SIMPLIFICATION BY PLANNING: THE THIRD WAY

Raimo Keloharju
Helsinki School of Economics
Runeberginkatu 14-16, Helsinki 10 Finland

ABSTRACT

This paper puts forward the idea that essential parts can be removed from a SD model without harm if the model is revised successively during the simulation. The revision requires an optimization package, called DYSMOD (Dynamic Simulation Model Optimizer and Developer). A multi-stage production model, developed by J.M.Lyneis, is used for demonstrating that the idea works in practice.

Conceptual background

Classical system dynamics can be described as being based on a concept of pre-understanding. The process starts by considering a reference mode over time concerning the real world behaviour of interest. A feedback model is conceptualized from the reference mode to explain the observed behaviour. This creates a dynamic hypothesis. The computer is then used as a fast calculating machine to check if the model can reproduce the reference behaviour and hence substantiate the hypothesis.

Revisions to model parameters are made manually until the model can achieve this objective. When it does so, the model is considered validated and appropriate for use in designing other system behaviour modes by making further parameter and structural changes. This process is shown as an iterative procedure in Fig.1.

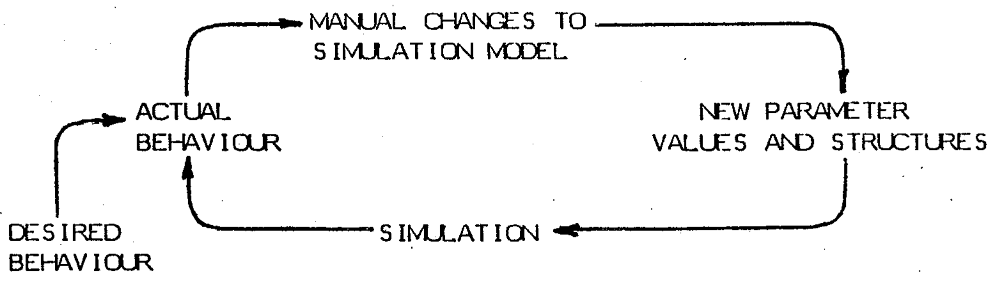
MANUAL CHANGES TO SIMULATION MODEL
ACTUAL BEHAVIOUR
NEW PARAMETER VALUES AND STRUCTURES
DESIRED BEHAVIOUR
SIMULATION

Fig.1. The process of model design in classical SD

The concept of dynamic optimization in SD is based on a belief that the manual procedure of system design given in Fig.1 can be automated (Keloharju, 1983). It is done by interfacing a heuristic optimization algorithm with a system dynamics simulation programme. The optimization algorithm used is the extended Search Decision Rule algorithm (SDR) and the system dynamics software used is the Dynamic System Modelling Analysis Programme (DYSMAP). The combined programme is known as DYSMOD (Dynamic Simulation Model Optimizer and Developer). The process of model design that uses it is shown in Fig.2.

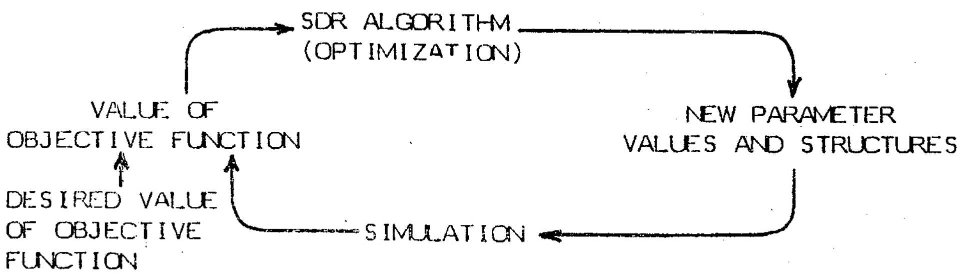

Fig.2. The process of model design in optimized SD

Fig.2 shows how the combination of optimization and simulation seeks a solution iteratively in optimized system dynamics. Before starting, it is necessary to take two steps. Firstly, an objective function must be defined within the simulation model which summarizes the overall model behaviour. Secondly, a number of parameters within the model must be chosen as candidates for optimization together with a range of feasible numerical values for each.

Each iteration starts with a simulation run which calculates the value of the objective function chosen within the initial conditions set for the simulation parameters. The SDR algorithm then treats these parameters as variables for optimization and optimizes them heuristically. In other words, the algorithm changes them one at a time using the objective function as a measure of performance. The output from the optimizer comprises a new set of parameter values.

Subsequent iterations repeat this cycle by using the modified parameters. The algorithm compares the value of the objective function at the end of the simulation run to the corresponding value from the best run received. Although the objective function may fluctuate from one iteration to another, the best solution is always kept stored in the computer memory. This guarantees that optimization never makes the initial situation worse.

The modeller must decide on the number of iterations which are appropriate for achieving some desired value of the objective function. The number chosen is, however, not too important as the whole procedure can be subsequently continued until the value of the objective function stabilizes.

Additional pseudo-parameters can be introduced into the simulation model for the purpose of optimization. This is a powerful use of optimization since it creates a means of carrying out structural rather than straight parameter optimization.

The optimizer revises the modelling culture in SD. If the optimizer knows which way to go during the search for better models, the initial model need not be of high quality. Pre-understanding is not necessary since the modeller can learn from the computerized modelling process. In this way, he acquires post-understanding.

System dynamicists have traditionally thought that the role of models is to represent complex sociotechnical systems. The opposite viewpoint is to accept models as thinking aids only. Two quotations from Forrester indicate that a search for balance between the opposite roles might be the next in line:

> "All systems that change through time can be represented by using only levels and rates. The two kinds of variables are necessary but at the same time sufficient for representing any system" (Forrester, 1971)

> "The proper balance between size and simplicity suggested simplification (in the National Model)" (Forrester, 1986)

Several methodological choices are available to solve problems of model simplification. Keloharju and Luostarinen have developed a 'black box' approach, based on extended use of the optimizer (Keloharju and Luostarinen, 1983). In this methodology, the computer simplifies a model structure by partitioning it in parameter space. The work is done by algorithms which calculate model sensitivity to various parameters. The procedure is fully automatic and has several steps. Wang and Yan have recently introduced a mathematical method which enables the partitioning of SD models for simplification (Wang and Yan, 1986).

So far, it has been assumed that a model has redundancy that should be removed. Suppose, however, that the reverse is true: a model is so simple that it is deficient. This paper will show that even such a model works well if it is revised from time to time. This is the third way of simplifying and it requires the explicit use of planning. A well known model by Lyneis (Lyneis, 1980) serves as a demonstration case in experiments that will be reported below.

The Lyneis model

Lyneis has a simple two-stage production model in his book 'Corporate Planning and Policy Design'. Fig.3 gives the influence diagram of the model, Appendix A lists it. The present author has added six pseudo parameters (A1,A2,B1,..,B4) and an objective function (OBJF.

The objective function accumulates production fluctuations (GOAL1), raw-material inventory (GOAL2) and the error term of finished inventory (GOAL3). The sum of these weighted (WG1,WG2,WG3) components is minimized.

Any influence diagram can be presented in open form. Fig.4 shows the Lyneis model after the required transformation. Connections between the information flows and the physical flow were cut off in the sources of information. The objective function has been added to the figure.

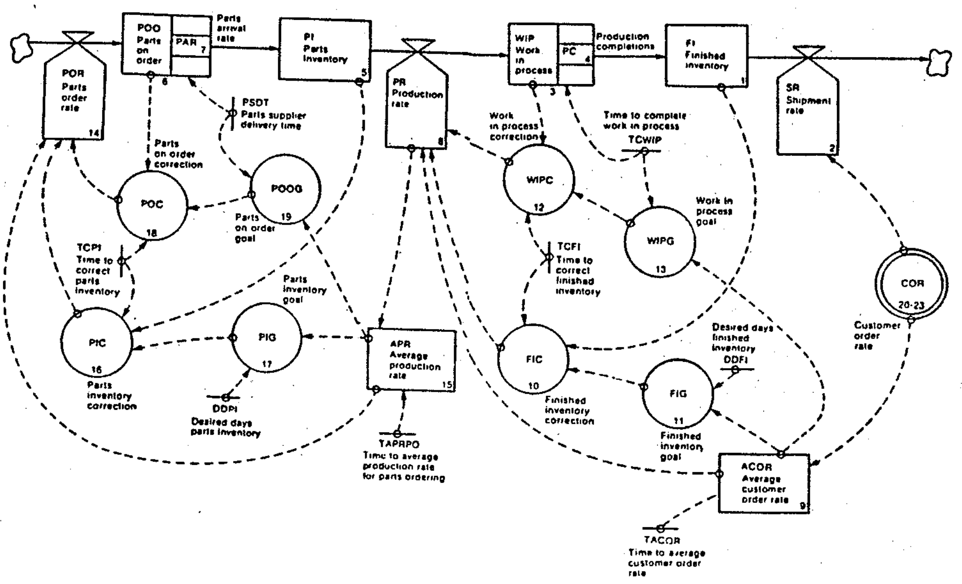
POO Parts on order 6
PAR 7
Parts arrival rate
PI Parts inventory 5
WIP Work in process 3
PC 4
Production completions
FI Finished inventory 1
POR Parts order rate 14
PR Production rate 8
SR Shipment rate 2
PSDT Parts supplier delivery time
Parts on order correction
POC 18
POOG 19
Parts on order goal
Work in process correction
Time to complete work in process
TCWIP
WIPC 12
WIPG 13
Work in process goal
TCPI Time to correct parts inventory
Parts inventory goal
TCFI Time to correct finished inventory
COR 20-23
Customer order rate
PIC 16
Parts inventory correction
PIG 17
DDPI
Desired days parts inventory
APR Average production rate 15
TAPRPO Time to average production rate for parts ordering
FIC 10
Finished inventory correction
FIG 11
Finished inventory goal
Desired days finished inventory
DDFI
ACOR Average customer order rate 9
TACOR Time to average customer order rate

Fig.3. The influence diagram of the Lyneis model

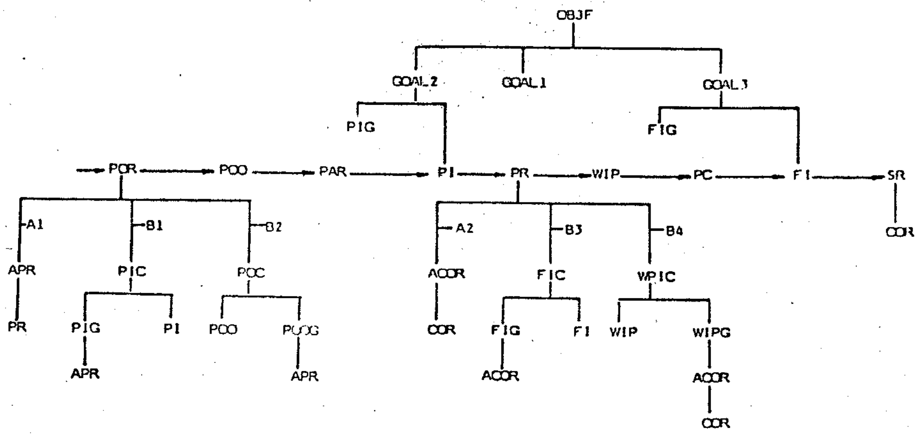

Fig.4. The Lyneis model after rearrangements

The model has three parts: objective function, physical flow and model structure. The physical flow flows from parts order rate (POR) to sales (SR) through intermediate stages. It is the horizontal flow in Fig.4. The objective function and its components are above the physical flow, the information network is below it.

Let us look at the part of the information network which defines Production Order Rate, POR. Suppose that parameters B1 and B2 are given the value of zero. The reduced model is an oversimplified version of the original model, which has now lost certain essential parts.

Double-dynamic optimization

An acceptable model must be created in parameter space before the 'real time' simulation begins. The optimizer does this work of dynamic optimization by repeated simulation. The cycle that starts at time 0 in Fig.5 is the same as in Fig.2. The time horizon of the model equals the length of the cycle.

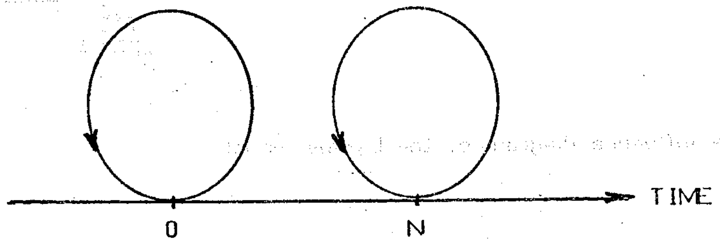

Fig.5 Double-dynamic optimization illustrated

Fig.5 is needed to understand the difference between dynamic and double-dynamic optimization. In double-dynamic optimization, the model is revised at least once during the simulation and that is shown by point N on the time axis. The length of the cycle is now called 'planning horizon'. The distance between two successive cycles, like 0-N, is called 'action time'.

Solution interval DT is a technical parameter which separates two groups of successive calculations from each other at times 'J' and 'K'. It creates the dynamics of a discrete model that should approximate the behavior of the corresponding continuous model. When DT is made too long, even a stable model may explode (Forrester, 1968).

The planning horizon gives the modeller the possibility to distinguish between two successive models which were developed by the computer. The action time is the distance between them. It is conceptually similar to DT and should be seen as a technical parameter. As the action time links models instead of variables of the model in time, it also has to react to unexpected changes of importance. DYSMOD has the capacity to do that but it will not be discussed here.

Fig.6 summarizes eight experiments that were performed to find out the usefulness of oversimplified models.

Run	Description		Total cost	Relative cost	Relative length of action time
1	simulation		21.9E12	100%	
2	dynamic optimization		75.1E11	34.2%	
	Double-dynamic optimization, Complete model				
	Planning horizon	Action time			
3	480	240	68.2E11	31.1%	10%
4	240	120	65.2E11	29.8%	5%
5	120	60	62.5E11	28.5%	2.5%
	Double-dynamic optimization, Reduced model				
	Planning horizon	Action time			
6	480	240	30.1E12	137.1%	10%
7	240	120	85.2E11	38.9%	5%
8	120	60	63.9E11	29.1%	2.5%

Fig.6. The cost comparison of eight experiments

Run 1 in Fig.6 relates to the original Lyneis model. The total cost, generated by objective function OBJF, is taken as 100% as it gives a yardstick for cost comparisons to follow.

The model was optimized in run 2 by taking the pseudo-parameters A1,A2,B1,..,B4 into optimization. Their ranges were defined from 0 to 3. As a compromise between convergence of the solution and computing time, the number of iterations was chosen as 150 in runs 2 to 8.

All pseudo-parameters were used for optimization in runs 3-5. In runs 6-8, A1 and A2 were optimization parameters and B1,..,B4 had the value of zero. In this way, the complete model was changed to a reduced version without feedback loops. To simplify the experiment, the length of planning horizon was always twice the action time.

Fig.6 shows that

(a) it was possible to improve the simulation model (run 1) by dynamic optimization. This bears out what we have learnt from experience namely that a simulation model can always be improved by the optimizer.

(b) double-dynamic optimization (runs 3-5) gave better results than dynamic optimization (run 2) when the relative length of action time varied from 2.5 to 10%

(c) the quality of double-dynamic optimization in the reduced models (runs 6-8) improved very rapidly when the action time was shortened. The cost exceeded the cost obtained from the simulation model when the action time was 10% of the total runlength. In the last run, the cost was close to the cost generated from the complete model with equal action time. This means that feedback information from the correction terms (PIC,POC,WPIC,FIC) had only marginal value!

System dynamicists extend their models by redefining some model parameters as new variables and, after that, by adding the necessary causal relationships. In runs 6-8, the 'extended' model was reduced, however, by allowing some model parameters (A1,A2) to take charge of omitted variables (PIC,POC,WPIC,FIC).

Reduced models were based on two kinds of trade-offs:

(a) Parameter A1 in run 8 corresponds to A1,B1 and B2 in run 5. Therefore there is a trade-off between parameter aggregation and the frequency of model revision.

(b) Either continuous local information or discontinuous global information can be used. The earlier is obtained from the correction terms (via B1,..,B4); the latter relates to the values of A1 and A2. Figures 7 and 8 show plottings from runs 1 and 8. The improvement looks impressive.

It is generally accepted that models should be as simple as possible. A double-dynamic model, however, can be oversimplified without harmful side-effects. Such a model works well for a short time; after that, change it!

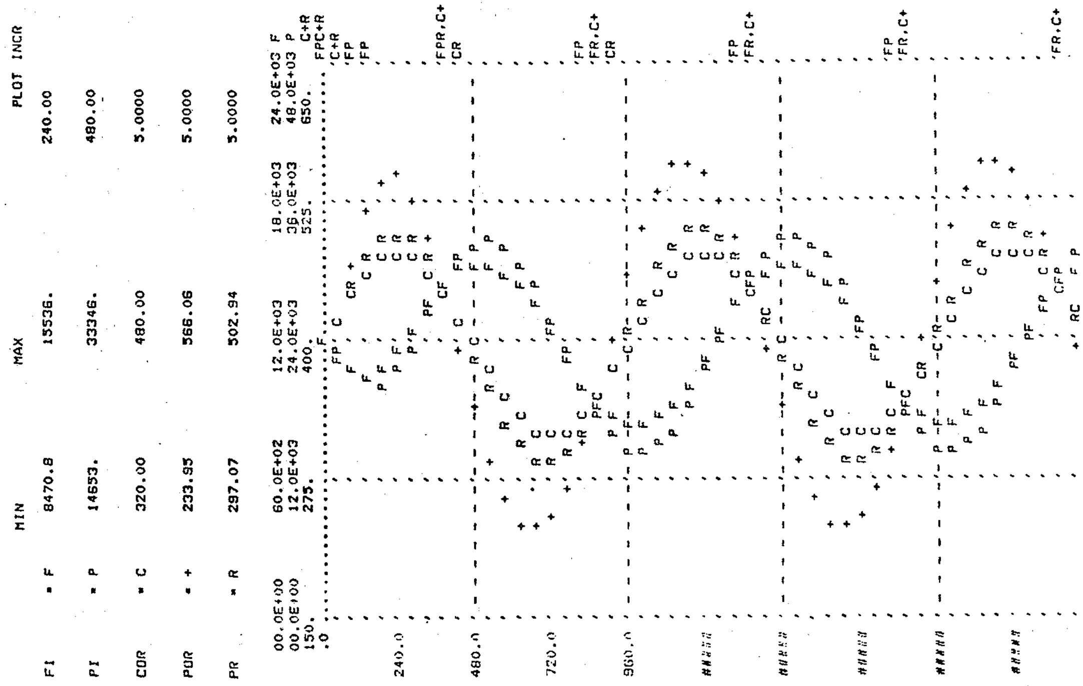

Fig.7. Some plottings from run 1

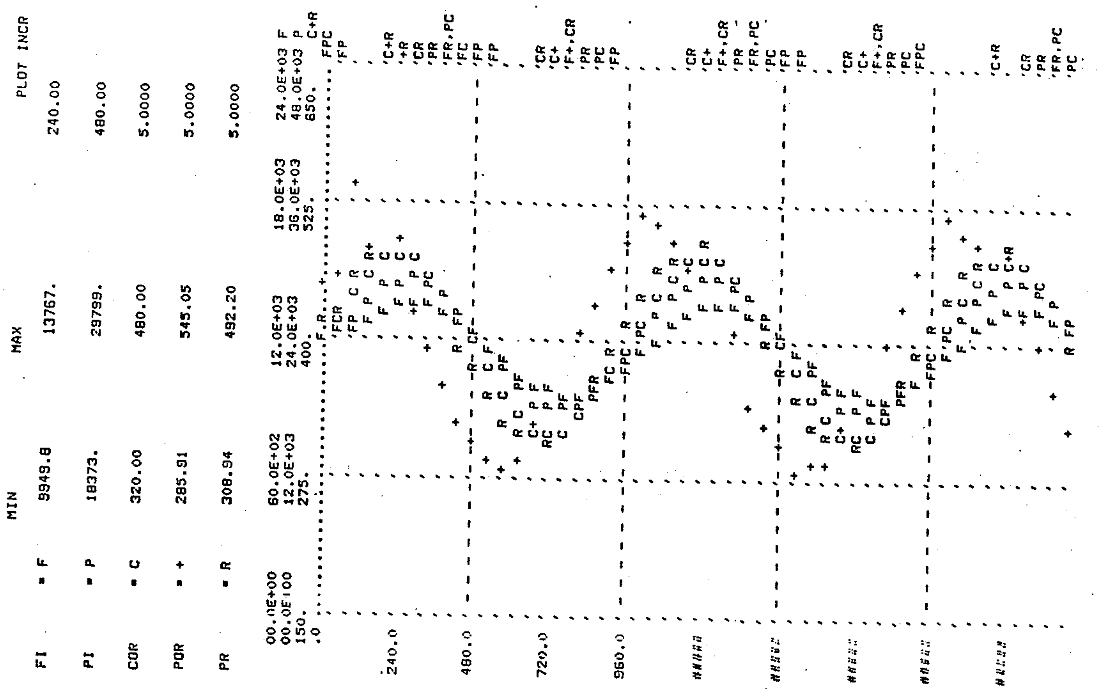

Fig.8. Some plottings from run 8

Conclusions

Conventional SD has two axioms, relating to the importance of model structure and feedback. The oversimplified model violates them since feedback loops were removed and model structure was deliberatly deficient. The control was transferred to implicit feedforward paths via the objective function. Here feedforward means the use of future information, derived from the past, to present needs.

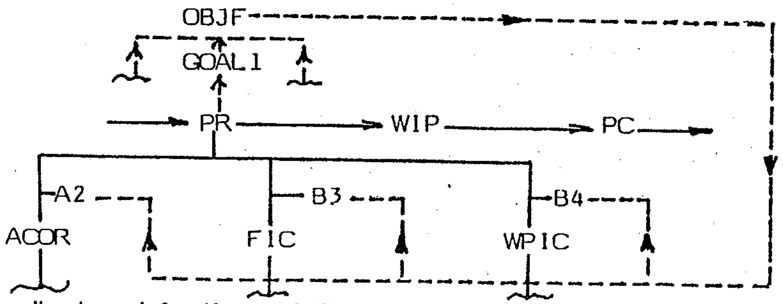

Fig.9. Feedback and feedforward illustrated

The relationship of feedback to feedforward is seen in Fig.9. The model structure with its feedback loops is shown as an 'organization chart', drawn in a continuous line. The dotted lines depit feedforward paths at the end of planning horizon.

A model which cannot be replaced has to be robust in order to safeguard against uncertainties. Reduced models have a life-length which equals the action time that was chosen. Therefore they need not be robust.

Figure 10 presents a framework which might be called systems cross'. It consists of two concept pairs and can be used for classifying different modelling approaches in SD.

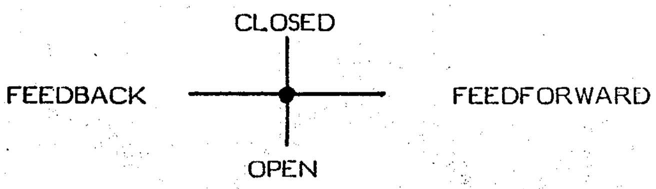

Fig.10. The systems cross

Fig.11 shows all configurations which can be derived from the systems cross. Their meanings will then be explained.

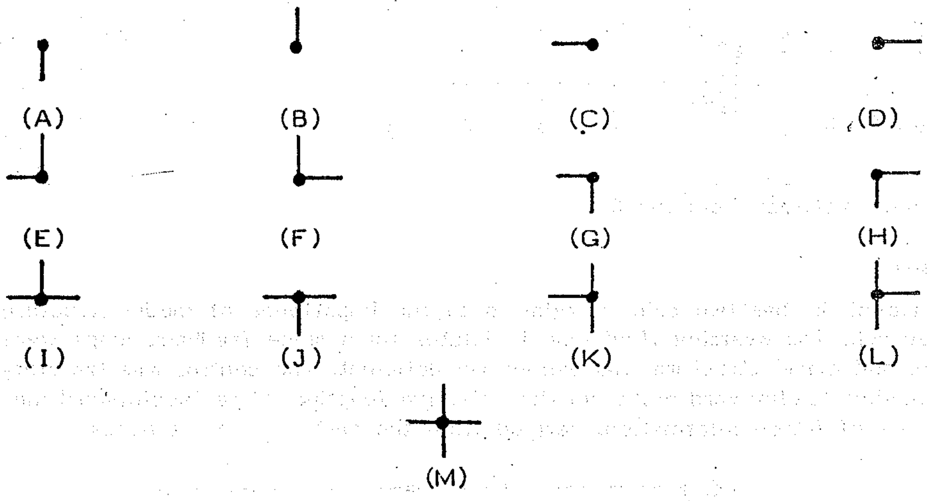

Fig.11. The configurations of systems cross

Fig.11 includes models which can be interpreted as follows:

(A) Open models without feedback or feedforward. Linear programming models, for example, belong to this group

(B) Closed models without feedabck. Not feasible.

(C) Feedback models which are not closed. Not feasible.

(D) Feedforward models which are neither open or closed. Not feasible.

(E) Closed feedback models. 'Purists' are system dynamicists who do not accept partially open models. Models of the purists belong to this group.

(F) Closed feedforward models without feedback. The objective function itself is now the model to be optimized. The idea was put forward by Taubert in his

(G) Open feedback models. Not feasible.

(H) Open feedforward models. All feedforward models are closed via implicit links from the objective function to optimization parameters. Not feasible.

(I) This is an extension of case (E). The model is solved by optimization.

(J) Open feedback models. Not feasible.

(K) Also this is an extension of case (E) as a non-purist also accepts exogeneous real-life variables to his model.

(L) A feedforward model which is both open and closed. Run 8 is an example of case (L).

(M) This can be interpreted either as the extension of case (K) or (L). The model is now partially open; see run 5.

The systems cross is reduced to a simpler form where 'open models' and 'closed models' are replaced by 'model structure' :

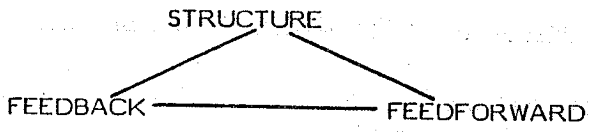

Figure 12. The systems triangle

Figure 12 summarizes the evolution of the SD paradigm. The classical SD needs two corner points of the triangle, i.e. feedback and structure. Reduced models need the remaining corner point of feedforward. They are complementary approaches and each may (or may not) work well.

Forrester has emphasized model acceptability instead of some statistical validation criteria (Forrester, 1961). A model is then valid when the structure of the model is meaningful, the model behaves as anticipated and it is accepted by the users.

If the model cannot be revised when in use, the validation procedure has to be based on descriptive validation. The model to be validated then describes a situation which is or was, not as it should be.

Since models are replaceable when needed, the validation procedure could be less formal than before. It is not any more a question of how man makes decisions but how man and the computer should make them in symbiosis. This viewpoint requires a great deal more maturity from decision makers than has been the case in the past.

Reduced models from double-dynamic optimization are created by a black-box, i.e. the optimizer. The idea may sound strange but it already has a real life counterpart. OPT (Optimized Production Technology), which combines the concepts of MRP and Japanese JIT-production, is based on similar principles (Goldratt and Cox, 1984).

REFERENCES

Forrester, J.F. (1961), Industrial Dynamics. The M.I.T. Press and John Wiley, New York, London

Forrester, J.W. (1968), Principles of Systems. The M.I.T. Press, Cambridge, Mass.

Forrester, J.W. (1971), World Dynamics. Wright-Allen Press, Cambridge, Mass.

Forrester, J.W. (1986), "Lessons from System Dynamics Modeling". The paper presented in the 1986 International Conference of the System Dynamics Society in Sevilla. October 1986, pp 1-16

Goldratt, E.M. and Cox, J. (1984), The Goal: Excellence in Manufacturing. Creative Output (Netherlands) BV. Printed in the United States

Keloharju,R. (1983), Relativity Dynamics. Acta Academiae Oeconomicae Helsingiensis, Series A:40

Keloharju, R and Luostarinen, A. (1983), "Achieving Structural Sensitivity by Automatic Simplification". Dynamica, Vol.9 Part 2, pp 60-65

Lyneis, J.M., (1980), Corporate Planning and Policy Design: A System Dynamics Approach. The M.I.T. Press

Taubert, W.H. (1968), "A Search decision Rule for the Aggregate Scheduling Problem". Management Science, Vol.14 N.o 6, pp B343-B359

Wang,Q and Yan,G (1986), "Studying the Relationship Among the Whole, Parts and Environment of a System Dynamics Model". The paper presented in the 1986 International Conference of the System Dynamics Society in Sevilla. October 1986, pp 501-509

APPENDIX A

```
* JANUARY 5,1987  **PRODUCTION MODEL BY LYNEIS**
R POR.KL=A1*APR.K+B1*PIC.K+B2*POC.K
L POO.K=POO.J+DT*(POR.JK-PAR.JK)
N POO=PSDT*CCOR
R PAR.KL=DELAY3(POR.JK,PSDT)
C PSDT=60
L PI.K=PI.J+DT*(PAR.JK-PR.JK)
N PI=DDPI*CCOR
A PIC.K=(PIG.K-PI.K)/TCPI
C TCPI=60
A PIG.K=DDPI*APR.K
C DDPI=60
A GOAL2.K=(PIG.K-PI.K)*(PIG.K-PI.K)
A POC.K=(POOG.K-POO.K)/TCPI
A POOG.K=PSDT*APR.K
A APR.K=SMOOTH(PR.JK,TAPRPO)
C TAPRPO=60
R PR.KL=A2*ACOR.K+B3*FIC.K+B4*WIPC.K
L PRX.K=PRX.J+DT*(PR.JK-PRX.J/DT)
N PRX=PR
A GOAL1.K=(PR.KL-PRX.K)*(PR.KL-PRX.K)
L WIP.K=WIP.J+DT*(PR.JK-PC.JK)
N WIP=TCWIP*CCOR
R PC.KL=DELAY3(PR.JK,TCWIP)
C TCWIP=20
L FI.K=FI.J+DT*(PC.JK-SR.JK)
N FI=DDFI*CCOR
R SR.KL=COR.K
A ACOR.K=SMOOTH(COR.K,TACOR)
C TACOR=60
A FIG.K=DDFI*ACOR.K
C DDFI=30
A GOAL3.K=(FIG.K-FI.K)*(FIG.K-FI.K)
A FIC.K=(FIG.K-FI.K)/TCFI
C TCFI=60
A WIPC.K=(WIPG.K-WIP.K)/TCFI
A WIPG.K=TCWIP*ACOR.K
A COR.K=CCOR*(1+STEP(COSH,COST)+ACOS*SIN(6.28*TIME.K/PCOS))
C CCOR=400
C COSH=0
C COST=0
C ACOS=0.2
C PCOS=960
C A1=1
C A2=1
C B1=1
C B2=1
C B3=1
C B4=1
NOTE OBJECTIVE FUNCTION
A OBJ.K=WG1*GOAL1.K+WG2*GOAL2.K+WG3*GOAL3.K
C WG1=1.5E4
C WG2=1
C WG3=1E3
L OBJF.K=OBJF.J+DT*OBJ.J
N OBJF=0
C DT=2
C LENGTH=2400
C PLTPER=48
C PRTPER=2400
PLOT FI=F(0,24000)/PI=P(0,48000)/COR=C,POR=+,PR=R(150,650)
PRINT GOAL1,GOAL2,GOAL3,OBJF
RUN
```