Autogenesis: Automatic Generation of a SAS Data Entry Application.

Derek Chalmers, GlaxoWellcome, Research Triangle Park, N.C.

Introduction

This paper describes a project whose objective was to pilot automatic creation of data entry applications; Autogenesis. Regardless of the software used, manual design, creation and validation of data entry applications can be a resource intensive process. Currently Medical Data Sciences (MDS) at GlaxoWellcome use EntryPoint 90 as the preferred software to build data entry applications. Programming, testing and validation takes typically 4-6 weeks and is carried out by a relatively small pool of available programmers. The company has developed an in-house database to manage the collection and processing of nearly all clinical trial data. The database, known as GOLD, is an Oracle-based product located on a VAX platform. GOLD is available to the corporate Clinical Data Management groups via three instances: one to serve the US, another in the UK and a third to handle European operating company studies. GOLD has a number of components including among others: Assessment Dictionaries, a Database Set-up library, a Processing database and a Reporting database; however, GOLD does not incorporate a data entry subsystem. When a new study protocol is being initiated, MDS will build a relevant database structure by adding to the Assessment Dictionaries and Database Set-up subsystems of GOLD. The challenge of autogenesis is to exploit the dictionary and set-up information; collectively known as Meta-Data, and use it to generate data entry applications automatically.

The Environment

The prototype application was developed on a PC using SAS version 6.08 for Windows 3.1. The application consisted exclusively of SAS/AF: Software Frame entries working in conjunction with various ancillary formats, macros and code generators. GOLD data was made available to the prototype by a two step process: first, all Oracle tables containing the meta-data were visualised through SAS/Access views using SAS version 6.08 for VMS on the VAX cluster. Secondly, SAS/Connect Remote Library Services (RLS) were exploited to make the VMS views available to the PC-SAS session. This bridging system allowed the prototype to read all required meta-data directly from the live database while avoiding any of the requirements usually associated with a permanent dataset library; such as nightly update cycles or large disk volumes.

The Source Meta-data

The meta-data is the driver of autogenesis. It is important to bear this in mind, as the effectiveness of the data entry child application produced by autogenesis is governed entirely by the quality and completeness of the meta-data. The meta-data can be thought of as an electronic specification of the case record form (CRF).

There are five main groups of meta-data information used by the prototype:

- **Therapeutic area & Protocol:** With around 60 therapeutic areas each containing an average of 15 distinct protocols these keys were essential to help subset the remainder of the meta-data.
- **Pages per protocol:** The CRF pages present, their order, and what clinical visit relates to each CRF page.
- **Sections per page (and their sequence):** A section is a group of related assessments; there are usually between 1 and 5 sections per CRF page. For example, a DEMOGRAPHY page may contain both CHILD BEARING POTENTIAL and SMOKING HISTORY sections. There may be more than one section per CRF page and there is corresponding meta-data available which stores the order that sections appear within a single CRF page.
- **Assessments per section (and their sequence):** An assessment is a single data point (Oracle column or SAS variable) for example SUBJECT is a patient's unique number; SEX is a patient's gender. The particular assessments per CRF section and the order that they present is stored within the meta-data.
- **Assessment attributes:** The name, length, type, label, format and valid-values associated with a CRF assessment when it is stored on the database.

The Autogenesis Model

The prototype application is known as the parent. On user selection of a protocol number, this application uses the information stored in the meta-data to generate a novel SAS catalog: the child application (See Figure 1). The child application consists of a child data entry application de novo takes around 4 minutes on a Pentium PC; all processing is generic and independent of additional user input.

By using autogenesis to generate data entry applications in this way only database set-up resource is required - not database and data entry application development.

Within the Child application data entry screens are presented to the operator in the order defined by the CRF. The cursor will follow the logical flow between assessments, even though the screens display the fields in a vertical format. Sections are displayed one screen at a time, in CRF page order.

Autogenesis excludes any 'customisation' of the data entry screens. Typically, screen painting activities can be used to make the data entry screen look more like the printed CRF page and inability to do this is recognised as one of the major limitations of autogenesis.
There were four key stages in the creation of a child data entry application using SAS as the Autogenesis software (See Figure 1):

1. **Generation of template datasets.**
   Template datasets were created on a one dataset per CRF-section basis: this would later ensure one data entry screen per CRF-section. The meta-data table containing the assessment attributes was joined with the sections table and the resulting dataset fed through a 'code-generator'. This program was simply a series of 'file-put' datasteps to generated ATTRIB statements that could be pulled by %INCLUDE into a dataset creation macro.

2. **Generation of SCL code.**
   The page and section details tables were joined to create a dataset holding the complete order of sections per page per protocol: once again 'file-put' code generation macros were used to create two types of SCL code based on this dataset. Firstly, a single piece of SCL code that would later be compiled behind a template 'front-end' frame entry and used to 'flow' all screens in the correct order. Secondly, the macros generated one SCL program per dataset - these contained instructions specific to the assessments included on any given section, as well as the SYMPUT and SYMGET SCL functions used to carry forward key information between separate data entry screens (e.g. SUBJECT).

3. **Generation of Data Entry Screens.**
   SAS /FSP-FSEDIT screens were generated using the template datasets already created. This was carried out from within an SCL DO-loop inside the parent application (See Figure 2). CALL EXECCMDI was used to issue the PROC FSEDIT command in conjunction with SCL variables holding the dataset and screen names. Various MOD commands were executed using CALL EXECCMDI so that the SCL code previously generated could be compiled behind each new FSEDIT screen. The LABEL option was used with PROC FSEDIT so that the value of the assessments label was used to describe the field on screen; instead of simply using the SAS variable name which would be meaningless to data entry operators.

In order to progress through the DO-loop, a dataset containing the names of all the screens was used to generate a format based on the observation number. This format could then be used to translate the value of the loop iteration variable (i) to the name of an FSEDIT screen. The final iteration of the DO-loop was determined using the total number of observations, i.e. the number of datasets/screens.
The user must first have selected a protocol identifier and hit a button called 'BUILD' to execute this code. The macros DSETS and FRAMETXT are code generators which create the template datasets and child application SCL code respectively. A DO-loop is used to cycle through all the datasets in turn, creating one FSEDIT screen per template dataset. The bounds of the DO-loop are extracted from the dictionary tables. The name of the dataset and screen in the loop is accessed from a format that was previously created using a dictionary view of the template datasets. The PROC FSEDIT command is issued using CALL EXCECMMDI functionality in order to pass a string of additional MOD commands: these facilitate inclusion of the SCL already generated by the code generators. The user's screen is updated with the progress of the autogenesis.

BUILD:  
* CREATING DATASETS AND WRITING APPLICATION CODE ;
text="Creating template datasets...";
refresh;
send continue;
   %dsets;
send submit;

text="Writing application SAS code...";
refresh;
send continue;
   %frametxt;
send submit;

text="Generating data entry screens...";
refresh;

* COUNTING THE NUMBER OF DATASETS TO LOOP THROUGH ;
send continue sql;
   reset noprint;
   select count(libname) into :dsetn
   from dictionary.tables
   where libname='DATA';
   quit;
send submit;

* LOOPING THROUGH ALL THE DATASETS TO BE PLACED BEHIND SCREENS ;
loop=symgetn('dsetn');
do i=l to loop;

* FORMATTING THE DATASET NAME BASED ON THE POSITION IN THE LOOP ;
call symput ('dset' ,put (L, secs.));

* OPENING AN FSEDIT SCREEN (+ LABELS) FOR EVERY LOOPED DATASET AND INCLUDING THE SPECIFIC SCL FOR EVERY LOOPED DATASET ;
call exccmdmi
   ('submit "proc fsedit data=data.&dset screen=desa.&prot..&dset..screen label modify;run;";3;inc "c:sas\desa\scl\&dset..txt";
   end;end;end; ') ,
call exccmdmi(' ');

text='Creating screen '||l||' of '||loop ;
refresh;
end;
text="Hit 'COMPILE' to complete the new application";
return;
4. 'Front-ending' and Compiling the Child Application.
PROC CATALOG was used to combine all the FSEDIT screens with a template frame entry that would serve as a front-end menu for the new child application. Once again, CALL EXECCMDI was used to compile the master SCL code written by the generator macros behind the template frame thus completing the child application functionality.

Pilot Assessment

The project was successful by virtue that autogenesis was proven to be an inexpensive way of generating a potentially resource intensive product. This pilot application was designed and constructed in approximately twelve weeks of intense development and as such was never intended to enter production as a clinical system without a further development phase. Various other components of a good data entry system were excluded from the pilot development; such as double independent entry capability and data export functionality. However, the pilot demonstrated sufficient potential to convince senior management that the autogenesis concept represents a possible strategic advantage and worthy of further resource investment. This project is ongoing to develop a production system by second quarter 1997, there is currently beta-release software undergoing formal multi-site testing with promising results.

As a postscript to this paper, the author must point out that the final software platform of choice for the autogenesis product was not SAS. Microsoft Access was chosen for a number of reasons. Principally, these included the licensing requirements for SAS: these would be required for every machine running a child application - an expensive option for some external data entry locations who have no previous interactions with SAS and may not have readily suitable hardware. Contrastingly, MS-Access will compile to a run-time version that merely requires Windows 3.1 installation. In addition, ODBC links between MS-Access and Oracle were favorable over using the SAS/Access and SAS/Connect RLS linkages.

Acknowledgments

I would like to thank the many contributors to the SAS-L internet discussion for their collective wisdom and entertaining debate, in particular Christian Le Bras who suggested the idea of using CALL EXECCMDI functions to automate FSEDIT screen modifications.

References

SAS, SAS/Access, SAS/Connect, SAS/AF and SAS/FSP are registered trademarks of SAS Institute Inc. VAX and VMS are registered trademarks of the Digital Equipment Corporation. Oracle is a registered trademark of the Oracle corporation. Entry Point 90 is a registered trademark of Datexel. Microsoft Access and Windows are trademarks of the Microsoft Corporation.