Section B. Editing and Processing

The Models4 System: Simplifying data management in Blaise 4 Windows

Boris Allan, Westat (USA)

Summary

The Models System for Blaise III[4] performed the following actions:

1. The data model was split into component data models (`.BLA` files), where each of the component data models was equivalent to an ASCIIRELATIONAL file. A batch file was created to drive the remaining steps.
2. Each data model produced as a result of the previous step was then prepared to create meta information (`.~MI`) files.
3. A chosen Cameleon translator was then applied to each meta information file.

We were able to take large data models, and produce SAS macro descriptions for the whole model, where each SAS macro corresponded to an ASCIIRELATIONAL file. The SAS macros were created by a special translator `SASMACRO.CIF`.

With the release of Blaise 4 Windows, some of the command-line parameters for Manipula and Cameleon changed slightly, and batch files did not seem to fit well with a Windows environment. I decided, therefore, to change the models methodology to incorporate these new features, which included an improved Cameleon interpreter. The Models4 System follows a different sequence of actions:

1. The data model is split into component data models (`.BLA` files), where each of the component data models is equivalent to an AsciiRelational file (the new version uses improvements in the Blaise 4 Cameleon interpreter). At the same time, a reference file is written (`.REF`), where each record in the reference file has the name of a component data model and its folder.
2. Each record in the reference file designates a data model to be prepared to create a meta information (`.~MI`) file.
3. Each record in the reference file also, therefore, designates the meta information file to which a chosen Cameleon translator is applied.

One consequence of this new approach is that it is possible to construct a reference file containing a list of files (without the `.BLA` extension), prepare the named files, then apply the same Cameleon translator to all these files (see *The elements of Models4*).

The background to Models4

At Westat, we often have to deal with very large instruments where we cannot write or read a file with records that contain all the fields for an instrument. We have to divide the fields into manageable chunks corresponding to blocks, and those chunks are AsciiRelational[5] files. As an example, one "very large" study has a data model with over 166 different AsciiRelational files and a meta information (`~MI`) file greater than 3.5 MB.

---

[4] *Automatic generation of data descriptions for ASCIIRELATIONAL files*, International Blaise User Group Newsletter, July 1998.

[5] I try to distinguish between MSDOS and Windows versions of (say) files by using all `UPPERCASE` for MSDOS and `MixedCase` for Windows.

Because of the work involved in managing this amount of information, we had to find a way of making the task less labour-intensive. For example, for the data model with 166 different files, the file produced by the Blaise 4 Cameleon translator `AsciiRel.CIF` has 8060 lines. Developing a Manipula setup to write AsciiRelational files is a simple task (one need only use the Wizard) and the fact that the Manipula setup might take a long time to write the files is unimportant. I, for one, can easily find other things to do whilst that is happening — usually involving food.

We needed an equivalent of the Manipula Wizard to produce data models that were equivalent to the formats of the AsciiRelational files. These formats are implicit in the output of `AsciiRel.CIF` but have to be made explicit, because all the data models are in one file and cannot be prepared until the file is edited, and each data model copied to a separate file. Once we have these separate files, we have to prepare each data model.

If you think of the amount of work involved in splitting a file into 166 data models, and preparing those 166 models, then you can see why automation sung sweetly. For many of our studies, Models4 is the only way we can conveniently manage the conversion of Blaise data into other forms for analyses.

The development of Models4

In the days of Blaise III, the original task was resolved by basing a Cameleon translator on the Blaise III `ASCIIREL.CIF` translator, with certain modifications. Called `BLKMODEL.CIF`, the translator had to overcome certain limitations in the Blaise III `CAMELEON.EXE` interpreter:

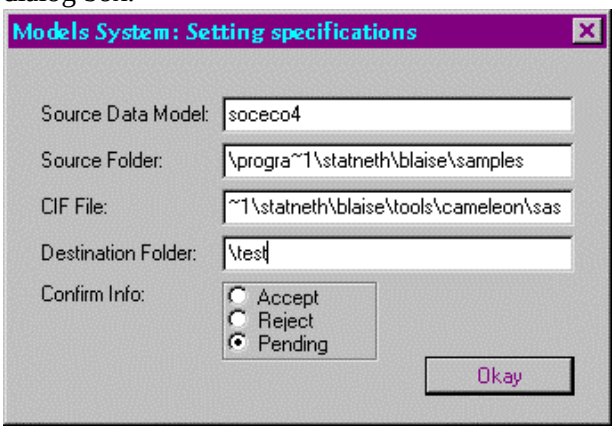
- One limitation was structural, in that you could not declare arrays of variables in the `[VAR ...]` section, and it was difficult to keep track of repeated types .
- Only types and embedded blocks at the top level were specified in the data model corresponding to Block 1 (the original data model). For data models corresponding to other blocks, type definitions (and similar) had to be copied from the top-level definitions.
- In a `TYPESLOOP ... ENDTYPESLOOP`, the only types tracked were types that had been defined at the level of the block, not types for the whole data model. Looping through types at the data model level did not include types defined at lower levels. `BLKMODEL.CIF` had to take a field at the block level, and search through the tree structure for enclosing blocks, until the top level was reached, to find the corresponding type definition (a reverse recursion). As each field had to be treated separately, and there was no central repository of types, type definitions were repeated.

  Another program, a Manipula setup called `CLEANTYP.MAN`, was used to remove duplicate definitions.
- `ASCIIREL.CIF` did not deal with classifications, and so a procedure to regenerate a classification was implemented.
- To deal with embedded blocks, and *nested* embedded blocks, special procedures had to be implemented for blocks at lower levels, if those blocks had fields referring to embedded blocks in their declarations.
- Various problems arose with the way the `CAMELEON.EXE` interpreter treated Blaise III `SET OF` declarations, and with the way the interpreter treated the Cameleon meta data distinction between `DEFINEDTYPENAME` and `PREVIOUSTYPENAME`.

Some of these points have been addressed in the Blaise 4 `Cameleon.EXE` interpreter, and the Blaise 4 `AsciiRel.CIF` translator. The Blaise 4 equivalent of my `BLKMODEL.CIF` is my `Create_Models.CIF`, which is based on ideas in the Blaise 4 `AsciiRel.CIF` translator, with some modifications. Generally speaking, Blaise 4 Cameleon is much more reliable than before, and this is reflected in the improved quality of the supplied sample translators.

Models_System.MAN: The interface to Models4

To start the Models4 System, you use the Maniplus setup `Models_System.MAN`, with the following dialog box:



In this example dialog:

- The Source Data Model is `SocEco4,` a variant of the `SocEco` sample provided with Blaise 4 — the listing of `SocEco4.BLA` is in the Appendix.
- The Source Folder is `\progra~1\statneth\blaise\samples` (using the abbreviation `progra~1` for `"program files"` because the version with the space can cause confusion for some of the Blaise 4 system command-line interpreters).
- The CIF File is at location `\progra~1\statneth\blaise\tools\cameleon\sas`.
- The Destination Folder for all the created models, meta information files, and Cameleon translations is `\test` (I believe in originality in choice of folder names).
- The Confirm Info: selection is used to stop the dialog being closed prematurely, if you hit an enter key by mistake. The dialog will only close if Accept or Reject is selected. The Maniplus setup runs the `Create_Models.CIF` translator to write a whole series of data model files, and to write a reference file (`SocEco4.REF`) in the working directory, containing the fully specified names of the data model files:

```
\test\soceco4_A01
\test\soceco4_A02
\test\soceco4_A04
\test\soceco4_A05
\test\soceco4_A06
\test\soceco4_A03
```

The first part of the name is the meta-information file name (`SocEco4`) and the second part (after the underscore) is the extension for the appropriate AsciiRelational file name. Listings of `SocEco4_A01.BLA` and `SocEco4_A06.BLA` are in the Appendix.
The SAS file written for `SocEco4_A06.BLA` is in the Appendix.

The elements of Models4

Apart from `Models_System.MAN`, the key elements of Models4 are the following:

- `Create_Models.CIF`
- `Generate_MetaInfo.MAN` (and, of course, `Generate_MetaInfo.MSF`)

Each file can be executed separately, and for `Create_Models.CIF` the command line is:

```
CAMELEON Create_Models /B /DSourceDataModelPath /PDestinationFolder
```

- The *`SourceDataModelPath`* is the fully specified path and name of the source data model — for example, `\progra~1\statneth\blaise\samples\soceco4`.

- There is one parameter: the *DestinationFolder* is the folder in which the models are created — for example, `\test`.

The command line for Manipula/Maniplus is (one line, though I have it split over two):

```
MANIPULA Generate_MetaInfo /ISourceDataModelName.REF /Q
    /PCifFilePath;DestinationFolder;DoPrepare;DoCIF /RSourceDataModelName.MSG
```

- The *SourceDataModelName* is the name of the source data model, to which is appended **.REF** — for example, `soceco4.REF` (this is the reference file). When `Generate_MetaInfo.MAN` is being run independently of the Models4 System, you can use any approriate file name.
- There are four parameters that follow the **/P**:

  CifFilePath (corresponds to CIF File in the dialog);
  DestinationFolder (corresponds to Destination Folder in the dialog);
  DoPrepare, that is, are the names in the reference file to be prepared? (Y if true);
  DoCIF, that is, are the names in the reference file to be translated by a CIF? (Y if true).
- The *SourceDataModelName* is the name of the source data model, to which is appended **.MSG** — for example, `soceco4.MSG` (this is the message file). When `Generate_MetaInfo.MAN` is being run independently of the Models4 System, you can use any approriate file name.

Appendix: Example listings

SocEco4.BLA (based on SocEco.BLA supplied with Blaise 4)

```
DATAMODEL SocEco4;     {Complete change from SocEco: TYPE and BLOCK}

PRIMARY PersonCode

TYPE
  TYesNo = (Yes,No)
  TFreqy = (Weekly, Monthly)
  TCode =   1..50
  TEarn =   0..10000

BLOCK PersonCodeBl
  FIELDS
    PersonCodeQ "What is your personal code?" : TCode
  RULES
    PersonCodeQ
ENDBLOCK

  BLOCK BSalary;
    FIELDS
      Income "Do you earn a regular salary?": TYesNo
      Amount "How much to you earn?": TEarn
      Period "Is this weekly or monthly?": TFreqy
    RULES
      Income
      IF Income = Yes THEN
        Amount
        Period
      ENDIF
  ENDBLOCK

  BLOCK BSeason
    FIELDS
      Income "Do you draw an income from seasonal work?": TYesNo
      Amount "How much do you earn per season?": TEarn
      RULES
        Income
        IF Income = Yes THEN
          Amount
          DUMMY
        ENDIF
  ENDBLOCK

  BLOCK BAllow
    FIELDS
      Income "Do you receive an allowance?": TYesNo
      Amount "How much do you receive?": TEarn
      Period "Is this weekly or monthly?": TFreqy
    RULES
      Income
      IF Income = Yes THEN
        Amount
        Period
      ENDIF
  ENDBLOCK
```

```
TABLE TIncome "Income specification"
  FIELDS
    Salary: BSalary
    Season: BSeason
    Allowance : BAllow
  RULES
    Salary
    Season
    Allowance
ENDTABLE

FIELDS
  PersonCode: PersonCodeBl
  Income: TIncome

RULES
  PersonCode
  Income
ENDMODEL
```

SocEco4_A01.BLA (An example data model)

```
DATAMODEL SocEco4 {FileNumber:= 1}

  FIELDS
    FPrimary: STRING[2]
    InstanceNumber: 0..99999
    PersonCode: 0..99999
    Income: 0..99999
ENDMODEL {SocEco}
```

SocEco4_A06.BLA (An example data model)

```
DATAMODEL BAllow {FileNumber:= 6}

  TYPE
    TYesNo = (Yes, No)
    TEarn = 0..10000
    TFreqy = (Weekly, Monthly)

  FIELDS
    FPrimary: STRING[2]
    InstanceNumber: 0..99999
    Income "Do you receive an allowance?": TYesNo
    Amount "How much do you receive?": TEarn
    Period "Is this weekly or monthly?": TFreqy
ENDMODEL {BAllow}
```

SocEco4_A06.SAS (An example SAS data description)

```
TITLE 'BAllow';

PROC FORMAT;

VALUE TE_1F
  1='Yes'
  2='No'
;

VALUE TE_2F
  1='Weekly'
  2='Monthly'
;
```

```
RUN;

DATA FILE;
INFILE 'soceco4_A06.ASC';
INPUT
    FPrimary $  1 -  2
    Instance    3 -  7
    Income      8 -  8
    Amount      9 - 13
    Period     14 - 14
;

LABEL
  FPrimary   = 'FPrimary'
  Instance   = 'InstanceNumber'
  Income     = 'Do you receive an allowance?'
  Amount     = 'How much do you receive?'
  Period     = 'Is this weekly or monthly?'
;

FORMAT
  Income    TE_1F.
  Period    TE_2F.
;

RUN;
```