MES: An Expert System for Reusing Models of Transmission Equipment
S. Roody Rosales
Prem K. Mehrotra
AT&T Bell Laboratories
Middletown, New Jersey 07748
480 Red Hill Road
(201) 615-4677 or 4535

ABSTRACT

The Modeling Expert System (MES) aids in the generation of models for new transmission equipment (unit) by reusing the models of existing transmission equipment. The concept of model reuse is based on "clones." An existing model is said to be a base model for generating a "clone" (the model of a new unit), if the transmission characteristics of the two models are similar but not identical. There may be some changes ("deltas") required in the base model to generate the model for the new unit.

The knowledge required for deciding what existing models can serve as the basis for cloning is empirical, ill-structured, and known only to a few human experts. The expert system programming facilitates preservation of such knowledge in software. The "clone" determination/generation knowledge for a few equipment families has been captured in MES.

The models of the transmission equipment are presently written in PL/I programming language. The code changes ("deltas") that are required in the PL/I code of the base model, to generate the code for the new unit, do not require extensive knowledge of the PL/I language or modeling. Instead, these "deltas" are specified as templates. MES also aids in the generation of these "deltas" by using the powerful macro facility of LISP.

MES is written in OPS5 and Franz LISP, and runs under AT&T UNIX operating system. The clone determination part is largely OPS5 rule-based, whereas the code-generation software consists of LISP functions. MES also uses a pattern-action language, AWX, for finding the lines where the code changes are to be made.

1. INTRODUCTION
At AT&T Bell Labs, we develop models for transmission equipment used in the AT&T network. These models are implemented in a software system, TIRKS©/COMPUTE, and are used in the design of telecommunication circuits. When a model of a unit of new equipment is to be developed, the model of an existing unit can, often, be largely reused as the basis for the model of the new unit. The new model is then termed a "clone" of an existing, or base, model. MES is an expert system to determine what models can serve as the basis for modeling a new unit. MES also generates the code changes that are required in the code of the base model to yield the model for the unit to be modeled. This paper describes the modeling domain, the engineering rationale for the existence of clones, the knowledge required for clone determination and code generation, and the implementation details of MES.

2. OVERVIEW OF MODELING DOMAIN
The Trunks Integrated Record Keeping System© consists of application programs and data bases designed to support the telephone circuit provisioning process and to meet the expanding need for easily accessed, up-to-date circuit and inventory records. The system is large scale and consists of millions of lines of code. The COMPUTE module in TIRKS performs equipment settings and transmission performance calculations for transmission equipment.

The principal feature of COMPUTE's computation technique is that each circuit component is a unit treated essentially as a two-port network where its input voltage and current are related to its output voltage and current by a transmission (or ABCD) matrix. These matrices are multiplied together to form a resultant matrix that represents the transmission characteristics of the entire circuit. Each circuit component or unit is a piece of equipment or a facility that makes up the circuit. A model of a circuit component is a PL/I subroutine that computes the ABCD matrix for that circuit component, along with the associated I/O interfaces (also PL/I subroutines).

Model developers use modeling, transmission, and programming expertise, as well as device expertise made available from equipment manufacturers documentation such as circuit diagrams and technical specifications, or consultation to generate acceptable models. These models have a table-driven or algebraic representation. A tabular model is one that uses measured data as part of its procedure to describe the behavior of the unit to be modeled. Carrier and Trans-Multiplexer models are examples of tabular models. In contrast, an algebraic model is defined as one that describes the behavior of the unit to be modeled as a frequency function with parameters derived from circuit analysis. The model description can, for example, be represented as algebraic equations obtained through nodal analysis of the circuit representing the unit. The D4 channel unit and MFT unit models are examples of algebraic models.

Existing models often serve as the basis for the new units to be modeled. This is particularly true if the unit to be modeled has transmission characteristics similar to those of an existing model. By cloning a model of a new unit from an existing model, a great deal of the existing code is reused.

1. TIRKS is a trademark of Bell Communications Research, Inc.
2. The Trunks Integrated Record Keeping System (TIRKS) was used by the Bell System to provision circuits prior to digitalization. At digitalization, a version of TIRKS was turned over to AT&T Communications where it has been evolving along a different path from the TIRKS version that remained at Bell Communications Research.
3. A D4 channel unit is a network component that provides the major function of converting an analog voice band signal (4 KHz) to a single pulse amplitude modulated (PAM) signal. These units are usually found in a bank where their PAM output is encoded, multiplexed, and a 24 (4 KHz) channel, bipolar, three multiplexed signal (US1) is produced and ready to be transmitted via a 71 carrier. There are many different types of D4 channel units. Their use is application dependent and each has to be adjusted to provide optimum transmission.
4. These designations represent different functional capabilities for telecommunication services provided by D4 channel units, each designation is a particular type of D4 channel unit:
   - 4FXO - Four Wire Foreign Exchange Office End
   - 4FXS - Four Wire Foreign Exchange Station End
   - 4EX - Four Wire Duple
   - 4E10 - Four Wire Eightfold Transmission Only
To determine if a base model exists for a new unit an analysis of the adjustment, or equalization, switch is physically labeled on the unit. For example, in the case of 4-channel units, relevant switches will provide gain, impedance adjustment, or equalization.

Modeling experts use various heuristics and rules to decide what properties a unit should have to serve as a clone of another unit. For example, if the \textit{D4 channel unit}\textsuperscript{2} to be modeled has an equalizer with band-width, height and slope switches, then it is likely that the D4 models of the existing 4FXO, 4FXS, 4DX or 4ETO units\textsuperscript{3} can be used as clones. It is valuable to preserve this expertise. The expert system programming languages can facilitate preservation of such expertise.

Once a base model is determined for cloning, the generation of the modeling code for the new unit involves making changes in the various routines of the base model. Even though numerous changes are involved, each change is simple in nature: editing a line of code, or insertion of a new block of code. Most of these changes are specified by templates, and do not require a user to have any deep knowledge of LISP programming or modeling. Automation of these code changes not only saves significant time, but also produces standardized code. It takes a human expert several days to fully debug and incorporate these changes; once automated, these changes are made in a few minutes.

This process of model generation will be referred to as \textit{model cloning} and the model resulting from cloning the existing model, a \textit{clone}.

2.1 \textbf{Engineering Rationale For Existence Of Clones}

Equipment models may be candidates for cloning if the equipment functions are identical or similar to those of the new unit, if their measured performance is close enough to that of the new equipment unit, and if their specifiable control options are similar enough to the new equipment unit. This is frequently true since the clone and its base model are designed to the same common specification.

Transmission networks and equipment can be separated into families that are classified by the function they provide. TRUNK\textsuperscript{4}/COMPUTE is primarily concerned with those functions affecting transmission performance (options) that are adjusted or tuned by the equipment installer or remotely programmed, e.g., D5 channel units. At present, the COMPUTE\textsuperscript{5} adjustable transmission functions found in any piece of equipment modeled (in COMPUTE) are as follows:

- Gain: in the AZ (left to right) and ZA (right to left) direction
- Equalization: in the AZ and ZA direction
- Nominal Impedance: at the A and Z side
- Balance: at the A and Z side
- End-Section (Line Build Out): at the A and Z side

Modeling code to simulate these functions is required within the subroutines used by COMPUTE. Since the modeling of each function is based on the particular physical characteristics of the switch controlling that function, it follows that all physical switches controlling a particular function will require essentially the same modeling code with at most some modifications. For example, the modeling code used to model attenuator switches from two different units, both having a range of 0.00dB to 16.5dB in steps of .1 dB, is identical (the only thing that may vary is how each attenuator is physically labeled). If the switches have different physical labels, then some code changes are required to take these differences into account. Since all equalizing \textit{D4} channel units use the same equalizer models, then any new equalizing unit will use the existing equalizer modeling code.

To determine if a base model exists for a new unit an analysis of the various adjustable transmission switches is required. This analysis identifies the range of values attainable by each switch and how each switch is physically labeled on the unit. For example, in the case of 4-wire \textit{D4} channel units, relevant switches will provide gain, impedance adjustment, or equalization.

Upon analysis of the modeling code and the physical units the code describes, a series of rules for adding new (and of course similar) units to the existing modeling code is derived and codified. These rules entail modifying existing code, and adding new code (lines or blocks) to the existing modeling code.

3. \textbf{SCOPE OF THE MODELING EXPERT SYSTEM}

The modeling expert systems consists of two parts: clone determination and code changes generation.

The clone determination part is an aid in finding what existing models can serve as the basis for creating a model for the new unit. As was stated earlier, the knowledge required in finding base models for cloning is heuristic, ill-structured and known only to a few human experts. Details of the clone determination process are presented in section 4.

Once a base model is determined, several changes (deltas) are required in the code of the base model to generate the modeling code for the clone (new unit). Most of these code changes are easily automated, in particular using symbolic languages like LISP that provide a rich set of facilities for template processing. Details of code-change generation is presented in section 5.

4. \textbf{CLONE DETERMINATION}

The models of equipment that can be used as a basis for clone generation are classified in terms of equipment families, e.g., D4 channel units, Trans-Multiplexers, D5 channel units, Customer Premise Equipment, etc. MES at present knows only about D4 and Trans-Multiplexer families. MES is designed to take full advantage of the user's supplied information to minimize its search space. MES first asks the user what family(ies), MES should search. For example, if the user thinks that the unit to be modeled is a D4 clone; MES will ask only D4 related questions and examine only D4 rules. If a user is not sure of a family, then MES tries all families one by one.

MES loads all the unit models in a family, for example nine units for the D4 family, in its working memory as the possibilities (hypotheses) for the clone determination. It then asks the user for more information on the unit to be modeled. For example, MES asks the user whether the unit has an equalizer. If the unit has an equalizer, then the unit have band-width, height and slope (BHS) switches? Does the user's unit have a -7dB attenuation switch? Does it have an impedance switch in transmit direction, etc? As was pointed out, MES's knowledge base consists of various rules specifying that certain units have certain components. Examples of the two rules are given in Fig 1. For example, only the 4FXO, 4FXS, 4DX and 4ETO units have equalizers that have BHS switches. If the user's unit did have an equalizer with BHS switches, then all the hypotheses other than the above four will be eliminated.

MES narrows down the clone possibilities, based on the user's responses. MES also provides the option for a "do-not-know" response. For example, if user does not know whether the unit has an equalizer, MES takes a default course of ignoring the question in its decision making. Once all the questions have been asked, MES's recommendations are all the possibilities that have not been eliminated.

The dialogue carried out by MES is context-sensitive and "intelligent". MES takes into account all the user's responses when asking a new question. It tries to ask a minimum number of questions. For example, if all the possibilities have been eliminated during the interactive dialogue, MES will immediately inform the user of this fact, and will not ask any questions related to that equipment family. The questions asked have also been ordered according to number of hypotheses they can eliminate. MES thus first asks questions which are likely to eliminate the maximum number of hypotheses, thereby minimizing the number of the questions asked.
4.1 Explanations

MES provides "why" type of explanation capabilities of Expert Systems [1]. In response to a "why", it prints out all the hypotheses that are being considered at that point, and informs the user that it needs the answer to that question to narrow down and/or confirm those possibilities. Some of the information used during clone determination, for example insertion losses and attenuation ranges, is also used during code generation; conversely some information is used only for code generation. MES informs the user about the purposes for a question, when a user wants to know why a question is being asked.

EXAMPLES OF MES RULES

1. IF UNIT TYPE IS D4
   AND UNIT HAS AN EQUALIZER
   AND UNIT HAS BHS SWITCHES
   THEN 4FXO, 4FXS, 4DX OR 4ETO UNITS
   CAN BE USED AS BASE MODELS

2. IF UNIT TYPE IS D4
   AND UNIT HAS AN ATTENUATOR
   IN TRANSMIT OR RECEIVE DIRECTION
   AND THERE EXISTS A HYPOTHESIS FOR A
   4PLR UNIT
   AND HYPOTHESES FOR OTHER D4 UNITS
   AND ATTENUATION RANGE IS BETWEEN
   0 AND 6.3
   THEN PICK 4PLR AS A BASE MODEL
   OVER OTHERS

Fig 1

5. GENERATION OF CODE CHANGES

As was stated earlier, some changes to the code of the base model are required to generate the modeling code for the new unit (clone). Typically, the modeling code for a unit consists of several PLI files, e.g. typed4, subna; as shown below in Fig 2.

In a subroutine, code changes (delta's) may be required at several places. For a D4 clone, there are more than twenty five places where code has to be changed.

Code changes are of two types: editing the code of the base model, and generation of the new code.

5.1 Editing Code of the Base Model

Some examples of the code editing are presented below. For example, the dimension of the legalnames array needs to be changed from 27 to 28

```plaintext
legalnames(27) char(13) var init
```

to accommodate the new unit in the subna file in TIRKS/COMPUTE. There is a name associated with each unit. If the length of the name of the new unit is more than 13, then the dimension of char in the above line must also be changed. Another example of the code editing is insertion of the user's unit name in the following line of the code. The name is to be inserted inside the init (....).

```plaintext
eq_code(14) char (8)
    init ('d3fxs', ..... 'd4sf', 'd4totr') static;
```

5.2 Generation of New Code

In addition to editing code of the base model, some new code must be added to the code of the base model to generate the code for the new unit. The new code generated can be a simple line, or a block of code or an entire subroutine.

5.2.1 Code Generation is Template Driven

The new code to be generated is generally template driven. Most of the code in a template is fixed, while some code is variable. The variables in the template are simple variables, or function calls that fetch the value of the variables.

An example of a new line of code template is:

```plaintext
when (<user-unit-name>) if (<base-ilos-c-name>)
    then return ("0'b);
```

![Diagram](image-url)
The variables shown inside the angle brackets are to be replaced by their values. For example, the name of the new unit should substitute `<user-unit-name>`, and the name of the base model subroutine for insertion loss should be substituted in place of `<base-iloss-name>.

The example for a subroutine generation is shown below:

```lisp
<user-unit-name>: proc returns (bit(1));
  xmt_sw = xmt_sw + (<get-transmit-loss>);
  rcv_sw = rcv_sw + (<get-receive-loss>);
  return <base-name>;
end <user-unit-name>:
```

Once again, the variables are shown in angle brackets. The "get-transmit-loss", in fact, denotes a function call. The value returned by the get-transmit-loss and get-receive-loss should be substituted in their respective lines. Also note that the line for transmit switch (xmt_sw = ...) is to be generated only when the new unit has a -TDB transmit switch. Similarly, the rcv_sw line is to be generated only when the new unit has a receive switch. These rules are all automatically enforced by MES. The base-name is the name of the selected base model (e.g., D4-EM).

5.2.2 LISP is Great For Template Processing

The symbolic manipulation capability of LISP is very useful in the generation of the code based on templates. The templates are essentially written as their English representation using the back-quote['] macro facility of LISP. LISP provides several features that make it very convenient for code generation. One does not have to declare strings and their lengths, convert integers into symbols, etc. when processing templates. Several strings can be concatenated using one function call. In order to generate code based on templates in a language like C, one has to write several printf statements and make numerous calls to the strcat function.

5.3 Knowledge Required For Code Generation

In addition to knowing the templates for the new code generation, and the lines that are to be edited, the following information may be required for the generation of the code changes:

- **Information from the User**

  During the code generation, MES may ask the user for information, e.g., name of the unit, documentary information about the unit, etc. For example, to generate the following line of new code:

  ```lisp
  <unit-name> fixed bin(15) init(417);
  /* <documentation> */
  ```

  the user will be asked to supply the unit-name and the corresponding documentation.

- **Information Supplied/Derived During Clone Determination**

  As was stated in section 4, some of the information asked during clone determination is also utilized during code generation, e.g., the insertion losses, and attenuation ranges. The code generation routines, in fact, need the difference of the insertion losses between the new unit and the clone. The difference is derived during code generation.

- **Extracting Information From the Existing Code**

  The number 417 in the line of the code discussed in section 5.3 was, in fact, derived by incrementing the number 416 in the last line of the code in the typed4 file of the clone. This file consists of following code:

  ```c
  d4sf_4 fixed bin(15) init(415),
  d4totr_4 fixed bin(15) init(416);
  /* 4w vendor-x st */
  ```

  MES automatically extracts the number 416, by parsing the last line of the include file, and increments it by 1 while generating a line of code for the new unit.

5.4 Adding Code Changes In the Files of the Base Model

The code changes generated must be added (made) in the code of the base model to generate the code for the new unit. When editing a line of code, MES generates a new line of code, and this line must replace the corresponding line in the code of the base model. When a new line or block of code is generated, it must be inserted at the proper place in the base model files.

5.4.1 Use of AWK Pattern Matching Language

MES uses AWK(14) procedures to select the line(s) that are to be edited, and to find line numbers where the code is to be placed. The "markers" are placed in the code of the base model around the lines where the code is to be edited or inserted. The AWK language is designed to find patterns in the files. The AWK procedures in MES look for these markers and save the extracted lines of code and/or line numbers on scratch files.

5.4.2 Use of Editor to Make Code Changes

Once it is known where the code is to be replaced/inserted, it is possible to write editor (e.g., sed(1)) scripts to make the code changes automatically. MES, at present, however does not make these changes automatically. Instead, MES takes the user to the places (lines) where the code changes are to be made using the automatic invocation of the VI editor command. MES then asks the user to make these changes manually. There are two reasons for this partial automation. One reason is that in the model development stage we want to make sure that new code generated gets fully debugged. The other reason is that the code generated by MES needs some beautification - the right justification of part of the code in a line of code to a certain column. The FRANZ LISP, OPUS (Version) 42.25(13), at present, does not have its right justification formatting capability working. This makes the placement of part of a code in a line at certain column very difficult to implement. Hopefully, in the newer OPUS's of FRANZ LISP, this problem will be rectified.

6. SOFTWARE AND HARDWARE

MES is mainly written in OPSS(10) and FRANZ LISP, OPUS 42.25, and runs on a VAX 8650 computer under AT&T UNIXTM System V, Release 11. The OPSS version used by us is written in FRANZ LISP. The clone determination part of MES is largely rule-based, and contains 120 OPSS rules. The code generation part is mainly written as LISP functions, and contains approximately 1000 lines of LISP code. Code generation also uses AWK scripts, and there are 25 AWK scripts in MES. These AWK scripts are used for selecting the lines to be edited, and finding the line numbers where the code is to be replaced/inserted. AWK was found to be very useful for finding the lines with certain patterns in a file.

7. CONCLUSIONS

In this paper, we have shown how the expert system methodology is useful in reusing models of transmission network equipment, where reuse is based on heuristics and ill-structured knowledge. We have also utilized the rich set of facilities of the LISP programming language for the "template-driven" code generation. The techniques presented in this paper should be applicable to reuse in other domains.
8. ACKNOWLEDGEMENTS

We are extremely grateful to our supervisors Abe Ososky and Peter Ting for their encouragement and support.

REFERENCES

5. Franz LISP, OPUS 42.14, "Franz Inc.", Alameda California.