A NEW GENERATION OF DYNAMOS

Alexander L. Pugh, III
D. Ross Hunter
Todd Sjoblom

ABSTRACT

The emergence of powerful personal computers and CAD/CAM machines offers a new opportunity for DYNAMO. Although users are generally satisfied with the language, a survey shows they want expanded simulation capabilities including single simulations, eigenvalue analysis, sensitivity analysis, and optimization by multiple simulation and hill climbing. Novices want easier access to models and simulation.

The modular version of DYNAMO now in development will meet these goals. It will break DYNAMO's normal functions into separate programs that users can reassemble in different ways. For example, one compiler will translate both conventional models and games. The simulation controller will work with a regular rerun, a game, or a sensitivity analysis package. The report generator will display output from any of these packages.

These modules will communicate through standard data files, which users can also access for other purposes such as statistical analysis.

The system requires seven different files:

Generic model	- includes menus to customize it
DYNAMO model	- standard DYNAMO source file
Object code	- machine dependent object code generated by DYNAMO compiler which may be linked or loaded with user's external functions
Generic rerun	- rerun statements and menus to customize them
Rerun file	- standard rerun output
Model results	- simulation output that can be used for report generation or read by another simulation
Definition file	- variable definitions which DOCUMENTOR or other program can use to display more readable information.

DYNAMO has been the simulation language of system dynamics for over 25 years. It has held that position by evolving simultaneously with the field and the hardware we use. With the advent of standard generic models, huge models and personal computers, DYNAMO must evolve further. This paper discusses some of the pressures forcing the creation of a new generation of DYNAMOs and the new features designed to ease these pressures.

In 1958, before FORTRAN was available, DYNAMO was created to avoid the necessity of writing a model in machine language. It was a pioneer in its thoroughness in checking a model before running it, and in ease of specifying output. Built-in macros simplified the inclusion of standard constructs such as third-order delays.

The advent of the IBM S/360 brought about the next generation, DYNAMO II. DYNAMO II supported any algebraic expression and user-defined macros, which simplified the construction of larger models.

The spread of systems dynamics in the 1960's resulted in a need for DYNAMO on other computers. DYNAMO II/F, a compiler written in FORTRAN and supporting the DYNAMO II language, met this requirement.

As system dynamics grew in self confidence the model size grew, and more advanced tools were needed to aid model construction. The addition of arrays to DYNAMO provided the necessary tools. Again, thorough checking by DYNAMO avoided two problems of most languages that support arrays: the element that is never computed and writing outside the array bounds.

In the 1970's another change in hardware affected the requirements for DYNAMO. Minicomputers, much cheaper than mainframes, became popular at many colleges teaching system dynamics. Mini-DYNAMO was created to run on 16-bit machines with modest memory. Mini-DYNAMO has amazing capacity given its memory requirements.

Mini-DYNAMO's architecture was so efficient of space that it could even be run on microcomputers with only 64 Kilobytes of memory. This version was named Micro-DYNAMO, in recognition of the computers it was being run on and the change in the source language from FORTRAN to Pascal.

What are the requirements of the 1980's? We build even larger models and need software that can accomodate them. We need special tools to analyze the reasons for behavior of complex models. More non-professionals use our models and require better

output facilities and easier ways to specify runs. We conceive more ways to test models and require software tools to support this effort. We relate models to data in more ways and need facilitating software. We produce standard, generic models for non-professionals and require software to reduce the time and effort required to tailor the standard model for a particular application. The hardware we use is even more diverse than before. Many personal computers offer as much power and more memory than the minicomputers of a few years ago; professional work stations such as the Apollo offer as much power as a large mainframe gives a single user.

This diversity of requirements calls for a new approach to DYNAMO. A monolithic program cannot answer all of these requirements. It would be cumbersome to maintain, difficult to modify to meet new requirements, and expensive for people not needing all of its features. On the other hand, a modular approach that breaks the total job into its natural subtasks and clearly specifies the boundaries between tasks, in the form of disk files, would simplify maintenance, could be packaged with as many modules as a particular user wanted, and could be extended by any computer professional to meet new requirements.

A modular DYNAMO written in a powerful computer language available on most computers would eliminate the need for the four distinct versions of DYNAMO. The C language, the source language of the UNIX operating system which has also been implemented outside the UNIX system, appears to be such a language.

The modules that have been identified to date are:

DYNAMO Editor - a text editor integrated with the DYNAMO compiler to simplify the creation and correction of DYNAMO models. It highlights each error in the model with its appropriate error message.

Model customizer - gives non-professional users a sequence of menus to select the type and number of sectors from a generic model, to create a DYNAMO model representing a specific real world system.

Compiler - translates a DYNAMO source code into intermediate code, independent of the particular machine on which it is to be run.

Native code translator - translates the intermediate code into the native code of the particular computer on which it is to be run. This module would be unique to each computer and, possibly, operating system for which DYNAMO was implemented.

Rerun customizer - shows users a series of menus to select the particular parameters of a run and output desired.

Simulator - runs a model with the specified parameters to produce intermediate (unformatted) results.

Report generator - creates reports based on the results of one or more runs according to user specifications. The reports can be laid out in any format the user desires, including graphical output.

Optimizer - optimizes model performance according to user-specified objective functions by repeated simulations and hill climbing.

Eigenvalue analysis - analyzes model behavior by computing eigenvalues, participation factors, and parameter elasticities.

Model documentor - integrates model source and definition file into an easily read document.

Flow diagram generator - interacts with user, model source, and definition file to produce a flow diagram of a model or sector.

These modules would communicate by means of a number of standard disk files. These files can be processed either by other DYNAMO modules or by user-written programs. Thus, users who do not find the particular facility they desire can build that faci-lity out of their own modules plus one or more DYNAMO modules.

Disk files accessible to the modules are:

Generic model - DYNAMO source plus menus to direct the creation of a DYNAMO model.

DYNAMO source - traditional DYNAMO source file (perhaps without PRINT and PLOT statements).

Model intermediate code - DYNAMO compiler output, independent of the particular host computer.

Model object code - output of native code translator which would be loaded or linked with the DYNAMO library and any user-supplied external functions to produce a running model.

Generic rerun - standard rerun instructions plus menus to create a rerun.

Rerun - standard rerun instructions.

Format instructions - instructions to report generator.

Unformatted results - simulation results that could be reprocessed by several other programs, such as a report generator, a graphics package or a statistical package.

Formatted results - tabular and graphic results generated by the report generator. The user might design this format to be compatible with non-DYNAMO programs.

Definitions - quantity definitions for inclusion in Documentor listing or flow diagrams, or for helping the user in reruns.

The evolution of DYNAMO has paralleled the evolution of system dynamics. Today, we need a very flexible tool on a wide variety of machines. The authors are writing a modular DYNAMO in the C language to meet this need.