EASDM: AN EXPERT AID FOR SYSTEM DYNAMICS MODELLING

González, J.C. y Fernández, G.
Laboratory of Cybernetics and Systems Theory
Escuela Técnica Superior de Ingenieros de Telecomunicación
Universidad Politécnica
Ciudad Universitaria s/n
28040 - Madrid (Spain)

Abstract: In this paper an Expert Aid for System Dynamics Modelling (EASDM) is introduced, a user friendly, interactive software tool which helps users unfamiliar with System Dynamics and computers, to construct their models from the formulation to the simulation. The most important feature that distinguishes this aid from its closest predecessor (ASDM) is the incorporation of an expert system capable of carrying out the conversion from the causal diagram to the Forrester schematics in a semiautomatic way. This is possible because, within the context of the causal diagram, there is an implicit set of "rules" which allows the classification of quantities. EASDM has been programmed in PROLOG and Pascal for personal computers with the MS-DOS operating system.

1. INTRODUCTION

The computer is a basic tool for System Dynamics. With its help, it is possible to simulate a model at low cost and in very short periods of time. Through simulation, we can compare the behaviour of the model with the behaviour observed in the actual system. So, we can determine its validity limits, i.e., a margin of conditions in which the model represents the system suitably.

But simulation is only one of the steps in the process of building models, and therefore some authors have envisaged the usefulness of extending the employment of computers to all this steps:

1. Specification of concerned quantities and couplings to establish the system boundaries and the causal diagram.
2. Quantities classification to construct the Forrester schematics.
3. Writing of model equations.
4. Simulation.

2. PRECEDENTS

In two papers, J.R. Burns (1977, 1979) described a theoretical framework consisting of a set of definitions, axioms and theorems from which he developed an algorithm for converting signed digraphs to Forrester schematics.

Based on these works, J.M. Ramos (1982, 1983) programmed the algorithm and noted some weeknesses when he began to work with some big models. After that, he introduced some modifications that gave an important increase of power to the process of classification.

Finaly, Ramos considered that it could be useful to computerize the whole modelling process. The result of his work was SIMDISCO, an integrated computer aided system for system dynamics modelling. It was programmed in APL for IBM 370 series computers.

The nest stage was to develop a similar tool for personal computers. This was the aim of ASDM (González, 1985).

3. ASDM: AN AID FOR SYSTEM DYNAMICS MODELLING

ASDM is the closest ancestor of the work we are presenting now. It was constituted by five modules:

3.1. Definition module.

Its objective is to let the user intoduce the quantities of interest for the model, as well as observed couplings between them.

3.2. Classification module.

Its aim is construction of Forrester schematics. Taking as starting point the existing algorithms, the one implemented here presents a fundamental improvement: it incorporates the necessary mechanisms to capture assignments lacking firmness during the classification process.

The algorithms presented by Burns and Ramos needed to start from a correct causal diagram. This restriction is very important, especially when the final objective is to make the construction of models easier for people unfamiliar with Forrester methodology. Moreover, many models elaborated by authors of whom a lack of experience couldn't be assumed (like WORLD-2), don't fit the axioms given by Burns in his works, yielding an incorrect classification of some quantities.

The new algorithm is able to detect and show errors in the structure of the causal diagram, also giving advice to the user if it can't finish the whole process by itself.

3.3. Equations writing module.

The user introduces model equations in the normal algebraic notation. This equations are interactively parsed by the system, that gives detailed notice of errors commited.

This module has two other remarkable characteristics: the possibility of declaring non-linear couplings between quantities in the form of tables and its ability to write state equations of the model automatically.

3.4. Processing module.

Its work consists in the elaboration of a simulation Pascal program for a model previously defined.

3.5. Simulation module.

This module allows to get results in numeric or graphic form. The integration method employed is Euler's method with a fixed interval of integration.

4. AN EXPERT SYSTEM FOR THE CLASSIFICATION OF QUANTITIES

As we have seen, the process of converting signed digraphs (or causal diagrams) to Forrester schematics could be operationalized because, within the context of the causal diagram, there is an implicit set of "rules" which makes possible the classification of quantities. Algorithms used in this process have been eventually enhanced to include new rules to the extent of detecting errors in the structure of the model. This successive improvements pointed out the lacking of an optimum and universal order for rule application. According to the model, some particular order will be more efficient than others.

The existence of such a set of rules, without an optimum order for its application, led us to carry out the process of classification with an expert system.

The knowledge representation technique used here is logic programming. We have employed PROLOG to program this module of EASDM.

4.1. Terminology

In order to represent the couplings of every quantity in the model, we will employ one predicate ("coupling") with three arguments. The causal diagram or signed digraph by which a model is to be represented will consist of a list of facts (headed Horn clauses) for that predicate, one fact for each quantity.

Just as they were defined by Burns, the terms affector and effector are used to characterize the direction of couplings and adjacency between quantities. The couplings inward-directed to a quantity q_i are called affector couplings of q_i and the couplings directed outwards from q_i are called effector couplings of q_i.

In the same way, the quantities associated with affector or effector couplings of q_i are said to be affector and effector quantities of q_i respectively.

Note that we are unaware of the coupling signs, something irrelevant for the process of interest.

This are the three arguments of the predicate "coupling":

1. The name of the quantity (q_i).
2. One list for effector couplings of q_i, whose two elements are:
   - One list with the names of effector quantities of q_i.
   - The type of these couplings.
3. One list for affector couplings of q_i, whose two elements are:
   - One list with the names of affector quantities of q_i.
   - The type of these couplings.

4.2. Consistency supposition and types of couplings

The consistency supposition establishes that all couplings directed towards a particular quantity must be of the same type, information or flow. The same thing must happen with couplings directed away from one specific quantity.

In clauses for "coupling" this is shown by de fact that we have associated a single type for each list of affector or effector quantities. When one of these lists was the empty list ([]), we will say that its type is the empty type. Before the begining of the classification we say that the type of all couplings is the "cun" type (unclassified coupling).

For example, the set of facts representing the causal diagram of figure 1 is:

```
coupling ( q1, [[q3]], cun], [[]], cun]).
coupling ( q2, [[q4]], cun], [[]], cun]).
coupling ( q3, [[q4]], cun], [[q1]], cun]).
coupling ( q4, [[q5]], cun], [[q2, q3, q5]], cun]).
coupling ( q5, [[q4, q6]], cun], [[q4]], cun]).
coupling ( q6, [[]], cun], [[q5]], cun]).
```

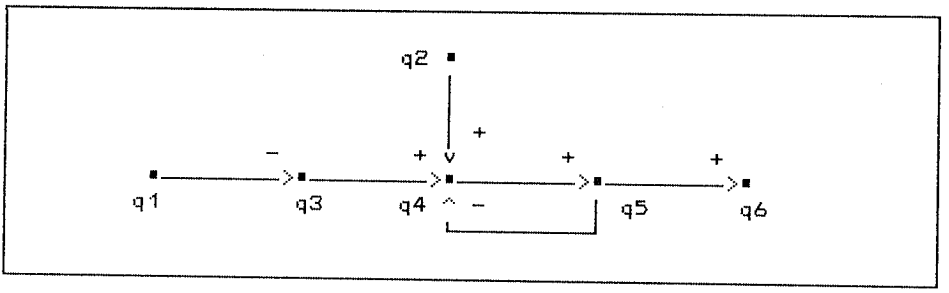

Fig. 1. Typical signed digraph.

4.3. Types of quantities

The information about the type of each quantity is stored in form of clauses too. There are eleven predicates:

- One for each type of quantity (input, output, auxiliary, rate, and level).
- One for each different type of error that can be detected during the classification (isolated, error_aux, error_rate, error_level, and error_level_rate).
- The last one, "qun" (unclassified quantities), to which every quantity belongs before classification.

All these predicates have a single argument, consisting of a list of quantities that belong to that type.

For the example of figure 1, the set of clauses for the types of quantities, before the classification, is:

```
qun ([q1, q2, q3, q4, q5, q6]).
input ([]).
output ([]).
auxiliary ([]).
rate ([]).
level ([]).
isolated ([]).
error_aux ([]).
error_level ([]).
error_rate ([]).
error_level_rate ([]).
```

4.4. Minor submodels

A minor submodel is a feedback loop consisting of two quantities, one of which is a rate and another one,that is a level.

Here, it is supposed that minor submodels have been previously identified. A new predicate has been defined, "min_submodels", that has as its only argument a list with the name of every quantity included in minor submodels. Thus, in our example there would be a clause like this:

```
min_submodels ([q4, q5]).
```

PROCEDURAL KNOWLEDGE

Until this moment, we have only presented the declarative knowledge of the problem: how to represent the particular facts known about a model. It remains now to explain how to implement the procedural knowledge, that allows to obtain conclusions from facts, and the control knowledge, that is how the procedural knowledge is applied to obtain those conclusions.

There are two works closely related in the classification process. The final objective is the classification of quantities, but we cannot achieve it without having their associated couplings classified. The basic mechanism that allows the chaining of this two works is the systematic application of the consistency supposition. This is the core of the system inference engine.

The set of clauses that makes reference to the quantities classification (the program for the predicate "class_q") is as follows:

```
class_q(Q) :- coupling( Q, [[], empty], [[], empty]),
              move( Q, qun, isolated),
              !.
class_q(Q) :- coupling( Q, [X, inf], [Y, inf]),
              fulfil( Q, min_submodels),
              move( Q, qun, error_aux),
              !.
class_q(Q) :- coupling( Q, [X, inf], [Y, inf]),
              move( Q, qun, auxiliary),
              !.
class_q(Q) :- coupling( Q, [X, inf], [Y, flow]),
              move( Q, qun, level),
              !.
class_q(Q) :- coupling( Q, [[], empty], [Y, flow]),
              move( Q, qun, error_level),
              !.
class_q(Q) :- coupling( Q, [SetEffQ, flow], [Y, inf]),
              length( SetEffQ, L),
              L > 2,
              move( Q, qun, error_rate),
              !.
class_q(Q) :- coupling( Q, [X, flow], [Y, inf]),
              move( Q, qun, rate),
              !.
```

```
class_q(Q) :- coupling( Q, [X, flow], [[], empty]),
              move( Q, qun, error_rate),
              !.
class_q(Q) :- coupling( Q, [X, inf], [[], empty]),
              move( Q, qun, input),
              !.
class_q(Q) :- coupling( Q, [[], empty], [Y, inf]),
              move( Q, qun, output),
              !.
class_q(Q) :- coupling( Q, [X, flow], [Y, flow]),
              move( Q, qun, error_level_rate),
              !.
```

Some auxiliary predicates have been defined: "move" is used to move a quantity from a quantity type clause to another one, "length" to calculate the length of a list, and "fulfil" to check if a particular quantity belongs to a list that is the only argument of a particular predicate.

The set of clauses about couplings classification (the program for "class_c") is as set out below. The predicates "class_aff" and "class_eff", in adition to classify the lists of affector and effector couplings, call the program of "class_q" to try the classification of the quantity whose couplings list has just been classified, and implement the consistency supposition.

```
class_c(Q) :- coupling( Q, X, [[], cun]),
              class_aff( Q, empty),
              !.
class_c(Q) :- coupling( Q, [X, cun], [[], empty]),
              class_eff( Q, inf),
              !.
class_c(Q) :- coupling( Q, [[], cun], X),
              class_eff( Q, empty),
              !.
class_c(Q) :- coupling( Q, [[], empty], [X, cun]),
              class_aff( Q, inf),
              !.
class_c(Q) :- coupling( Q, [SetEffQ, cun], [X, inf]),
              length( SetEffQ, L),
              L > 2,
              class_eff( Q, inf),
              !.
class_c(Q) :- coupling( Q, [X, cun], [Y, inf]),
              fulfil( Q, min_submodels),
              class_eff( Q, flow),
              !.
class_c(Q) :- coupling( Q, [X, inf], [Y, cun]),
              fulfil( Q, min_submodels),
              class_aff( Q, flow),
              !.
class_c(Q) :- coupling( Q, [X, flow], [Y, cun]),
              class_aff( Q, inf),
              !.
```

```
class_c(Q) :- coupling( Q, [X, cun], [Y, flow]),
              class_eff( Q, inf),
              !.
```

We will not describe programs for the inference engine, conceptually less interesting, longer and more complex.

CONCLUSIONS

In this paper EASDM, an Expert Aid for System Dynamics Modelling, is introduced. Its most important features are:

1. It is a computer based interactive aid which include every intermediate steps of the modelling process, from quantities specification to simulation.

2. The quantities classification is carried out with an expert system. These are some advantages of the system:

   - It is capable of explaining how a conclusion has been made or why a particular question is stated.
   - It allows to incorporate new knowledge to the expert system in a simple manner.
   - It is of assistance to the users showing errors in the structure of the causal diagram and giving advice if the system can't finish the process by itself.

3. In relation with design, it has been programmed in PROLOG and Pascal for personal computers with the MS-DOS operating system.

REFERENCES

Aracil, J. (1978). Introducción a la Dinámica de Sistemas. Alianza Editorial, Madrid.

Burns, J.R. (1977). Converting Signed Digraphs to Forrester Schematics and Converting Forrester Schematics to Differential Equations. I.E.E.E. Trans. on Systems, Man and Cybernetics, vol. SMC-7, n. 10, pp. 696-707.

Burns, J.R. (1979). An Algorithm for Converting Signed Digraphs to Forrester Schematics. I.E.E.E. Trans. on Systems, Man and Cybernetics, vol. SMC-9, n. 3, pp. 115-124.

Fernández, G. (1980). Modelos Matemáticos y de Simulación para Sistemas Continuos. E.T.S.I. Telecomunicación, Madrid.

González, J.C. (1985). Sistema Programado en Ordenador para Ayuda a la Elaboración de Modelos de Dinámica de Sistemas. M.Sc. Thesis. E.T.S.I. Telecomunicación, Madrid.

González, J.C. y Fernández, G. (1985). AMDS: Un Sistema Programado en Microordenador para Ayuda a la Elaboración de Modelos de Dinámica de Sistemas. Actas del VI Congreso de Informática y Automática.

Ramos, J.M. (1982). Diseño y Realización de un Sistema para el Modelado en Dinámica de Sistemas Asistido por Ordenador: SIMDISCO. Ph.D. Thesis. E.T.S.I. Telecomunicación, Madrid.

Ramos, J.M. (1983). Some Modifications to the Burns Algorithm. I.E.E.E. Trans. on Systems, Man and Cybernetics, vol. SMC-13, n. 1, pp. 108-110.