CHAPTER 13.

A WORKED EXAMPLE

The following example is given in terms of SIMULA. It demonstrates a technique of aggregating individual items into groups of items in order to increase program efficiency. The system described is a simplification of an actual case study carried out at the Norwegian Computing Center, Oslo.

Given a job shop consisting of machine groups, each containing a given number of identical machines in parallel. The system will be described from a machine point of view, i.e. the products flowing through the system are represented by processes which are passive data records. The machines operate on the products by remote accessing.

The products consists of <u>orders</u>, each for a given number of product <u>units</u> of the same <u>type</u>. There is a fixed number of product types. For each type there is a unique routing and given processing times.

For each machine group (number <u>mg</u>) there is a set <u>avail[mg]</u>of idle machines and a set <u>que[mg]</u>, which is a product queue common to the machines in this group. The products are processed one <u>batch</u> at a time. One batch consists of a given number of units, which must belong to the same order. The batch size depends on the product type and the machine group.

A product queue is regarded as a queue of <u>orders</u>. The queue discipline is essentially first-in-first-out, the position of an order in the queue being defined by the arrival of the first unit of that order. However, if there is less than an acceptable

---

This chapter is part of a series of lectures originally given by O.-J. Dahl at the NATO Summer School on Programming Languages, Vilard-de-Lans, 1966.

of them. An opart process will reference the one at the next step through an element attribute "successor". An order is thus represented by a simple chain of opart records. The one at the head has no successor, the one at the tail has its attribute "last" equal to true. The chain "moves" through the system by growing new heads and dropping off tails.

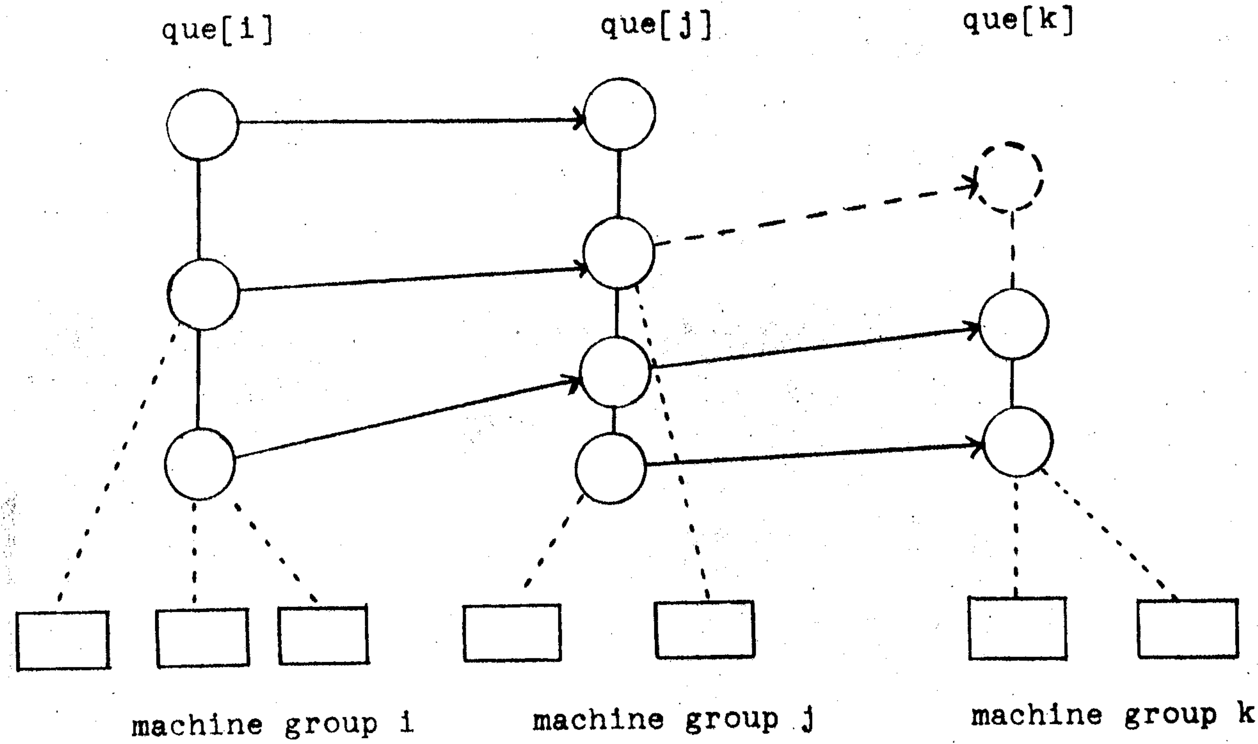

Fig. 3.

Three consecutive steps in the schedule of products of a given type. A product queue consists of oparts (circles) connected by vertical lines. Oparts belonging to the same order are connected by horisontal lines. Machines are represented by squares. A dotted line between an opart and a machine indicates a batch of units in processing. When the batch of the third opart in que[j] is finished, a new opart receiving this batch will be generated and included in que[k].

The following piece of program is part of the head of a SIMULA block describing the above system. A machine activity is given. For clarity only statements essential for the behaviour of the model are shown. The program is not complete.

```
set array que, avail [1:nmg]; integer U;
integer procedure nextm (type, step); integer type, step;....;
real procedure ptime (type, step); integer type, step;....;
integer procedure bsize (type, mg); integer type, mg;....;
activity opart (ono, type, step, nw, np, last, successor);
  integer ono, type, step, nw, np;
  Boolean last; element successor;;
activity machine (mg); integer mg;
  begin integer batch, next; Boolean B; element X;
serve: X:= head (que[mg]);
    for X:= suc (X) while exist (X) do
    inspect X when opart do
    begin batch := bsize (type, mg);
      if nw < batch then begin
        if last then batch := nw else go to no end;
      nw := nw - batch; np := np + batch;
      if last ∧ nw = 0 then remove (X);
      activate first (avail[mg]);
      hold (batch × ptime (type, step)×uniform (0.9, 1.1, U));
      np := np - batch; B := last ∧ nw + np = 0;
      next := nextm (type, step);
      inspect successor when opart do
        begin nw := nw + batch; last := B end
      otherwise begin successor :=
        new opart (ono, type, step + 1, batch, 0, B, none);
        include (successor, que [next]) end;
      activate first (avail [next]);
      go to serve;
no: end;
    wait (avail [mg]); remove (current); go to serve end;
```

Comments.

Line 1. The sets will contain oparts and idle machines respectively. The variable U defines a pseudo-random number stream (line 19).

Lines 2-4. The functions "nextm" and "ptime" specify the next machine group and the current processing time for a given product type and step in the schedule. "bsize" determines the batch size, given the product type and machine group number. The three functions are left unspecified.

Lines 5-7. The meanings of the attributes of opart processes have been explained above. The activity body is a dummy statement: an opart process is a data record with no associated actions.

Line 8. The machine activity extends to and includes line 30. The parameter mg is the machine group number. Machines belonging to the same group are completely similar.

Line 9. "batch" is the size of the current batch of units, "next" is the number of the next machine group for the units currently being processed, the meaning og "B" is explained below (line 20), and "X" is used for scanning.

Line 10. Prepare for scanning the appropriate product queue.

Line 11. Scan. The controlled statement is itself a connection statement (lines 12-29).

Line 12. There is only one connection branch (lines 12-29). Since a product queue contains only opart records connection must become effective. The attributes of the connected opart are accessible inside the connection block.

Line 13. Compute the standard batch size.

Lines 14, 15. A smaller batch is only accepted if the opart is at the tail end of the chain. In this case "nw" is nonzero (cf. line 17), and the units are the last ones of the order. Otherwise the next opart is tried.

Line 16. "batch" units are transferred from the waiting state to the in-processing state by reducing nw and increasing np.

Line 17. The opart is removed from the product queue when processing has started on the last units of the order.

Line 18. The current machine has found an acceptable batch of units, and has updated the product queue. There may be enough units left for another batch, therefore the next available machine in this group (mg) is activated. If there is no idle machine, the set avail[mg] is empty and the statement has no effect. See also lines 27 and 30.

Lines 19. The expected processing time is proportional to the number of units in the batch. The actual processing time is uniformly distributed in the interval ± 10% around the expected value. The sequence of pseudo-random drawings is determined by the initial value of the variable U.

Line 20. Processing is finished; np is reduced. The Boolean variable B gets the value true if and only if the last units of an order have now been processed. In that case the connected opart should drop off the chain at this system time (see comments to line 28). It follows that B is always the correct (next) value of the attribute "last" of the succeding opart (lines 23, 25).

<u>Line 21</u>. Compute the number of the machine group to receive the current batch of units.

<u>Line 22</u>. The <u>element</u> attribute "successor" is inspected. The connection statement, lines 22-26, has two branches.

<u>Line 23</u>. This is a connection block executed if "successor" refers to an opart. The latter is a member of the product queue of the next machine group. It receives the processed batch of units, which are entered in the waiting state. The attribute "last" is updated. Notice that the attributes referenced in this inner conection block are those belonging to the successor to the opart connected outside (X).

<u>Lines 24, 25</u>. If the connected opart (X) is at the head of the chain the value of "successor" is assumed equal to <u>none</u>, and the <u>otherwise</u> branch is taken. A new opart is generated, and a reference to it is stored in "successor". The new opart has the same "ono" and "type" as the old one, and its "step" is one greater. It has "batch" units in the waiting state and none in processing. Its attribute "last" is equal to "B". Since the new opart has become the head of the chain, its "successor" should be equal to <u>none</u>. Notice that the initial value of "last" may well be <u>true</u>, e.g. if the order contains a single unit.

<u>Line 26</u>. The new opart is included at the end of the product queue of the next machine group.

<u>Line 27</u>. The current machine has now transferred a batch of units to the product queue of next machine group. Therefore the first available machine (if any) of that group is activated. If that machine finds an acceptable batch it will activate the next machine in the same group (line 18). This takes care of

<u>Line 29</u>. The end of the connection block and of the statement controlled by the *for* clause in line 11.

<u>Line 30</u>. If, after having searched the entire product queue, the machine has found no acceptable batch, it includes itself in the appropriate "avail" set and goes passive. Its local sequence control remains within the wait statement as long as the machine is in the passive state. When the machine is eventually activated (by another machine: line 27 or 18), it removes itself from the "avail" set and returns to scan the product queue. The "avail" sets are operated in the first in-first out fashion.

The mechanism for feeding orders into the system is not shown above. This is typically done by the Main Program or by one or more "arrival" processes, which generate a pattern of orders, either specified in detail by input data, or by random drawing according to given relative average frequencies of product types and order sizes.

An arrival pattern defined completely "at random" is likely to cause severely fluctuating product queues, if the load on the system is near the maximum. The following is a simple way of rearranging the input pattern such as to achieve a more uniform load. The algorithm is particularly effective if there are different "bottle-necks" for the different types of products.

```
31. activity arrival (type, mg1, pt);
32.   integer type, mg1; real pt;
33. begin integer units;
34. loop: select (units, type); id := id + 1;
35.       include (new opart (id, type, 1, units, 0, true,
                                                none), que[mg1]);
36.       activate first (avail [mg1]);
37.       hold (pt×units); go to loop end;
38. procedure select (n, type); value type; integer n, type;....;
39. integer id;
```

Comments.

Line 31. There will be one "arrival" process for each product type. "mg1" is the number of the first machine group in the schedule of this type of product. "pt" is a stipulated "average processing time" per unit, chosen so as to obtain a wanted average throughput of units of this type (see line 37).

Line 34. The procedure "select" should choose the size, "units", of the next order of the given type, e.g. by random drawing or by searching a given arrival pattern for the next order of this type. "id" is a non-local integer variable used for numbering the orders consecutively.

Line 35. An order is entered by generating an opart record which contains all the units of the order. The units are initially in the waiting state. The order is filed into the appropriate product queue. The set membership is the only reference to the opart stored by the arrival process. Consequently this opart will leave the system when it becomes empty of units, as assumed earlier (line 28).

Line 36. A machine in the appropriate group is notified of the arrival of an order.

Line 37. The next order of the same type is scheduled to arrive after a waiting time proportional to the size of this order, which ensures a uniform load of units (of each type).

The "output" of units from the system can conveniently be arranged by routing all products to a dummy machine group at the end of the schedule. It contains one or more "terminal machines" (not shown here) which may perform observational functions such as recording the completion of orders.

The dynamic setup of the system is a separate task, since initially the Main Program is the only process present. The Main Program should generate (and activate) all processes which are "permanent" parts of the system, such as machines, arrival processes and observational processes. The system can be started empty of products, however, a "steady" state can be reached in a shorter time if orders (opart records) are generated and distributed over the product queues in suitable quantities.

The experimental results are obtained by observing and reporting the behaviour of the system. Three different classes of outputs can be distinguished.

1) On-line reporting. Quantities describing the current state of the system can be printed out, e.g. with regular system time intervals: lengths of product queues in terms of units waiting, the total number of units in the system, the number of idle machines in each group, etc. A more detailed on-line reporting may be required for program debugging.

2) Accumulated machine statistics. By observing the system over an extended period of system time averages, extrema, histograms, etc., can be formed. Quantities observed can be queue lengths, idle times, throughputs, and so on. The accumulation of data could be performed by the machine processes themselves.

   Example. To accumulate a frequency histogram of the idle periods of different lengths for individual machines, insert the following statements on either side of the "wait" statement of line 30:

   "tidle := time" and "histo" (T, H, time - tidle, 1)", where "tidle" is a local real variable, and T and H are arrays. T[i] are real numbers which partition observed idle periods (time - tidle) into classes according to their lenths, and H[i] are integers equal to the number of occurrences in each class. The system procedure "histo" which will increase H[i] by one (the last parameter), where i is the smallest integer such that T[i] is greater than or equal to the idle period "time - tidle". T and H together thus define a frequency histogram, where T[i] - T[i - 1] is the width of the i'th column, and H[i] is the column length.

3) Accumulated order statistics. During the life time of an opart record the "history" of an order at a given machine group can be accumulated and recorded in attributes of the opart. The following are examples of data which can be found.

The arrival time of the first unit of the order at this machine group is equal to the time at which the opart is generated. The departure time of the last unit is equal to the time at which the variable B gets the value true (line 20 of a machine connecting the opart).

The sum of waiting times for every unit of the order in this queue is equal to the integral with respect to system time of the quantity nw (which is a step function of time). The integral can be computed by the system procedure "accum". The statements "nw := nw ± batch" (lines 16 and 23) are replaced by "accum (anw, tnw, nw, ± batch)", where the real variables anw and tnw are additional attributes of the opart process, with initial values zero and "time" respectively. The procedure will update nw and accumulate the integral in anw. It is equivalent to the statements: anw := anw + nw * (time - tnw); tnw := time; nw = nw ± batch;

It is worth noticing that arrival times, waiting times, etc., can not in general be found for individual units, unless the units are treated as individuals in the program. Neither can the maximum individual waiting time for units in an order. The average waiting time, however, is equal to the above time integral divided by the number of units in the order.

The complete history of an order in the shop is the collection of data recorded in the different oparts of the order. These data can be written out on an external storage medium at the end of the lifetime of each opart. I.e. an output record could be written out before line 28, whenever B is true, containing items such as the order number, ono, the sum of waiting times, anw, the current system time, etc. When the simulation has been completed, the data records can be read back in, sorted according to order numbers, and processed to obtain information concerning the complete order, such as the total transit time, total waiting time etc.

The same information can be obtained by retaining the complete opart chain in the system until the order is out of the shop. However, this requires more memory space. The chain can be retained by making the arrival process include the initial opart in an auxiliary set, or by having a pointer from the opart currently at the head of the chain back to the initial one. The opart chain can be processed by the terminal machine. (The order is completely through the shop at the time when the attribute "last" of the opart in the terminal product queue gets the value true.) In the former case the terminal machine should also remove the appropriate opart from the auxiliary set, in order to get rid of the opart chain.

1107 - SIMULA.
Library procedures.

A. Alphabetic order.

Name	Result Arithmetic	No.of.param.	Report reference
ACCUM	none	4	8
CANCEL	-	1	4.8
CARDINAL	integer	1	3.7
CLEAR	none	1	3.7
CURRENT	element	0	4.4
DISCRETE	integer	2	7.2
DRAW	Boolean	2	7.2
EMPTY	-	1	3.7
EQUAL	-	2	(no reference)
EVTIME	real	1	4.4
EXIST	Boolean	1	3.7
FINISHED	-	1	4.4
FIRST	element	1	3.7
FOLLOW	none	2	3.7
HEAD	element	1	3.4.2
HISTD	integer	2	7.2
HISTO	none	4	8
HOLD	-	1	4.3
IDLE	Boolean	1	4.4
INCLUDE	none	2	3.7
LAST	element	1	3.7
LINEAR	real	3	7.2
MEMBER	element	2	3.7
NEGEXP	real	2	7.2
NEXTEV	element	1	4.4
NORMAL	real	3	7.2
NUMBER	element	2	3.7
ORDINAL	integer	1	3.7
PASSIVATE	none	0	4.2
POISSON	integer	2	7.2
PRCD	none	2	3.6
PRECEDE	-	2	3.7
PRED	element	1	3.4.2
PROC	-	1	3.4.1
PSNORM	real	4	7.2
RANDINT	integer	3	7.2
REMOVE	none	1	3.6
SAME	Boolean	2	3.5
SIMILAR	-	2	3.5
SUC	element	1	3.4.2
SUCCESSOR	-	2	3.7
TERMINATE	none	1	4.2
TIME	real	0	4.4
TRANSFER	none	2	3.7
UNIFORM	real	3	7.2
WAIT	none	1	4.2

B. Aritmetic order.

1. No arithmetic.

Name	N.of p.	Param. types
ACCUM	4	
CANCEL	1	E
CLEAR	1	S
FOLLOW	2	E,E
HISTO	4	A,A,R,R
HOLD	1	R
INCLUDE	2	E,S
PASSIVATE	0	-
PRCD	2	E,E
PRECEDE	2	E,E
REMOVE	1	E
TERMINATE	1	E
TRANSFER	2	E,S
WAIT	1	S

2. Integer.

Name	N.of p.	Param. types
CARDINAL	1	S
DISCRETE	2	RA,I
HISTD	2	RA,I
ORDINAL	1	S
POISSON	2	R,I
RANDINT	3	I,I,I

3. Real.

Name	N.of p.	Param. types
EVTIME	1	E
LINEAR	3	RA,RA,I
NEGEXP	2	R,I
NORMAL	3	R,R,I
PSNORM	4	R,R,I,I
TIME	0	-
UNIFORM	3	R,R,I

4. Boolean.

Name	N.of p.	Param. types
DRAW	2	R,I
EMPTY	1	S
EQUAL	2	E,E
EXIST	1	E
FINISHED	1	E
IDLE	1	E
SAME	2	E,E
SIMILAR	2	E,E

5. Element.

Name	N.of p.	Param. types
CURRENT	0	-
FIRST	1	S
HEAD	1	S
LAST	1	S
MEMBER	2	E,S
NEXTEV	1	E
NUMBER	2	I,S
PRED	1	E
PROC	1	E
SUC	1	E
SUCCESSOR	2	I,E