieved and may be stopped until it is diminished to the fraction mentioned. The processing conditions are mutually exclusive ranges of a property (moisture content) and consist of the

Table 5.55 Initial section of class MATERIAL.

```
I     i     n     i     t     i     a     l;

FLD_Q:- new Head;
OPRTN_PROC_Q:- new Head;
OPRTN_DLVR_Q:- new Head;
STATE_NEXT:= STATE:= DOWN;
M_POS:= MAT_NO * 13;

inner;                                         !initialisation of subclasses, if any;
```

Table 5.56 Input related to objects of class PROCESSED-MAT.

```
        INITIAL MATERIALS
        *****************

               Parameters of material, processed material                                           ,subclasses
               Name    Type of shift fur        Number of       Number of timeliness    Number of workable time
               (24char)availability in Urgency  Processing      Functions, Dates used   Conditions given for
                       Periods Week      calcul. conditions     ()=1)   ()=2)           ()=0)  1-Days

PARAMETERS MAT.:
Wheat (winter).......... 1       1       1       2              1       6               8       8
               ----------------------------------------------------------------------------------------------

               General data
                       Expected         Expected price  Field delivered         Available quantity of material
                       capacity         of material     in processing queued    Maximum Delivering prohibited
                       [ha/h]           [Dfl/ha]        at 'tail' / 'head'      [ha]    until    ... (fract. max.)

DATA MAT.1:            .68              2500.0          tail                    100.0            1.0
                       --------------------------------------------------------------------------------------

               Processing conditions (exclusive categories, 1/line; union of categories defines workability)
                       Type of material        Range of property (e.g. moisture content)
                       delivered (0=no effect) lower bound     upper bound

DATA MAT.PRC.:         6                       19.0            23.0
                       7                       0.0             19.0
                       --------------------------------------------------------------------------------------
```

Table 5.56 (continued)

```
                 Timeliness data to derive urgency of processing
                        Disurgency of   Timeliness losses       Minimum urgency is
                        delivering used derived from functions  from area
                        ('yes' / 'no')  (1.0 / 0.0 = no)        [ha]

DATA MAT.3:             no              1.0                     1.0
                        ------------------------------------------------------------------------------------

                        Relative dates (1-n lines) and Fractions of yield at optimum date 0 (1-n lines/function)
                        item:   1       2       3       4       5       6       7       8       9       10 (+n*10)

DATA MAT.TML.:                  -3      -2      0       16      40      100

                                .15     .93     1.0     .96     .84     .0
                                ----------------------------------------------------------------------------

                 Workability data
                        Long term workable      Fraction of workable time expected for processing this material in each
                        duration Conversion     period of shift period
                        [h/d]   factor          1       2       3       4       5       6       7       8       9       10 (+n*10)

DATA MAT.5:             5.0     1.6             1.0     0.6     0.6     0.6     0.6     0.6     0.6     1.0
                        ------------------------------------------------------------------------------------

                        Expected workable durations on subsequent days fur several conditions (1-n lines/condition)
                        current day 1   day:2   3       4       5       6       7       8       9       10 (+n*10)

DATA MAT.WRKBL.:                7.6     3.3     1.9     2.3     3.9     4.1     2.8     3.8
                                5.3     2.7     2.4     2.9     3.8     3.7     3.5     4.2
                                0.8     2.1     3.0     3.4     3.7     3.9     3.9     4.5
                                0.0     0.8     1.5     2.0     2.4     2.9     3.7     4.3
                                9.8     8.1     7.1     6.4     6.0     5.8     6.1     5.9
                                7.0     6.6     5.9     5.5     5.0     5.0     4.4     4.0
                                1.9     4.5     4.9     4.6     4.7     4.6     4.6     3.9
                                0.1     1.5     3.0     4.2     4.8     4.7     4.8     4.6
                                ----------------------------------------------------------------------------
```

Table 5.57 Initial section of class PROCESSED-MAT.

```
!      i      n      i      t      i      a      l;

procedure INIT_INPUT_;                              !----  -----  ----   ----   ----;
inspect PARAMETERS do begin
    FIND_DATA_AT ('DATA MAT.1');
    CAP_PROC:= Inreal;
    PRICE_HA:= Inreal;
    TAIL:- Copy('tail');
    Lastitem;                                       !scans to next character;
    FLD_AT_TAIL :=Intext (4) = TAIL;                !tail = 'tail', no capital letters in input!!;
    QUANT_MX:= Inreal; QNT_MX_FACT:= Inreal;        !0.0<...<=1.0;
    FIND_DATA_AT ('DATA MAT.PRC.');
    for I_M2:= 1 step 1 until PROC_CNDTNS do begin
        MAT_DLV_CNDTN [I_M2]:= Inint;
        PRC_CNDTN_LB [I_M2]:= Inreal;
        PRC_CNDTN_UB [I_M2]:= Inreal;
    end;
    FIND_DATA_AT ('DATA MAT.3');
    Lastitem;
    DISURG_USFD:= Intext (3) = YES;
    LOSS_MULT:= Inreal;
    QNT_ALT_FLD := Inreal;
    FIND_DATA_AT ('DATA MAT.TML.');
    for DATE_NO:= 1 step 1 until TML_DATES do TML_DATE [DATE_NO]:= Inint;
    for TML_CNDTN:= 1 step 1 until TML_FNCTNS do
    for DATE_NO:= 1 step 1 until TML_DATES  do TIMELINESS [TML_CNDTN,DATE_NO]:= Inreal;
    FIND_DATA_AT ('DATA MAT.5');
    T_WRKBL:= Inreal;                               !>0.0 for mat. processed;
    WT_MULT:= Inreal;
    J1:=  SH_PRD.PERDS;
    for PERD_MAT:= 1 step 1 until J1 do T_FRAC_PROC [PERD_MAT]  :=Inreal;        !>0.0!!;
    T_FRAC_PROC [J1 + 1]:= T_FRAC_PROC [J1];
    if CNDTNS_W_T > 0 then begin
        FIND_DATA_AT ('DATA MAT.WRKBL.');
        for CNDTN_W_T:= 1 step 1 until CNDTNS_W_T do
        for DATE_NO:= 1 step 1 until DAYS_WT_DATA do T_WRKBL_EXP [CNDTN_W_T,DATE_NO]:= Inreal;
    end;                                            !may be                                0.0;
end of inspect parameters;

INIT_INPUT_;
for J1:= 1 step 1 until PROC_CNDTNS do M_PRC_CNDTN [J1]:= true; !redefined in attr_updt_m_;
MAT_DLVR:= MAT_DLV_CNDTN [IMIN (1, PROC_CNDTNS) ];
TML_CNDTN:= IMIN(1, TML_FNCTNS);                    !defaults, redefined in attr_updt_m_;
CNDTN_W_T:= IMIN (1, CNDTNS_W_T);
PERD_MAT:= 1;
if SH_URGN =/= none then new RECORD_MAT (this MATERIAL).Into (SH_URGN.MAT_Q);
```

material delivered and the lower and upperlimit of the property; the delivered materials have to be ordered with low sequence numbers first (6 before 7). The timeliness data concern the use of disurgency to prevent too much delivery of this material, a variable that is 1.0 when the function is also used to derive the timeliness losses or is 0.0 when the timeliness losses are derived from more specific data (only possible in a specific subclass with a modified virtual procedure ATTR-INTG-F-, Table 4.46), and a minimum area for which an alternative urgency is calculated (Tables 4.42 and 4.43). The actual timeliness data consist of a row of dates (relative with respect to the optimum date for processing) and some lines with fractions valid at these dates (one line for each timeliness function considered). The last date must not exceed 100 but has to be such that a timeliness loss will be calculated even if the processing is delayed considerably. The workability data are only used to calculate urgency. First there is some long term estimate of the workable time per day and a conversion factor to justify the estimate to another length of working day or to another definition of workability (depends on the processing conditions). This factor is mainly used to tune the urgencies to achieve minimum costs over the years. For each period considered in SHIFT-PERD, an estimate is given of the fraction of the workable duration spent on processing this material; such a fraction can be known from your experience, from linear-programming results for the same problem or from considering the relative capacities of the available gangs. For example, assume that harvesting, baling and gathering is performed during the same workable time and requires 1.5, 0.8 and 1.7 h/ha, respectively; the relative use is 0.37, 0.2 and 0.42 and the fractions become 0.74, 0.4 and 0.84, respectively because both men can perform the job. If the workable time for gathering is 1.5 times that for baling, the fractions become 0.74, 0.4 and 0.56 (i.e. 0.84/1.5). After the wanted harvesting period (some weeks) the fractions are reduced and that of the subsequent stubble-ploughing operation is raised. Fractions lower than 0.1 are not advisable. The fraction of the first period in the year, before the harvesting season when the material is not yet available, is irrelevant and assigned as 1.0; the fraction of the period, after the harvesting season, Period 8, the last one defined in SH-PRD (Table 5.39), is 1.0. The fraction of Period 9, from 31 Dec. to the last day of a year, 500 or 1000, becomes equal to that of Period 8. A low fraction of processing results in greater expected delay of processing and so in a higher urgency than with a higher fraction.

Finally workable durations for a number of days and some conditions are shown. These data are used to calculate the urgency (Table 4.42) for the current valid condition (default value of condition is 1 and can be justified in a specific subclass only). Procedure ATTR-UPDT-M-, Table 4.45, has no influence on the timeliness condition nor on the workable time condition; if you want an influence of material properties on them, redefine the 'virtual' procedure in a subclass of a specific material in the experimental frame (for instance, Table 7.2). The use of an excess capacity of processing is also only possible in a subclass; such extra capacities may depend on properties of the material processed.

The input for objects of class FINAL-MAT is limited to the name (a parameter; other parameters are assigned default values referring to non existing shifts, Table 5.33). The initialization is shown in Table 5.58.

Initialization of some objects is continued now by producing a list of values of the random number generator drivers used in the machines for failure, repair and service and by activating the objects that met a call to Passivate such as machines and operations (Table 5.59). Using Passivate and Activate ensures that further initialization occurs after all the objects are created; such a guarantee is not possible if Hold (0.0) is used instead of Passivate.

Table 5.58 Initial section of class FINAL-MAT.

```
!       i       n       i       t       i       a       l;

DLVR_ALLWD:= true;
FLD_AT_TAIL:= true;
FLD_EXPCT_M_;                                   !to create fld_d;
```

Table 5.59 Final part of the initialisation of an experiment.

```
inspect ADMNSTR.REPORT_PERD do begin
    Outtext (EXPER_IDF); Outimage;
    Outtext ('Machine Random number driver U_FL_RP       U_SRVC and driver U');
    Outtext (' of drivers at time:'); Outfix (Time,6,12); Outimage;
    for I:= 1 step 1 until MACHNS do inspect MACH [I] do begin
        Outtext (NAME_COMP); Outint (U_FL_RP,12); Outint (U_SRVC,12); Outint (U,12);
        Outimage;
    end;
end;
for I:= 1 step 1 until MACHNS do activate MACH [I];          !now decide exists;
for I:= 1 step 1 until OPRTNS_MT do activate OPR_MAT [I];    !now materials exist and;
!                   !become known in operations, also operations known in material;
```

The initialization of the main program is now considered, i.e. the block referenced by MAIN in the simulation. It consists of an initial and a dynamic part. Table 5.60 shows:

- the value of END-EXPRMNT, LAST-DAY-YEAR (a convenient value larger than 365, for instance, 500 or 1000) and a counter of years, YR-N;
- a message to the terminal giving the value of the driver of random number drivers;
- a call to INIT-EXPER, a procedure that creates and initializes the required objects in the scheduling system as described in Section 5.2.1 and shown in Tables 5.31 – 5.33;
- the message to the terminal that the set up is completed and the simulation starting;
- the shifts over periods are activated at Time; so they are scheduled after this current process MAIN.

Table 5.60 Initial section of the main program belonging to an experiment.

```
!       i       n       i       t       i       a       l       of experiment;

for I10:= 0,1,2,3,4 do INFILE_TXT10 [I10]:- INF_TXT [I10];
EXPER_IDF:- EXP_IDENTIF;
END_EXPRMNT:= false;
LAST_DAY_YEAR:= 500;
YR_N:= 1;
Outimage;
Outtext ('Current value of driver U of random number drivers is:');
Outint (U,12); Outtext ('  at time:'); Outfix (Time,6,12); Outimage;
INIT_EXPER;                             !further initialisation of experiment;
Hold (0.0);                             !complete initialisation ;
Outimage;
Outtext ('**-->Setup of experiment completed, simulation starts');
Outimage;
for I:= 1 step 1 until SHS_PERD do activate SH_PERD [I] at Time;
!                                       !now decide and components exist;
```

In several classes, there is a specific definition of the 'virtual' procedure INIT-INPUT-. This allows a redefinition of the input in subclasses you want to add. In that case the creation of objects concerns objects of those new subclasses.

The following section describes the dynamic part of MAIN; it handles several seasons and resets variables in objects.

5.2.3 Seasons

Within one simulation run several experiments can be performed (different number of men, machines, different processing conditions, etc.) and within one experiment several seasons can be used. The dynamic section of process MAIN is shown in Table 5.61. The file EXP-FILES (Table 5.35) shows after the keyword 'WEATHER DATA FILES' the name(s) of the file(s) with weather data. The variable END-SEASON is set to false and NEW-FILE in

Table 5.61 Dynamic section of the main program belonging to an experiment.

```
PAR:-EXP_FILES;
inspect EXP_FILES do begin
    KEYWORD ('WEATHER DATA FILES');
    Inimage;

    ! d    y    n    a    m    i  , c     of seasons in an experiment;

    while not END_EXPRMNT do begin
        if not Lastitem then begin
            END_SEASON:= false;
            PAR:- PARAMETERS:- EXP_FILES;
            WTHR.NEW_FILE;
            PAR:- PARAMETERS:- INIT_FIELDS;
            INIT_FIELDS.Open (Blanks (132));
            KEYWORD ('FIELD PROPERTIES');
            RELOAD;                     !zeroing variables and reads initial fields;
            INIT_FIELDS.Close;
            Outimage;
            Outtext ('**--..>Experiment continues for year');
            Outint (WTHR.YEAR,6); Outimage;
            Hold (LAST_DAY_YEAR);       !until end of year;
            YR_N:= YR_N + 1;
            Outimage;
            Outtext ('Current value of driver U of random number drivers is:');
            Outint (U,12); Outtext ('  at time:'); Outfix (Time,6,12); Outimage;
            inspect ADMNSTR.REPORT_PERD do begin
                Outtext (EXPER_IDF); Outimage;
                Outtext ('Machine Random number driver U_FL_RP      U_SRVC and driver U');
                Outtext (' of drivers at time:'); Outfix (Time,6,12); Outimage;
                for I:= 1 step 1 until MACHNS do inspect MACH [I] do begin
                    Outtext (NAME_COMP); Outint (U_FL_RP,12); Outint (U_SRVC,12); Outint (U,12);
                    Outimage;
                end;
            end;
            TIME_1JAN:= Time;           !time at 0.0 o'clock;
        end;
        Inimage;
        if Endfile or Image.Strip.Length = 0 then END_EXPRMNT:= true;
    end !! of year or season;
    activate ADMNSTR;                   !close files in terminal section;
end of inspect exp_files with seasons of weather data;
```

object WTHR of class WEATHER is called (Table 5.3) to read the file name and to activate the dynamic section of weather that updates attributes of materials. For each season RELOAD is called that calls RESET in objects referenced as COMPNT[.]. Table 5.62 shows procedure RELOAD; it is rather simple but in Chapter 7 'Extensions' some extensions on administration will be included. The objects referenced by COMPNT[.] are shown in Tables 5.32 and 5.33 and concerns objects of the classes LABOUR, EQUIPMENT, MAN-MACH-SET, GANG-SET, OPRTN-MATRL, SRVC-REPR, INITIAL-MAT, INTERMDT-MAT and FINAL-MAT. Table 4.4 shows procedure RESET defined in class COMPONENT; it makes cumulative costs and time duration equal to zero, updates time variables and calls the 'virtual' procedure RESET- that is defined in subclasses such as RESET- of class EQUIPMENT (Table 5.44; also used during the initialization to assign values to variables). Table 5.63 shows RESET- of class MAN-MACH-SET and makes the cumulative set-up duration equal to zero. Some cumulative variables of objects of class OPRTN-MATRL are made zero in RESET-, Table 5.64. Table 4.48 shows RESET- from class PROCESSED-MAT; it sets variables on appropriate values for a new season and calls RESET-M- to provide the material with fields as is shown in Table 4.49. This procedure uses the file with initial fields that is referenced by PARAMETERS (at this moment) and by INIT-FIELDS (Table 5.61). The input shown in Table 5.65 is used to create four initial fields as in Tables 4.49-4.51 and consists of a number of initial fields and the data for each field, such as area, the dates when that material becomes processable and achieves its optimum yield, and the name of the field; these data are given for all the initial materials. RESET- is not defined in the classes LABOUR, GANG-SET, SRVC-REPR and FINAL-MAT; in such cases the empty definition in class COMPONENT: procedure RESET-; ; is used (Table 4.4).

Table 5.62 Procedure RELOAD of the main program.

```
procedure RELOAD;                              !----  ----  ----  ----  ----;
begin                                          !called in dynamic of seasons;
   for I:= 1 step 1 until C_N do COMPNT [I].RESET;
end;
```

Table 5.63 Procedure RESET- of class MAN-MACH-SET.

```
procedure RESET_;                              !----  ----  ----  ----  ----;
SETUPTIME:= 0.0;                               !called in reset;
```

Table 5.64 Procedure RESET- of class OPRTN-MATRL.

```
procedure RESET_;                              !----  ----  ----  ----  ----;
begin                                          !called in reset;
   QNT_NOT_PROC:= QUANT_PRC_OPR:= 0.0;
   CAP_FRAC:= 1.0;
end;
```

Table 5.65 Input related to fields belonging to an object of class INITIAL-MAT.

```
            FIELD PROPERTIES
            ****************

            Number of           Data of each field (1 line/field)
            initial fields      Area        Month / Date (24.00h) material is      Name of field
                                [ha]        processable    <=    at optimum yield   (<= 30 char.)

DATA LIST of FLD.:    4
                                15.0        08      05            08      05        FIELD A
                                15.0        08      05            08      05        FIELD B
                                15.0        08      05            08      05        FIELD C
                                15.0        08      05            08      05        FIELD D

**************************************************************************************************
```

After reloading a message to the terminal shows the year to which the season belongs, Table 5.61 and process MAIN is rescheduled on the very last day of the year (initialized at 500 or 1000 days, Table 5.60). At the end of the year, a message to the terminal shows the value of the driver of random number drivers and a list of drivers for each machine (on a file, Table 5.23) to show whether the values change or not. The number of years is increased and the simulation time 'Time' at the beginning of the next year is recorded. In EXP-FILES the system looks after another weather data file within the experiment. The list of seasons is ended by a blank line and then END-EXPRMNT becomes true and ADMNSTR is activated to close output files. Table 5.66 shows the program which sends a message to the terminal that the experiment is completed. Now the main program looks after the file EXP-FILES if the end of the file is achieved (Table 5.29; while not Endfile do ...) and stops the simulation with a message that the simulation is completed (Table 5.66); the current date and hour of the execution are also printed. Otherwise another set of files is found labelled with the keyword EXPERIMENT; the simulation starts another experiment by creating and initialising objects and by creating a new time axis and variable Time (starting at 0.0).

The system variable Time increases from season to season. An example occurred where the year starting with 10000.0 has almost the same output as the first year starting at Time = 0.0 (both years in the simulation represent the same season!). The resulting costs of the year starting with Time = 20000.0 were slightly different. Because the year starting with Time = 10000.0 shows a lot more transfers of fields of almost 0.00 ha, it is not recommended to use more than 20 seasons when LAST-DAY-YEAR = 500. Several identical experiments can be defined, each with some seasons or even one season to prevent problems arising from accuracy. Two other ways are possible: (i) to use long real variables associated with 'Time'; (ii) to reset the simulation time 'Time' to zero after each season; these ways are not always available in a machine-dependent version of SIMULA and are therefore not used.

Table 5.66 Final part of main program.

```
            end ** of agropexperiment;
        end of inspect environment of experiment;
        Outimage;
        Outtext ('**-->Experiment completed'); Outimage;
    end  ** of simulation run with some experiments;

    ! t      e      r      m      i      n      a      l;

    EXP_FILES.Close;
    Outimage;
    Outtext ('**>Simulation completed at'); Outtext (TODAY); Outtext ('  '); Outtext (DAYTIME);
    Outimage;
    !************************************************************************************;
end ** of main;
```

5.2.4 *Input data restraints*

There are many interactions between input data. This section describes the restraints; although it is impossible to cover every thinkable situation, the most relevant limitations are described. In general an input file consists of:
- keywords
- data
- explanatory comment.

The keywords have to be as in the program; the main keywords in front of the data of several objects may be indented but has to appear as the first word on a line (Table 5.28 procedure KEYWORD). The intermediate keywords just in front of particular data (within an object or a list of objects) appear at the very beginning of a line and are followed by ':' and the data (Table 5.34 procedure FIND-DATA-AT). After the main keyword (or after data) and before the following intermediate keyword, explanatory comment is possible (table headings). The input data are given in six files related to:
- experiment
- environment
- man-machine system
- materials
- initial fields
- weather data and material properties.

The experiment file (Table 5.35) starts with the main keyword 'EXPERIMENT' and a text to identify the experiment. This line is used as an identification in output files. The following four lines each start with a file name of 12 characters; after the twelfth character some comment is possible. The main keyword 'WEATHER DATA FILES' starts a list of files (seasons) with weather data and material properties (for instance, moisture content). The explanatory comment on a file may not exceed the line that starts with the file name. The list is ended by a blank line and a subsequent experiment and list of weather data files may appear.

The environment file defines the general context of the experiment and starts with the main keyword 'GENERAL DATA' (Table 5.36). Some headings as explanatory comment may precede the intermediate keyword 'PARAMETERS EXP.' After the character ':' the following parameters are shown:
- number of categories of men and machines (at least the maximum used in the man-machine system);
- number of men (+);
- number of machines (+);
- number of sets of men-machine systems (+);
- number of gangs (+);
- number of combinations (+);

- number of operations for materials (+);
- number of operations for service and repair (+);

  (+) these numbers are used to read lists of objects from the man-machine system; a longer list is skipped; a shorter list is ended with a blank line and adjusts the parameter; the number of operations must not exceed the number of gangs;
- number of initial materials (x);
- number of processed materials (x), *including initial material;*
- number of materials (x), including processed materials,

  (x) these numbers of objects are required in the material file;
- number of shifts to control the availability over periods (y);
- number of shifts to control the availability within weeks (y);
- number of shifts to calculate urgencies (y),

  (y) these numbers of objects are required in this file.

Table 5.39 shows the following main keyword 'SHIFT PERIOD'. The intermediate keyword 'PARAMETERS PRD.' is requested (Table 5.31). After ':' the number of periods is shown (at least one). The following keyword is 'DATA PRD.' (Table 5.37) and after ':' a number of subsequent dates are shown that are preceded by '+' or '-' to indicate the availability of the period ending at that date; with more than 10 periods more lines with dates are used. The end of the last period has to exceed the last date with weather data. If more shifts are used, both intermediate keywords and the data are given more times in the file.

The following main keyword 'SHIFT WEEK' (Table 5.31) is shown in Table 5.39. Each shift required starts with the intermediate keyword 'PARAMETERS WK.' and follows with 'DATA WK.CTG.' and 'DATA WK.COSTS' related to parameters (all are at least one), category of days and costs, respectively. Usually the regular time has no extra costs, but you may do so and distinguish between men with different costs. Note the influence on the urgency of a gang. Table 5.38 shows that for each period considered the keywords 'DATA WK.SHIFTS' and 'DATA WK.4' are used to find data on the period, a series of increasing clocktimes (with a 24 hour cycle) and on cost categories per shift and per type of day (preferably on separate lines). The end of a period is independent of the periods used in SHIFT-PERD. Only for the first object of this weekly shift 'DATA WK.5' must contain the regular begin and end of a working day.

If more shifts are required to control the availability of objects within a week, for instance, one for men, another for cattle or installations, the sequence of keywords starts again with 'PARAMETERS WK.'.

Table 5.41 shows the following main keyword 'SHIFT URGENCY' (requested in Table 5.31) and the two intermediate keywords 'PARAMETERS URG.' and 'DATA URG.' followed by the number of points per cycle and the series of (increasing) clocktimes (usually with a 24 hour or weekly cycle) to calculate urgencies. The last two keywords and the data are repeated when several shifts are considered for different materials. The number of points is at least one: the first calculation takes place at point zero, i.e. the moment the material becomes available (according to its SHIFT-PERD).

The man-machine system file consists of lists of men, machines, etc. and each list is followed by a blank line. A list is used until sufficient objects are created to meet the requirement presented under 'GENERAL DATA' of the environment file; more objects are skipped. If the list is shorter than the requirement, the requirement is adjusted in the program. The list contains parameters for each object on a new line (without blank lines between objects) and the data on one or more lines; only on the line with the last data element some explanatory comment is possible (in general the sequence number of a reference variable). Three parameters (a name of the object, the type of shift for periods and within a week) are found in all lists except for operations. The name has to contain exactly 24 characters. The type of shift can be zero; a reference by the element zero of the reference array is a reference to 'none', an object that does not exist. The shift of periods is needed for men and may be used also for the sets of man-machine systems and for some machines (if they are not available in all periods). The shift within a week is needed for men only and may be used for machines (if such a pattern in a week is appropriate, for instance, for a milking parlour). If a weekly shift is used then a period shift is obligatory. The type of shift may not exceed the available number of shifts, as given in the environment file.

Table 5.32 shows the creation of objects from this man-machine system file. The first main keyword 'SETS OF MAN-MACHN.SYSTEMS' is in Table 5.42 followed by some comment lines, the intermediate keyword 'PAR. LIST of SETS' and ':'. The following lines define two objects with three parameters (name, period and week shift) and show the reference MM-S-SET [1] and MM-S-SET [2], respectively, as explanatory comment. The list is followed by a blank line.

The following main keyword is 'LABOUR'; Table 5.43. The parameter headings precede the intermediate keyword 'PAR. LIST of MEN' and ':'. The list contains for each object the usual parameters (name and two types of shift), a category and explanatory comment (a reference to MAN [.]). The category is one of the ten available numbers, as defined by the first parameter given in the environment file (Table 5.36). Independent of the explanatory comment are references given by the program in sequential order (Table 5.32, a loop with index I from 1 until MANN), so the comment is appropriate if the references start with one and increment by one.

The following main keyword is 'EQUIPMENT' (Table 5.45). The intermediate keyword 'PAR./DATA LIST of MACHN.' is preceded by lines with parameter and data headings and followed by ':' and a list of lines closed by a blank line. The four parameters (name, two shifts and category) are as for men; the category number of identical objects is the same, i.e.

Tractor 1 and Tractor 2 are exchangeable in a gang just as are Man 1 and Man 2. The ten data concern:

- the costs per hour; these costs are inevitable with the use of the machine, for instance, the fuel costs of a drier, but the costs are avoidable by deciding not to use the machine; these costs influence the urgency of a gang containing the machine;
- the storage capacity of trailers (some bale trailers are handled as one machine) and installations; this figure can be used by materials (for instance, wet grain) to handle an appropriate maximum quantity and to redefine the maximum prescribed with the material;
- the lower and upper limit of a failure free duration (whose length is equally distributed between these limits); zero is interpreted as no failure will occur for this machine (Table 5.44);
- the minimum duration a machine has to be used (running time) before a service is needed; zero is interpreted as no service needed;
- the lower and upper limit of the uniform distribution of the repair duration;
- the lower and upper limit of the uniform distribution of the service duration;
- the type of service, repair operation; i.e. a reference to object OPR-S-R [.]; zero is interpreted as no failure and no service needed.

The decimal data need at least one digit behind the decimal point! Digits before a decimal point are not obligatory in the input.

The keywords related to the gangs are 'GANGS' and 'PAR./DATA LIST of GANGS'. After ':' a list of parameters, data and comment is shown (Table 5.47) closed by a blank line. The parameters concern the three usual ones (name and two shifts) and a parameter giving the type of set of man-machine systems to which this gang belongs. This type of set is one of those defined in the experiment and cannot be larger than the list of sets in Table 5.42 or the number of sets wanted (Table 5.36 fourth parameter). Two sets are used: one with those gangs that need men and another with the grain-drying gang that uses only the automatic grain-drier. In this case no types of shift are prescribed (zero); this means that all the gangs are available permanently but that their use is controlled by the availability of each of the men and machines involved (this is updated by the program each moment that availability changes). The data concern:

- the processing capacity of the gang /rate of operation (the program has to be extended to change this capacity according to properties of a processed material);
- the set-up durations for the first time on a day and a lower one for sequential times;
- a number of different categories involved in the gang;
- a list of pairs of category and number of elements required; a category must be given only once and may not exceed the number (10) given as

first parameter in the environment file (i.e.in Table 5.36); the number of elements required is free but if it is larger than the number available then the gang cannot be used; if more than five categories are used, another line is used to continue the list of pairs.

The gangs are referenced by GANG [.] and by man-machine system MM-S [.]: this information is comment only.

The keywords related to combinations are 'COMBINATIONS-G' and 'PAR./DATA LIST of COMB-G'. The list is closed by a blank line (Table 5.48). Each line of the list contains the parameters, the data and the reference. The parameters are the name, the two shifts, the type of set of man-machine system and the number of gangs involved in this combination. The type of set has to be the same as is used for the gangs involved. The data only concern the types of gang involved and that type refers to the references GANG [.]; so no gang can be included more than once in a combination. The type of gang may not exceed the available types, i.e. the minimum of the parameter (Table 5.36 fifth parameter, 11) and the length of the list (Table 5.47). One can define a combination that include gangs requiring more elements of men or machines than available; such a combination will never be used. The comment shows the reference to this combination by COMB-G [.] or by the man-machine system NM-S [GANGS + .]. The program can also generate all the possible combinations from the gangs contained in a set; in that case the list of Table 5.48 is then skipped.

The following main keyword 'OPERATIONS MATERIAL' and the intermediate keyword 'DATA LIST of OPR-MAT' precede the list of data and comment for objects of operations processing and producing materials (Table 5.52). No parameters are needed because default values are used (Table 5.32). The data concern:

- a type of gang; i.e. a reference to GANG [.] and that type must exist;
- the number of materials processed (less than four);
- for each material processed, its type (a reference to MAT-PROC [.]) and three related processing conditions that tell which conditions are appropriate for this operation; the type must appear in the materials file and not exceed the number of processed materials (6, Table 5.36, Parameter 10) and the conditions must appear in that type of material; the type of material and the conditions (repeat an appropriate condition to meet three data elements) of the processed materials are given on one line per material under the appropriate headings;
- the number of materials produced simultaneously;
- the number of produceable materials (less than four); this includes the materials that are produced always and a number of materials from which only one is produced (this number of alternative materials is related to a processing condition in a processed material);.for example, for cereal harvesting, straw and grain are produced; moisture content determines whether it is wet or dry grain; so two materials are produced at a given

moment and three are produceable: straw, wet grain and dry grain;
- the types of materials, first those produced always and subsequently the alternative materials beginning with the lowest type; a type must not exceed the total number of materials (9, Table 5.36, Parameter 11).

The comment shows the references OPRTN [.] and OPR-MAT [.].

The following keywords 'OPERATIONS SERVICE-REPAIR' and 'DATA LIST of OPR-S-R' precede the list of service-repair operations (Table 5.54). The data concern only a type of gang (a reference to GANG [.]) that must be contained in the list (Table 5.47) and in the number of gangs (11, Table 5.36, Parameter 5).

Finally, if a gang is not related to an operation, such a gang will never be selected and no difficulties are expected from such a situation. The number of operations, however, may never exceed the number of gangs.

The materials file contains only two main keywords (Table 5.33) 'INITIAL MATERIALS' and 'INTERMEDIATE MATERIALS'; both categories of materials can be processed and produced (consumed and delivered) but only the initial materials have fields at the begin of a season. Each material has six intermediate keywords and one optional (Table 5.56):
- 'PARAMETERS MAT.', preceding the parameters;
- 'DATA MAT.1', preceding some general data;
- 'DATA MAT.PRC.', preceding the processing conditions;
- 'DATA MAT.3', preceding some data related to urgency and timeliness losses;
- 'DATA MAT.TML.', preceding the timeliness functions;
- 'DATA MAT.5', preceding data about the workable time;
- 'DATA MAT.WRKBL.', optional, used only when the parameter for the number of workable time conditions is more than zero; it precedes the workable time for a number of days and conditions.

The parameters concern:
- a name of 24 characters;
- a type of shift over periods, necessary (+);
- a type of shift within a week; usually zero; only in specific situations related to the shifts of men to record data per shift (+);
- a type of shift for urgency calculations; though useful, urgency will be calculated without it when a period ends because of the shift over periods (+);

  (+) the type may not be more than the number given in the environment file;
- a number of processing conditions, at least one;
- a number of timeliness functions, at least one;
- a number of dates for the timeliness functions, at least two;
- a number of workable time conditions, usually zero;
- a number of days with workable times, only relevant and at least zero if

the conditions exceed zero.

The general data concern:

- an expected capacity of processing this material (some weighted sum or the average of the processing capacities of the gangs involved); it is only used for the urgency calculation;
- an expected price of material without any timeliness loss;
- a field delivered becomes part of a queue of fields and is queued at the tail with 'tail' (small letters!) and otherwise at the head, for instance, with input of 'head' (or even with 'TAIL' or 'tai ');
- a maximum;
- a fraction of the maximum quantity that must be achieved before delivery may be continued.

For each condition, the processing conditions concern:

- a type of material that can be produced, such as wet or dry grain;
- a lower and upper limit of a property of this material; the property is given in the weather data file; the limits of several conditions may not overlap, i.e. the conditions must be mutually exclusive.

The conditions with a lower type of material are shown first.

The general timeliness data concern:

- a disurgency of delivery; if used, then 'yes' (small letters!) otherwise, for instance, 'no' it is not used (even with 'YES' or 'y');
- a timeliness loss is derived from the timeliness function with 1.0; otherwise 0.0 is required, indicating that no timeliness loss is considered or is calculated in a specific way (only in extended models);
- an area to calculate a minimum urgency; it can also be used to handle some constant level.

The timeliness functions concern:

- a list of relative dates (increasing integers including zero); negative dates are appropriate if processing may start before yield is optimum;
- a list of fractions of the recoverable value (Fig. 2.1), with 1.0 at date zero and dates after zero when the optimum continues for days.

The workability data concern:

- a long-term estimate;
- a conversion factor to justify the long term estimate to other conditions and to tune the urgencies for creating a schedule with optimum costs;
- a list of expected fractions a material will be processed within the workable time; the number of fractions has to be at least equal to the number of periods given in the shift over periods (Table 5.39).

All these workability data are only used to calculate the urgency of materials. These urgencies are transformed by the program to an urgency of gangs; costs (overtime costs of men and fuel costs of a drier if given) are subtracted from the urgency of the gang and may cause a negative urgency and no work! The expected workable durations for a number of days and conditions (according to the parameters) concern the duration per day (zero or positive). All the materials processed follow this pattern; the final materials (not processed, only produced) have only default parameters (Table 5.33).

The file of initial fields contains the main keyword 'FIELD PROPERTIES' (Table 5.61 and 5.65). This file is used each season to provide the initial materials with the appropriate fields. For each initial material an intermediate keyword 'DATA LIST of FLD.' is used followed by ':' and the number of initial fields (Table 4.49). For each field, a subsequent line gives:
- the area (Tables 4.50 and 4.51);
- the date when the material on this field will be processable (at 24:00);
- the date when the material on this field achieves its maximum yield (at 24:00); logically this date equals or exceeds the processable date (Fig. 2.1);
- a name; at most 30 characters.

The weather data file (Table 5.5) contains:
- a date (month and day);
- a clocktime until which the properties remain valid;
- a type of day (1 = Monday, etc.); this type is also used in the shift within a week to find the appropriate cost categories on such a day; the type given on the first line is used to find the type of day on 1 Jan. and that in its turn is used to record in output the current type of day;
- the amount of rain since the preceding 'moment' (i.e. preceding line or record);
- one property for each material processed.

The date on each line is the date at 00:00; the type of day, however, is related to the clocktime (beat this in mind when you aggregate these data over seven days and the clocktime reaches 168:00 and the date remains the same).

The above description completes the Chapters 4 'Base model' and 5 'Experimental frame' that give the complete simulation program. The entire program is not printed; a program copy and files are available on a floppy, Annex A. The following chapter verifies parts of the program and Chapter 7 'Extensions' describes extensions to the simulation program and ideas about modelling other scheduling problems.

6 VERIFICATION AND VALIDATION

The correctness of program and model are now considered. A process with three stages is used to establish correctness:

(i) the first stage, verification of program, is based on the internal structure of the program, its consistency, logic and plausability (rationalism);

(ii) the second stage, validation of model, is based on the acceptance of the behaviour of the model and related to the performance of forecasting and its sensitivity to parameters (empirism); the model is structurally valid (Zeigler, 1976) when it produces observed behaviour and reflects the way in which the real system operates;

(iii) the third stage, use of model, is based on the acceptance as a tool in practical management or in research (pragmatism).

The last stage is not part of this publication. The reader has to decide when to accept the model in some environment (research, extension, teaching etc.) on the basis of stages one and two.

6.1 Verification

6.1.1 Decision tables and system matrices (SMX)

The structure of a program is supported in SIMULA by using indentation of blocks (begin, end), short procedures with clear tasks, classes and comment. This, however, does not cover everything in dynamic simulation with activating and deactivating procedures in processes such as (re)activate, passivate, hold, wait and cancel. These procedures may cause problems and loss of sight on the program execution due to the interruption of the sequential flow of executing statements. Two methods are used to check the logic of the program: decision tables and system matrices.

Decision tables are useful to show the meaning of a complex 'if' statement. A simple example from SHIFT-CHNGE- of men and machines (Table 4.54) where the statement reads:

```
STATE-NEXT: = if not AVLB-COMP then DOWN
                                else if STATE = RUN then RUN
                                                        else PASSIVE;
```

Table 6.1 shows the decision table with all the possible situations given in comlumns. The situations Rff, Pff and Dtt are not shown because they contain a contradiction, which make them impossible.

Table 6.1 Decision table for the next state of men and machine.

Variables	Situation		
STATE	R R	P P	D D
AVLB_COMP was	t t	t t	f f
AVLB_COMP becomes	t f	t f	t f
	Result		
STATE_NEXT results as	R D	P D	P D

(R = RUN, P = PASSIVE, D = DOWN, t = true, f = false)

Exercise: Can you find the contradictions?

One can check if the decision table is correct and if the statement has the same effect. The decision table shows that the old value of the variable AVLB-COMP has no influence on the result and therefore only the current value is used in the statement. Other examples of controlling the local programming logic have already been mentioned in chapters 4 'Base model' and 5 'Experimental frame'.

System matrices can also be used to check the programming logic on a broader scale (local and dynamic). First the power of system matrices (SMX) to represent the logic in a procedure is illustrated and this example is used to explain SMX as developed by Jaederlund.

Table 6.2 shows a system matrix derived from START-OPRTN- (Table 4.86) and is used to illustrate the method. The matrix exists of four qua-

Table 6.2 System matrix of START-OPRTN-.

```
                             Variables               Conditions
                             ---------               ----------
                           *STATE                  *STATE=PASSIVE
                             *STATE_NEXT             *STATE<>PASSIVE
                             . *RUN_PHASE            . *RUN_PHASE=INACTIVE
                             .   *GANG_6.STATE       .   *RUN_PHASE<>INACTIVE
                             .     *operation scheduled    *PROCESSING
                             .     . *MAT_PROC[i].PROCESSING *not PROCESSING
                             .     .                 .      .
Execution functions          .     .                 .      .
------------------           .     .              :  .      .
  GANG_6.START_WORK_6      U U   U .              :  . 1    .
  MAT_PROC[i].ATTR_FLD_M_    .     .              :  .      . 1
  MAT_PROC[i].ATTR_INTG_F_   .     .              :  .      . 2
  activate this OPRTN_MATRL  .     U              :  .      1 3
    at Time                  .     .              :  .      .
                             .     .              :  .      .
                         ---------------------------------------------
                             .     .              :  .      .
Control functions            .     .              :  .      .
-----------------            .     .              :  .      .
  Entry                      .     .              :0 .      .
    STATE=PASSIVE?         I .     .              :10 0     .
    RUN_PHASE=INACTIVE?      . I   .              :1 . Y  N .
    MAT_PROC[i].PROCESSING?  .     . I            :  . 2    0 0
                             .     .              :  .      .
  Exit                                            :  1    1 2 4
```

drants (I – IV); four catalogs are added and contain the attributes (A), the conditions (B), the execution or process functions (C) and the control or logic functions (D) according to the following scheme.

	A	B
C	I	II
D	IV	III

The upper left part of the matrix, Quadrant I, shows (Table 6.2) that the execution functions use attributes (I = input) and change others (U = update, O = output). The lower left part, Quadrant IV, shows the input (I) for the control functions. The lower right part, Quadrant III, shows the logic, the relation between a control function and a condition. The right part, Quadrants II and III, can be read as follows: start at line 'Entry', go to the column with '0' and then to the next number '1' in that column on line 'STATE = PASSIVE?', now one is forced to select the prevailing condition from the columns with '0'; if STATE <> PASSIVE then the next number in that column is on line 'Exit', otherwise STATE = PASSIVE and the next number in that column is on line 'RUN-PHASE = INACTIVE?'; the selection of that condition is required in columns with '0' or in this case with 'Y' and 'N' (representing yes and no); if the answer is 'N' then the next number is on line 'Exit', otherwise the function START-WORK-G of GANG-G is executed as indicated by the next number '1' in column 'Y', the STATE of the gang and of the operation are updated (U) as shown in Quadrant I, and the next number '2' is on line 'MAT-PROC[i].PROCESSING?' where a selection is required; if PROCESSING of that material is true then the operation is scheduled by 'activate this OPRTN-MATRL at Time' and the next number '2' is on line 'Exit'; if PROCESSING of that material is not the case then the same sequence of execution is preceded by updating some attributes of the material. Before the scheduling procedure 'activate ...' is called, it is known from the above that the state of the operation is STATE = RUN and RUN-PHASE = INACTIVE; this situation continues also after exit of START-OPRTN- and can be used as a condition when the dynamic section of the scheduled operation is executed. 'T' and 'F' as true and false can replace 'Y' and 'N'. You may use your own symbols if needed, for instance, 'Z' for making a variable zero. The same method could be used to develop a system matrix of STOP-OPRTN- (Table 4.92), but in this case we can already learn from the program directly that before the scheduling procedure 'reactivate ...' is called the situation is STATE <> RUN (due to STOP-WORK-G and STATE-CH-OP-) and RUN-PHASE is unknown. 'Reactivate ... Time prior' forces an immediate execution of the dynamic section of the operation and interrupts the calling process, the dynamic part of process DECISION. So at the exit of STOP-OPRTN- the dynamic section of operation is already executed and the situation updated.

There is no need in general to transform procedures described in Chapters 4 and 5 into decision tables or system matrices. A complex procedure in Chapter 5 that derives the applicability of combinations of man-machine systems, however, can be explained with the help of system matrices. The aim of procedure APPLCB-MM-SYSTMS (Table 5.10) is to find out if all gangs of the system are applicable already or if not and the reason is that some material processed is not available then find out whether that material will be delivered by other gangs selected to work. Table 6.3 shows a simple system matrix where sentences shown under 'functions' contain a lot of detail that is still hidden (for instance, no attributes mentioned) and can be explained in submatrices and sub-submatrices etc. If one understands the system matrix, then it is not too difficult to check whether the meaning of the sentences is reflected in the statements of the procedure.

The verification of a program with dynamic elements such as activating and deactivating of processes requires special attention. The convention was adopted to use the procedures Passivate, Hold and Wait only in the dy-

Table 6.3 System matrix for deriving the applicability of man-machine systems.

```
------------------------------------------------------------------------------------------------------
           APPLCB_MM_SYSTMS                    Attributes                 Conditions
           ----------------                    ----------                 ----------

Derive the applicability of a              *  APPLY_M_DLVR true if    *  delivering is an objection
man-machine system with some                  delivering of material   *  other objections
gangs.                                        is an objection          . *  no objections
                                            *  APPLCBL true if no objections
                                            . *  APPLICABLE              .    *
                                            .                            .      *
                                            .                            .      . *  material needed
                                            .                            .      .    is not produced
                                            .                      :     .      .
           Process functions                .                      :     .      .
           -----------------                .                      :     .      .
                                            .                      :     .      .
Save materials produced in                  .                      :     .    1 .
    gangs if the gang is                    .                      :     .      .
    applicable.                             .                      :     .      .
For the gang that is not applicable         .                      :     .    2 .
    find if the material processed          .                      :     .      .
    and not available is produced           .                      :     .      .
    in gangs                                .                      :     .      .
APPLICABLE: = false                         . F                    :     1      . 1
APPLICABLE: = true                          . T                    :     . 1    1
                                          ------------------------------------------------------------
                                            .                      :     .      .
           Control functions                .                      :     .      .
           -----------------              . .                      :     .      .
                                            .                      :     .      .
Entry                                       .                      :0    .      .
    APPLY_M_DLVR?                         I .                      :1 T F       .
    APPLCBL?                                I                      :  1 . T F   .
    Material needed is produced?            .                      :     .    3 Y N
Exit                                        .                      :     2 2    2 2
------------------------------------------------------------------------------------------------------
```

namic section of a Process because they have effect on the current object. Using them in the dynamic section ensures that the current object is the same Process to which the section belongs. The procedures (Re)Activate and Cancel require an explicit reference to an object and can be safely used in procedures. Because of different deactivation points in a dynamic section and different sources of activating an object, it is not self-evident where the execution will continue in a specific case. In general such a situation leads to the introduction of a state variable in the dynamic section. That variable can be checked in procedures for proper execution of its statements. Such a situation is shown in Table 4.93, where RUN-PHASE is used in the dynamic section of the key-process: operation of material. It is hardly possible to gain insight into the relationships between the procedures shown in Tables 4.86 – 4.92 and the dynamic section. To gain more confidence system matrices are used, which have been developed to describe several related processes. The method is a tool to draw attention to gaps in the program.

In the following sections the logic of the program when 'events' occur is checked. Because the main aim is to show the effects of an event, the procedures involved are listed and the execution and control functions are mixed in the system matrix as in the program. The matrix is reduced to two quadrants when the execution sequence under conditions is replaced by a top-down sequence (Table 6.4). The functions are indented if they are called from a preceding function. The range of a control function ends at an inserted blank line. With these system matrices the consequences of an event can be followed and it can be concluded whether it agrees with the model or not. In the latter case an attempt can be made to fill the gap between program and model.

6.1.2 Events from materials and weather

The first event arises from the environment, the weather and it effects, for instance, the moisture content of materials, the workability and processing conditions. Table 6.4 lists the procedures (called if certain conditions occur) and the relevant consequences on attributes of materials and other components such as decision and operation. The component WEATHER (Table 5.4) calls mainly two procedures: ATTR-UPDT-M- and Reactivate. Within ATTR-UPDT-M- of a specific material the processing property, processing conditions and the workability are updated; the state of the material is changed accordingly with consequences for decision. If the material that can be produced from the processed material changes (for instance, grain with a moisture content of 21% can only result in wet grain and with 16% results in dry grain) then the consequences for the operations are inserted. After the name of a procedure or attribute the component involved (material, operation, decision) is sometimes mentioned to prevent ambiguity, for instance, 'material produced until now' means that DLVRY-STOP is called in the

Table 6.4 Event in dynamic section of WEATHER.

```
                                                          Attributes
                                                          **********
Originated by change of attributes and
processing conditions of materials. The             * PROC_PROPRTY       (material
materials produced (wet grain to                    . * M_PRC_CNDTN [i] (
dry grain, due to moisture content) or              .   * WORKBL          (
the state of the material processed are updated.    .     * STATE         (         )
                                                    .     . * MAT_PASS          (decision
                                                    .     .   * MAT_DOWN        (
                                                    .     .     * PRC_CND       (         )
                                                    .     .     . * STATE     (operation
                                                    .     .     .              processing)
                                                    .     .     .
ATTR_UPDT_M_      (of material)                     U U   .     .
   CH_WORKBL                                        .   U .     .
   STATE_CH_MAT                                     .     U     .
      (if state changed from DOWN to PASSIVE        .     I     .
       then:)                                       .     .     .
      DECIDE.MAT_PASS:= true                        .     . T   .
      Activate DECIDE at Time                       .     .     .
                                                    .     .     .
      (if state changed from RUN to DOWN then:)     .     I     .
      DECIDE.MAT_DOWN:= true                        .     .   T .
      Activate DECIDE at Time                       .     .     .
                                                    .     .     .
      (if state was or becomes DOWN then:)          .     I     .
      STATE_CH_OP_   (operations processing)        .     .     . U
                                                    .     .     .
   (if materials produced change then:)             .     .     .
   PROCESS_MAT                                      .     .     .
   MAT_PROD_CHNG (of operations processing mat.)    .     .     .
      DLVRY_STOP    (material produced until now)   .     .     .
      SUPPLY_EXPCT  (    "        "    afterwards)  .     .     .
   STATE_CH_OP_ (operations processing mat.)        .     .     . U
      DECIDE.PRC_CND:= true                         .     .     T
      Activate DECIDE at Time                       .     .     .
                                                    .     .     .
Reactivate this WEATHER at ...                      .     .     .
```

material produced until now, which was called by MAT-PROD-CHNG of the operation.

The second event (Table 6.5) is related to fields. It occurs when a field is consumed entirely and PROCESS-MAT (Table 4.34) is called in the dynamic section (Table 4.33). Within this procedure a distinction is made between a part related to the delivery and consumption of materials and a part related to the continuation of processing if an appropriate field is available or delivery of the material is expected (STATE <> DOWN). Table 4.34 adds some additional conditions such as (AVLBL or DLVRY-SETUP) and PROCESSING only for safety reasons and better understanding; the conditions are superfluous because the first one is implied by STATE <> DOWN and the second was already true at the very beginning of the procedure and should not be changed in the meantime.

The third event occurs when the maximum quantity of a material is achieved, for instance, when a grain drier is filled with wet grain (Table 6.6). The procedure DELIVER-MAT (Table 4.34) updates the delivery and consumption of materials and warns decision that a maximum quantity of a material is true. Table 6.6 also contains the complement: the fourth

Table 6.5 Event in dynamic section of MATERIAL; field is processed.

```
                                                     Attributes
                                                     **********
Originated by terminating the consumption of a
field or even of all the fields processable.     * QUANT_AVLBL       (material
                                                 . * AVLBL           (
                                                 .   * STATE         (consumed)
                                                 .     * MAT_DOWN      (decision)
                                                 .     . * STATE       (operation)
                                                 .     .   * EVTM_FLD  (material consumed)
                                                 .     .     * EVTM_MAX(material delivered)
                                                 .     .     . * MAT_PRD_RORD (operation
                                                 .     .     .                 delivering)
(If EVTM_FLD = Time and PROCESSING then:)        .     .     .
PROCESS_MAT                                      .     .     .
   UPDAT_QNT   (operation consuming material)    .     .     .
      Activate this OPRTN_MATRL at Time prior    .     .     .
         QNT_TRANSFER                            .     .     .
            DELIVERY_M_  (material produced      .     .     .
            CONSUMPTN_M_ (material processed)    U U   .     .
               STATE_CH_MAT                      .   U .     .
                  (if STATE = DOWN then:)        .   I .     .
                  DECIDE.MAT_DOWN:= true         .     T     .
                  Activate DECIDE at Time        .     .     .
                  STATE_CH_OP_                   .     . D   .
                                                 .     .     .
      Passivate (of OPRTN_ MATRL)                .     .     .
   (if STATE <>DOWN and                          .     .     .
   (not SUPPLY_NDD or DELIVERING) then:)         .   I .     .
   CONTINUE (operation consuming material)       .     .     .
      Activate this OPRTN_MATRL at Time          .     .     .
         GO_ON_                                  .     .     .
            GO_UNTIL_MP  (material processed)    .     .   U .
            ACCEPT_UNTIL (material produced)     .     .     U
      Passivate (of OPRTN_MATRL)                 .     .     .
                                                 .     .     .
   (if STATE <>DOWN and SUPPLY_NDD and           .     .     .
    not DELIVERING then:)                        .     .     .
   MAT_PRD_RORD:= true (operation delivering mat.).    .     . T
                                                 .     .     .
Reactivate this MATERIAL at ... else Passivate   .     .     .
```

event occurs when the quantity is decreased so far (for instance, to half of the maximum) that delivery can be continued; a decision is required.

The last event occurs at the moment when the calculation of the urgency is requested (Table 4.108) from a shift defining such moments in a day or period (Table 6.7). The urgency is updated according to the current situation after delivering and processing the fields involved. When the state of a gang is passive and the urgency positive, then a decision is requested.

6.1.3 *Events related to men and machinery*

The first two events related to men and machinery arise from SHIFT-WEEK (Table 4.105) and SHIFT-PERD (Table 4.103), respectively. The last one controls the availability in periods of a year and the former one controls the availability and costs pattern on days in a week. Both components call SHIFT-CHNGE- in all the objects with that shift pattern. Table 6.8 illustrates the changes in the state of men or machines and the activation of an object UPDT-MM-SYS that updates man-machine systems accordingly

Table 6.6 Event in dynamic section of MATERIAL; maximum achieved or obsolete.

```
                                                          Attributes
                                                          ==========
Originated by achieving or exceeding the maximum
quantity of material or termination of such a          * QUANT_AVLBL      (material
situation.                                             . * DLVR_ALLWD     (delivered)
                                                       .   * MAT_MAX_QNT   (decision
                                                       .     * MAT_DLV_OK  (
                                                       .     .
(i)                                                    .     .
(If EVTM_MAX = Time and DELIVERING then:)              .     .
DELIVER_MAT                                            .     .
   UPDAT_QNT  (operation delivering material)          .     .
      Activate this OPRTN_MATRL at Time prior          .     .
         QNT_TRANSFER                                  .     .
            DELIVERY_M_  (material produced)           U     .
               MAX_QUANT_M_                            . F   .
            CONSUMPTN_M_ (material processed)          .     .
      Passivate (of OPRTN_MATRL)                       .     .
   (if DLVR_ALLWD = false then:)                       . I   .
   DECIDE.MAT_MAX_QNT: = true                          .   T .
   Activate DECIDE at Time                             .     .
                                                       .     .
Reactivate this MATERIAL at ... else Passivate         .     .
                                                       .     .
(ii)                                                   .     .
(If EVTM_DLVR_OK = Time then:)                         .     .
TERMNT_NO_DLVR                                         .     .
   MAX_QUANT_M_                                        . T   .
   (if DLVR_ALLWD then:)                               .     .
      DECIDE.MAT_DLV_OK: = true                        .     T
      Activate DECIDE at Time                          .     .
Reactivate this MATERIAL at .... else Passivate        .     .
```

Table 6.7 Event in dynamic section of SHIFT-URG.

```
                                                          Attributes
                                                          ==========
Originated by a request to update the urgency of
materials.                                             * URGENCY_PROC      (material
                                                       . * DISURGNC_DEL    (          )
                                                       .   * URGNC_CALC    (decision)
                                                       .     * TML_LOSS_EXP (material)
                                                       .     . * URGENCY   (gang)
URG_MAT_PRC_                                           .     .
   DELIVER_MAT                                         .     .
   PROCESS_MAT                                         .     .
   TML_LOSS_EXP_                                       .     U
URG_MAT_EXT_                                           U U   .
DECIDE.URG_GANGS_                                      .     . U
   (if a gang passive and urgency positive then:)      .     .
   URGNC_CALC: = true                                  .   T .
   Activate DECIDE at Time                             .     .
                                                       .     .
Reactivate this SHIFT_URG at ...                       .     .
```

by checking whether sufficient men and machines are available to perform their task and by changing the costs (for instance, overtime). These patterns are developed to control the men and machines; they can be used also for gangs and materials, for instance, to control the availability of contract work.

The third event occurs when a machine failure occurs during work (Table 4.61). Table 6.9 shows that repair is needed and perhaps a service is

Table 6.8 Event in dynamic section of SHIFT-WEEK or SHIFT-PERD.

```
                                                              Attributes
                                                              **********
Originated by a change in availability or costs
within a week or availability in a period within  * AVLB_COMP      (component
a year for men and machines.                      . * STATE        (         )
                                                  .   * LE_NMB[i] number available
                                                  .     of men, machines of category i
                                                  .     * AVL_LE      (man-machine
                                                  .     . * COSTS_H  (
                                                  .     .   * STATE  (     system)
                                                  .     .     * SHIFT_CH  (decision)
                                                  .     .     .
                                                  .     .     .
   (If costs or availability changes then:)       .     .     .
   SHIFT_CHNGE_       (of component)              U     .     .
      STATE_CH_LE_      (for men, machines)       . U U .     .
   (re)activate UPDT_MM_SYS at Time (prior)       .     .     .
      AVLBLTY_TEST          (man-                 .     U     .
         STATE_CH_MMS       (machine              .     .   U .
      COST_CHANGE           (system)              .     . U   .
                                                  .     .     .
   DECIDE.SHIFT_CH:= true                         .     .     T
   Activate DECIDE at Time                        .     .     .
Reactivate this SHIFT_WEEK /SHIFT_PERD at ...     .     .     .
```

Table 6.9 Event in dynamic section of EQUIPMENT due to a failure.

```
                                                              Attributes
                                                              **********
Originated by a breakdown or failure of the
machine during work                               * RPR_ND
                                                  . * RPR_DUR
                                                  .   * SRVC_ND
                                                  .     * SRVC_DUR
                                                  .     . * STATE
                                                  .     .   * LE_NMB[i] number of available
                                                  .     .     machines of category i
                                                  .     .     * MACHN_FLR (decision)
                                                  .     .     .
                                                  .     .     .
   (If FLR_AT_RUNT = TIME_USED [RUN] and          .     .     .
    STATE = RUN then:)                            .     .     .
   RPR_ND: = true                                 T U   .     .
   SERVICE_NEED                                   .   U U     .
   STATE_CH_LE_                                   .     . U U .
      Activate UPDT_MM_SYS at Time                .     .     .
   DECIDE.MACHN_FLR: = true                       .     .     T
   Activate DECIDE at Time                        .     .     .
                                                  .     .     .
Passivate (of EQUIPMENT)                          .     .     .
```

planned simultaneously. The state of the machine and the man-machine systems involved are changed and decision is activated. Table 6.10 shows the procedures involved to update the attributes of a man-machine system and its related operation by the dynamic section of UPDT-MAN-MCH (Table 4.75), due to such a failure. The complementary event of failure occurs when a service repair operation completes the repair and service, see next section.

Table 6.10 Event in dynamic section of UPDT-MAN-MCH.

```
                                                  Attributes
                                                  **********
Originated by shifts of men, a machine failure
or service need and by repair or service of a   * STATE   (man_machine system)
machine is completed.                           . * STATE (operation)
                                                .
                                                .
STATE_COSTS                                     .
   AVLBLTY_TEST    (man_                        .
      STATE_CH_MMS (machine systems)            U
         STATE_CH_OP_    (operation related)    . U
            (if state was RUN or DOWN and       . I
             changed then:)                     .
            Activate DECIDE at Time             .
                                                .
   COST_CHANGE                                  .
Passivate (of UPDT_MAN_MCH)                     .
```

6.1.4 *Events related to operations*

Before the events in operations are discussed, the heart of the scheduling system, the dynamic aspects of the operation, is considered. Table 6.11 illustrates the connection of the dynamic section and the procedures of an operation processing and producing materials. Only some attributes are given, others can be added if necessary. Conditions are not mentioned in the catalog when they are clear from the answer on the control function; 'Y' and 'N' in such columns suffices. For references to the columns the letters a,b,c,...,r are used. The system matrix closely resembles the program (Table 4.93) with separation of the control and execution functions. On behalf of the system matrix the control function 'RUN-PHASE?' was added, which is not needed in the dynamic section of the program where just before each possible deactivation (by Hold, Passivate, Reactivate or Cancel) RUN-PHASE is updated and clearly known. RUN-PHASE is used in procedures such as GO-ON- (Table 4.88) to check the phase of the process. This system matrix draws my attention to, for instance, where QUANT is made zero or updated and where M.DLVRY-SETUP, M.PRCSS-SETUP, M.DELIVERING and M.PROCESSING becomes true or false. M.xx refers to an attribute xx of a material. M.DLVRY-SETUP is set to true at the very beginning of an operation to allow the start of another operation processing that delivered material at the same moment; M.DELIVERING and M.PROCESSING are set to true in GO-ON- after the set-up of the operation. The number of operations processing/delivering a material PRC-OPRS/DLR-OPRS is updated in GO-AHEAD and TERMNT. The capacity of the operations producing/processing a material M.CAP-OPRS-DLV/-PRC is updated in GO-ON- and in TERMNT (termination). The updating in TERMNT however is only required when GO-ON- is passed and therefore we meet in DLVRY-STOP (Table 4.22) and STOP-PRCSNG (Table 4.32) of a material a condition that RUN-PHASE is greater or equal to BUSY (i.e. BUSY or GO-ON-WAIT). The system matrix was extended with a list of procedures

Table 6.11 System matrix of the dynamic section of an operation of materials.

```
                         Variables                Conditions
                         ----------               ----------

                       *STATE                     *RUN_PHASE=INACTIVE
                         *RUN_PHASE                 *       =SETUP_WAIT
                         . *QUANT                   . *      =MAT_WAIT
                         .   *scheduled             .   *     =BUSY
                         .     *                    .     *    =GO_ON_WAIT
                         .     . *M.DLVRY_SETUP/M.PRCSS_SETUP
                         .     .   *M.PROCESSING/M.DELIVERING
                         .     .     *M.CAP_OPRS_DLV/_PRC
                         .     .     . *M.QUANT_ARRVD/_AVLBL/_PROCSD
                         .     .     .   *M.PRC_OPRS/M.DLR_OPRS
Execution functions      .     .     .       :      .     .
-------------------      .     .     .       :      .     .
  GO_AHEAD               . Z   . T   .   U   :      .     . 1   .
  RUN_PHASE:=SETUP_WAIT  U     .     .       :      .     . 2   .
  Hold                   .   T .     .       :      .     . 3   .
  RUN_PHASE:=MAT_WAIT    U     .     .       :      1     .     .
  WAIT_MAT_PRC           .   U .     .       :      .     .     1
  GO_ON_                 .     .   T U       :      .     .     .   1 .
  RUN_PHASE:=BUSY        U     .     .       :      .     .     .   2 .
  Passivate              .   F .     .       :      .     .     .   3 .
  QNT_TRANSFER           . U   .     . U     :      .   1 .     .     .
  RUN_PHASE:=GO_ON_WAIT  U     .     .       :      .   2 .     .     .
  Passivate              .   F .     .       :      .     .     .     . 1   .
  TERMNT                 .     . F F U   U   :      .     .     .     .     1
  RUN_PHASE:=INACTIVE    U     .     .       :      .     .     .     .     2 1
  Passivate              .   F .     .       :      .     .     .     .     .   1
                         .     .     .       :      .     .     .     .     .
                       ------------------------------------------------------------
Control functions        .     .     .       :      .     .     .     .     .
-----------------        .     .     .       :      .     .     .     .     .
  Entry                  .     .     .       :  0   .     .     .     .     .
    RUN_PHASE?           I     .     .       :  1 0 0 0 0 0     .     .     .
    STATE=RUN?         I .     .     .       :      .     . Y N .     .     .     1
    STATE=RUN?         I .     .     .       :      2     .   1 Y N   .     .     .
    STATE=RUN?(while)  I .     .     .       :      . 1   1     . 1 Y N   1 .     .
    STATE=RUN?         I .     .     .       :      .   3 .     .     . Y N .     .
    STATE<>RUN?        I .     .     .       :      .     .     .     1     Y N   .
    STATE<>RUN?(while) I .     .     .       :    1 .     .     .     .     3 2 Y N
  Exit                   .     .     .       :      .     . 4   2   4 . 2   .   2 .
                         .     .     .       :      .     .     .     .     .     .
    (Column reference:                          a b c d e f g h i j k l m n o p q r)
                       -------------------------------------------------------------
                                                    .     .     .     .     .     .
Dynamic is      See            Situation            .     .     .     .     .     .
activated by    Table    STATE      RUN_PHASE   Sequence of columns
                                                    .     .     .     .     .     .
START_OPRTN_    4.86       RUN     INACTIVE     1 2 .     . 4   .     .     .     3
Hold            4.93       RUN     SETUP_WAIT   1   2     .     3     .     .     .
WAIT_MAT_PRC    4.87       RUN     MAT_WAIT     1   . 2   .     .   3 .     .     .
WAIT_MAT_PRC    4.87       RUN     GO_ON_WAIT   1   .     2     .   3 .     .     .
UPDAT_QNT       4.90       RUN     BUSY         1   .   2 .     .     . 3   .     .
CONTINUE        4.90       RUN     GO_ON_WAIT   1   .     2.    .   3 .     .     .
STOP_OPRTN_     4.92   not RUN     INACTIVE     1 2 .     .     .     .     .   3 .
STOP_OPRTN_     4.92   not RUN     SETUP_WAIT   1   2     .     . 3   4     5   6 .
STOP_OPRTN_     4.92   not RUN     MAT_WAIT     1   . 2   .     .     3     4   5 .
STOP_OPRTN_     4.92   not RUN     BUSY         1   .   2 .     .     4   3 5   6 .
STOP_OPRTN_     4.92   not RUN     GO_ON_WAIT   1   .     2     .     3     4   5 .
```

which activate the dynamic section; after the possible situations of STATE and RUN-PHASE the resulting sequences of columns are given. Now it is possible to check if the flow of program executions caused by an activation is satisfactory or not. The activation from START-OPRTN- finds the situation STATE = RUN and RUN-PHASE = INACTIVE and the columns a, b, r and g satisfy the conditions in that sequence and result in column g in

GO-AHEAD, RUN-PHASE:= SETUP-WAIT, Hold and Exit. An activation after Hold results in RUN-PHASE:= MAT-WAIT, WAIT-MAT-PRC and Exit because the columns a, c and i satisfy the situation with STATE = RUN and RUN-PHASE = SETUP-WAIT. The activation from WAIT-MAT-PRC results in GO-ON-, RUN-PHASE:=BUSY, Passivate and Exit. In both cases where UPDAT-QNT and CONTINUE are called from DELIVER-MAT (Table 4.34) and PROCESS-MAT (Table 4.34) cause the execution of QNT-TRANSFER, RUN-PHASE:=GO-ON-WAIT, Passivate, Exit and GO-ON-, RUN-PHASE:=BUSY, Passivate and Exit, respectively. The activation from STOP-OPRTN- with STATE <> RUN and RUN-PHASE unknown is divided up for each of the possible phases with the following effects:

if RUN-PHASE= ...	then
INACTIVE,	Passivate;
SETUP-WAIT,	RUN-PHASE:=MAT-WAIT, TERMNT, RUN-PHASE:=INACTIVE, Passivate;
MAT-WAIT,	TERMNT, RUN-PHASE:=INACTIVE, Passivate;
BUSY,	QNT-TRANSFER, RUN-PHASE:=GO-ON-WAIT, TERMNT, RUN-PHASE:=INACTIVE, Passivate;
GO-ON-WAIT,	TERMNT, RUN-PHASE:=INACTIVE, Passivate;

and in the system matrix also Exit. These effects are adequate in the different situations and all end in RUN-PHASE=INACTIVE and the operation not scheduled. If a sequence of calculations was inappropriate, more adequate conditions can be used before the activation (as done in START-OPRTN-, Table 4.86) or add specific control functions in the dynamic section (as now is done repeatedly with the test: if STATE = RUN then ...).

The events and their consequences are now considered. To enlarge the scope of the system more procedures are incorporated than shown in Table 6.11; procedures called directly or indirectly in the dynamic section of an operation are added and the consequences of the procedures on the state of the system are mentioned. In Table 6.12 three events are shown due to the start of the operation, the end of set-up and end of waiting for materials, respectively. At the left hand side the control functions and the procedures are listed in sequential order and indented if called from another procedure. Some comment is added to show if a procedure belongs to a material processed or to a material produced. At the right hand side more attributes are listed than was acceptable in Table 6.11 and the change of value due to procedures is shown. Table 6.12 shows that procedure GO-AHEAD calls PRCSNG-EXPCT in the material processed, which changes the variable STATE of that material to RUN (abbreviated by R). In the materials produced SUPPLY-EXPCT is called, which makes the variables DLVRY-SE-

Table 6.12 Three events in dynamic section of OPRTN-MATRL; start (a), set-up ended (b) and wait for materials ended (c).

```
                                                            Attributes
                                                            ==========
Originated by (a) START_OPRTN_, (b) end of setup
or (c) end of waiting for materials                 * RUN_PHASE         (operation
                                                    . * OPRTN.STATE   (              )
R = RUN                                             .   * STATE                 (material
                                                    .     * PROCESSING          (
                                                    .     . * EVTM_FLD          (consumed)
                                                    .     .   *  DLVRY_SETUP    (material
                                                    .     .      * AVLBL        (
                                                    .     .      . * DELIVERING (
                                                    .     .      .   * EVTM_MAX (delivered)
                                                    .     .      .     * EVTM_MAX  (material
                                                    .     .      .     . * EVTM_DLVR_OK
                                                    .     .      .     .           (consumed)
                                                    .     .      .     .   * STATE (material
                                                    .     .      .     .           (delivered)
                                                    .     .      .     .     * OPRTN.STATE
                                                    .     .      .     .(operation processing
                                                    .     .      .     .     .  (the material
                                                    .     .      .     .     .    (delivered)
(a)                                                 .     .      .     .     .
GO_AHAED                                            .     .      .     .     .
   PRCSNG_EXPCT     (for materials processed)       .  R  .      .     .     .
   SUPPLY_EXPCT     (for materials produced)        .     .   T  T     .   U U
RUN_PHASE:= SETUP_WAIT                              U     .      .     .     .
Hold (setup duration)                               .     .      .     .     .
                                                    .     .      .     .     .
(b)                                                 .     .      .     .     .
RUN_PHASE:= MAT_WAIT                                U     .      .     .     .
WAIT_MAT_PRC                                        .     .      .     .     .
   Cancel (this OPRTN_MATRL)                        .     .      .     .     .
   (wait until on next call, all materials          .     .      .     .     .
    processed are available then:)                  .     .      .     .     .
   Reactivate this OPRTN_MATRL at Time              .     .      .     .     .
                                                    .     .      .     .     .
(c)                                                 .     .      .     .     .
RUN_PHASE:= PREPARED                                U     .      .     .     .
GO_ON_                                              .     .      .     .     .
   START_PRCSNG     (material                       .     T      .     .     .
      ACCEPT_UNTIL (processed)                      .     .      .     U     .
   DLVRY_STARTD     (material produced)             .     .      . T   .     .
      (if STATE = RUN and not PROCESSING then:)     .     .      .     .     .
      WAIT_MAT_PRC (processing delivered material). .     .      .     .     .
                                                    .     .      .     .     .
   (if MAT_PRD_RQRD then:)                          .     .      .     .     .
   DLVRY_CONTND     (material produced)             .     .      .     .     .
      (if STATE = RUN and SUPPLY_NDD then:)         .     .      .     .     .
      WAIT_MAT_PRC (processing delivered            .     .      .     .     .
                    material; again)                .     .      .     .     .
                                                    .     .      .     .     .
   GO_UNTIL_MP     (material processed)             .     . U    .     .     .
      (if not DLVR_ALLWD then:)                     .     .      .     .     .
      NO_DLVRNG_UNTIL                               .     .      .     . U   .
                                                    .     .      .     .     .
   ACCEPT_UNTIL   (material produced)               .     .      .   U .     .
RUN_PHASE:= BUSY                                    U     .      .     .     .
Passivate (of OPRTN_MATRL)                          .     .      .     .     .
```

TUP and AVLBL true (if it was not already the case). When STATE of this delivered material becomes PASSIVE (i.e. it can be processed immediately), then the STATE of operations processing this delivered material can also be updated to PASSIVE. The variable RUN-PHASE is updated to SETUP-WAIT and Hold is called to delay the progress of this operation until the set-up duration is passed. At that moment the set-up is completed, the dynamic section is activated again and continues with updating RUN-PHASE to MAT-WAIT and calling WAIT-MAT-PRC which cancels this operation from the list of events and waits until the materials processed become available; (for instance, wet grain drying operation has to wait until the combine-harvesting operation delivers the wet grain) in that case the operation is reactivated, placed on the list of events at that very moment 'Time'. The dynamic section continues with updating RUN-PHASE to PREPARED for operating and calls GO-ON-. This procedure calls START-PRCSNG of the material processed to make PROCESSING true and to update the event time when a maximum can occur by ACCEPT-UNTIL. The next procedure called in GO-ON- is DLVRY-STARTD of the materials produced to set DELIVERING to true. In the case one of the materials produced is waiting for delivery (to start processing) then WAIT-MAT-PRC is called. If MAT-PRD-RQRD is true i.e. waiting for further delivery (to continue the processing) then DLVRY-CONTND is called, which in turn activates the processing operation by calling WAIT-MAT-PRC, for instance, if during the set-up of the baling operation the gathering operation collects all the available bales then gathering has to wait for bales and baling is aware of it for MAT-PRD-RQRD is true (the bales that will be produced are required); when the set-up of baling is completed the delivery of bales starts and the gathering operation continues; it checks in WAIT-MAT-PRC whether waiting for the processed materials i.e. bales is no longer needed and reactivates the gathering operation (as in Table 6.12 after b). The three following procedures called in GO-ON- are GO-UNTIL-MP, NO-DLVRNG-UNTIL and ACCEPT-UNTIL to update event times when the termination of a field will occur, when delivery becomes possible again if delivery is not allowed and when a maximum quantity will be achieved respectively. Finally RUN-PHASE becomes BUSY and Passivate is called, which takes this operation from the list of events.

Tables 6.13 and 6.14 can be interpreted likewise, where the first one updates the quantities processed and delivered and may continue with the operation and the second one stops the operation. It is already known from Table 6.11 that STOP-OPRTN- can be called in different situations i.e. in different phases: during some wait (RUN-PHASE = SETUP-WAIT or = MAT-WAIT or = GO-ON-WAIT) or during the processing itself (RUN-PHASE = BUSY). In the second case the quantity consumed and delivered is transferred and in both cases the dynamic section calls TERMNT, which tells the materials processed and produced that processing stops (PROCESSING becomes false, STATE becomes PASSIVE if the reason to stop processing was not from that material) and delivery stops (DELIVERING and DLVRY-SETUP become false and STATE is updated), respectively. Finally RUN-PHASE becomes INACTIVE and the operation is taken from the list of events by calling Passivate.

Table 6.13 Event in dynamic section of OPRTN-MATRL; processing and delivering.

```
                                                       Attributes
                                                       **********
Originated by materials processed or produced
by this operation due to consuming the field  * RUN_PHASE   (operation
completely, producing the maximum quantity or . * STATE     (of
a request to update the quantities processed  .   * QUANT   (material)
or produced.                                  .     * QUANT_ARRVD     (material
                                              .     . * QUANT_AVLBL   (
                                              .     .   * DLVR_ALLWD  (delivered)
                                              .     .     * EVTM_MAX               (material
                                              .     .     . * QUANT_AVLBL          (
                                              .     .     .   * QUANT_PROCSD       (
                                              .     .     .     * STATE            (
                                              .     .     .     . * EVTM_FLD       (
                                              .     .     .     .   * EVTM_DLVR_OK (consumed)
                                              .     .     .     .
QNT_TRANSFER                                  .   U .     .     .
   DELIVERY_M     (material produced)         .     U U   .     .
      MAX_QUANT_M_                            .     .   U .     .
   CONSUMPTN_M    (material processed)        .     .     . U U U
RUN_PHASE:= GO_ON_WAIT                        U     .     .     .
Passivate (of OPRTN_MATRL)                    .     .     .     .
                                              .     .     .     .
(Continuation of processing follows if        .     .     .     .
   - the state of this operation is run       . I   .     .     .
   - material processed is available and ready.     .     .     I
   - materials produced can be delivered)     .     .   I .     .
GO_ON_                                        .     .     .     .
   GO_UNTIL_MP     (material processed)       .     .     .     . I
      (if not DLVR_ALLWD then:)               .     .     .     .
      NO_DLVRNG_UNTIL                         .     .     .     .   U
                                              .     .     .     .
   ACCEPT_UNTIL    (material produced)        .     .     U     .
RUN_PHASE:= BUSY                              U     .     .     .
Passivate (of OPRTN_MATRL)                    .     .     .     .
```

Table 6.14 Event in dynamic section of OPRTN-MATRL; stopping.

```
                                                       Attributes
                                                       **********
Originated by a decision to call STOP_OPRTN_.
This may interrupt the process (a) during setup   * RUN_PHASE   (operation
or waiting for material or (b) when busy with     . * STATE     (          )
processing.                                       .   * STATE           (material
                                                  .     * PROCESSING    (consumed)
P = PASSIVE                                       .     . * DELIVERING      (material
                                                  .     .   * DLVRY_SETUP   (
                                                  .     .     * STATE       (delivered)
                                                  .     .     .
(a)                                               .     .     .
(If RUN_PHASE = SETUP-WAIT or = MAT_WAIT)         I     .     .
   or = GO_ON_WAIT then:)                         .     .     .
TERMNT                                            .     .     .
   STOP_PRCSNG     (material processed)           .   P F     .
   DLVRY_STOP      (material produced)            .     . F F U
RUN_PHASE:= INACTIVE                              U     .     .
Passivate (of OPRTN_MATRL)                        .     .     .
                                                  .     .     .
(b)                                               .     .     .
(If RUN_PHASE = BUSY then:)                       I     .     .
QNT_TRANSFER                                      .     .     .
   DELIVERY_M      (material produced)            .     .     .
   CONSUMPTN_M     (material processed)           .     .     .
RUN_PHASE:= GO_ON_WAIT                            U     .     .
TERMNT                                            .     .     .
   STOP_PRCSNG     (material processed)           .   P F     .
   DLVRY_STOP      (material produced)            .     . F F U
RUN_PHASE:= INACTIVE                              U     .     .
Passivate (of OPRTN_MATRL)                        .     .     .
```

Table 6.15 Event in dynamic section of SRVC-REPR; machine is repaired.

```
                                                    Attributes
                                                    **********
Originated by finishing a repair of failure
or a service to a machine                       * RUN_PHASE  (operation
                                                . * STATE     (          )
                                                .   * STATE   (machine)
                                                .     * MACHN_OK   (decision)
                                                .     . * LE_NMB[i] number of available
                                                .     .             machines of type i
                                                .     .
                                                .     .
TRANSFER                                        .     .
   SRVC_RPR_DON     (of machine)                .     .
      STATE_CH_LE_                              .   U . U
         Activate UPDT_MM_SYS at Time           .     .
      DECIDE.MACHN_OK: = true                   .     T
      Activate DECIDE at Time                   .     .
   STATE_CH_OP_                                 . U   .
RUN_PHASE:= GO_ON_WAIT                          U     .
                                                .     .
(if state <> RUN then:)                         .     .
RUN_PHASE:= INACTIVE                            U     .
Passivate (of SRVC_REPR)                        .     .
                                                .     .
(if state = RUN then:)                          .     .
GO_ON_      (with next repair or service)       .     .
RUN_PHASE:= BUSY                                U     .
Reactivate this SRVC_REPR at ...                .     .
```

Tables 6.12-6.14 incorporate many procedures and their consequences due to an event; they show an overview of many system-matrices here and there from the program. Table 6.11 shows less detail of the consequences and much more detail on the logic of a specific dynamic part of the program. Both forms are supplementary to each other and can draw attention to an incorrect flow of computations.

To complete the verification of events due to operations, the consequences of an event in a service-repair operation must be shown. Table 6.15 shows that at the moment a service is completed or a failure repaired, TRANSFER is called, which 'transfers' the machine from the queue of machines waiting on a repair or service to its original queue of available machines with the consequences for the man-machine systems containing such a machine. A decision is requested and the state of the operation updated; if a following machine for repair is available to continue this operation then RUN-PHASE becomes BUSY and the next event of the operation is scheduled at the moment the repair or service will be completed; otherwise RUN-PHASE becomes INACTIVE and the operation is removed from the list of events by Passivate.

6.1.5 Remaining events

Some events from the experimental frame have to be checked. The first one concerns the decision making; in this case with the heuristic urgency strategy. Table 6.16 shows the list of procedures called and the effects when a decision is required due to events of men, machines, weather or materials.

Table 6.16 Event in dynamic section of URGENCY-DCSN.

```
                                                          Attributes
                                                          ==========
Originated by several events from shifts of man,
machines, weather and materials.                       * GANG.APPLICABLE
                                                       . * COMB_G.APPLICABLE
R = RUN                                                .   * MM_S_NEW
                                                       .     * MAN/MACH.STATE
                                                       .     . * GANG.STATE
                                                       .     .   * COMB_G.STATE
                                                       .     .     * OPRTN.STATE
                                                       .     .     .
                                                       .     .     .
URG_GANGS_                                             .     .     .
APPLCB_GANGS                                           U     .     .
APPLCB_MM_SYSTMS                                       . U   .     .
SELECT_MM_S                                            . I O .     .
STOP_START_OPRTNS                                      .     .     .
   STOP_WORK_C_, STATE_CH_MMS                          .     .   U .
   START_WORK_C_, STATE_CH_MMS                         .     .   R .
   STOP_OPRTN_                                         .     .     .
      STOP_WORK_G                                      .     .     .
         STOP_ACTVTY, STATE_CH_LE_                     .     U     .
      STATE_CH_MMS, STATE_CH_OP_                       .     . U   U
      Reactivate this OPRTN_MATRL at Time prior        .     .     .
         QNT_TRANSFER                                  .     .     .
         TERMNT                                        .     .     .
      Passivate (of OPRTN_MATRL)                       .     .     .
   START_OPRTN_                                        .     .     .
      START_WORK_G                                     .     .     .
         START_ACTVTY, STATE_CH_LE_                    .     R     .
      STATE_CH_MMS, STATE_CH_OP_                       .     . R   R
      Activate this OPRTN_MATRL at Time                .     .     .
Passivate (of URGENCY_DCSN)                            .     .     .
```

In URG-GANGS- the urgency of a gang is calculated. The applicability of gangs and combinations is derived in APPLCB-GANGS and APPLCB-MM-SYSTMS; that depends on urgency, availability of the gang and its men and machinery, the availability and workability of the materials processed, etc. The selection of appropriate man-machine systems is done in SELECT-MM-S; in STOP-START-OPRTNS some operations are stopped, others are started and others continue. With the stopping of an operation in STOP-OPRTN- stops the work of a gang and the activity of men and machines involved; the states of the men, the machines, the man-machine system and the operation are updated and the processed quantity is transferred (QNT-TRANSFER) before the operation is terminated (TERMNT) and removed from the list of events by calling Passivate. The start of an operation in START-OPRTN- involves the start of the gang, the men and machines and the activation of the operation (Table 6.12). After such a decision the object is removed from the list of events by calling Passivate and waits until another decision is requested.

The second event occurs once a day in ADMINISTRATION and is shown in Table 6.17; attributes are not mentioned. DAILY-ADMINISTRATION is performed and results displayed if required. Periodically output is presented on the use and costs of each object of the system. Finally the activation is delayed for one day by calling Hold (1.0). At the end of a

Table 6.17 Event in dynamic section of ADMINISTRATION.

```
Originated by a request to record and report
once a day the cumulative or daily data of
components.


(If not END_SEASON then:)
   DAILY_ADMINISTRATION
      (if DISPLAY then:)
      DISPLAY_OUTPUT

      (if PERD_OUT and EACH_PERIOD then:)
      CUM_USE_COSTS
      USE_COSTS

   PERD_OUTPUT
   Hold (1), one day
```

Table 6.18 Event in dynamic section of MAIN.

```
Originated by setup of an experiment or
within an experiment by change of year/season


(While a next experiment is given then:)
INIT_EXPER, creates objects of classes

   (while a next year is given then:)
   WTHR.NEW_FILE     (weather)
   RELOAD
      RESET          (component)
   Hold (LAST_DAY_YEAR)

Activate ADMNSTR     (close files)
```

season the administrative object is removed from the list of events and waits for a new season in the experiment.

The last event is shown in Table 6.18 and occurs in the main program (reference MAIN). An experiment is set up by creating the necessary objects belonging to the system in INIT-EXPER. For each year or season a new file with weather data is started, the system is reloaded which requires a reset of each object to an initial stage. The main program is scheduled at the last day of a year (LAST-DAY-YEAR) by calling Hold(.). This sequence is repeated for each season involved in the experiment and for each experiment required in the simulation. Finally output files are closed.

The description of events in the scheduling system is an efficient way to check the logic of this program. It is also a necessary task because the flow of computations is not completely obvious from the program itself. By using a system matrix and its submatrices gaps in the program can be discovered and solved until the result pictures a correct program. It is hoped that the use of decision tables and system matrices also convinced the reader that the program is correct, at least under certain circumstances and conditions. One can describe a scheduling system completely with a hierarchy of system matrices. However the system matrices have been used in a selective way.

6.2 Validation

The behaviour of the model is shown in Table 5.19 where the display output shows the gangs selected; the behaviour during the season is shown in Fig. 5.1 where the area and costs are presented for each day. It is rather subjective to accept or reject a behaviour. Part of it depends on the input data, but the part depending on the model did not call to question the reliability of the model. The sensitivity of the model to parameters is discussed in an earlier monograph published by Elderen (1977). The relation to linear programming is described in Elderen (1980) and is still the subject of a research project along with the evaluation of heuristic strategies. Differences between seasons and between experiments or techniques are described in both publications. To demonstrate the influence of weather and of input, four situations are defined:

(a) wheat ripe on 6 August; dry and wet grain can be harvested;
(b) wheat ripe on 6 August; only dry grain can be harvested because no drier is available;
(c) wheat ripe on 20 August; dry and wet grain can be harvested;
(d) wheat ripe on 20 August; dry and wet grain can be harvested; sprouting of grain in the field is expected (depending on wheat variety and weather in preceding weeks).

Grain harvesting, straw baling, bale gathering and stubble ploughing of 30 ha are restricted to workable time and to regular worktime (07:00-17:00) or overtime of men (17:00-22:00 and Saturday). Grain drying can occur at any time.

Table 6.19 shows the workable time and the rainfall from 6 August to 5 September 1962. The first week, 6-12 August, has only one hour for harvesting of dry grain and a small number (15 hours) for wet grain. The amount of rain from 6-8 August results in a soil moisture content above 47% and no workable time of ploughing for several days. The workable time for straw baling mostly exceeds the time for harvesting but not always. The other weeks offer more workable time. The total workable time during four weeks (6 August – 2 Sept.) and within the worktime of men (07:00-22:00, except on Sunday) is for dry-grain harvesting 50 hours, for dry and wet grain 168 hours, for straw baling 194 hours, for bale gathering 346 hours and for stubble ploughing 161 hours.

The area harvested in the four situations (a-d) is shown in Table 6.20 that clearly shows the delay in harvesting in Situation b where no drier is available compared to Situation a with a drier. The completion of all the work is delayed two days only because a lot of baling, gathering and ploughing could be performed already at hours that harvesting was prohibited. With better weather and more workable time after than before 20 August, 24 days required to complete all the work (and 18 days for the harvesting) in Situation a are reduced to 16 days in Situation c. In Situation d, the urgency

Table 6.19 Workable time in the cereal harvest in 1962 between 7:00 and 22:00.

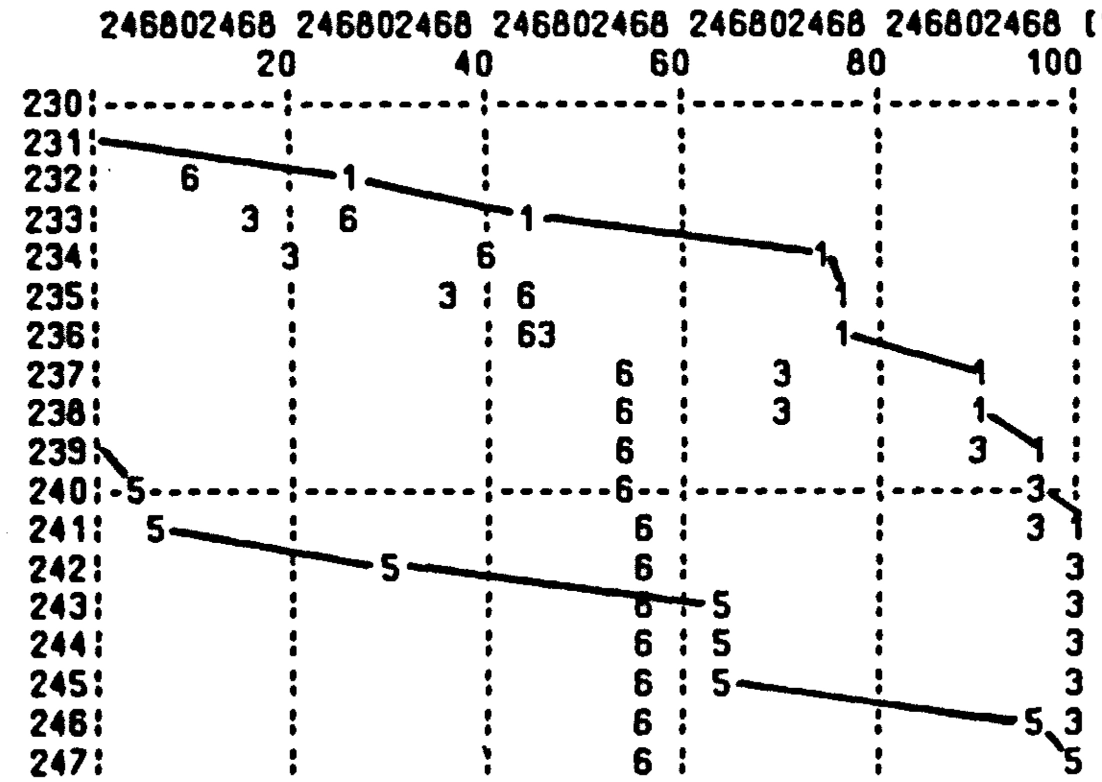
Date	Day	Workable time [h] for: harvesting, dry grain, 0-19% m.c. 7:00-17:00	dry grain, 0-19% m.c. 17:00-22:00 *	wet grain, 19-23% m.c. 7:00-17:00	wet grain, 19-23% m.c. 17:00-22:00 *	total	baling, straw < 25% m.c. no moist	bale gathering, rain < 1 mm/h	ploughing soil < 47% m.c. rain < 1 mm/h	Rain [mm]
6 August	M	1		7	1	9	8	15	15	9.4
7	T							10		13.8
8	W							10		6.4
9	T						8	15		
10	F						6	15		3.2
11	S *			4	3	7	12	15		
12	(S)			(9)	(2)	(11)	(9)	(15)		0.1
13	M	4	4	6	1	15	15	15	15	
14	T	9	1		1	11	11	15	15	0.6
15	W			4	4	8	8	15	6	3.0
16	T		2	10	1	13	13	15	13	0.8
17	F							11		7.2
18	S *			5	2	7	9	15		
19	(S)			(5)	(2)	(7)	(9)	(15)	(10)	0.1
20	M	2	2	4	1	9	9	15	15	0.3
21	T			5	5	10	11	15		3.7
22	W	2	3	6		11	11	15		0.3
23	T	2		7		9	7	15	8	9.2
24	F						3	15		6.0
25	S *			4	4	8	15	15	8	
26	(S)	(4)	(4)	(6)	(1)	(15)	(15)	(15)	(15)	
27	M	10	5			15	15	15	15	0.1
28	T			5		5	5	15	2	3.2
29	W			4	2	6	9	15	5	0.9
30	T			4	1	5	7	15	13	0.1
31	F			5	4	9	11	15	15	
1 Sept	S *	1	2	7	1	11	11	15	15	
2	(S)	(3)	(2)	(2)	(1)	(8)	(8)	(15)	(15)	
3	M	5	4	3	1	13	13	15	15	
4	T	10	2			12	12	15	15	0.6
5	W						4	13	13	5.2

* overtime in the evenings or on Saturday (S) no worktime on Sunday

Figure 6.1 Areas processed and costs of the harvest of sprouting cereal.

```
        EXPERIMENT  with aggregated DAILY data, 'Simulation Monograph' Validation            (d)          1986-02-24  11:07:32
QUANTITIES and COSTS vs. TIME in year:  1962

Date  Day    QUANTITY processed [ha]  /  Quantity expected  30 [ha]          COSTS of Operations,     Crops,                 Total DAY
      number                                                                   men,mach.,gangs,wheat,timeliness losses,         num
                                                                                              exc.c wheat,straw,bales           ber
```



Date	Day number	Day								Total	DAY number
Aug 18	230	Sat	0	0	0	0	0	0	0	0	230
	231	Sun	0	0	0	0	0	0	0	0	231
	232	Mon	75	72	147	96	27	0	0	270	232
	233	Tue	212	174	386	96	75	19	11	485	233
	234	Wed	302	251	553	228	227	31	13	1051	234
	235	Thu	302	274	576	228	253	77	25	1159	235
	236	Fri	407	274	681	228	253	124	27	1313	236
	237	Sat	854	367	1221	228	368	249	27	2094	237
	238	Sun	854	367	1221	228	368	249	27	2094	238
	239	Mon	881	367	1247	228	498	361	27	2362	239
Aug 28	240	Tue	881	367	1247	228	498	379	30	2382	240
	241	Wed	881	390	1270	228	629	379	30	2537	241
	242	Thu	881	390	1270	228	629	381	30	2539	242
	243	Fri	881	390	1270	228	629	381	30	2539	243
	244	Sat	881	390	1270	228	629	381	30	2539	244
	245	Sun	881	390	1270	228	629	381	30	2539	245
	246	Mon	881	390	1270	228	629	381	30	2539	246
	247	Tue	881	390	1270	228	629	381	30	2539	247

1 = Wheat (winter)..........
2 = Straw swath in field....
3 = Bales in field..........
4 = Bales on trailers.......
5 = Stubble field...........
6 = Wetgrain in dryer store.

Table 6.20 Area processed in 1962 in four situations (a-d).

Date	Day	Area of wheat harvested (ha) wheat ripe on 6 August drier (a)	no drier (b)	wheat ripe on 20 August drier (c)	drier sprouting (d)
6 August	M	4.7	0.7		
7	T	4.7	0.7		
8	W	4.7	0.7		
9	T	4.7	0.7		
10	F	4.7	0.7		
11	S	8.5	0.7		
12	S	8.5	0.7		
13	M	17.3	8.8		
14	T	22.2	14.8		
15	W	22.5	14.8		
16	T	26.1	16.8		
17	F	26.1	16.8		
18	S	26.1	16.8		
19	S	26.1	16.8		
20	M	27.3	20.0	7.2	7.7
21	T	27.3	20.0	11.3	12.7
22	W	28.5	25.5	16.5	22.2
23	T	30.0	25.5	19.9	23.3
24	F		25.5	19.9	23.3
25	S		25.5	19.9	26.7
26	S		25.5	19.9	26.7
27	M		30.0	26.7	29.4
28	T			26.7	29.4
29	W	I		26.7	30.0
30	T			26.7	
31	F		I	26.7	
1 Sept	S			26.7	
2	S			26.7	
3	M			29.8	
4	T			30.0I	I
5	W				

I All work (harvesting, baling, bale gathering, ploughing) finished

of harvesting is increased because damage from sprouting grain in the field is expected; the completion of the harvesting is already achieved in 10 days. Figure 6.1 clearly shows that ploughing is performed after harvesting, baling and gathering. Harvesting of grain is accelerated on Monday 20 August and Wednesday 22 August by allowing extra capacity (a higher speed of the machine on the field) and the consequence of extra grain losses at the costs of *f* 96 and *f* 132 (fourth column of costs). The wheat area remaining on Thursday 23 August (6.7 ha) is not urgent enough to prevent baling and bale gathering from about 5 ha. The amount of bales is limited in the simulation to a maximum of 4 ha and even that is reduced if the weather expectation is bad; this forces the gathering operation and allows combine-harvesting to fill the gap with 1.1 ha only. The expectation of workable time for harvesting becomes bad on Friday 24 August and that results in enough urgency to harvest 3.4 ha on Saturday during the four workable hours.

Table 6.21 Costs of the cereal harvest of 30 ha in four situations (a-d).

Table 6.21: Costs of the cereal harvest of 30 ha in four situations.

	Wheat ripe on 6 August		Wheat ripe on 20 August	
	drier (a)	no drier (b)	drier (c)	drier sprouting (d)
Men, overtime only	528	445	630	880
Machines, energy only	295	0	277	389
Timeliness losses of				
wheat	1481	2472	842	628
straw	191	166	191	381
bales	15	19	34	30
Loss of grain due to faster driving the combine harvester	0	0	0	228
Total	2510	3101	1975	2537

The resulting costs of the four situations are shown in Table 6.21. The better weather after 20 August reduces the costs from *f* 2510 for Situation a to *f* 1975 for Situation c, mainly by less timeliness losses of wheat (1481 to 842), although more overtime costs are made (528 to 630). Without a drier, Situation b, the machine costs become zero but the timeliness losses of wheat increase by about *f* 1000. The expected sprouting, Situation d, results in a reduction of timeliness losses of wheat (842 to 628) at the costs of more overtime costs (630 to 880), more drying costs (277 to 389), more timeliness losses of straw (191 to 381) and the use of extra capacity of the combine-harvester and associated grain losses on the field of *f* 228.

These four examples show the influence of the weather (a, c), the workable time related to wet and dry grain and to dry grain, respectively (a, b no drier) and of the timeliness function (c, d). This is no complete validation of the simulation model but it shows that work is limited by workability and that the input can define several experiments. With the description of the input (Section 5.2), the reader must be able to define experiments suitable in a particular case and to test whether output and behaviour of the model is acceptable.

7 EXTENSIONS

This chapter describes extensions of the scheduling problem and the related model, of the programs and of the use of utilities. The program is adapted to the wheat harvesting. Some other specific scheduling problems are discussed in Section 7.2. The last section (7.3) concerns some convenient utilities. The author is interested to learn your applications and extensions and willing to support a further development of such models.

7.1 Wheat harvesting

The wheat-harvesting experimental frame is programmed in a class: SFO-BASE-MODEL class SFOWHEXP; and replaces the subclass SFO-EXPERIMENT described in Chapter 5 'Experimental frame'. The frame contains extended subclasses of the materials and their fields, of weather and of administration. The main purpose of this extension is to be far more specific in the use of material properties, for instance, to use the straw moisture content to influence the capacity of combine-harvesting of wheat, to use a ripening date to revise moisture contents and to use weather information for sprouting expectations.

7.1.1 Materials and weather

The definition of class FIELD (Table 4.9) has been extended with several subclasses related to specific materials (Table 7.1). This allows, for instance, the storage of moisture contents of grain, straw and swath of straw; these are used to initialize the moisture content of wet grain for drying, the date the straw swath becomes processable and the moisture content of bales, respectively. Now these data can be used to redefine the workability or the capacity of processing: not drying at an average rate as if the moisture content is always 22%, but each field dried has its own moisture content and gets its appropriate capacity. The use of these options will be demonstrated for the subclass WINTER-WHEAT.

The weather data file now contains data for month, day, clock, daytype, rain (as in Table 5.5), and for a weather expectation, a condensation index and moisture contents of grain, straw, swath and soil. These data are used to update properties of wheat, straw, bales and soil. For the winter wheat two procedures are involved; ATTR-UPDT-M- updates attributes of the material and is called from weather and ATTR-FLD-M- updates attributes of the field that will be processed. Table 7.2 shows ATTR-UPDT-M-. It first integrates the quantity with the current values, then updates moisture

Table 7.1 Subclasses of FIELD in class SFOWHEXP.

```
FIELD class FLD_WHEAT;
begin
    real MCGRN_INPROC, MCSTR_INPROC;         !weighted moisture content of quantity in process;
end;

FIELD class FLD_SWATH;
begin
    real DATE_WHTRIPE,                       !date when wheat was combine ripe;
    MCSWT_INPROC;                            !m.c. of swath;
end;

FIELD class FLD_BALES;
begin
    real MC_BALING,                          !moisture content (wet basis) of bales at baling;
    RAINSUM_BLNG,                            !cum.amount of rain in mm until baling;
    VALUE_BLNG;                              !value of swath at baling;
end;

FIELD class STRG_WET_GRN;
begin
    real MC_GRAIN_WET;                       !moisture content (wet basis) of grain at harvest;
end;
```

contents and the condition of workable time expected depending on MC-GRAIN and WTHR.EXP-RPRTD (for instance, Elderen, 1977, Table 5). By changing the moisture content of grain, the material delivered can change from wet to dry or the reverse. This initiates a change in all operations processing winter wheat by calling MAT-PROD-CHNG ( ), Table 4.91. If the material to be delivered is dry grain then the use of excess capacity is considered by calling CAP-EXC--M- (it balances losses and urgency). The expected costs for drying in [f/ha] are derived from the fixed costs and capacity of the gang processing wet grain. Workable and the state of winter wheat are now revised. When processing conditions change, then all operations involved update their state (STATE-CH-OP-) and a decision may be requested after giving specific signals to DECIDE such as DRY-GRAIN and WET-GRAIN.

The attributes of the processed field are updated, Table 7.3. The moisture content of unharvested straw on the field, MC-STRW-FLD, is derived from a ripe and an unripe straw moisture content and the ripening date. The capacity ratio, CAP-RATIO depends on this calculated moisture content. The consequences of a change in capacity are reported to operations by calling CAP-QNT-CHNG. This more appropriate approach requires also redefined procedures of 'virtual' procedure DELVR-FLD-M- (,) as shown in Table 4.47. An example is shown in Table 7.4 that belongs to subclass STRAW-SWATH. A ripeness date is derived from wheat and the date the straw becomes processable depends on the moisture content of the straw as calculated in MATRL [WHEAT]. This shows clearly that properties can now be related appropriately and are not restricted to one property per material (as in Table 5.5) and to assumptions about other properties and capacities of processing.

Table 7.2 Procedure ATTR-UPDT-M- of class WINTER-WHEAT.

```
procedure ATTR_UPDT_M_;                              !----  ----  ----  ----  ---;
!                                                    !called from weather, reset_;
begin
    if PROCESSING then QNT_INTGR_M;                  !calls attr_intgr_f_ to integrate m.c.;
    MC_GRAIN:= WTHR.MC_GRAIN_WTH;
    MC_STRW:= WTHR.MC_LWR_STRW;
    if WTHR.REPORT then
    CNDTN_W_T:= IMAX (1, IMIN (4, Entier ( (MC_GRAIN - 10.0) / 5.0) ) ) +
    4 * (WTHR_CATGR_G (WTHR.EXP_RPRTD) - 1); CNDTN_W_T:= IMIN(CNDTN_W_T, CNDTNS_W_T);
    MAT_DLVR_PRV:= MAT_DLVR;
    MAT_DLVR:= if MC_GRAIN <= MC_UB_DRY then DRYGR else WETGR;
    if MAT_DLVR <> MAT_DLVR_PRV then begin
        if PROCESSING then PROCESS_MAT;
        REC_OPR_C_D1:- OPRTN_PROC_Q.First;
        while REC_OPR_C_D1 =/= none do
        inspect REC_OPR_C_D1.OPR_MT do begin
            MAT_PROD_CHNG (MAT_DLVR_PRV, MAT_DLVR);
            REC_OPR_C_D1:- REC_OPR_C_D1.Suc;
        end;    !after this point an operation continues with go_on, see process_mat;
        if MAT_DLVR = DRYGR then CAP_EXC__M_ else CAP_LEVEL:= 0;
    end;
    COSTS_EXPCTD:=
    if MAT_DLVR = DRYGR then 0.0
    else G1.COSTS_H_FXD * (MC_GRAIN - 17.0)/ 5.0/ G1.CAPACITY;  ![fl/ha];
    ATTR_FLD_M_;
    WORKBL_NEXT:= MC_GRAIN <= MC_UB_GRAIN and not WTHR.MSTR_PLNT;
    CH_WORKBL;
    STATE_CH_MAT;
    M_PRC_CND_CH:= false;
    for I_M2:= 1 step 1 until PROC_CNDTNS do begin
        M_PRC_CND_PRV [I_M2]:= M_PRC_CNDTN [I_M2];
        M_PRC_CNDTN [I_M2]:= MC_GRAIN >= PRC_CNDTN_LB [I_M2] and
        MC_GRAIN <= PRC_CNDTN_UB [I_M2];
        M_PRC_CND_CH:= M_PRC_CND_CH or not (M_PRC_CND_PRV [I_M2] eqv M_PRC_CNDTN [I_M2]);
    end;
    REC_OPR_C_D1:- OPRTN_PROC_Q.First;
    if M_PRC_CND_CH then
    while REC_OPR_C_D1 =/= none do begin
        REC_OPR_C_D1.OPR_MT.STATE_CH_OP_;
        REC_OPR_C_D1:- REC_OPR_C_D1.Suc;
    end;
    DECIDE qua URGENCY_DCSN.DRY_GRAIN:= M_PRC_CNDTN [DRY] and not M_PRC_CND_PRV [DRY];
    DECIDE qua URGENCY_DCSN.WET_GRAIN:=  M_PRC_CNDTN [WET] and not M_PRC_CND_PRV [WET] ;
    if (M_PRC_CNDTN [WET] and M_PRC_CND_PRV [DRY] and not M_PRC_CNDTN [DRY] and STATE = RUN)
    !                     !dry-->wet;      or       !wet-->dry;
    (M_PRC_CNDTN [DRY] and not M_PRC_CND_PRV [DRY] and STATE = PASSIVE)
    then activate DECIDE at Time;
end ** of attribute update due to input;
```

Another application of greater flexibility is the use of a maximum quantity of material depending on clocktime or weather expectation (for instance, the area of the bales in the field is limited in this way) or an urgency of processing that depends on clocktime (for instance, unloading of bales on trailers in the evening is delayed until the next morning by manipulating the urgency).

An almost indispensable feature is the appropriate calculation of timeliness losses based on physical properties, for instance, amount of rain in bales. This replaces the crude method of using the expected values of losses represented in the timeliness function. The timeliness functions are intended to allow the calculation of urgencies and are used also to calculate losses when no better ways are available.

Table 7.3 Procedure ATTR-FLD-M- of class WINTER-WHEAT.

```
procedure ATTR_FLD_M_;                          !----  ----  ----  ----  ----;
!                     !called in attr_updt_m_, cap_exc__m_, start_oprtn_, consumptn_m;
if FLD_C =/= none then begin
   RIPE_6:= FLD_C.DATE_OPT_YLD;
   DURTN_RIPE1:= RIPE_6 + 3.0 - TIME_YR;       !duration until full ripeness of straw;
   MC_STRW_UNRP:= 18.0 + 0.8 * MC_STRW;
   MC_STRW_FLD:= if DURTN_RIPE1 < 0.0 then MC_STRW
   else if DURTN_RIPE1 > 5.0 then MC_STRW_UNRP
   !; else MC_STRW_UNRP + (DURTN_RIPE1 - 5.0) / 5.0 * (MC_STRW_UNRP - MC_STRW);
   CAP_RTIO_PRV:= CAP_RATIO;
   CAP_RATIO:= (if MC_STRW_FLD <= 20.0 then 1.0
   else         if MC_STRW_FLD > 65.0 then 0.1
   !;           else 1.0 - (MC_STRW_FLD - 20.0) * 0.02)
   * (1.0 + CAP_EXC_FRAC [CAP_LEVEL]) / 0.8;
   REC_OPR_C_D1:- OPRTN_PROC_Q.First;
   if CAP_RTIO_PRV <> CAP_RATIO and CAP_RTIO_PRV > 0.0 then
   while REC_OPR_C_D1 =/= none do begin
      REC_OPR_C_D1.OPR_MT.CAP_QNT_CHNG (CAP_RATIO / CAP_RTIO_PRV);
      REC_OPR_C_D1:- REC_OPR_C_D1.Suc;
   end;
end;
```

Table 7.4 Procedure DELVR-FLD-M- of class STRAW-SWATH.

```
procedure DELVR_FLD_M_
(QNT_PROD__M_, FLD_ORIGINAL_);                  !----  ----  ----  ----  ----;
real   QNT_PROD__M_;                            !called in delivery_m_;
ref (FIELD) FLD_ORIGINAL_;
inspect new FLD_SWATH do begin
   FLD_Q_SQN:= FLD_Q_SQN + 1;
   FLD_Q:- this FLD_SWATH;
   if FLD_AT_TAIL then Into(FLD_Q) else Follow (FLD_Q);
   DATE_WHTRIPE:= MATRL [WHEAT].FLD_C.DATE_OPT_YLD;
   QUANT_F_PRD:= QUANTITY:= QNT_PROD__M_;
   DATE_PRODCD:= TIME_YR;
   DATE_PROCSBL:= DATE_OPT_YLD := TIME_YR +
   (MATRL [WHEAT].FLD_C qua FLD_WHEAT.MCSTR_INPROC - 20.0) / 10.0;
   MAKE_NAME (FLD_Q_SQN, MAT_NO);
   FLD_AREA:- FLD_ORIGINAL_.FLD_AREA;
end ** of delivering a swath of straw by harvesting;
```

7.1.2 Decision

The description of the urgency decisions as given in Section 5.1.2 'Decision' is almost adequate. Table 7.5 shows some additions to the calculation of the urgency of gangs as shown in Table 5.8. When, for instance, the quantity is less than QUANT-TRMN the urgency increases with URG-TRMN or the quantity is below QUANT-MIN then the URGENCY becomes zero. The urgency is multiplied by STIM-GNG-RUN (1.0) to stimulate gangs already in use. The activation of DECIDE is more restrictive by requiring that the urgency increased by URG-INCR etc.

7.1.3 Administration

The display output and the periodical output (Section 5.1.3) are extended. There are options to create a file for:

Table 7.5 Procedure URG-GANGS- of class URGENCY-DCSN.

```
procedure URG_GANGS_;                               !----   ----   ----   ----   ----;
!                                                   !called in shift_urg, this dynamic;
for K:= 1 step 1 until GANGS do inspect GANG [K] do begin
    real URG_G_PRV;
    URG_G_PRV:= URGENCY;
    URGENCY_GANG_;
    inspect OPR_MT__G do                            !specification from processed materials;
    for I4:= 1 step 1 until MATRLS_PRCSD do inspect MAT_PROC [I4] do begin
        real URG_M;
        if QUANT_EXPCTD = QUANT_ARRVD and QUANT_AVLBL < QUANT_TRMN
        or FLD_C.QUANTITY < QUANT_TRMN or FLD_C.QUANT_F_PRC > FLD_FRCTN_PRC * FLD_C.QUANT_F_PRD
        then URG_M:= URG_TRMN
        else    if QUANT_AVLBL < QUANT_MIN then URG_M:= - URG_PROC_T;
        URGENCY:= URGENCY + URG_M / CAP_PROC;
        URGENCY_CORR:= URGENCY_CORR + URG_M / CAP_PROC;
    end;
    if STATE = RUN then URGENCY_CORR:= URGENCY_CORR * STIM_GNG_RUN;
    if OPRTN__G.STATE = PASSIVE and this DECISION =/= Current and (not MM_S_SELECT or
    URGENCY > URG_G_PRV * URG_MULT  and  URGENCY > URG_G_PRV + URG_INCR) then begin
        URGNC_INCREASE:= true;
        activate DECIDE at Time;
    end;
end;
```

- showing the use of gangs with moments of start and stop and the states of materials and operations;
- quantities processed and delivered vs. daytime with indications of a new field and end of field;
- timeliness losses at each moment it is updated during processing;
- urgencies of materials and gangs each time they are calculated.

These options are so specific that it is sufficient to draw attention to them as means of checking the calculation of the simulation model over time.

In addition another mean is used to show more details about the scheduling system. It concerns the definition of a specific class to record data of a component. For that reason a class COMP-ADMINISTRATION was defined in the base model, Table 7.6, with a 'virtual' procedure TIME-C-ACC-to accumulate time durations and costs. That 'virtual' procedure TIME-C-ACC- in an object of COMP-ADMINISTRATION (referenced by COMP-ADMIN) is called from TIME-C-ACCUM (Table 4.3). Such an object is created if required at the beginning of each new season. In the wheat-harvesting model the empty virtual procedure TIME-C-ACC- was redefined in a subclass as shown in Table 7.7. It records data of a component concerning (i) the variation, mean and number of observations of the time and costs during the three states RUN, PASSIVE and DOWN and in each category of costs considered in a week as defined by the related shift (Section 4.4.2 'Shifts within a week'). A similar facility is available to show more details of the materials. The base model refers to the object by M-ADMN (Table 4.13) and calls the 'virtual' procedures announced in the base model, Table 7.8. In the wheat model a subclass of MATRL-ADMN redefines those procedures. The records made by these two types of classes are used to create output in the class ADMINISTRATION.

Table 7.6 Class COMP-ADMINISTRATION of the base model.

```
class COMP_ADMINISTRATION (COMP);                   !**************************************;
ref(COMPONENT)COMP;
virtual:procedure TIME_C_ACC_;
begin
    procedure TIME_C_ACC_;;                 !called in time_c_accum;
end;
```

Table 7.7 Class COMP-ADMN-YR of the wheat harvest model.

```
COMP_ADMINISTRATION class COMP_ADMN_YR              !**************************************;
(CTG_LB, CTGRS);
integer CTGRS,CTG_LB;
begin
    !d     e     c     l     a     r     a     t     i     o     n;

    real                                            !----------------------;
    COSTS_MD_PRV,                           !cum costs made, previous time;
    COSTS_D_MEAN, COSTS_D_VAR,              !costs made per day: mean and variance;
    TIME_PSV_DAY, TIME_TOTAL,
    TIME_USD_DAY, RUNT_MEAN, RUNT_VAR,      !runtime used per day;
    LT_RO, LT_R, LT_CST;                    !lasttime of update of time used/ costs;
    integer                                         !----------------------;
    COSTS_D_OBSR, RUNT_OBSRV;               !number of observations > 0.0;
    real array                                      !----------------------;
    TIME_USD_PRV,                           !cumulative time used in state: 1=run, 2=passive;
    !                                       3=down, /at previous time;
    TM_DAY_MEAN, TM_DAY_VAR [1:3];          !time used per day in state i: mean and variance;
    real array                                      !----------------------;
    TIME_CTG [1:3,CTG_LB:CTGRS], COSTS_R_CTG [CTG_LB:CTGRS];
    integer array                                   !----------------------;
    TM_DAY_OBSRV [1:3];                     !number of observations with duration > 0.0;

    procedure TIME_C_ACC_;                          !----  ----  ----  ----  ----;
    inspect COMP do                                 !called in component.time_c_accum;
    if AVLB_COMP then begin
        if STATE <> STATE_NEXT or (if SH_PRD =/= none then
        Abs (TIME_YR - SH_PRD.PERD_END [SH_PRD.PERD])
        < 1.0&-4 else false) then begin             !no accumulation if state continues;
            if STATE = RUN and Time - LT_RO > 1.0&-4 then
            SIGMEAN (RUNT_VAR, RUNT_MEAN, RUNT_OBSRV, Time - LT_RO);
            LT_RO:= Time;
        end;
        if STATE = RUN then Accum (COSTS_R_CTG [CTGR], LT_CST, COSTS_D, 0.0)
        else LT_CST:= Time;
        Accum (TIME_CTG [STATE,CTGR], LT_R, LEVEL1, 0.0);
    end
    else LT_RO:= LT_R:= LT_CST:= Time;
    !end of time_c_acc_;

end ** of component adminstration;
```

Table 7.8 Class MATRL-ADMN of the base model.

```
class MATRL_ADMN (MATERL);                          !****************************************;
ref (MATERIAL) MATERL;
virtual: procedure ADMN_M_STATE_, ADMN_AVLBL_, ADMN_WRKBL_, ADMN_READY_,
ADMN_M_UPDATE_, ADMN_M_LT_;;
```

7.2 Specific scheduling problems

In this section some ideas are presented about modelling a scheduling problem in cases not described in the previous chapters. It includes extensions of the program, extensions of the input as well as simplifications. It is merely a list of hints to encourage the tackling of problems and does not present ready made solutions.

7.2.1 Men and machinery

To introduce contract work into the model, one can define an item of equipment (including men if required), a special gang and operation just to perform the task appointed to the contract worker. If a contract worker is not permanently present then a SHIFT-WEEK and SHIFT-PERD can be used to describe the time it is available or when the presence is stochastic failures and repair of equipment can be 'used' to control the possibilities of performing work. On big farms men and women do not perform all the work, but they are specialized. Such a case is well covered by the use of different MM-SYSTMS-SETs (Table 4.77). And when groups of men or machines always operate together, then such a group can be defined as one object; some trailers are already given is one object (Table 5.45). It is also possible to introduce special men for special tasks, for instance, milking and other operations needed for cattle, with MM-SYSTMS-SET. A restricted amount of time of men per day or week may be controlled by 'using' service of equipment; a man is created as if he behaves as an object of class EQUIPMENT.

7.2.2 Materials and fields

If a number of fields are related to a storage and other fields to another storage, then the storage has to become an attribute of a field; even the distance (Table 4.11) can be considered as a factor influencing the capacity of processing in the same way as moisture content of straw controls the combine-harvester capacity. Where the same crop is grown at different locations it seems better to introduce more attributes of a field then to define identical materials at each location (that necessitates the use of parallel gangs and operations). There may be a good reason to revise occasionally, daily, the ordering of the fields to a criterion which changes with the development of the material (ripeness, moisture content).

7.2.3 Development of crops

The growth of a crop depends on moisture, temperature, radiation, etc. To represent such a development of a material a special process is preferred such as shifts for materials as described in Section 4.4.3 for regular urgency calculations. Each material can have its own object of such a shift for growth. The time step of integration can be fixed or depending on the integration routine used and can also be influenced by certain events in the weather or the crop (for instance, irrigation, cutting). When development stages are required, then some concepts of shifts of periods (Section 4.4.1) must be incorporated in class 'shift of growth'. In SIMULA use can also be made of external procedures in other languages such as FORTRAN, so even existing parts of programs or packages can be handled, except for I/O. The dynamic section of the class 'shift of growth' calls integration and updating procedures on behalf of the material or even on behalf of each field of a material if fields are considered with different circumstances.

Sometimes a crop disease occurs and influences the growth of the plant. In that case a 'shift of disease' would be developed similar to 'shift of growth' with its own dynamic behaviour and influence on the material. So the communication of growth and disease is via the material and its fields. It is supposed that distributional patterns of a disease can also become attributes of a field. If the development is straightforward one can use the 'virtual' procedure MAT-DVLPMNT- that is already available in the base model; this alternative to 'shift of ...' has to be defined in a subclass of MATERIAL.

7.2.4 Grass and cattle

How to model grazing cattle? The grass crop is a material and is related to a shift of growth. Some materials are considered; one with fields selected for grazing and another with fields intended for harvesting as silage or hay. The urgency of processing these two materials is different as are the operations. Cut grass also needed two materials; one with fields in need of drying and operations such as tedding, windrowing and another with fields for baling and gathering. The drying process can be formulated in a 'shift of grass drying' similar to shift of growth. Cattle can be considered as a material with identical properties of growth. Different type of animals result in different materials used to represent calves, heifers, beef cattle and milking cows. Pigs and poultry can be handled identically. Cows need a second shift to control milk production for the herd or for each individual animal (comparable with fields). The same shift can control the urgency of milking. Feeding animals is a two step process; the supply of feed is a normal operation performed by men and machines, the eating is the second step, which is independent of men or machines. Grazing is such a step and a good approach would be to introduce an automatic operation 'grazing' which consumes grass on a particular field on behalf of a herd of cattle, where the grazing intensity depends on the cattle and the daily pattern of grazing and resting. The pattern can be controlled by a 'shift of pattern' similar to shift of week. The grazing operation also updates the quantity of grass consumed and trampled; the remaining amount is relevant for regrowth and growth; both depend on the spatial pattern of grass that is still unused, is partly consumed, completely consumed and/or trampled or stained with faeces.

7.2.5 Simplification of input

Although input may be a minor problem some one may be interested in the minimum. Materials, operations, gangs and combinations are always required. But it seems possible to remove machines as objects in the system. Another way is to reduce array dimensions and the related input data such as number of periods, shifts per week, processing conditions, timeliness dates and functions, workable time expectations and number of initial fields.

7.3 Miscellaneous facilities

With the DEC 10 – SIMULA some special facilities are available to make the simulation easier. The database management system SIMDBM and a package VISTA to use a display with cursor control are mentionned. The communication with external procedures includes also FORTRAN subroutines or functions; this may allow perhaps the use of CSMP models within the scheduling environment.

7.3.1 Conversational input

DEC 10 – SIMULA has a package SAFEIO to create a conversation on a display terminal with checking of answers, using manual input or file input. To serve the user of the general scheduling model, a separate SIMULA program based on SAFEIO will be developed. The explanation will be described in the program itself. The system will show a question and a default answer (from the program or a file). The answer is the default, the information from the file or your own reply. The program will contain checks on the range or type of data (Section 5.2.4 'Input data restraints'); when everything is correct, it will produce the output files required by the simulation program, including keywords and headings to explain the data (Section 5.2.2 'Initialization of objects'). This program is especially useful in instructional situations, for instance, to demonstrate the sensitivity of the result (costs in several seasons) to the men and machines selected.

7.3.2 Debugging with SIMDDT

An indispensable facility is the elegant debugging package SIMDDT. It is called automatically when a runtime error occurs (array index out of bounds, a reference denoting to no object, etc.). More valuable is its use at the request of the user. The loaded program can be told to stop or to create output when a certain line is met during execution each time or only when a condition is true. At such a stop, one can ask for the chain-of-procedure calls leading to the current execution or for the objects scheduled on the time axis or to display source program text lines. It is even possible to use a file that contains some SIMDDT commands. This facility may be used to check the scheduling of objects in the simulation, for there are no standard tracing facilities available as in DEMOS (Birtwistle, 1979).

7.3.3 Graphic data

Display of the development of the scheduling system is possible with VISTA. The costs or area of materials vs. time and the use of operations each day are ideas that will be built into a special program that will be developed.

7.4 Other computers

The same scheduling program has been modified and tested on a VAX computer. Because the simulation program consists almost entirely of only standard SIMULA, it should be not difficult to convert it to IBM-SIMULA, CDC-SIMULA or SIMULA on other mainframes. SIMULA for microcomputers is not yet available, but it is announced.

Definitions

– combination	– a set of gangs that can work simultaneously with the available men and items of machinery.
– dynamic system	– 'a system to which events occur, whose state changes over time' (Ackoff, 1971).
– event	– 'a change in one or more structural properties of the system (or its environment) over a period of time of specified duration; that is a change in the structural state of the system (or environment)' (Ackoff, 1971).
– failure	– 'the inability of a machine to perform its function under specified field and crop conditions'. (ASAE, 1985, 4.4)
– gang	– the men and items of machinery required to perform an operation with a specific set of materials according to a method.
– material	– an entity of the biological subsystem of a farm which is processed/consumed or produced/delivered/supplied by an operation.
– processable	– a property, an attribute of a material changing only one time for each field; for instance,
processable	= true if material is ripe, = false otherwise.
– ready	– a property, an attribute of a material depending on processable and workable;
ready	= true if (processable = true and workable = true), = false otherwise.
– repair	– 'restoring a machine to operative condition after breakdown, excessive wear, accidental damage.' (ASAE, 1985, 4.8).
– scheduling	– 'determining the time when the various operations are to be performed. Availability of time, labor supply, job priorities, and crop requirements are some important factors'. (ASAE, 1985, 2.7.2).

– service	– 'periodic activities to prevent premature failure and to maintain good functional performance'. (ASAE, 1985, 4.6).
– state	– the set of relevant properties of a system at a time (Ackoff, 1971).
– strategy	– a procedure prescribing a decision at each decision moment, based on the state of the system at that moment.
– timeliness function	– recoverable value of a material as a function of time.
– timeliness (of operation)	– 'ability to perform an activity at such a time that crop return is optimized considering quantity and quality of product'. (ASAE, 1985, 2.14).
– workable	– a property, an attribute of a material depending on variable properties such as moisture content, which depends in its turn on the variations of the weather;
workable	= true if relevant properties are within a range appropriate for processing by specific operations, = false otherwise.

Literature

Ackoff, R.L., 1971. Towards a System of Systems Concepts. Management Science 17: 661-671

ASAE, 1985. Uniform terminology for agricultural machinery management. ASAE Standard S322.1. Agricultural Engineers Yearbook American Society of Agricultural Engineers, p.217-218.

Birtwistle, G.M., 1979. DEMOS. A system for Discrete Event Modelling on Simula. The Macmillan Press Ltd, Londen, 214pp.

Birtwistle, G.M., O.J. Dahl, B. Myhrhaug & K. Nygaard, 1973. SIMULA Begin. Auerbach Publishers Inc., Philadelphia, 391 pp.

DEC system 10 SIMULA Language Handbook, Part I, II & III. Swedish National Defense Research Institute, Stockholm, 1976.

Elderen, E. v. & S.P.J.H. v. Hoven, 1973. Moisture content of wheat in the harvesting period. J.Agric.Eng.Res. 18: 71-93.

Elderen, E. v., 1977. Heuristic strategy for scheduling farm operations. PUDOC, Wageningen NL, 217 pp.

Elderen, E. v., 1980. Models and techniques for scheduling farm operations; a comparison. Agric. Systems 5: 1-17.

Franta, W.R., 1977. The process view of simulation. Elsevier North-Holland, Inc., New York, 244 pp.

Jaederlund, Chr., Systematrix; complete SMX handbook; Holistic system development. (CAP-Gemini Nederland; Copyright 1982 from Jaederlund Box 3369, S-163 03 Spanga, Sweden)

Kindler, E., K. Prokop & S. Chochol, 1981. Dynamic modelling of transport in agricultural systems. Elektronische Informationsverarbeitung und Kybernetik 17, 645 – 657.

Pidd, M., 1984. Computer simulation in management science. John Wiley & Sons, Chichester, 237pp.

Stuff, R.G. & R.F. Dale, 1973 A simple method of calender conversion in computer applications. Agric. Meteorology 12: 441 – 442.

Thesen, A., 1978. Computer Methods in Operations Research. Academic Press, New York, 268 pp.

Zeigler, B.P., 1976. Theory of modelling and simulation. John Wiley & Sons, New York, 435 pp.

Appendix A Floppy with programs and input

A floppy can be requested from the author:

dr. E. van Elderen
Institute of Agricultural Engineering
P.O.B. 43
6700 AA Wageningen
the Netherlands

at the costs of 25 Dutch florins (or 10 US dollars) per floppy (incl. postage).

The IBM-formatted floppy can be read on a micro-computer under MS-DOS.

Content of floppy A: Base model and Experimental frame (incl. input)

- Three programs (used on a DEC-10 computer) with line numbers (increment 10):
  SFOBAS.SIM – program of the base model,
  SFOEXP.SIM – program of the experimental frame,
  SFOSIM.SIM – main program of the simulation;
- Six input files used in EXAMPLE (see Tables 5.35 – 5.56):
  EXPERI.MNT – input file of files,
  ENVRMT.XPL – input file of experiment,
  MMSSSF.XPL – input file of man machine subsystem,
  MATWWT.XPL – input file of biological subsystem,
  WHTFLD.XPL – input file of initial fields,
  GDLR62.XPL – input file of weather data.

Content of floppy B: Extended experimental frame and VAX-version of the base model

- Two programs used on a DEC-10 computer to simulate the wheat harvest:
  SFOWHE.SIM – program of the wheat-harvesting experimental frame,
  SFOWHS.SIM – main program of the wheat-harvesting simulation.
- Three programs without comment and with a line length less than 80 characters, accepted by the compiler on a VAX8600-computer:
  SFOBAS.VAX, SFOEXP.VAX and SFOSIM.VAX.

Appendix B Classes, References and Procedures

Name of Class	Table	Reference	Line numbers (incr. 10) in program files (see Appendix A) SFOBAS	SFOEXP	SFOSIM
ADMINISTRATION	5.14	ADMNSTR		4010-7450	
ADMINISTRATOR	5.13	ADMN	30220-30290		
AREA	4.11	FLD_AREA	29090-29220		
COMPONENT	4.2	COMPNT	03360-04310		
COMP_ADMINISTRATION	7.6	COMP_ADMIN	04430-04480		
COMP_ADMN_YR	7.7				
DECISION	4.101	DECIDE	16560-16640		
EQUIPMENT	4.58	MACH	05500-06950		
FIELD	4.9	FLD, FLD_C, FLD_D, FLD_E	29340-29880		
FINAL_MAT			28680-28990		
GANG_SET	4.72	COMB_G	09650-09990		
GANG_SET_GNRTD	4.78		10140-11190		
INITIAL_MAT			27920-28410		
INTERMDT_MAT			28530-28570		
LABOUR	4.57	MAN	05220-05330		
LABR_EQPMNT	4.53		04640-05050		
Main program	5.28	MAIN			0340-4580
MAN_MACH_SET	4.67	GANG	08240-09500		
MAN_MACH_SYS	4.62	MM_S	07180-08060		
MATERIAL	4.12	MATRL	17640-24120		
MATRL_ADMN	7.8	M_ADMN	24240-24270		
MATRL_ADMN_YR					
MM_SYSTMS_SET	4.77	MM_S_SET	11370-11450		
OPERATION	4.82	OPRTN	12240-12400		
OPRTN_MATRL	4.83	OPR_MAT	12640-15260		
PLOT_ADMIN				7500-7550	
PROCESSED_MAT	4.37	MATRL_PRC	24430-27800		
RECORD_COMP	4.1		03490		
RECORD_LB	4.1		05160		
RECORD_LE	4.1		04600		
RECORD_MAT	4.1		17580		
RECORD_MM_S	4.1		07110		
RECORD_OPR	4.1		12580		
RECORD_SR_RP	4.1		15370		
SFOBASE_MODEL	4.1, 4.5		00530-31140		
SFOEXPERIMENT	5.1			0310-7590	
SHIFT_PERD	4.102	SH_PERD	01230-01880		
SHIFT_URG	4.107	SH_URG	16790-17420		
SHIFT_WEEK	4.104	SH_WKLY	02060-03330		
SRVC_REPR	4.94	OPR_S_R	15440-16410		
UPDT_MAN_MCH	4.73	UPDT_MM_SYS	11620-12080		
URGENCY_DCSN	5.6	DECIDE_URG		1480-3860	
WEATHER	5.2	WTHR		0550-1320	
WEATHER_MATRL	4.8	WTH_MAT	30010-30090		

The scheduling of operations on a farm is described as a system and the theory and models used are presented. The program which simulates the scheduling system is written in SIMULA. A base model contains the basic components of the system such as men, machines, operations and crops. An experimental frame describes the input and output and defines the simulation, An example is given of the scheduling of operations during wheat harvesting. Verification of the program and validation of the model are discussed. Extensions valid for wheat harvesting are mentioned and suggestions for use in other circumstances and applications are described.

Pudoc Wageningen