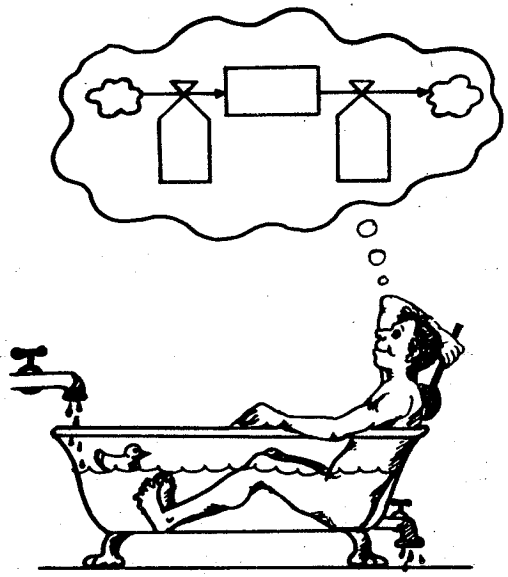
EXPERIENCES IN TEACHING SYSTEM DYNAMICS

Leif Gustafsson and Miroslaw Wiechowski
Uppsala University Data Center
Box 2103, S-750 02 Uppsala, SWEDEN

ABSTRACT

The scope of this paper is to present our views on teaching System Dynamics and Dynamo in our courses in Systems Analysis at Uppsala University. We treat the pedagogical aspects as well as the hardware and software system we built around Dynamo. A large part of this article is devoted to ideas and constructive criticism of System Dynamics and Dynamo which we have acquired from our experiences in education and research.



1. INTRODUCTION

The Systems Analysis Group at Uppsala University is located in the Institute of Technology and at the University Data Center in Uppsala (UDAC). The Group has, during the last five years, grown very rapidly and is now working with a large number of projects, especially in the fields of medicine, biology, and agriculture. Several courses for students of mathematics, technology, biology, and agriculture, as well as for graduate students from various faculties are regularly held.

In both our research projects and courses, System Dynamics and Dynamo play an important part. We have found it an ingenious tool for teaching model building, simulation and also various techniques such as identification, optimization and prediction, often in the form of a project. Therefore, for education and research purposes we have built up a large system of software and course material.

2. TEACHING SYSTEMS ANALYSIS

Our group works with education and research in Systems Analysis. We usually subdivide this field into:

1. Theory and concepts.
2. Working methodology.

3. Techniques.

The introductory course covers all these items whereas other courses may emphasize one or several techniques like OA-methods, simulation, or other topics.

Course materials are largely based on our books: "System och modell" (ref. 1) which deals mainly with System Dynamics ideas and "Modellbyggnad och simulering i Dynamo" (ref. 2) which fills the prerequisites for learning Dynamo and covers the complete Dynamo language. It also includes a number of examples with solutions provided in the form of flow diagrams and programs.

For educational purposes we have developed a number of exercises, especially from the fields of biology, agriculture, and medicine and also a number of laboratory experiments for e.g. teaching Dynamo language, optimization, prediction, sensitivity analysis, identification and other techniques.

In the theory section we deal first with the concepts of system, system conceptualization, and model formulation. We stress that the whole modelling process is controlled by the purpose of the study. The underlying philosophical strength of the holistic view is also accentuated.

We then examine various system classifications: static vs dynamic, linear vs nonlinear, deterministic vs stochastic, time-discrete vs continuous and descriptions: internal vs external (black-box).

Subsequently we discuss system description techniques: causal-loop diagrams and Dynamo flow diagrams, which we use extensively. Students in mathematics, science and technology are also given the equivalent descriptions in mathematical terms. In other courses these diagrams constitute the fundamental tools for analysis of dynamic systems.

We then stress the importance of exposition of relationships between the structure and the behaviour of dynamic systems on the basis of the System Dynamics approach. The systems are classified as open or closed and such notions as positive and negative feedback and order of the system are considered. Laboratory exercises on these topics are also included in several courses.

We consider a number of notions from automatic control, e.g. feedback, load, stability, sensitivity, feedforward etc. and we also deal with quantitative descriptions of systems through their characteristics such as stability, equilibrium, accuracy, sensitivity, controllability, observability, and concepts like step response, rise time, transient, and stationary behaviour.

Finally we have a section on complex systems and counter-intuitive behaviour in the classical System Dynamics style.

In many courses the students are required to carry out a project on a certain theme and to present the work as a written report.

Much attention is focused on approaching a project in a systematic way. Following this method we perform the steps from

problem identification and formulation, via modelling, validation and solution to result evaluation and presentation, where we also stress the iterative nature of this work. In this context we treat various aspects of cooperation within the frame of a project. System Dynamics and flow diagrams are then often useful, especially for communication and documentation.

Systems Analysis contains a vast number of techniques. In most courses we choose to put emphasis on continuous simulation using Dynamo. Discrete simulation, another field of our interest, is taught through DEMOS (ref. 3) and/or SIMULA (ref. 4).

In courses at the Departments of Mathematics and Engineering, Operational Analysis methods are important.

Continuous models and simulation are the dominating elements in several of our courses. Here we use System Dynamics and Dynamo almost exclusively. An exception is made when demonstrating the differences and similarities of various continuous simulation languages. Then we use SIMNON (ref. 5) which implements a state space approach and works with high-resolution screens.

Model construction and simulation constitute the foundation to which concepts and techniques, such as objective function, sensitivity analysis, optimization, identification, and prediction are added. This is also practiced generally in laboratory exercises.

The education scheme described above puts demands on:

- written materials such as books, manuals and laboratory guides, (already treated above).
- hardware and software.
- pedagogical approach.

3. SYSTEM DESCRIPTION - HARDWARE AND SOFTWARE

The computer system we use in an IBM/370 compatible BASF 7/68 machine running under MVS 03.8 with which users communicate through the Gothenburg University Timesharing System GUTS (ref. 6). The software is based on Dynamo II/370 version 8 (ref. 7) as the nucleus of a fairly complex but user-friendly system containing various extensions and utility programs. See figure 1.

As most of our students have a rather limited experience of computers, we prepared a number of easy to use dedicated command procedures and selected a well-defined subset of editor commands to be taught (ref. 8). Therefore we are able to concentrate our efforts on the main subject of model building and simulation, paying only a little attention to the intricacies of the computer operating system, file handling, job control language etc.

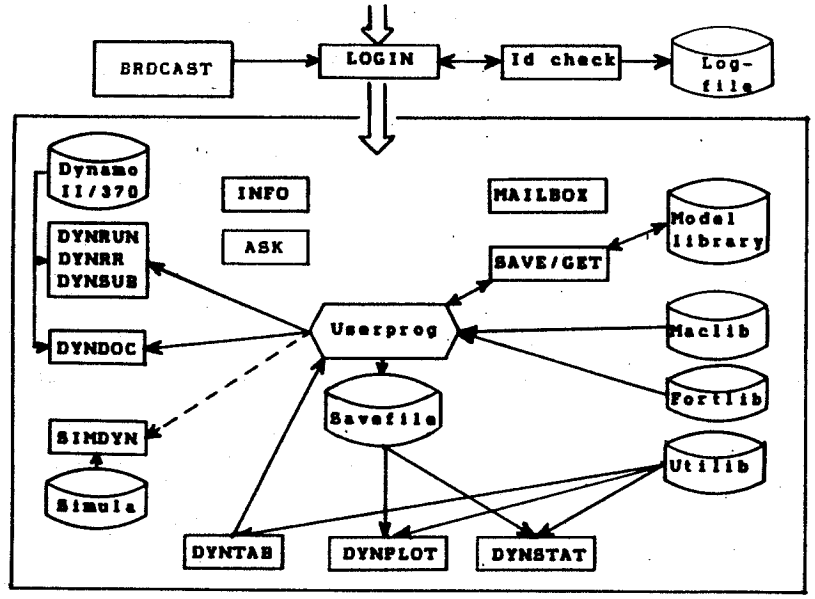


Figure 1: The integrated system provides many facilities in an easy to handle Dynamo environment (DYNRUN - interactive run, DYNRR - interactive reruns, DYNSUB - batch run, DYNDOC - documentation, SIMDYN - Dynamo in Simula, DYNTAB - table handler, DYNPLOT - high-resolution plot package, DYNSTAT - statistical package).

As soon as the user gets the terminal connected to the system, he or she can log in through the LOGIN procedure which performs an identity check, makes a record in a logfile and calls our mail systems BRDCAST to display current messages for this particular user as well as for the group of users he or she belongs to. Once the login procedure is complete, the user can access the INFO procedure to get concise information on any

131 facilities available in the system. If any problem occurs and no instructor is present, a special ASK procedure may be called to put questions into the MAILBOX file, which is regularly monitored, and answers are given through BRDCAST.

The other procedures concern running Dynamo programs and associated utilities. A great effort was made to facilitate easy use. Missing parameters of any importance are asked for and fairly detailed self-explanation may be obtained from each procedure.

The most commonly used command procedures are DYNRUN, DYNRR and DYNSUB which call the Dynamo II/370 compiler and run time system for user program execution. DYNRUN is the general procedure used in the interactive mode in our GUTS terminal system. DYNRR is a special variant of DYNRUN designed for interactive reruns. DYNSUB is used mostly for large jobs as it submits programs to batch processing. The results can be fetched with the help of the DYNOUT procedure or the job can be aborted by DYNCAN.

The above procedures give the user automatic access to two external libraries: MACLIB, where a number of useful macros have been stored, and FORTLIB, containing Fortran functions that may be called from within user programs. We implemented e.g. the BOXCAR facility through a Fortran function which was then wrapped around by a macro definition to make it easier to use and stored in MACLIB.

There is also a library of complete System Dynamics models MODLIB. Each model can be brought into the system with the help

of the GET procedure and immediately executed as a Dynamo program.

For documentation purposes we developed the DYNDOC procedure which calls the standard Dynamo utilities, Documentor and Number. However, we regard this kind of documentation as rather poor and prefer well documented flow diagrams with all quantities separately fully described as the main documentation tool. A well structured and commented Dynamo list is merely an appendix.

Our version of Dynamo has the ability to generate special, "database-like" SAVE-files where various data concerning selected quantities may be stored for subsequent use. In 1981 we conceived a plan for making use of these files for post-processing purposes and received information on their internal structure from Pugh-Roberts Associates, Inc. The plan proved to be productive and became an important factor in linking various system utilities together. We have been able to develop two major interactive, command driven programs: DYNPLOT and DYNSTAT that access SAVE-files and transform the data stored there into easily intelligible information. DYNPLOT creates plots on a high resolution screen, providing the opportunity of obtaining phase-plane curves as well as time-function displays. DYNSTAT performs a number of statistical calculations.

Another program worth mention is a Dynamo-table handler and generator, DYNTAB, which allows a number of operations on one- or

twodimensional tables to be performed, such as summing, multiplication and inversion, and new tables may be generated from old ones according to the Dynamo (T-card) or DARE P format.

Usage of the main procedures is monitored and recorded in the logfile giving us useful information on the basis of which we have been able to improve the system and keep track of student activities.

There are also other procedures and utilities which are of no great relevance to an "average" user as they were intended for system maintenance only (e.g. adding new macros or compressing the library).

We have also implemented the program package "Dynamo in Simula" based on Hegna (ref. 9) and developed in Regionales Rechenzentrum Erlangen in Germany (ref. 10) on our computer system. This powerful package, which we call SIMDYN, makes it possible to write Dynamo-like programs in Simula giving us facilities comparable with those of the full Dynamo III language. What is more, the source code is accessible and can be easily extended (e.g. to implement new functions) or altered (e.g. to change the Dynamo integration rules or add facilities for optimization or identification).

Besides Dynamo, Dynamo in Simula and SIMNON we also have access to CSMP (ref. 11) and DARE P (ref. 12) packages for

Continuous System Modelling and Simulation. Both of these languages can be run with the help of easy to use procedures. They take advantage of Fortran as an intermediary language which means that many error messages are given in terms of the Fortran compiler or its run time system rather than in those of CSMP or DARE P, which confuses novice users. This is one reason that neither of these languages have achieved the same popularity as Dynamo, despite some interesting features.

4. EXPERIENCES IN SYSTEM DYNAMICS AND DYNAMO

It becomes clear from the above that System Dynamics and Dynamo hold a central position in the course content, in our teaching, and as technical tools.

For students trained in mathematics System Dynamics provide an interesting alternative with easy to understand schematicism. It should be pointed out that graduate engineers and mathematicians also need a tool for communicating with mathematically less trained people about dynamic systems. For biologists, agriculturists and quite a few economists it is the only realistic way of coping with dynamic systems.

Our experience from System Dynamics and Dynamo in education is very positive. Their advantages have been praised in many articles and we share the positive opinion of the basic philosophy of System Dynamics and its realisation in Dynamo. Our criti-

cism which follows is aimed at suggesting some improvements as result of our experience from education and research.

4.1. Some suggestions on System Dynamics

The basic idea behind System Dynamics is to describe the structure of dynamic systems with the help of levels and rates so that loops of different kinds appear clearly. For this purpose causal-loop diagrams are used to give a rough picture showing components, causal relations and the loop structure.

The flow diagram is a more detailed system description. Its strength derives from the description of dynamic structures with the simple concepts of level and rate. This diagram is a tool for communication, documentation and, above all, a basis for a simulation program, traditionally but not necessarily, written in Dynamo.

Although we make use of causal-loop diagrams to introduce the loop concept and as a first modelling step, the flow diagram is the main description tool.

The standard flow diagram symbols are ingeniously simple and clear. However, in common praxis there are some points of confusion that could be eliminated.

In our opinion the symbols and their use should be simple but strict. At the same time the connection to simulation programs should be regarded. We suggest the following:

a. It is very important to define a positive direction of a flow and to follow this convention strictly. The arrow should therefore define this positive direction. This does not imply that the flow is unidirectional but it means that a flow in opposite direction is negative. Experience shows that this convention eliminates a lot of confusion in the modelling phase. Double arrows should be absolutely banned.

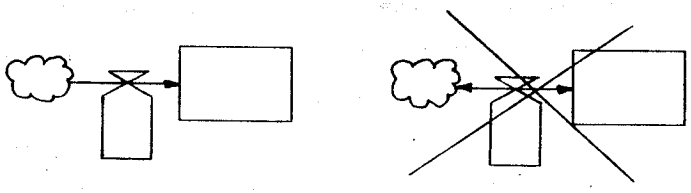


Figure 2: The arrow defines the positive direction according to our convention.

b. The flow diagram is multiplicative by its nature and any other operations (such as addition, subtraction or division) should be explicitely stated, which is often left undone. A typical example is the two kinds of constants shown in figure 3. It would be more precise to indicate time constants by 1/T rather than T so that the multiplicative convention can be consistently applied.

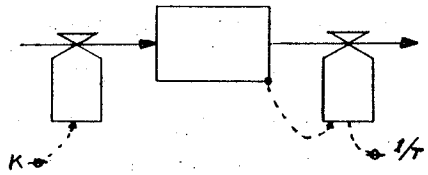


Figure 3: All constants should be multiplicative if nothing else is indicated on the diagram.

c. Time delay  is a fairly common and practical symbol, which really means  +  taken once or more. We noticed that the similarity of this symbol with  give rise to various kinds of errors when the students wish to put a time-delay in their diagrams. They find it difficult to understand why the symbol must be preceeded by a rate and followed by a level.

This inconvenience can be eliminated if we put the rate symbol  into the delay symbol which gives . This is merely a simple trick, but it works - the students are immediately able to handle the symbol in the proper manner.

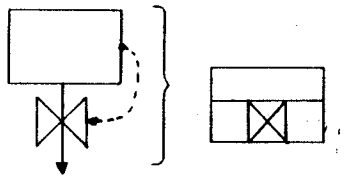


Figure 4. Stressing that the delay is in fact composed of two symbols  +  helps to avoid mistakes.

d. We would like to propose that the System Dynamics symbols should be standardised as follows:

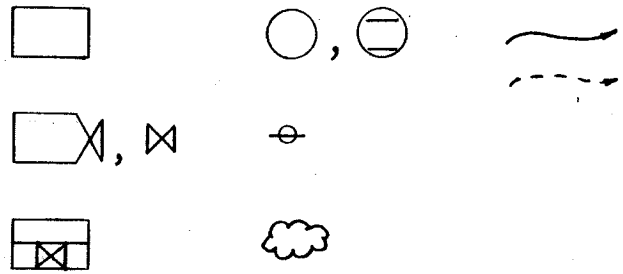


Figure 5: A proposal for flow diagram symbols.

Using of other symbols should not be encouraged.

4.2. Connection between flow diagrams and simulation programs

Flow diagrams, through the close connection between their symbols and program statements, constitute an excellent basis on which simulation programs can easily be constructed. Various continuous system simulation languages as Dynamo CSMP, DARE P and SIMNON do not differ much in their structure. The most essential difference is the form of the state equations. The flow diagrams can therefore be a basis not only for Dynamo but also for other continuous systems simulation languages. The Center for Agriculture in Wageningen, Netherlands, has for example used System Dynamics and flow diagram symbolism together with CSMP for a long time.

The explicit Euler integration form that Dynamo adopted for its state equations is excellent from the viewpoint of pedagogy, especially for students without mathematical background. In our lectures we point out the coupling between the Dynamo level equation and the concepts of derivative and integral, also showing alternative state equation forms that other languages make use of. We achieve this through the following argumentation:

The level equation in Dynamo is:

$$\square.K = \square.J + (\Sigma \boxed{IN}.JK - \Sigma \boxed{OUT}.JK)*DT$$

Substituting $\boxed{NET}.JK = (\Sigma \boxed{IN}.JK - \Sigma \boxed{OUT}.JK)$ we get:

$$\square.K = \square.J + \square.JK * DT$$

which is equivalent to:

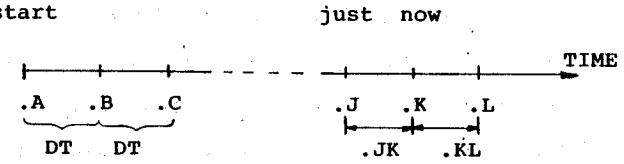
$$\frac{\square.K - \square.J}{DT} = \square.JK$$

This expression corresponds to the definition of derivative when DT->0. This can also be written as:

$$\frac{d\square}{dt} = \square \quad \text{or} \quad \dot{\square} = \square$$

which is the form used by DARE P.

Labelling the time axis as below:



we can successively develop the level equation according to

□.K = □.J + □.JK*DT

□.J = □.I + □.IJ*DT

□.I = □.H + □.HI*DT

⋮

□.B = □.A + □.AB*DT

which gives:

$$\square.K = \square.A + (\square.AB + \square.BC + \ldots + \square.JK)*DT$$

i.e. $$\square.K = \square.A + \sum_{A}^{K} \square*DT$$

or $$\square.K = \square.A + \int_{A}^{K} \square\, dt$$

A mathematician would call the last symbol an integral with respect to time.

This last form is used e.g. in CSMP where you write:

□.K=INTGRL(□,□.A)

where □.A is the initial value of □.

4.3. Comments on Dynamo

In order to hold a course in modelling, simulation, systems behaviour and demonstrate various techniques such as sensitivity analysis, optimization and identification, all within strict time limits and with practical applications in mind, we needed an adequate programming language available on our computer equipment.

Our choice, Dynamo II/370, was motivated by the following advantages of Dynamo:

1. A good, natural connection between the flow diagram and Dynamo program. Even though the flow diagrams can be applied to other languages, Dynamo level equations are the most pedagogically formulated, which is a matter of great importance for understanding the concept of System Dynamics.

2. Simplicity and clearness of the language. The statements can be easily coded from the diagram and have a plain and common form. The time indices (.J, .K, .JK, etc) are syntactically unnecessary but give the programmer an additional help.

3. The system is reliable. The error and warning messages are well formulated and informative.

4. Dynamo II/370 is roughly 5-8 times faster than CSMP or DARE P when running small and medium scale models which are common in the student environment. Dynamo is also much easier to use.

5. The control statement SPEC and the standardised output are very easy to learn and do not distract the user from the task of model building.

6. For more advanced applications there are several useful facilities like reruns, comparative plots, user-defined macros and external Fortran functions.

Some of the disadvantages of using Dynamo are:

a. Its closeness - a user cannot insert an "own code block" (which is only partly compensated for by macros and external Fortran functions) - and unavailability of the source code (which hasn't much to do with the language itself).

b. The lack of optimization and identification facilities is a major drawback of Dynamo. Both these require that a criteria function, which can be easily written in Dynamo, may be evaluated with an algorithm in order to set new program parameters for the next rerun. This could be achieved if we had the means to control reruns with criteria function value obtained from the previous run.

c. The Euler integration rule, in spite of its rather bad reputation, works surprisingly well in practical applications. However, to be considered a mature language (and perhaps to avoid the not always competent critique), Dynamo should be equipped with alternative integration algorithms, like Runge-Kutta's method and why not an algorithm for stiff problems.

On a more detailed level we have found a number of irritating deficiencies, most often quite unnecessary. Several of them could be attributed to Dynamo being an American rather than an international language, while others could be explained as design flaws. From this viewpoint we will suggest some improvements we find desirable.

d. Quantity names should be meaningful which is difficult or impossible to achieve within the limit of 6 characters imposed by Dynamo (as a matter of fact, Dynamo II/370 can distinguish names of up to 60 characters, truncating them to 7 characters on output, an undocumented feature). Names like FMFAF and like are inconvenient.

e. Another restriction is put on the so called national characters which most European languages abound in. We do not see any reason for this since no standard ASCII characters which share codes with national characters are used in Dynamo for any other purpose (e.g. 64, 91-94).

f. Logical flaws and lack of consistency lead to unnecessary confusion. Dynamo sometimes deviates from the common sense:
- LENGTH parameter ought to either denote simulation time length as in DYSMAP (ref. 13) or be renamed to something that would imply its real meaning: the value of TIME when the run is to be terminated.

- PRINT and PLOT statements should be treated in the same manner. However, two separate PLOT statements will generate two separate plots while two separate PRINT statements will often combine into one table. Our older version of Dynamo II and also Mini-Dynamo are more consistent in this respect.
- The scalling letters which Dynamo sometimes uses to indicate powers of ten are a real peculiarity without a counterpart in any of the other languages we know. Using e.g. R for tRillions (which means E+12 in the USA and E+18 in other countries) is both unsuitable and difficult for a student or researcher to learn. A better alternative could be to output E+12, even if that requires more space.

g. The Dynamo run time system gives very good and clear error and warning messages, zero division being the exception in some versions. Moreover, some warnings are unnecessary and even irritating, e.g. "DYN ASSUMING..." (if the expression is not overparenthesized) or, in some cases "UNUSUAL FORMAT FOR LEVEL..." (which is raised by e.g. L L.K=L.J-DT*R.JK).

h. The date format Year/Day/Month does not adhere to ISO standard: Year/Month/Day.

i. The cumulative rounding errors - e.g. getting 74.99 instead for 75 - which become visible during the output of TIME values (caused by the algorithm used: TIME.K=TIME.J+DT) are unpleasant and one should try to remove or minimize them. We wrote several Pascal and Fortran continuous systems simulation programs modelled after Dynamo and found that the following algorithm works satisfactorily:

1. Let TIME.0 and TIME.END denote the initial and end time values.
2. Compute LENGTH=TIME.END-TIME.0.
3. Compute NSTEPS=INT(LENGTH/DT+0.5).
4. TIME.K=TIME.0+K*LENGTH/NSTEPS

This means, in effect, that the DT value cannot be changed during one run. In many cases this is of little importance. However, any user's attempt to change DT in his or her program (either explicitly or through the choice of an alternative integration rule which works with variable integration step length) could be easily detected by the compiler and the decision to use the "old" time increment algorithm could be made.

j. Computer graphics is gaining in popularity as the prices of high resolution screens fall. It is a comparatively easy matter to add a post-processor of SAVE-files providing for graphic displays and thus the internal organization of SAVE-files should be described in the manual.

k. External macros and Fortran functions play an important role in advanced Dynamo. These are conveniently stored in libraries. Unfortunately, it is not possible to access a single macro from within the program as the whole library will be inserted in memory at once. As several library macros may call Fortran functions and the number of external Fortran calls may be limited by the Dynamo implementation (as in our case: max 3), this may cause abnormal program end. This problem can be solved from case to case but a more general solution would be welcome.

4.4. Some practical tips and suggestions

Dynamo contains on the whole a well selected set of functions which can be completed with macros and Fortran routines. In our work we found the following functions especially useful:

a. Macro DURING(FROM,TO) implemented through the CLIP built-in function - simple but very useful in a vast number of cases.

b. Array function TABXY which will fetch values from Dynamo tables (T-statements) and simulate 2-dimensional arrays. This function is also implemented as a macro and utilises CLIP functions for selection of up to 16 T-rows.

c. BOXCAR function, implemented through Fortran. The main function is wrapped around by a macro definition for easy user interface.

When implementing a model with many reruns it may sometimes be desirable to skip these reruns during the test phase. We discovered that putting QUIT after the first RUN statement in the program source file works perfectly well, just as if given interactively from the keyboard.

In our system a blank line functions like a NOTE statement, which enhances, in our opinion, readability of the program.

5. CONCLUSIONS

The above described pedagogical approach has proved to be a flexible and efficient instrument for teaching System Dynamics and simulation. During the last five years more than 450 student including 150 graduate students of Uppsala University and the Swedish University of Agricultural Sciences (also situated in Uppsala) have completed our courses. A number of them now use these methods in their research.

Although there are strong limitations in Dynamo, the language has also been shown to be surprisingly well suited to most research projects, although the lack of posibility to control Dynamo II/370 from external routines prevents optimization and identification in an easy way.

In this article we have also presented some criticism of System Dynamics and Dynamo in connection with some solutions and suggestions we have found useful or desirable.

6. REFERENCES

1. Gustafsson L., Lanshammar H., Sandblad B. "System och modell", (System and Model), Studentlitteratur, Lund, Sweden 1983.

2. Gustafsson L. "Modellbyggnad och simulering i Dynamo", (Model Building and Simulation in DYNAMO), Uppsala 1983.

3. Birtwistle G.M. "DEMOS A System for Discrete Event Modelling on Simula", The Macmillan Press Ltd 1979.

4. Birtwistle G.M., Dahl O-J., Myrhaug B., Nygaard K. "SIMULA BEGIN", Petrocelli/Charter, New York 1975.

5. Elmqvist H. "SIMNON An Interactive Simulation Program for Nonlinear Systems - User's Manual", Dept. of Automatic Control, Lund Institute of Technology 1975.

6. Gothenburg University Timesharing System GUTS v3.6 Reference Manual, Gothenburg 1983.

7. Pugh A.L. III "Dynamo User's Manual", The MIT Press 1983.

8. Gustafsson L. "Att köra Dynamo på UDAC", (How to run Dynamo at UDAC), Uppsala 1982.

9. Hegna H. "Dynamo in Simula 67 - A Rough Outline of a Simple Implementation", Norwegian Computing Center, Olso 1974.

10. Simon K.-H. "Dynamo in Simula - Simulationsinstrumente auf Simula-basis", Regionales Rechenzentrum Erlangen, IAB Nr. 118, Erlangen 1980.

11. System/360 Continuous System Modelling Program User's Manual, IBM Application Program, New York 1972.

12. Wait J.V., Clarke D. "DARE P User's Manual Version 4.1", University of Arizona - College of Engineering, Dept. of Electrical Engineering 1977.

13. Ratnatunga A.K. "DYSMAP User Manual", System Dynamics Research Group, University of Bradford 1980.