AN INTERACTIVE GRAPHICS SYSTEM FOR MODELLING AND RESEARCH BY NON-PROGRAMMERS

DOUG SEELEY
DHIRENDRA CHEDDA
UNIVERSITY OF BRITISH COLUMBIA

ABSTRACT

Systems dynamics models have become a popular method of exploring the behaviour of complex feedback systems such as those found in ecology, urban development, and resource economics. This paper and video presentation describes an interactive graphics program called CAM that allows a user (who does not have to be versed in programming) to create and edit large systems dynamics models, cause them to create CSMP source and object code, run the programs, and view the plotted outputs, all in the same session at a graphics terminal.

CAM provides a method and tool whereby non-programmers can carry out complex modelling activity in a highly productive manner because of its on-line setting, its graphic orientation, and the completeness of the experimental facility it provides. The paper will discuss the program's design, the unique problems encountered in implementation, and some of the considerations involved in providing access to a powerful graphics tool to a non-programmer.

SYSTÈME GRAPHIQUE AVEC INTERVENTION D'OPÉRATEUR POUR LA MISE EN MODÈLE ET LA RECHERCHE À L'USAGE DES NON-PROGRAMMEURS

ABRÉGÉ

Les modèles de la dynamique des systémes sont devenus une méthode très répandue pour l'étude du comportement de systèmes complexes à contre-réaction comme ceux que l'on rencontre en écologie, en urbanisme et en économie des ressources. Ce rapport, ainsi que la présentation vidéo, décrivent un programme graphique avec intervention d'opérateur, appelé CAM, permettant à l'utilisateur (qui n'a nul besoin d'être expert en programmation), de créer et d'éditer des modèles de la dynamique de vastes systèmes, de telle sorte qu'ils créent une source CSMP et un code d'objet, qu'ils passent les programmes et visionnent les diagrammes de sortie, le tout en une seule séance, à un terminal graphique.

Le système CAM est à la fois une méthode et un instrument qui permet à des non-programmeurs de réaliser des modèles compliqués de manière très productive, grâce à son réglage en direct, a sa visualisation graphique et à ses moyens complets d'expérimentation. Le rapport discutera de la conception du programme, des problèmes particuliers auxquels on eut à faire face durant sa mise en oeuvre, et exposera quelques-uns des points à considérer lorsqu'on met un puissant outil graphique à la disposition d'un ingénieur non-programmeur.

Interactive computer graphics can be a powerful tool when it comes to the design of objects whose composition or structure can be described pictorially, for instance, circuit boards, aeroplane wings, and ship hulls. However, in most installations access to this tool by those unfamiliar in programming is severely limited. Frequently, experience in a high level algebraic language, a graphics subroutine package, and some graphic hardware knowledge is required. Even some of those programmes familiar with only one language often balk at the extra information needed to carry out a useful applications project.

However, creative design is a pursuit that is usually expressed in some graphic or constructed form in which the designer is facile at the concrete expression, exploration, and modification of ideas. Such systems exist for interactive graphics, expecially in technical design such as for electronic circuits. However, many folks doing creative design or more socially-oriented research have strong biases against the intellectual symbol-manipulative skills that mathematics and applied science requires. This limits the useful audience for a graphics terminal.

More and more studies are using computer modelling and simulation as a testube in order to explore the behaviour of various models of systems found in the world. However, there is often a technological priesthood of expertise between the folks who should be asking the "what if" questions of simulation and the opportunity to "muddle" around with the assumptions, values, and structure of the model under investigation. If this dependence upon over-specialization can be lessened and more open access to the simulation tool achieved, perhaps applications that are more humane in the long run can ensue.

The Domain: Systems Dynamics Models

Simulation is a widely used tool for investigating phenomena ranging from information flow in business organizations to the dynamic behaviour of complex mechanical and ecological systems. As the systems modelled increase in size and become more complex, one has had to switch from anologue computers to special digital simulation languages in order to obtain numerically accurate results. Systems like Forrester's "World Model" (4) have been modelled and simulated on digital computers in an attempt to understand the dynamic behaviour and interaction of system components under various conditions. This would nearly be impossible without the digital simulation of continuous systems. In order to simulate such complex systems, one must model and write a simulation program for the model, which may often turn out to be a tedious task involving other work like learning a simulation language.

With the advent of interactive graphic displays, it is now possible to directly communicate and interact with a program in order to direct the flow of the program in many computer-aided design applications. Such on-line simulation allows the user to

directly formulate his model on a Graphic Display for simulation. "BIOMOD" developed at The Rand Corporation (5) is an example of such an on-line continuous system simulation of biological and pharmaceutical models. It allows the user to represent a model by analogue-computer-like elements, algebraic, differential or chemical like equations, and Fortran statements. The system described herein, CAM, was constructed along similar lines for the systems dynamics models that are typically written for DYNAMO and CSMP using the graphical conventions of Jay Forrester (3).

Systems dynamics is a method of system analysis, dealing with the time-varying interaction between components of a system within closed loop structures. These closed systems have interlocking feedback loops, which interact to produce growth, fluctuations, and changes in system components. A broad definition of an information feedback system is presented by J.W. Forrester (2). He states:

> An information-feedback system exists whenever the environment leads to a decision that results in action which affects the environment and thereby influences future decisions.

A structure is essential, if we are to inter-relate and comprehend our observations and knowledge of such dynamic systems. The laws of physics form a structure to explain our observations of nature, but industrial, management and biological systems are merely descriptive. A unified basic structure to describe such systems did not exist until recently. Forrester has approached industrial systems and their structure using elements of control system theory in order to produce such a methodology.

A similar approach can be taken in order to analyze many biological and mechanical systems. In these systems, a problem with many interacting components can be isolated and a formal graphical model describing the interaction and feedback between components of a problem and their cause-effect relationships can be constructed. The results obtained from the model simulation can be compared with the known results, and the model's structure can be respecified in order to best approximate known behaviour.

In Figure 1, the basic structure of a feedback loop and the graphical representation of its components are illustrated. Such a loop is a closed path, having an alternating structure of reservoirs or levels (state variables), interconnected by controlled flows of material and information flow. A decision generates an action which affects the level or state of a system. The information about this level is measured, a process which may have discrepancies and may get distorted over time. The information of this apparent level is obtained by the decision process, which is the basis for new action. Other delays or distortion in information may be encountered within larger feedback loops.

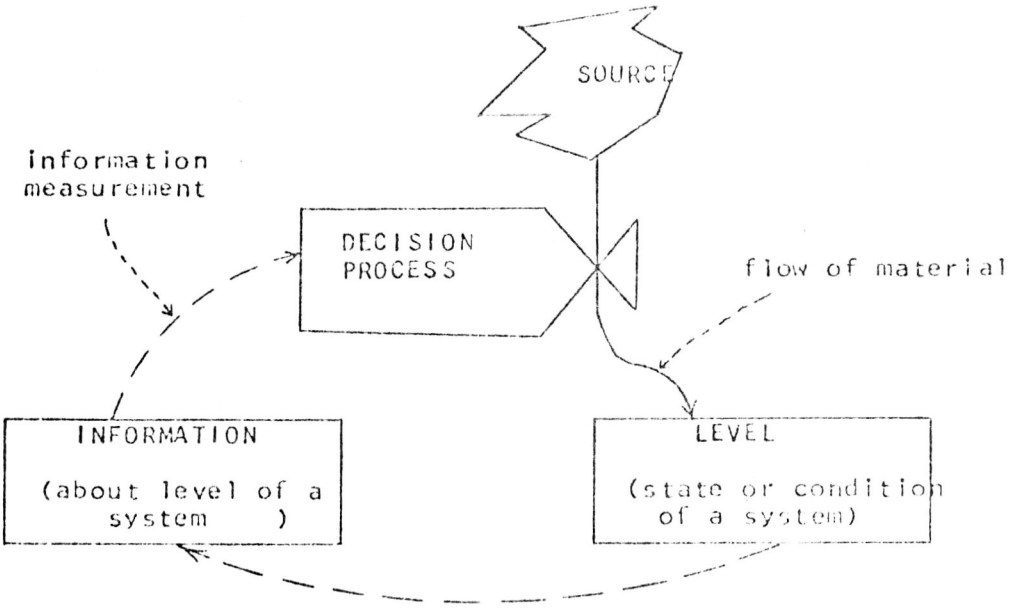

Figure 1 - The Structure of a Feedback Loop

System Components and Graphical Representation

The behaviour of some systems can be described by models comprised of verbal statements or beliefs, or by equations describing the separate component interactions of a system. The former is awkward, hard to communicate and lacks precision.

From the above discussion feedback loops can be formed from four basic components: levels, flow rates, decision processes and information flow. The causal structure of a system consisting of interconnected feedback loops can be represented succinctly by a graphical flowchart if the various system components are represented by graphical symbols and their interdependence by flow lines. If such a flowchart has a well-formed structure it can be "translated" into the statements and equations of a continuous system simulation language (provided necessary numerical data is supplied). The interactive graphics system described here provides a vehicle for defining systems using these symbols and carries out the translation and execution under command control.

The simulation language used by CAM is S/360 CSMP (Continuous System Modelling Program).

Another such programming language that has been in use is DYNAMO (6). It was developed by the Industrial Dynamics group at M.I.T. to simulate models of industrial systems; but it has notational difficulties and is awkward to program in. S/360 CSMP incorpor-

ates a number of advanced features, and provides a significantly more powerful and sophisticated tool for simulation.

As an example of a typical system and its graphical description, regard Figure 2, which is a hardcopy version of a CAM-generated model component (subnet). It gives rise to two basic types of equations: level equations and rate equations, with other auxiliary equations which add convenience and clarity in representing complex systems.

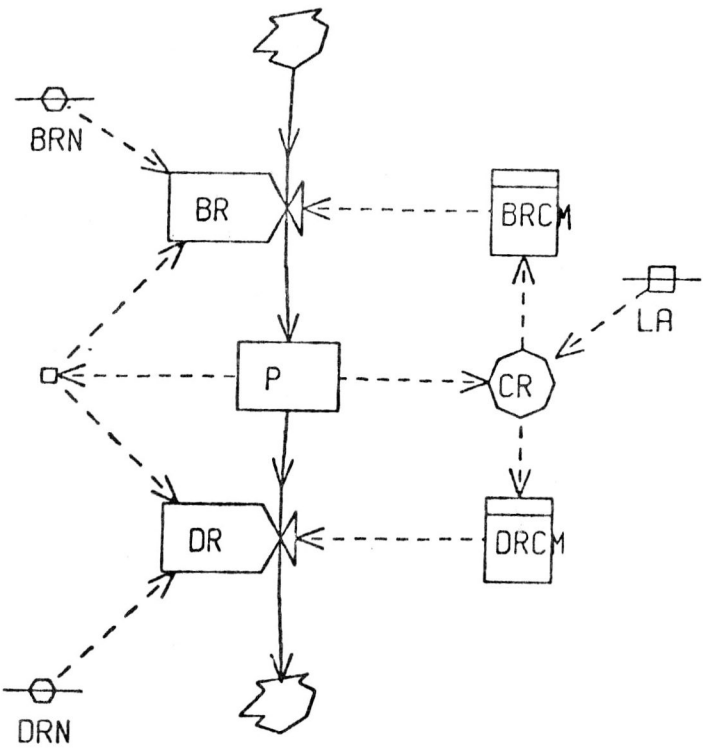

Figure 2 - Effect of Crowding on Population (World Model)

Levels: Level variables describe the state of a system at any point in time, and levels continue to exist, even when the system is at rest. Levels accumulate the results of actions produced by decision processes. Normally, level variables feed information to rate variables, which cause changes in other level variables. Therefore, level and rate variables must alternate along any path through a system structure. The existence of a level, which is described graphically by a large rectangle (P in Figure 2) generates an integral equation using the in and out flows from the flowchart. (For the mathematical description and representations in CSMP and DYNAMO see (1).)

Flow Rates: Rates define the present instantaneous flows between the levels in a system. Rates reflect activities or actions which

change the state of a system. Rates of flow or actions are determined by the levels of a system, according to the rules defined by the decision processes. Values of a rate variable can be computed by using the values of its associated levels and constants and neither depends on its previous value, nor on the time interval between computations. Rate variables can only interact through their influence on system levels.

A Rate is represented graphically by a valve-like symbol (BR and DR in Figure 2). The inputs to a rate equation may be a simple mathematical relationship between levels, numeric constants, and summary variables called multipliers that may represent complex system interrelationships.

Auxiliary variables: These quantities are used for convenience in systems where the decision process involves a number of intermediate steps. An octagon represents a rate multiplier (CR in Figure 2) and generally is formed by a product of its inputs. A square with a rectangle bar represents a function multiplier (BRCM & DRCM in Figure 2). It is an arbitrary function of a single variable where the functional relationship is described by interactively drawing an explicit curve.

Constants and Switches: These values can be changed at the beginning of a new simulation run. Constants are independent of the system behaviour and remain unchanged, with dynamic changes in the system. A simple constant graphically represented by a small square with a line through it (e.g. LA in Figure 2). The other type of constant is a "switch"; it has an initial constant value, and it assumes a new value at a specified time in a simulation. It is represented by an octagon with a line through it (BRN in Figure 2).

Sources and Sinks: Sources and sinks lie outside the closed boundary of a system. Closed boundary of a system is such that nothing from within the system flows across that boundary, which may affect system behaviour. An infinite source provides any "material" flow that is required by model equations. Conversely, an infinite sink absorbs any flow that is put out by a system. Both are represented by an irregular shaped blob.

Connectors: Connectors are intermediate points of flow in a network or a model. They are used to minimize the cross overs of flow lines in a planar view of a flow diagram. The first is called a planar connector and is graphically represented as a small square. The second is referred to as a subnet connector and is graphically represented as a small triangle. These allow a user to build up large models, consisting of several smaller segments which can be interconnected by these subnet connectors. Due to the small size of a display screen, building a large model (as for example, the "World Model"), in a single network can get cluttered and confusing on the screen. With the aid of subnet connectors, a user can divide up his model into small subnetworks or segments.

Flow Lines: There are two different types of flow lines that is used in organizing information. Material flow is represented by solid lines as shown in Figure . Material flow occurs when an action generated by a flow rate or decision process is fed into or out of a level to change the level. Information flow is graphically represented by dashed line and is used to measure the information flow of a level or a decision process. Information may be copied many times without affecting its source.

Interactive Modelling with CAM

The CAM system was developed on an IBM system/370 model 168 under the MTS (Michigan Time Sharing) operating system. It is dependent upon the software and the hardware facilities available at the University of British Columbia. The hardware configuration is shown in Figure 3. The program is run under MTS utilizing a partition of approximately 250,000 bytes. It operates in parallel to an Adage AGT-10 graphics computer which is monitored by a graphics supervisor.

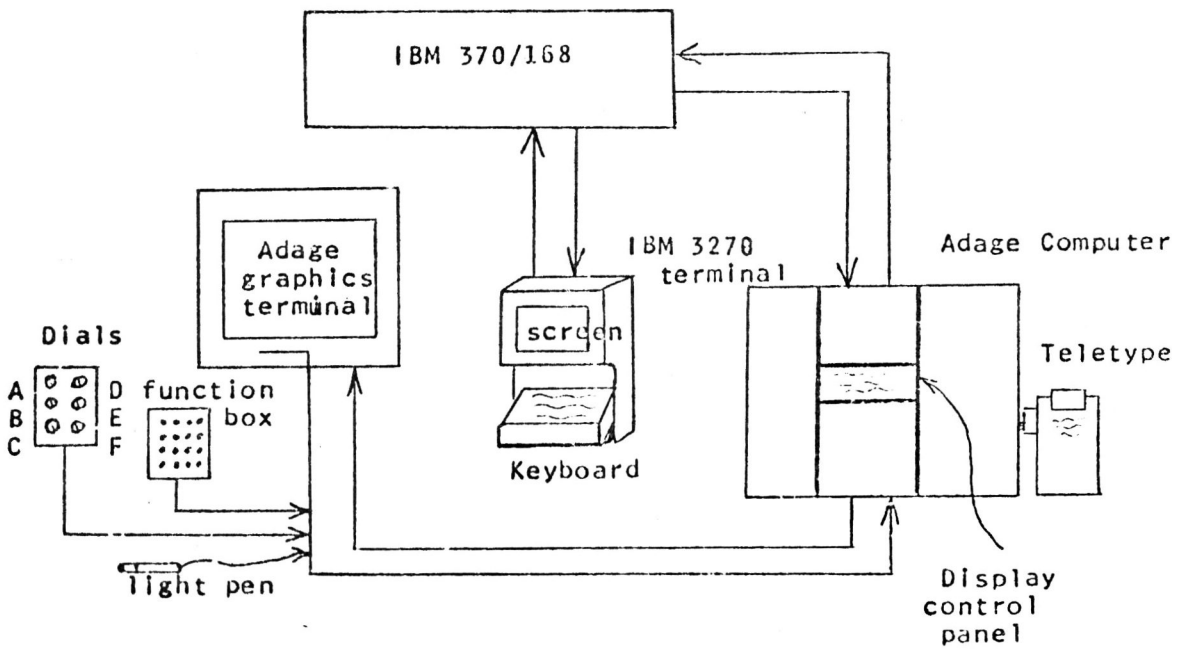

Figure 3 - Hardware Configuration of the System

CAM was written in ALGOL W, and upon running the command subsystem and entering the command HELP, the following is displayed on the 3270:

FOLLOWING COMMANDS ARE CURRENTLY AVAILABLE

CREATE	*TO CREATE THE MODEL*
EDIT	*TO MAKE CHANGES TO THE MODEL*
SAVE	*TO SAVE THE CREATED MODEL ON THE DISK*
RESTORE	*TO RESTORE THE MODEL FROM THE DISK*
RESET	*TO RESET THE DISPLAY*
*SUBNET **	*TO CREATE NEW SUBNET OR TO DISPLAY SPECIFIED SUBNET*
SYMBOLS	*FOR DESCRIPTION OF EACH SYMBOL USED IN THE MENU*
NAMES	*TO GET LIST OF VARIABLES USED IN THE MODEL*
ENAMES	*TO GET LIST OF VARIABLES AND THEIR DEFINITION*
EQNS	*TO GET EQUATIONS OF THE MODEL BEING DISPLAYED*
*SCALE **	*TO CHANGE THE SCALE OF THE DISPLAY (0.0<*<1.0)*
CSMP	*TO GET CSMP PROGRAM FOR THE MODEL*
PLOT	*TO GET A HARDCOPY OF THE MODEL*
RUN	*TO EXECUTE THE CSMP MODEL*

STOP OR END TO STOP THE PROGRAM EXECUTION

There are essentially four phases to the system; (i) the modelling phase, (ii) the specification phase, (iii) the execution phase, and (iv) the editing phase. One may enter any of these by stopping the current phase and entering an appropriate command. Together, they provide the user with a tight design system for on-line creation and behaviour exploration of a range of models.

CREATE, EDIT, and RESTORE will put the user into model creation phase. Here, in order to create symbols on the screen, the user selects the position on the screen by using dials lavelled X-hair and Y-hair, and then points at the desired symbol in a menu. CAM responds by creating a copy of that symbol at the selected position. In this fashion the user "converses" with the graphics display until all the required symbols for the model's flowchart have been created.

All of the symbols mentioned in the above section are included in the menu, along with the graphic commands DELETE and SPECIFY.

The next task is to draw the lines for information flow and material flow in the model. For material flow the user selects a solid directed line from the menu, and enters the material flow mode. When a mode is entered by a light pen hit on its name (or symbol), the menu disappears and a short prompting message appears at the bottom of the screen. In all cases the user is reminded what to do next and is told (or can see on the picture of a model) that his requested action has been completed. Yet the screen is never cluttered with extraneous information. Blinking has also proved very effective as an acknowledgement and as a prompting device.

In the flow line mode, the user is first prompted with a message "FROM WHERE ? ". The user may respond by pointing at any component in the model. The system acknowledges this by blinking that symbol and changing the message to read "TO WHERE ? ".

Pointing at another component results in an unidirectional flow line being drawn from the blinking component to the component just pointed. Repeating this cycle, all the flows may be assigned. In a like manner, dotted lines may be drawn in order to assign information flows, by entering the mode represented on the menu by a dotted directional line. At any point to terminate the mode, the user can simply point at "END".

DELETE mode can be used to remove any component or a flow line by pointing at it with the light pen after entering DELETE mode. The user can select or terminate any mode without any restriction. By using the light pen, dials and the menu in this way, the user can build or edit a representation of a model on the screen.

Once the model has been built in the above manner, the user can specify proper variable names (labels), values and mathematical relations must be assigned to the components. This can be achieved by pointing at SPECIFY in the menu. This results in the display of the second phase of the menu which has six modes: LABELS, VALUES, RELATIONS, FUNCTIONS, REDEFINE and END. LABELS mode can be used to assign labels to the components. VALUES mode can be entered in order to assign values to the constants and integrals (levels). RELATIONS mode allows one to specify a mathematical relationship on the screen for components in terms of other model components using the labels assigned in LABELS mode. FUNCTIONS can be used to sketch and edit graphs (curves) for function multipliers. REDEFINE switches on the first part of the menu in order to add or delete symbols or flow lines. END terminates the user interaction with the graphics display and the mode of interaction is switched to the keyboard.

If one selects any of the first four modes by the light pen, the system responds by prompting the user with a message "ALL OR SELECT ? ". This is to ask the user whether he wants to choose specific components for assigning labels, values, equations or functions. The user may do so by pointing at SELECT. In case of ALL the user is lead by the program to assign the selected specification to all the required components. This is done by blinking each component in turn and then prompting the user to take action, until all the components have been assigned the selected specifications. In SELECT mode when the user selects the component, the program checks to see, whether it needs to be assigned the particular specification, and then either presents the user with an error diagnosis, or acknowledges by blinking the symbol and prompting him to take action[1]. Once again the user can terminate any of the

---

1 For example, if the user picks a flow rate to assign the value, the program produces an error diagnosis stating that "Flow rates do not require values to be assigned to them." If instead he picks a constant to assign the value, the program, after checking the proper type, blinks that symbol in an acknowledgement.

selected modes by pointing at END.

Having created and specified a complete model or subnet the user has several choices for their next move. They may save the model as an MTS file (as a graphic data structure) for later re-working or experimentation and they may request a CAL COMP hard copy for reference, documentation, and reworking. However, they may choose to obtain the CSMP source translated from the graphic data structure, which may also be saved, and then run their model under MTS. Upon completion, they may view the DYNAMO like plots of the simulations output. The system prompts the user from the 3270 for details on the simulation run and any subsequent plotting. After viewing the output, the user may return to the model to alter a specification of a parameter or edit the feedback structure before running again. In this manner, they may quickly explore the resultant behaviour of different model assumptions. The fact that this can be done at one sitting greatly enhances the user's ability to acquire intuition about model behaviour.

Implementation Details

The display list for placing pictures on the Adage screen is created by "down-compiling" the compound data structure that represents the model. Conversely changes in the model as a result of interactive editing are "up-compiled" from a temporary display file to this data structure. Since the data structure must also contain relevant information for creating a simulation program it is somewhat complex. This was the major reason for choosing ALGOL W as the implementation language, and consequently ALGOL W records for the models' data blocks. These records or data blocks are used to store all the information about the entities or objects displayed on the screen. Relational-linkage pointers are used to link blocks whose corresponding objects are participating in diverse types of relationships. Identifiers are associated with each different data block to describe the block, and for fast identification of blocks when interrupts are handled to extract information about that symbol.

Each of the nine different symbols used in the formulation of a model (see Chapter 2) use the same basic data block with different identification tags. Each one has a different attribute list. The identifier is the first component of a data block while the second assigns a unique serial number to the symbol it represents in that symbol group. The next two components store the pointers to the display list for that symbol. They point to the first and the last display instructions for that symbol in the display list. These are stored in order to speed up search of any entity when interrupt by the light pen occurs to alter or delete that entity. The next two locations of an object block specify its X and Y co-ordinates on the screen. This speeds up computations for drawing flow lines. In addition to this, each one has a pointer to the next data block of the same model. Thus, all data blocks are serially connected. The data block also has up to four more pointers to the data blocks

of different structures in order to store various other non-pictorial (but semantic) information about a symbol such as its name, its values if needed, its equation, and connectivity information.

Since each of the symbols has different characteristics, different types of additional data blocks are required in order to store these characteristics. The four additional pointers in the basic data block point to these special data blocks. These four different types of data blocks or record classes are: i) value record, ii) name record, iii) from record, and iv) to record. Connectivity information in the data blocks is used in order to buide a user while he is forming equations. If the modeller does not assign any specific relationship, then this flow of connectivity information is used to form the basic default equations when writing the CSMP program. This information is also used to remind a user of proper connections to other subnets if the model consists of more than one subnet. These subnets allow the user to build up a large model and concentrate on any one part of a system, by visualizing one subnet at a time. All subnets internally form one large model.

The various graphical and semantic information of the model is stored in different data blocks associated with each symbol, as described earlier. The information about each symbol is used in order to formulate a CSMP program for that model. The CSMP program has three segments: INITIAL, DYNAMIC and TERMINAL. For the INITIAL phase, CSMP statements are produced for levels, constants, and switches in that order from their value records. For the DYNAMIC phase, CSMP equations for function multipliers are first formed from the co-ordinate array pointed to by their value records. Next equations for each of the switches are produced since these are parameters which are modified at some point in simulation time. Their equations are produced for the dynamic behaviour of levels, flow rates, and rate multipliers. If the user has not specified an equation, a default one is formed using connectivity and making standard assumptions. The TERMINAL phase is defined interactively or by default by the user.

Conclusions

What has been described in this paper is an interactive graphics system which enables a user familiar with systems dynamics notation to build and specify complex feedback models of the world. Not only may models be constructed but they may also be simulated, the results displayed, and the model modified and re-run all at the same sitting. Such an environment greatly enhances the experimenters' capability to explore, gain intuition, and to concentrate on the validity of important model assumptions.

The proof of the utility of such a tool is in the usage it attracts, especially from non-programmers. A particular active use of CAM has been made by the Political Science Department at UBC (7) for peace research modelling. Especially valuable for them has been the flowchart approach and the function multipliers which graphically can represent complex assumptions regarding the dependence of model components. Other users have been an interdisciplinary "systems analysis" group at UBC and by ourselves to gain insight into the role of communication in the behaviour of dynamic systems.

As more systems like CAM are developed, the accessibility of interactive graphics is enlarged. As access broadens, so does the hope for humane application of the technology.

Bibliography

(1) Chheda, D.P. 'CAM, A Computer-Aided Modelling Program for Systems Dynamics Models', M.Sc. Thesis, Computer Science, University of British Columbia

(2) Forrester, J.W. Industrial Dynamics, M.I.T. Press, 1961, p. 13.

(3) Forrester, J.W. Principles of Systems, Wright-Allen Press, 1968, pp. 1-9 to 2-4.

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

(5) Groner, G.F.; Clark, R.L.; Berman, R.A.; DeLand, E.C.; 'BIOMOD - An interactive computer graphics system for modeling', AFIPS Conference Proceedings, 1971, FJCC, Vol. 39.

(6) Pugh, A.L. III, DYNAMO - User's Manual, third edition, M.I.T. Press, Cambridge, Massachusetts, 1970.

(7) Wallace, M.D. 'Resources and International Conflict: The Onrushing Crisis', General Systems, Vol. 19 (1974).