Paper TS06

Reduce Programming time with Reusable Templates

Cathal Gallagher, d-Wise, Manchester, UK

ABSTRACT

Having validated programming templates for raw CRF data to SDTM domains and from SDTM domains to ADaM can save huge amounts of programming time. There are very few study specific variables, most are standard, and therefore can be pre-programmed and validated into a template. Simply add your raw CRF data to a template, add the code for study specific variables if there are any and run. Having these fully validated templates allows you to add your data and run the job in minutes. Having only the study specific variables left to validate results in minimal re work. Using CDI templates, snapshots can be run at any time with no programming required. This paper discusses the flow of data from raw data to an SDTM, to an ADaM standard and how templates facilitate that flow.

Introduction

Study programming takes a long time, and a lot of the programming is repetitive, both within a study and between studies. Here we will discuss some problems within study programming, how a template can help address these problems, what is involved in a making a template in base SAS and in DI Studio, and finally we state the advantages and drawbacks of using templates.

The Problem

A new study arrives and everyone starts programming from scratch each time. There are many repetitive tasks within each study and indeed across studies. All Studies are different and can take varying lengths of time to complete. The test study d-Wise used took roughly 3600 programmer hours to complete. This was programmed with no template, taking data from CRF tables to SDTM domains, and further from SDTM domains to ADaM domains. Within the single study programmers found they were having to type out the same code several times in order to work out variables that are common within a study. Such as VISITNUM and USUBJID. With a standardised CRF, and now SDTM and ADaM domains this repetitiveness wastes thousands of programmer hours in every study.

current programming method

A view from the outside would suggest that a programmer gets a specification, starts a new program window in SAS and types every single line of code from scratch.

Rarely do we actually program from scratch. Everyone usually has a text file on their desktop with repetitive code snippets, or a macro library set up to work out common variables for us such as USUBJID, RFSTDTC, xxSTDY or xxENDY. Despite this, reading the entirety of the specs is necessary, then understanding what the spec writer means. Then putting the code snippets or macro calls in order and doing the individual programming for other study specific variables. These things all take time.

WHAT COULD WE DO?

If we consider that a Pharmaceutical company has a standardised CRF, then we should be using validated templates. A program should be produced for each domain required e.g. AE,DM,DS,LB etc. This program will have logic included for core variables (the variables that are calculated the same way for every study). This program should then be validated with an independent replication program. Study programmers are then each given the same template, they point the libraries at the input folders and they program study specific variables for each domain. Study programmers would not have to waste time programming core variables as they have already been done. They will not even have to do macro calls or copy and paste code snippets. The program is mostly complete, they simply have to fill in the blanks. Validation of these variables will of course still occur, but if the template code has not been edited they will always match. Therefore programming time is reduced, validation time is reduced, and re work is reduced.

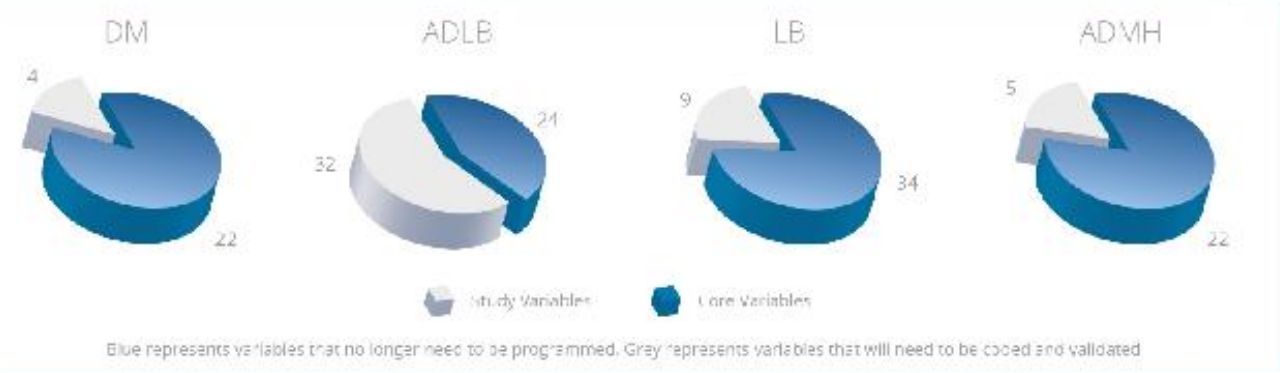
SDTM.DM has 26 variables of which only 4 are study specific. If study programmers were tasked with programming this domain, then in essence they only have to program 4 variables. They would take a copy of the DM template that has 22 variables fully coded. This code has already been validated to the specification for DM and therefore does not need to be checked. They would point the libraries at the location of their input data. Then they would find the location in the program where they can insert code to calculate the remaining 4 study variables, complete this code and run the program. With only 4 variables for them to program, they will complete this domain in much less time than it would take them to start from a blank programming window. Validation of this domain would take much less time as there is only 4 variables to check, and rework would be much easier as any mistakes should be located within the code for these 4 variables. This means the user does not have to search any of the template code for mistakes.

STUDYID	INVID
DOMAIN	INVNAM
USUBJID	BRTHDTC
SUBJID	AGE
RFSTDTC	AGEU
RFENDTC	SEX
RFXSTDTC	RACE
RFXENDTC	ETHNIC
RFICDTC	ARMCD
RFPENDTC	ARM
DTHDTC	ACTARMCD
DTHFL	ACTARM
SITEID	COUNTRY

STUDYID	INVID
DOMAIN	INVNAM
USUBJID	BRTHDTC
SUBJID	AGE
RFSTDTC	AGEU
RFENDTC	SEX
RFXSTDTC	RACE
RFXENDTC	ETHNIC
RFICDTC	ARMCD
RFPENDTC	ARM
DTHDTC	ACTARMCD
DTHFL	ACTARM
SITEID	COUNTRY

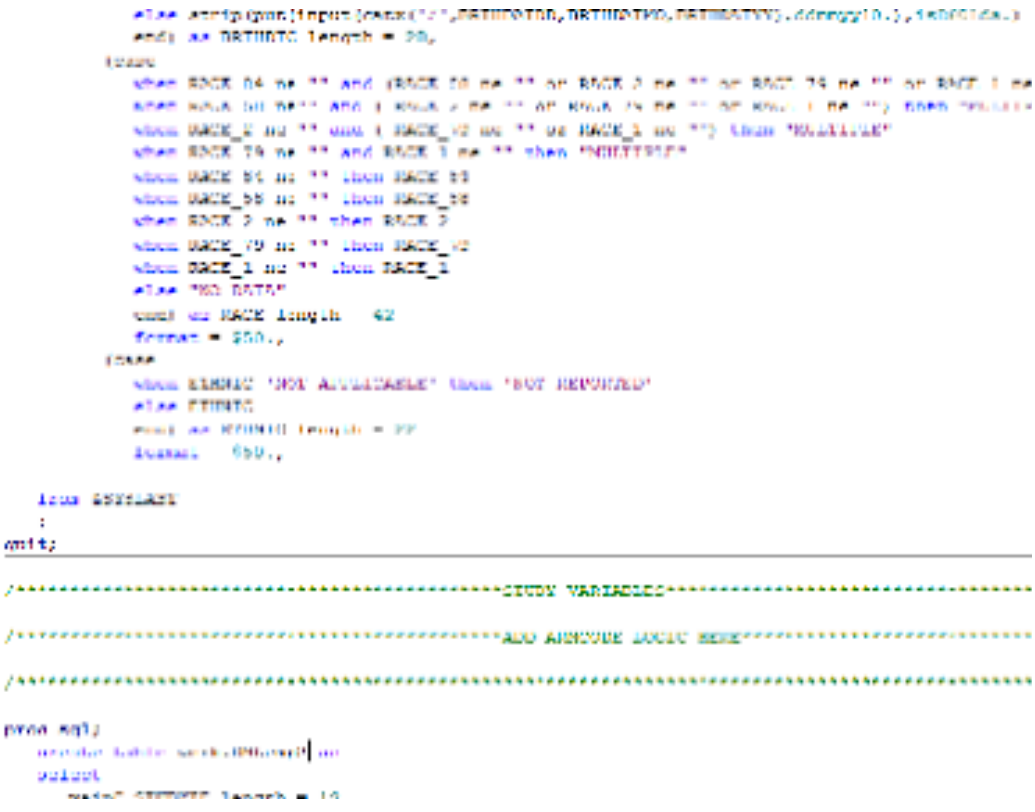
ADAM.ADLB has 56 variables. 24 of these are core variables that will always be calculated in the same way, which

leaves 32 study variables that need to be coded. This is a domain that requires much more work at the study level, but saving study programmers from having to code, validate and re work 24 variables will save them time. Even if there is only a couple of variables that a study programmer no longer has to code, validate and re work then surely this will save time. A starting structure in place can help motivate programmers, as it reduces the feel of starting from scratch.



HOW IS A TEMPLATE CREATED?

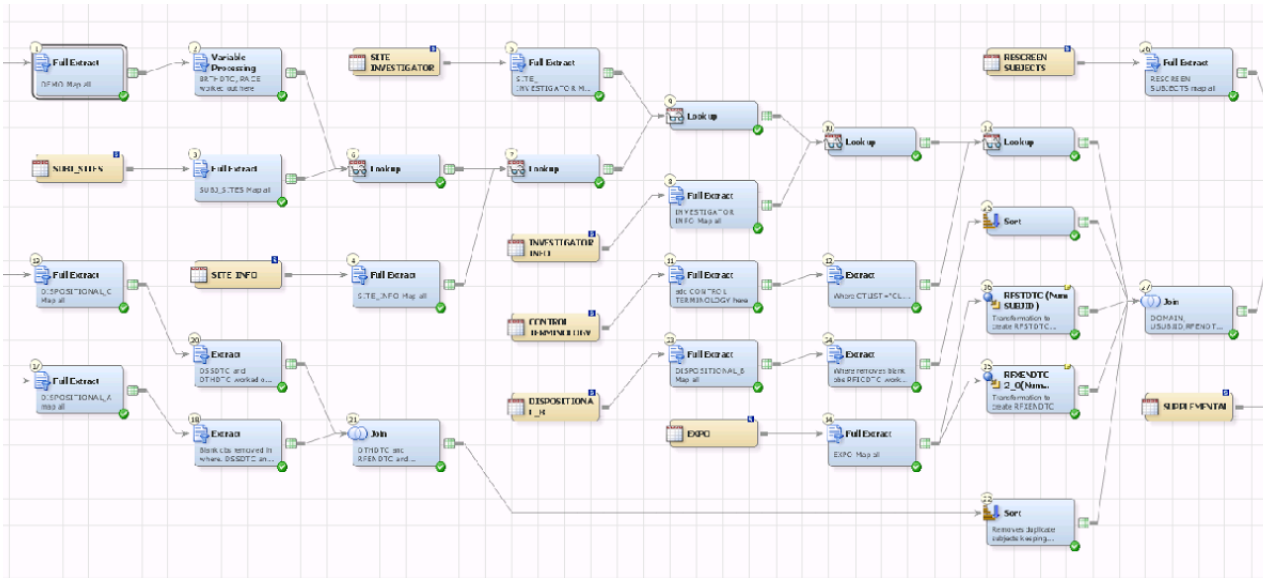
I use DI Studio, but you can use base SAS. Simply write out the code in base SAS for all the core variables in a domain, and have notes to indicate where the study programmer should put in code for the study specific variables. Another programmer will need to do exactly the same program so that the primary program can be validated. When a new study arrives, the study programmer will take a copy of the newly developed template. They will update the libname statements to point to the relevant study data and output datasets. They can then concentrate on locating the areas within the template for coding the study specific variables. In the example below most of the code is done for the study programmer with a placeholder left that indicates where they should insert logic to work out ARMCD

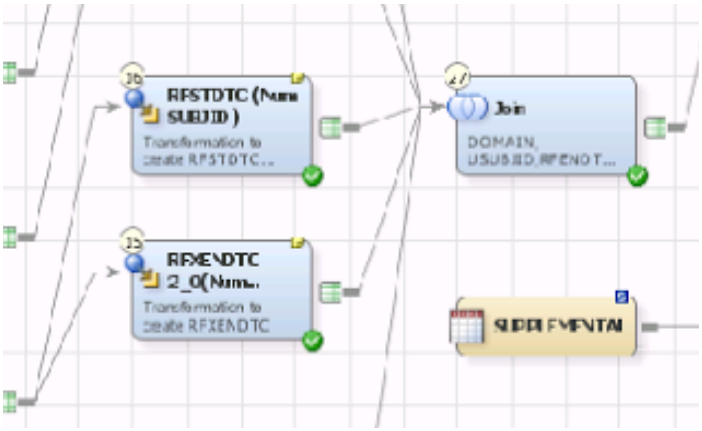


During the validation process the core variables should always match, the primary and the replication programmers have been using the same template therefore they are using the same code. Therefore any problems that are discovered shall be with the study programmer's code. This means there is much less code to examine in order to find and resolve errors. It not only reduces the time it takes to validate datasets but it reduces the difficulty in finding

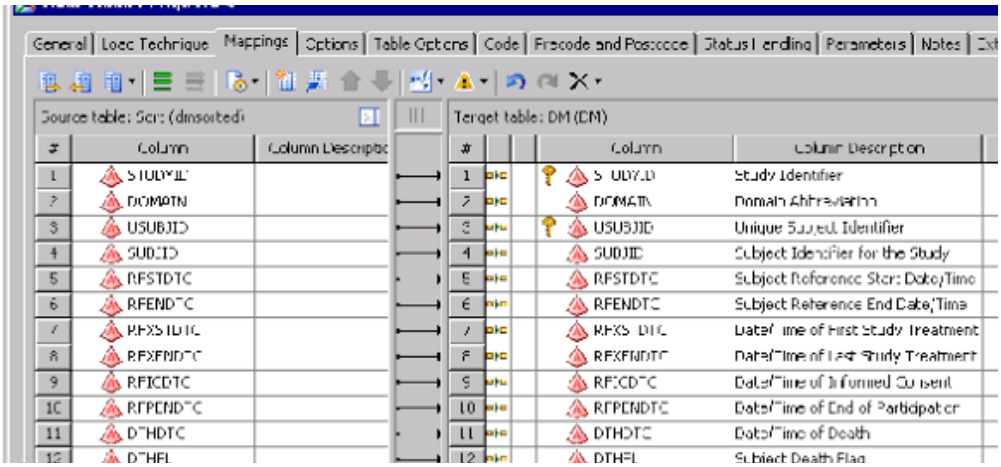
and resolving issues with the code.

In DI Studio I have created templates that offer a visual representation of the entire program (job in DI). I then have Sticky labels and comments that tell study programmers where they should put in their study variable code. These jobs will run in their current state and populate all the core variables to the output datasets. DI studio offers several advantages over base SAS. The visual representation makes it very easy to understand where the data is coming from and where it joins together. Data can be tracked visually through the program which can make it easier to find and resolve problems. Even without reading the comments it can be easy to understand the basics of how the program works. With base SAS, a programmer has to read a lot of code, which can sometimes be code from another programmer, to understand where the joins occur and how variables are getting populated. This can be time consuming and frustrating. DI studio offers a folder structure window illustrating the datasets available in each library, here you can drag and drop these datasets as well as transformations into jobs. Using the stock transformations means a lot of the code is already written, expressions simply have to be updated with code snippets, in order to achieve the desired results.
Comments are then placed on each transformation that indicate what variables are calculated where. This makes it very easy for a programmer to pick it up and begin working almost immediately.





DI Studio also has a clinical option that offers output Domains. This means a user does not have to worry about programming the formats and the labels of the output variables, they simply map to them and this step is done for them. Integrity constraints are automatically assigned according to the SDTM and ADaM models. These can be edited according to study requirements or turned off entirely. Although stock data models are supplied with CDI (Clinical DI Studio) it is possible to create bespoke versions for each study. Integration with SDD allows programmers to publish their jobs (programs) in an SDD folder as well as populating datasets directly in SDD folders from within CDI using webdavs.



WHAT ARE THE METRICS?

D-wise has calculated that an entire study from CRF to SDTM to ADAM takes 3600 programmer hours using base SAS with no templates or code snippets. Using a template system in CDI, D-wise has calculated that doing the same study can be reduced to 400 programmer hours. The reason you see very round numbers, is because every study is different and therefore the numbers can vary.

PROBLEMS

User acceptance is always the main risk with using any new system. Programmers always prefer to write code themselves rather than using other people's code. It takes time to get programmers used to using templates and ignoring and trusting the code that is already there. With experience programmers can achieve huge time savings when completing an entire study.

CRFs that change a lot can cause huge problems. Templates that call data from these CRFs will always need to be updated whenever a CRF changes. If a standard CRF model is not in place this can reduce the usefulness of templates. Unless a schedule is used to update the CRFs and the templates, then templates can become out of date very quickly.

THE FUTURE

Industry standard templates. Every pharmaceutical company using CDash SDTM and ADaM. With industry templates will speed up programming time, reduce re work and will get drugs out there quicker.

CONCLUSION

Template programming will reduce the amount of time required for studies. Even if a template only has code for one variable, then that is one less variable that a study programmer has to code and has to validate. Using templates means less programming time, less validation time, and less rework. Less variables that need to be coded reduce the programming times required. Less variables that need to be validated reduce the validation time required. Less code to check through in order to find and resolve issues that have arisen means less re work will be required. If a company is programming to SDTM and ADaM standards, the next step should be to standardise their CRF and then use programming templates. Using current programming methods can take 3600 programming hours per study. If the same study output can be achieved in only 400 hours then all possible should be done to overcome the user acceptance problems that are likely to arise.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Cathal Gallagher
d-Wise
Suite 2, 3rd Floor
61 Mosely Street
Manchester / M2 3HZ
Work Phone: +44161 236 0961
Fax:
Email:Cathal.gallagher@d-wise.com
Web: www.d-wise.com