Proceedings
of the 3rd International Modelica Conference,
Linköping, November 3-4, 2003,
Peter Fritzson (editor)

Jerzy Mikler and Vadim Engelson
PELAB, Linköping University; Royal Institute of Technology, Sweden :
Simulation for Operation Management: Object Oriented Approach using Modelica
pp. 207-214

Paper presented at the 3rd International Modelica Conference, November 3-4, 2003, Linköpings Universitet, Linköping, Sweden, organized by The Modelica Association and Institutionen för datavetenskap, Linköpings universitet

All papers of this conference can be downloaded from
http://www.Modelica.org/Conference2003/papers.shtml

Simulation for Operation Management : Object Oriented Approach using Modelica

Jerzy Mikler,
IIP, The Royal Institute of technology, Sweden,
jerzy@iip.kth.se

Vadim Engelson,
IDA, Linköping University, Sweden,
vaden@ida.liu.se

WWW: http://www.ida.liu.se/~vaden/or

Abstract[1]

Operation management provides methods and tools for decision making in production systems. There are both mathematical and simulation-based methods for finding optimal production parameters. Simulation models are based on both continuous and discrete event simulation. These models can be reused on both component level and pattern (template) level. Modelica fits well into these requirements; one of case studies revealing related problem is discussed in this paper.

1 Introduction

Simulation is an important tool for executives in their day-to-day decision making activities. The problems occurring in sales departments, on the shop floor, and all the other departments of a company are more often difficult and complex. They cross the functional boundaries, and are dynamical in nature. Figure 2 shows a simplified causal loop diagram (CLD) of a generic assembly system. It shows all the identified factors relevant for the system flexibility, together with their relations.

The relations (arrowed lines connecting the factors) mean that the one factor affects the other[2]. The problem is yet more complicated by the fact, that the influence is most often delayed, adding the time dimension, what in turn makes the understanding of the behaviour (the consequences) impossible without aid of simulation. To simulate the behaviour of this system one has to build system dynamic (SD) model of the CLD above. SD approach has been initially defined by Forrester[4]. Relations between the factors in SD are described with a system of differential algebraic equations (DAE) with delays. Such a model of a part of the system in Fig. 2 (the "operations") is shown in Fig. 1. A Modelica library for system dynamic modelling has been developed by Fabricius [3].

These kinds of models are typical for *analysis* tasks – i.e. when the real structure of the system is not known, and we examine the patterns of behaviour to identify the structure. In this case the suitable tools are system thinking and system dynamics (based on time continuous simulation). Another type of problems arises when we know the components and *synthesize* them to a system in the aim to achieve a desired behaviour of this system. In this case a suitable tool is discrete event simulation. These kinds of problems are typical for shop floor design and operations. The inventory problem evaluated later in this paper is an example of that (see even Fig. 3).

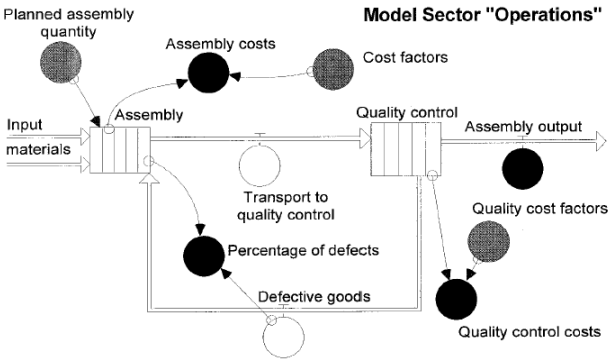

Fig. 1. The system dynamics model of the sector "Operations" on the CLD in Fig. 2, modelled in the Stella environment. System Dynamics library in Modelica uses similar graphical notation. Boxes denote levels (stocks), circles denote rates (e.g. flows between levels). Source: Zahn [1]

[1] This article describe ongoing work carried out in the **VISP** project supported by Vinnova foundation, Sweden. Information about the current status can be found at **`http://www.ida.liu.se/~vaden/or`**

[2] Arrows indicate the direction of causality. Signs (+ or -) imply the polarity of relationships: a'+' indicates that an increase in the independent variable causes the dependent variable to increase. A loop is identified by number and sign (-) or (+). (-) indicates restoration of balance by an action, whereas (+) implies either reinforcing vicious cycle or virtuous cycle.

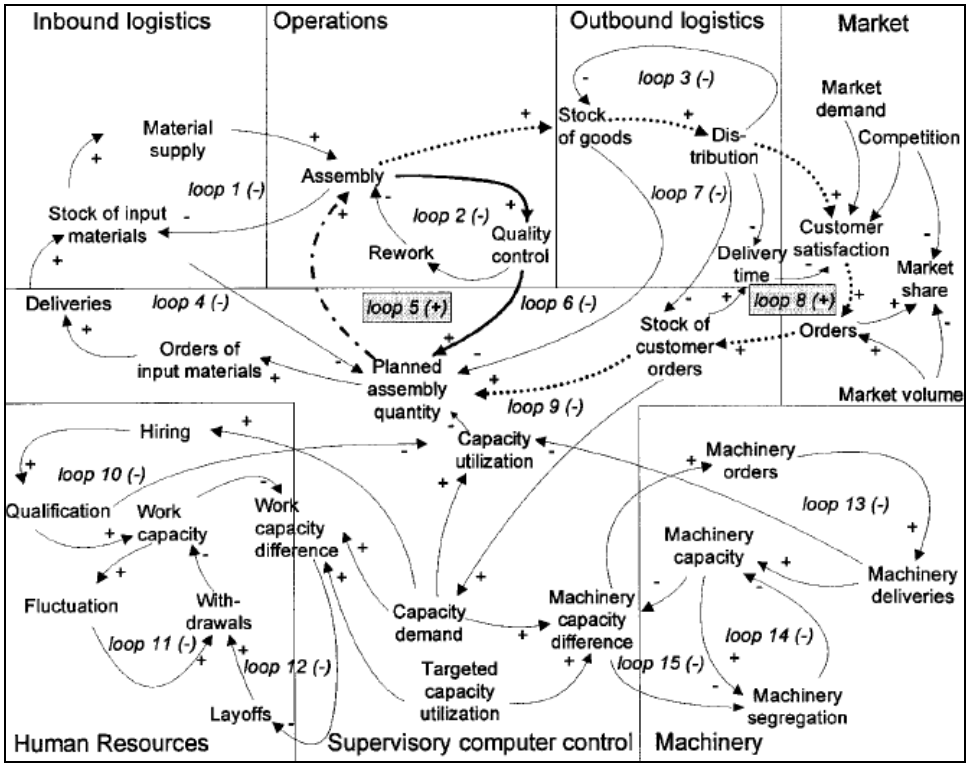

Fig. 2. Causal loop diagram of generic assembly systems. Source: Zahn [1]

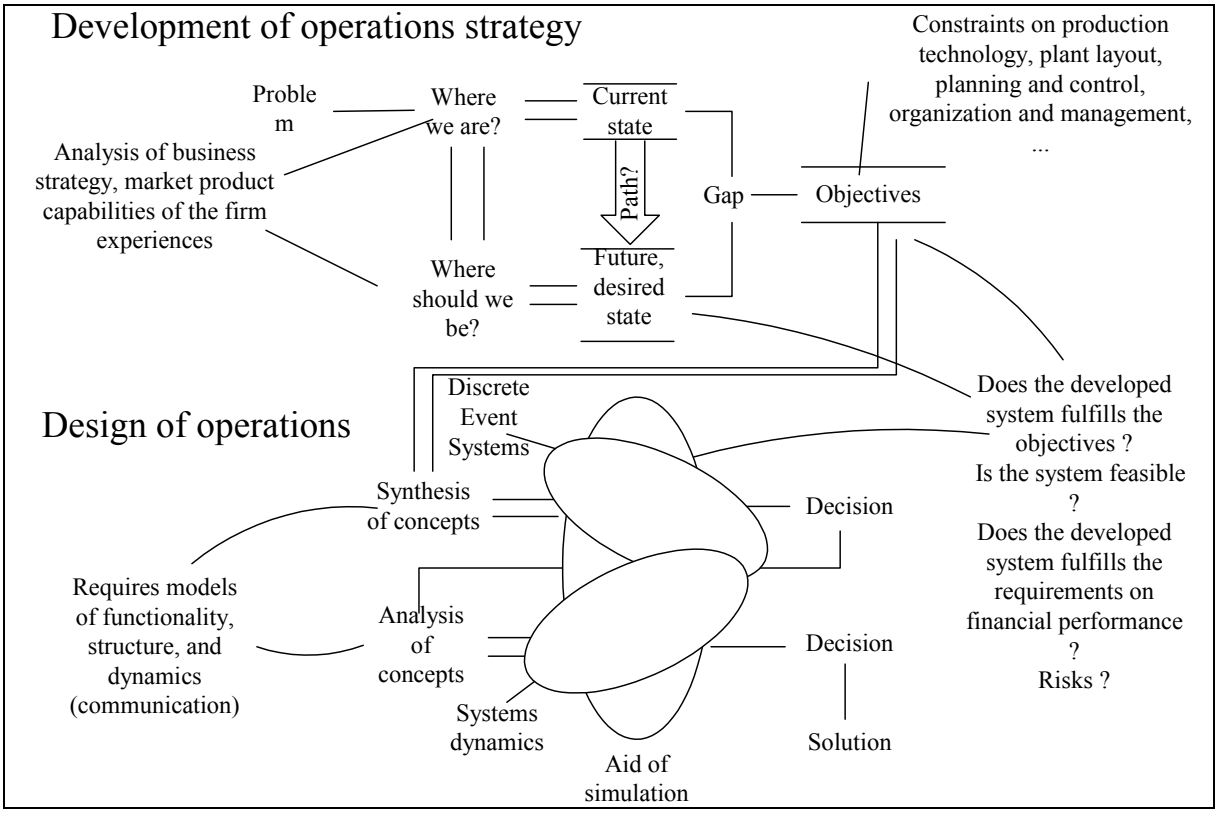

Fig. 3. Structure of a design process, based on work of B.Wu.[13]

2 . Operation Management

Operations Management is an established discipline providing insights, methods and tools facilitating decision-making in production systems. It spans the whole spectrum of managerial problems in a company – from the formation of strategy, trough product and process design, to the run time operation. These decisions are usually divided into the four categories [5]:

1. *Strategic choices* – determination of competitive priorities and strategies how to best design the product and production processes that fit the priorities (operations strategy)

2. *Product and Process design* –

a. Product design[3] (decisions concerning concept generation, screening, preliminary design, evaluation and improvement, prototyping and final design)

b. Process design

   i. Network design – considers the whole network of operations for a given product ("from dirt to dust"). At this stage we have to consider three issues:

      1. The shape of the network, distribution of influence and responsibilities leading to make/buy decisions (vertical integration),
      2. Decide upon location for each node, and
      3. Decide upon capacity of each node (chasing strategy, level and balancing)

   ii. Layout and flow – the physical location of all the machines, equipment, and stuff, as well as the flow of materials and information.

   iii. Process technology – decisions upon

      1. what technologies to use,
      2. the scale of automation to use
      3. the degree of integration of the technology

   iv. Job design – decisions upon the work methods to be used. The work methods define liaison between the people the used technology.

3. *Operating decisions* – production planning and control– decisions during the day-to-day operation after the production system was build; coordination of activities in the internal and external supply chain, forecast of demand, inventory control, resource planning, scheduling (prioritising of jobs).

4. *Improvement* – methods of improving existing processes.

Until recently, much of the research work in the production system area was directed towards problem solving rather than theory building. Models developed within the Operation Research area (OR) give us valuable insights in basic trade-offs when analysing the problems, but it is neither an explanatory nor predictive method. Also, the proposed solutions frequently fails, because models, often adapted to capability of the chosen solution techniques, cover only certain aspects of the analysed problems, neglect the effect of numerous factors and most of all lucks an ability to generate understanding (and this requires synthetic thinking [6]).

Recently the OM research community noticed the increasing importance of theory and theory driven empirical research and we observe an explosive transformation of operations management to a science discipline. This implies development of explanatory and predictive models of the operational processes – that is models, that could be used to explain or predict the output or performance of the process as a function of process characteristics, process states, and inputs to the process, as a main object of study in OM [7].

Joining these efforts, we positioned our research as a model-based approach to knowledge generation by building models explaining dynamic behaviour of real-life operational processes (an axiomatic descriptive research) and decision-making problems (an axiomatic normative research) for design and operation of manufacturing systems.

Our research is based on computer simulation and experimental design. We expect that much more complex problems may be studied than in the case of using mathematical models. The aim of our work is to create library with standard components and patterns that can easily be embodied and configured for solving particular OM problems.

As the OM decision types listed above are typical for most businesses, well demarcated, and may be defined on high level of abstraction, a tempting idea is to represent them in form of parameterised patterns consisting of primitive components. Use of object-oriented paradigm would considerably simplify the modelling and simulation process during evaluation of alternatives due to specialisation, class replacement and override mechanisms. Also the possibility of merging both continuous (system dynamics) and discreet time simulation in one and the same environment is crucial as well. The reason why we choose Modelica for the VISP project is, that it provides both of the features.

[3] Not discussed in this paper

3 Simulation support for decision-making.

Benefits of implementing simulation support should include increased quality of decision-making, knowledge preservation and thorough understanding of the structure of the problem as well as the decision-making process. Figure 3 shows a formalized model of the OM problem solving process [7,8]. Usually, such a process is an iterative process of incremental refinement aimed to guide the designer throughout the design process and ensure that all the necessary aspects were considered. Decision-making is associated with choice between a finite numbers of feasible alternatives generated in the synthesis and the analysis process respectively.

There are three facts indicating, that our efforts should end with promising results – firstly, *decision criteria* (cost, quality, dependability, flexibility, speed [5]), and inference mechanisms are to great extend recognised, and available in the OM literature, secondly, the evaluation of the design concepts may be categorised – typically we have to evaluate *feasibility* (is the concept feasible?), *acceptability* (how much does it cost?), and *risks* (what risks are connected with a particular solution?), and thirdly, recent development in knowledge engineering give us a solid ground in structuring of the decision making process [12].

4 The case study – the case specification

As an example of the method proposed above, in this paper we will evaluate an inventory system design problem called "one-machine-multi-storage-point" shown in Fig. 4. This is one of very typical problems in operation management. The inventory system has a number of products (that should be manufactured), each with associated variable daily demand, with different and variable lead times[4], and required service level[5] on the demand side. Manufacturing is organized in batches of identical products. To meet the required demand levels we have to determine safety stocks for each product, as well as a suitable prioritising strategy for the production process (the sequencing of jobs, i.e. production orders).

The *traditional approach* to modelling an inventory with continuous review (the well-known Wilson formula) has a number of presumptions: stock outs are prohibited; demand is known with certainty and is constant over time; lead-time is known and constant; the cost of the products is fixed; adequate capacity and capital exist to implement the suggested strategy; the strategy does not affect other products the organisation handles. Unfortunately, as we can realise no one of these presumptions holds in our case, and solving it mathematically is very hard if not impossible. Also Operation Research cannot suggest any solvable mathematical model of this problem (queuing networks are not applicable here).

Our approach is to build suitable simulation model to gain the understanding of the problem, generate possible solution alternatives, perform stochastic simulation with different input parameters and then choose the optimal alternative and optimal parameters.

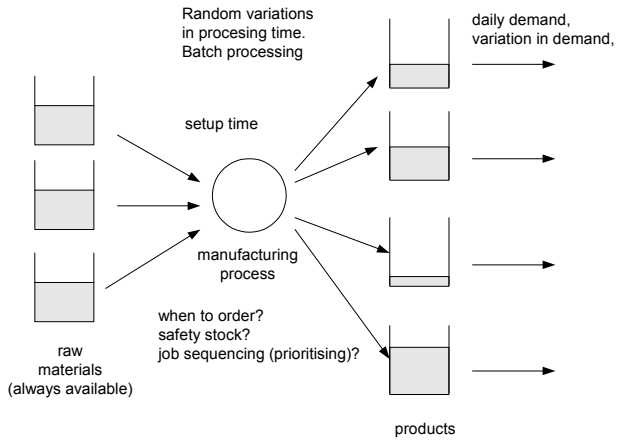

Fig. 4. The "one-machine-multi-storage-point" model.

With this example we assess the benefits of using Modelica as a simulation platform in operations management decisions.

The simulation model is shown in Fig. 5, and the object interaction diagram in Fig. 6 respectively.

[4] Lead time: the time required between placing an order and receiving the ordered product

[5] Service level: the desired probability of not running out of stock in any one ordering cycle, which begins at the time an order is placed and ends when it arrives in stock

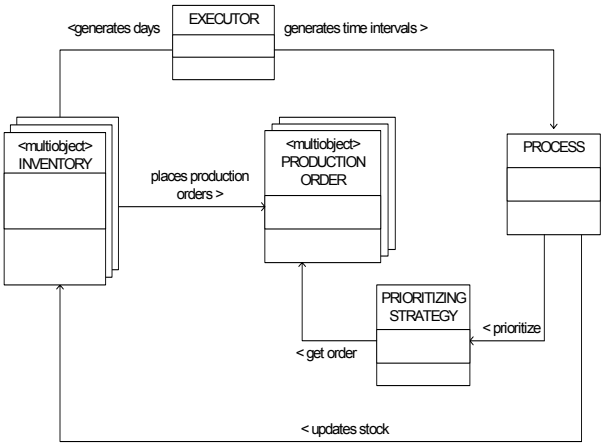

Fig. 5. The simulation model for "one-machine-multi-storage-point" problem, in UML notation. Note that in Modelica notation a specific prioritising strategy (such as "FIFO" or "least processing time – first") should replace the placeholder for "Prioritising Strategy".

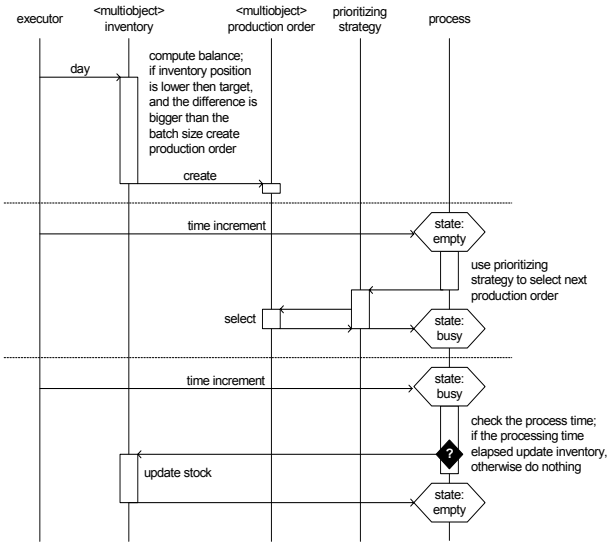

Fig. 6. The object interaction diagram for the simulation model in UML notation[6]

To find the dependency between specific prioritising strategy choice, size of safety stocks and the demand characteristics for each product we use tree-variable experimental design model [2], running each combination of values a sufficient number of times to achieve required confidence interval.

[6] The pictures 4 and 5 do not show data collection functionality that is "hidden" to keep simplicity

5 Using Modelica for the case study

A Modelica library for discrete event simulation targeted for operating decision support is under construction. The library contains components of business or manufacturing process, such as scheduled and random event generators, queues, delays, triggering actions, sequence schedulers, gathering statistics, etc. Therefore declarative style of modelling, high level of abstraction and ease of combining components into a working system is achieved. Some components, e.g. sequence schedulers are modelled using Modelica algorithms and functions. Signals should be used for synchronisation between the components. Since there is no explicit signal send/receive in Modelica, event-triggering mechanism will be used.

The DEVS Modelica library has been previously designed by Bunus [8]. The DEVS formalism is defined by Zeigler [9]. It describes basic discrete event entities from which complex models can be built. Atomic DEVS components are state machines; state depends on internal (timeout) and external (change in input port) components.

In addition to basic primitives, the library will contain *patterns*, i.e. collection of ready-made complex models, traditionally used in operation research. In order to simulate such patterns the user has to modify them by replacing necessary *replaceable* classes, as well as by parameterisation.

Since just triggering signals is not sufficient, in the library some non-trivial connectors should be used for communicating information about incoming orders and product movement.

Possible difficulties we expect with Modelica use for the operating decision simulation is connected with data handling. As noted by Remelhe [10], an event list of a scheduler has to be realized by a fixed length vector. If just queue length is simulated, an integer variable is sufficient to describe a queue. However if queue elements contain some essential information (product identification, time stamps), a dynamic data structure would be very convenient. These should be unnaturally modelled by globally accessible vectors, indexing mechanism and predetermined maximal length.

Adding full scale dynamism to data structures would certainly break nice declarative properties of Modelica, complicate the type system and would make implementation more difficult. As a first step to limited dynamism we would suggest introduction of arrays with dynamically changing size. Such array size should be a discrete integer variable. For

DES systems it is sufficient if all variables within dynamic arrays are discrete. Number of equations and variable is not known when system simulation starts but it is always a linear combination of all dynamic array lengths. For continuous systems things become more complex.

6 Test model

In order to test our ideas about future library a test package has been constructed in Modelica in order to simulate the "one-machine-multi-storage-point" model. The UML notation above has been taken as model specification. First a rapid prototype has been created in Mathematica in completely functional programming style. It took just few hours to write such model and test it for typical situations.

It has been more difficult task to do it with Modelica. We attempted to use declarative Modelica style for this package. Unfortunately most of computations here take place in conjunction with an inventory database. Such databases can be modelled in Modelica as arrays of records and records of arrays. However the visual structure of UML notation cannot be represented here directly. Our Modelica model contains two databases and one "sampler". The databases are used as input (**ddata** and **dbase**) and output (**dbase**) arguments of many functions. Separation of these databases to smaller components would lead to more inconvenient design. The **sampler** can be divided into two samplers – one for production time and one for day start. The sampler is based on an algorithm in with **when**-section.

Some functions (**normal**, **makeCriteria**) can be replaced by user-defined functions if necessary.

Simulation results are shown in Figure 8.

```
package ormodel13
  final constant Integer nr=2 "Number of products";
```

```
record ddata "data which are not returned from update functions"
  parameter Real[nr] demand "average product demand";
    parameter Real[nr] demandDistr
     "normal distribution of product demand";
    parameter Real[nr] processingTime
     "average processing time for a batch";
    parameter Real[nr] processingTimeDistr
      "its normal distrib ution";
    parameter Integer[nr] batchSize "required batch size";
    parameter Real[nr] wantedLevel "safety stock level";
    Real thetime "time of last event";
end ddata;
```

```
record dbase "data which returned from update functions"
  discrete Real[nr] level "current stock level";
  discrete Real[nr] backorder "current missed deliverables";
  discrete Real[nr] todayCustomerOrder "today's demand";
  discrete Integer[nr] orderedBatches "computed nr of batches";
  discrete Real[nr] criteria "computed criteria for sorting";
  discrete Integer[nr] ordering "ordering of producion orders after sorting";
  discrete Integer orderindex "currently processed production order";
end dbase;
```

```
function busyFun "actual (random) proceessing time for each batch"
  input dbase db;
  input ddata dd;
  output Real busyVar;
algorithm
  busyVar := normal ( dd.processingTime [ db.ordering [ db.orderindex ]], dd.thetime);
end busyFun;
```

```
function normal
  "random number, distributed in some way; should be replaced" " by some more meaningful distribution law"
  input Real mean;
  input Real thetime;
  output Real randval;
algorithm
  randval := mean + mod(thetime, 0.5) - 0.25;
end normal;
```

```
function updateinventory
  "updates inventory when production order has been fullfilled"
  input dbase dbin;
  input ddata dd;
  output dbase dbout "returns updated database here";
protected
  Real production;
  Integer productidx;
algorithm
  dbout := dbin;
  productidx := dbout.ordering[dbout.orderindex];
  production := dbout. OrderedBatches [ dbout . orderindex ] * dd
.     batchSize[ productidx];
  (dbout.level[productidx],dbout.backorder [productidx ] ) := update ( dbout , productidx, production);
end updateinventory;
```

```
function update "updates inventory information for a single product"
  input dbase db;
  input Integer pindex "product index";
  input Real production "amount of produced product";
  output Real newlevel;
  output Real newbackorder;
algorithm
  newbackorder := db.backorder[pindex] + db . todayCustomerOrder[pindex];
  if (production + db.level[pindex] >= newbackorder) then
    newlevel := db.level[pindex] + production - newbackorder;
    newbackorder := 0;
  else
    newbackorder:=newbackorder-(production+db.level[pindex] );
    newlevel := 0;
  end if;
end update;
```

```
function endday "statistics can be gathered here"
  input dbase dbin;
  output dbase dbout;
algorithm
  dbout := dbin;
```

```
  end endday;
```

```
  function makeCriteria "computes criteria; can be replaceable function"
    input dbase dbin;
    input ddata dd;
    input Integer i;
    output dbase dbout;
  algorithm
    dbout := dbin;
    dbout.criteria[i] := dd. ProcessingTime [i] * dbout . orderedBatches [i]
      "criteria is shortest processing time first. Here the prioritising strategy is defined";
  end makeCriteria;
```

```
  function sort "bubble-sorting"
    input Real[:] qinp "sequence of criteria values";
    output Integer[nr] ordering "permitation";
  protected
    Real qtmp; Integer itmp; Real[nr] q;
  algorithm
    q := qinp;
    for i in 1:nr loop   ordering[i] := i;  end for;
    for i in 1:nr loop     for j in 1:nr - 1 loop
       if (q[j] > q[j + 1]) then
         qtmp := q[j];q[j] := q[j + 1];q[j + 1] := qtmp;
         itmp := ordering[j];ordering[j] := ordering[j + 1];
         ordering[j + 1] := itmp;
       end if;     end for;    end for; end sort;
```

```
  function startday "preparations at start of the day"
    input dbase dbin;
    input ddata dd;
    output dbase dbout;
  algorithm
    dbout := dbin;
    for i in 1:nr loop
      dbout.todayCustomerOrder[i] := normal(dd.demand[i], dd. thetime ) ;
    end for "today's demand randomly chosen";
    for i in 1:nr loop
      dbout.orderedBatches[i] := integer((dd.wantedLevel[i] + dbout.backorder[i]+dbout.todayCustomerOrder[i] - dbout.level [i] ) / dd.batchSize[i]);
    end for "necessary number of batches";
    for i in 1:nr loop
      dbout := makeCriteria(dbout, dd, i);
    end for;
    dbout.ordering := sort(dbout.criteria);
    dbout.orderindex := 0
      " production orders are performed according to this ordering now ";
  end startday;
```

```
  model sampler "production unit and sampler"
    outer dbase db;
    outer ddata dd;
    discrete Real nextSampling(start=0.1) "when unit will become free";
    discrete Real busyTime(start=1) "duration  of production time";
    discrete Real daystart(start=0) "when day started";
    Boolean signal "the signal could be sent over to other components,
  but this is not needed";
  algorithm
    signal := time >= nextSampling;
    when (pre(signal)) then

      if db.orderindex < nr then
        db.orderindex := db.orderindex + 1 "start next order";
        dd.thetime := time "used for randomizer";
        daystart := pre(daystart);
        db := updateinventory(db, dd) "we assume that order is done";
        busyTime := busyFun(db, dd) "production time for a new order";
      else
        busyTime := 1 "stay idle one more hour";
      end if;
      nextSampling := pre(nextSampling) + busyTime;

      if (time - daystart > 24) then
        daystart := time;
        db := endday(db);
        db := startday(db, dd) "new day started";
      end if;
    end when;
  end sampler;
```

```
  model factory
    inner dbase db(
      backorder(start={0,0}),  level(start={3,4}),
      todayCustomerOrder(start={10,12}),
      orderedBatches(start={0,0}),   criteria(start={0,0}),
      ordering(start={1,2}),  orderindex(start=1));

    inner ddata dd ( demand={5,6}, demandDistr={2,2}, processingTime={2,3} , processingTimeDistr = {2,2}, batchSize = {3,3} ,  wantedLevel={5,5});

    sampler S;
  end factory;
```

```
end ormodel13;
```

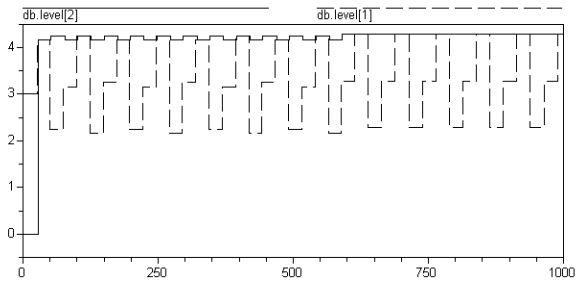

Fig. 8. Result of simulation of the "factory" model for 1000 hours. Production stock levels do not exceed the safety stock ({5,5}), and no backorder occurs.

An alternative sampler based on an **when**-equations would allow to build a structure consisting of models and blocks instead (or in addition to) functions. This approach is the next step in our ongoing work.

7 Future research

The plans for research include design of multiple modelling patterns for typical operation management problems. Also it is necessary to create user-friendly modelling and simulation tools that can easily be configured for solving particular problems of business operation management.

8 The VISP project

The VISP project [11] is a cooperation between the departments of Machine Design and Production Engineering at KTH, The Dept. of Computer Science at Linköping University, IVF, The Institute for Engineering Science at Skövde University College, and a number of Swedish companies.
The expected output is the development of an information platform for industry-adapted product realization based on a common, integrated map over workflows and data access during concurrent development of a product program and a manufacturing system. The work includes development of methodology, modularisation and configuration of simulation models of products and production systems, a pilot installation of the methodology with a commercial software, and evaluation of the achieved results in several real industrial cases.

References

[1] Zahn, E., Dillerup, R., Schmid, U.: "Strategic evaluation of flexible assembly systems on the basis of hard and soft decision criteria". System Dynamics Review Vol.14, Winter 1998.
[2] Bergman, B. "Industriell försöksplanering och robust konstruktion", Studentlitteratur 1992
[3] Fabricius, S. "SystemDynamics Modelica Library", available from www.modelica.org, 2002
[4] Forrester J. "Principles of Systems, Wright-Allen Press, Cambridge, U.S.A, 1969
[5] Slack, N., et al. "Operations Management". Prentice Hall 2001
[6] Ackoff. Russell, L. Operation Research: „after the post mortem". "System Dynamics Review" Vol. 17. No4 Winter 2001
[7] J Will M Bertrand; Jan C Fransoo. "Operations management research methodologies using quantitative modeling", International Journal of Operations & Production Management, Year: 2002 Volume: 22 Number: 2 Page: 241 – 264
[8] Bunus P., Fritzson, P., DEVS-Based Multi-Formalism Modeling and Simulation in Modelica. Proc. of the 2000 Summer Computer Simulation Conf. (Vancouver, Canada, Jul. 16-20).
[9] Zeigler, B.P., Multifaceted Modeling and Discrete Event Simulation, Academic Press. London, 1984
[10] Remelhe, M.A.P., Combining discrete event models and Modelica – General thoughts and a Special Modelling environment. Proc. of Modelica 2002 conf. (Oberpfaffenhofen, Germany)
[11] VISP Project site, http://extra.ivf.se/VISP/
[12] Schreiber, G., at al, "Knowledge engineering and management", MIT Press, 2000.
[13]. Wu, B., "Manufacturing systems design and analysis", Chapman&Hall, 1992.