ADSL - Friend or Foe?

Jørgen Mangor Iversen
Principal SAS Programmer
Statistical Programming, Biometrics
LEO Pharma A/S

June 7, 2019

Dermatology
beyond the skin

L E O

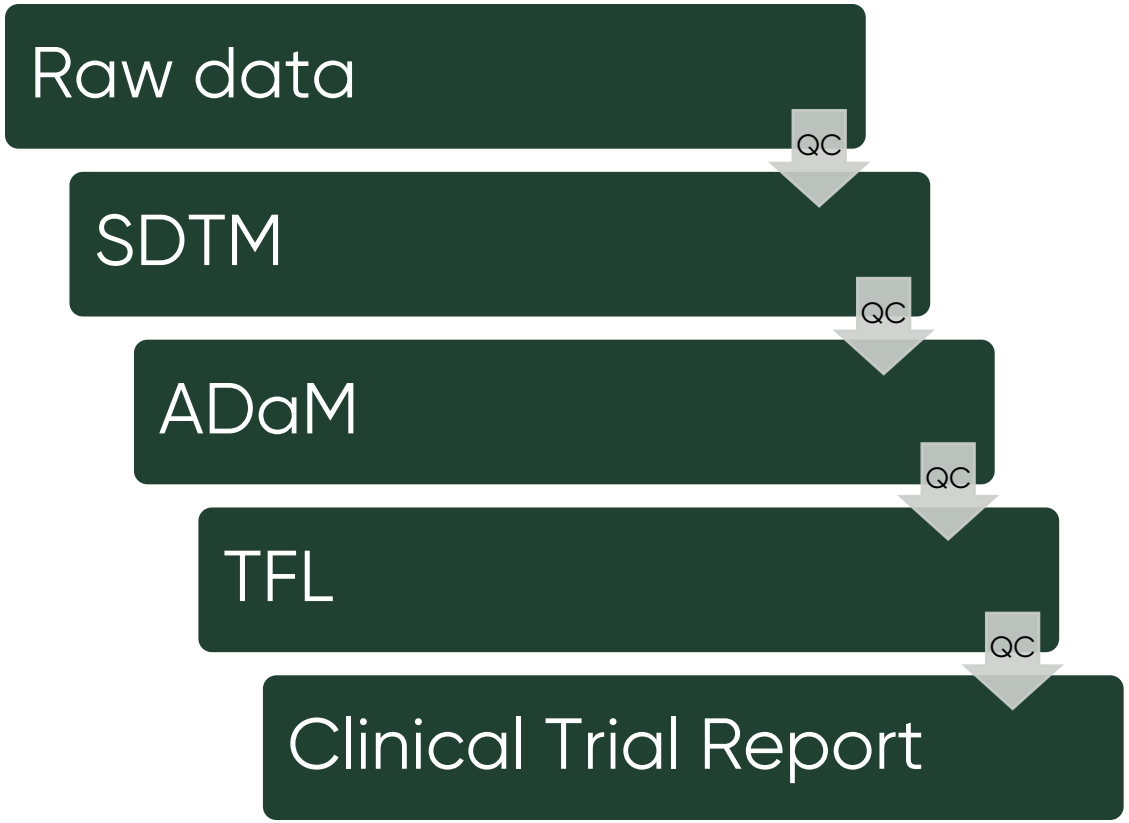
Content

- Introduction
- Problem
- Solution
- Perspective
- Conclusion

Introduction

Data Flow

- Creation of ADaM purely from SDTM
  - SDTM is signed off as submission ready
  - ADaM contains most complexity for reporting
  - TFLs directly from ADaM
    - Temporary mock ADaMs may occur



Introduction

Working example

ADSL	ADAE	ADLB
• USUBJID • DTHDT	• USUBJID • TRTEMFL	• USUBJID • CRIT1

Introduction

Variable Classes

- Creation of ADaMs (1.1)
  - Common variables
    - USUBJID Unique Subject Identifier — Needs the same length and label across ADaMs
  - ADSL
    - DTHDT Date of Death — May be updated during the trial
  - Occurrence data model (ADAE)
    - TRTEMFL Treatment Emergent Analysis Flag — May be updated by Medical Expert
  - Basic Data Structure (ADLB)
    - CRIT1 Analysis Criterion 1 — May be updated/created during reporting

Problem

Dependencies

ADSL

- All Common Variables
  - USUBJID
- ADSL specific Variables
  - DTHDT
- Last run: DateTime 1

ADAE

- Subset of Common Variables
  - USUBJID
- ADAE Specific Variables
  - TRTEMFL
- Last run: DateTime 2

ADLB

- Subset of Common Variables
  - USUBJID
- ADLB Specific Variables
  - CRIT1
- Last run: DateTime 3

Problem

Dependencies

ADSL

- All Common Variables
  - USUBJID
- ADSL specific Variables
  - DTHDT (New Value)
- Last run: DateTime 4

ADAE

- Subset of Common Variables
  - USUBJID
- ADAE Specific Variables
  - TRTEMFL
- Last run: DateTime 2

ADLB

- Subset of Common Variables
  - USUBJID
- ADLB Specific Variables
  - CRIT1
- Last run: DateTime 3

ADSL is now younger than it's dependants

Problem

Consequences

- Regardless of File Share or Repository or Database
- Misaligned dates is the first thing an inspector looks for
  - Together with version number sequences
- How do you guarantee that the changes to ADSL does not affect dependant datasets?
  - You have to re-run all dependant ADaMs to line up the DateTimes
  - You have to Re-QC all dependents!
  - Maybe not in total, but still…

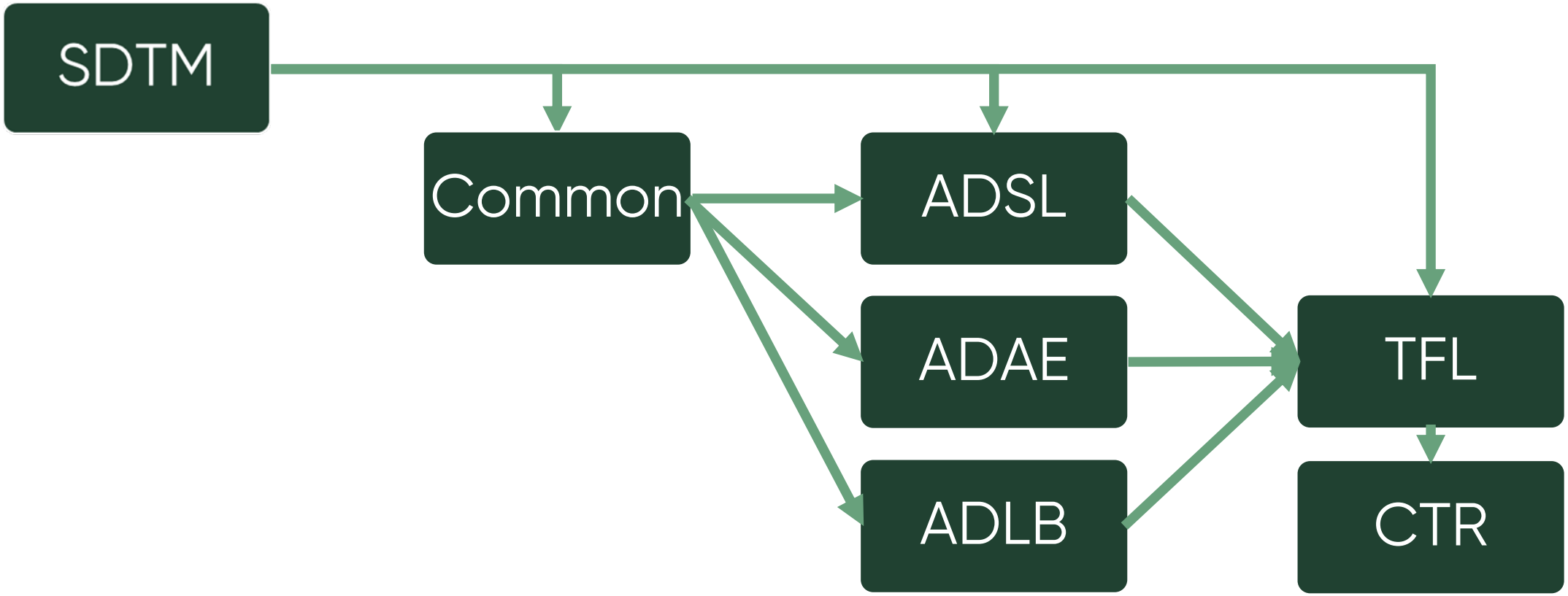
Solution

Central idea

- In stead of having ADaMs dependant of each other, isolate all common variables into a separate program or dataset, and let all ADaMs be dependant of a common source
  - ADSL specific variables can be updated independently
  - Other ADaM specific variables can be updated independently
  - When a common variable need updating, you have to re-run (and re-QC) everything anyway

Solution

Structure



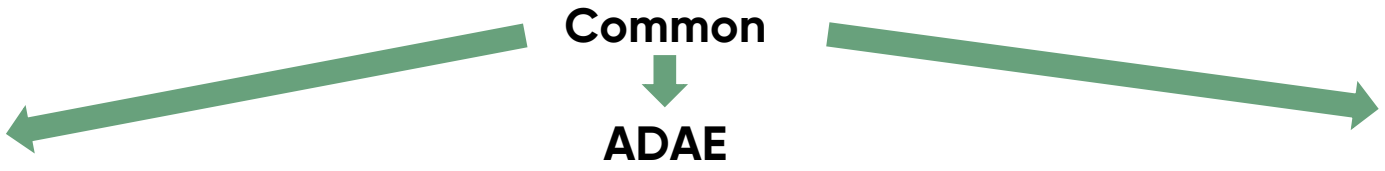
Solution

Details

- Common is a WORK dataset at LEO
  - No reason why it cannot be a permanent dataset
  - Choice depends on performance of rerunning common
  - Having it in WORK eliminates all dependencies

Solution

Without Dependencies



ADSL

- All Common Variables
  - USUBJID
- ADSL specific Variables
  - DTHDT (New Value)
- Last run: DateTime 4

ADAE

- Subset of Common Variables
  - USUBJID
- ADAE Specific Variables
  - TRTEMFL
- Last run: DateTime 2

ADLB

- Subset of Common Variables
  - USUBJID
- ADLB Specific Variables
  - CRIT1
- Last run: DateTime 3

ADSL is now younger, but independent of the other ADaMs

Solution

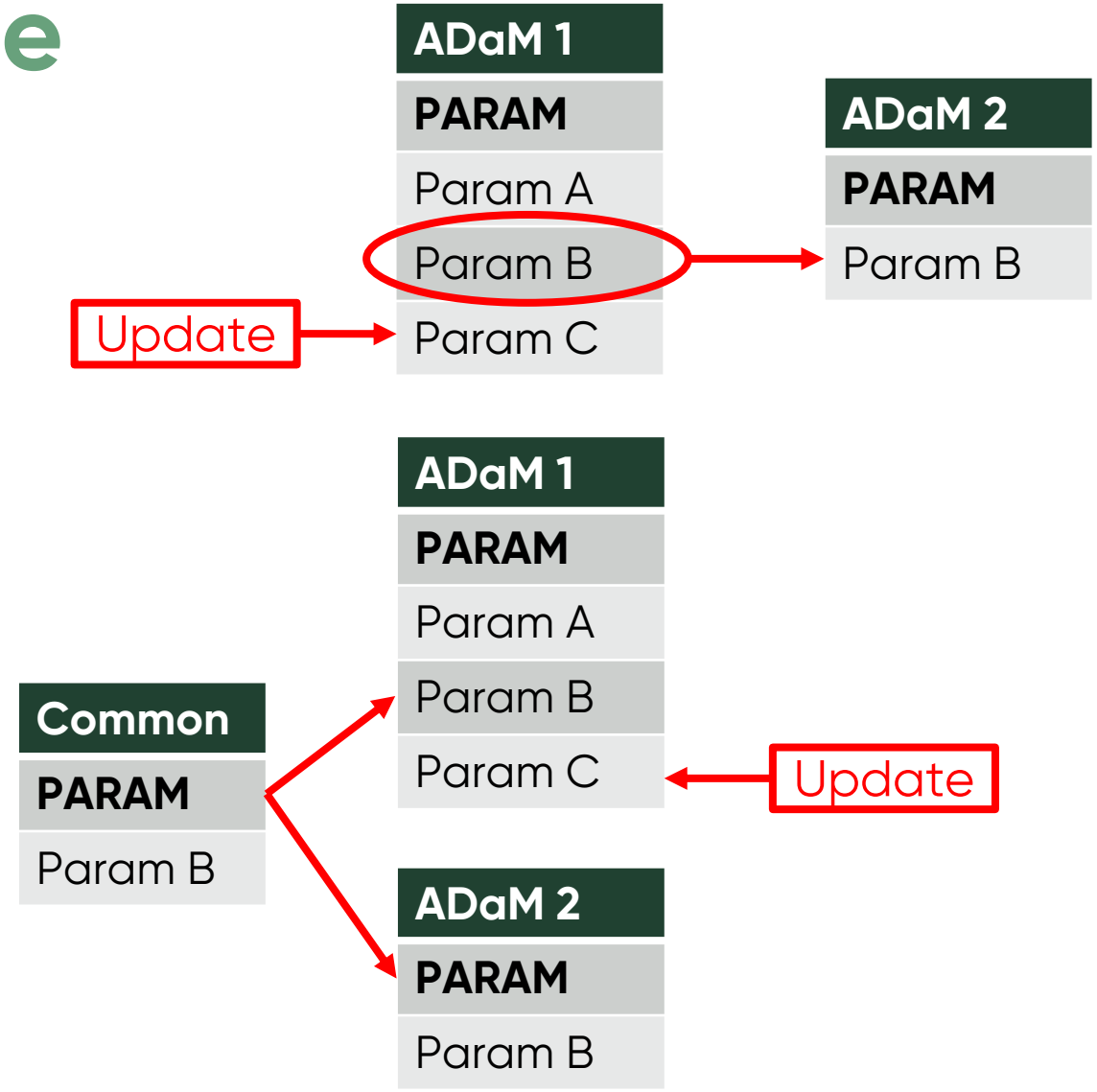
Details

- All ADaMs can be changed/updated independent of each other
- Any one update requires only rerun and re-QC of that single ADaM
- DateTimes of runs and versions numbers will always line up
- At LEO we have saved weeks of re-QC for trials utilizing this technique, compared to similar trials not doing so
- Benefit at it's maximum when finding late stage issues causing loop-backs

More complex example

Not only ADSL can have dependencies

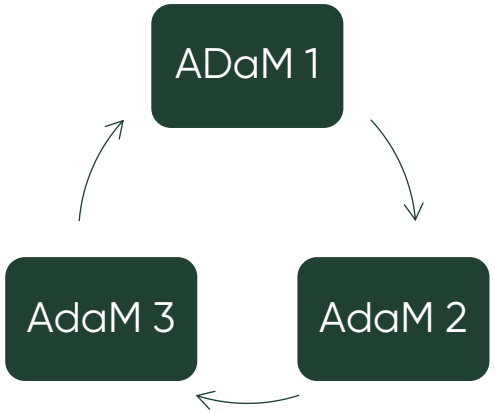
- ADaM based on PARAM subset
  - PARAM collects many discrete subsets
  - One or more subset needs further processing
  - What if other PARAM subsets needs updating?
- Create a common dataset for relevant subsets of PARAM
  - Use dependences to decide which subsets
  - Create resulting ADaMs independent of the common dataset

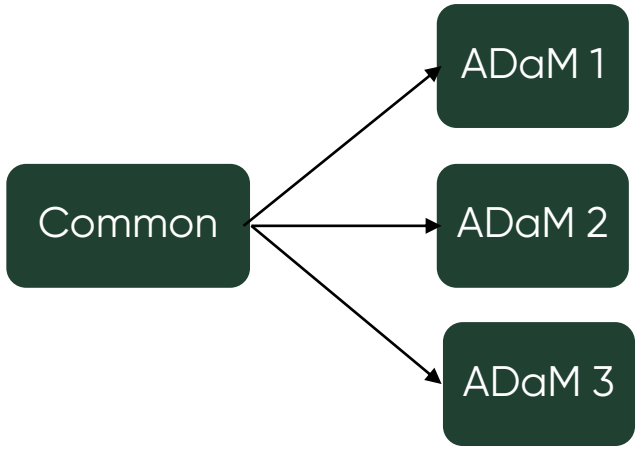


Hypothetical example

Circular refences

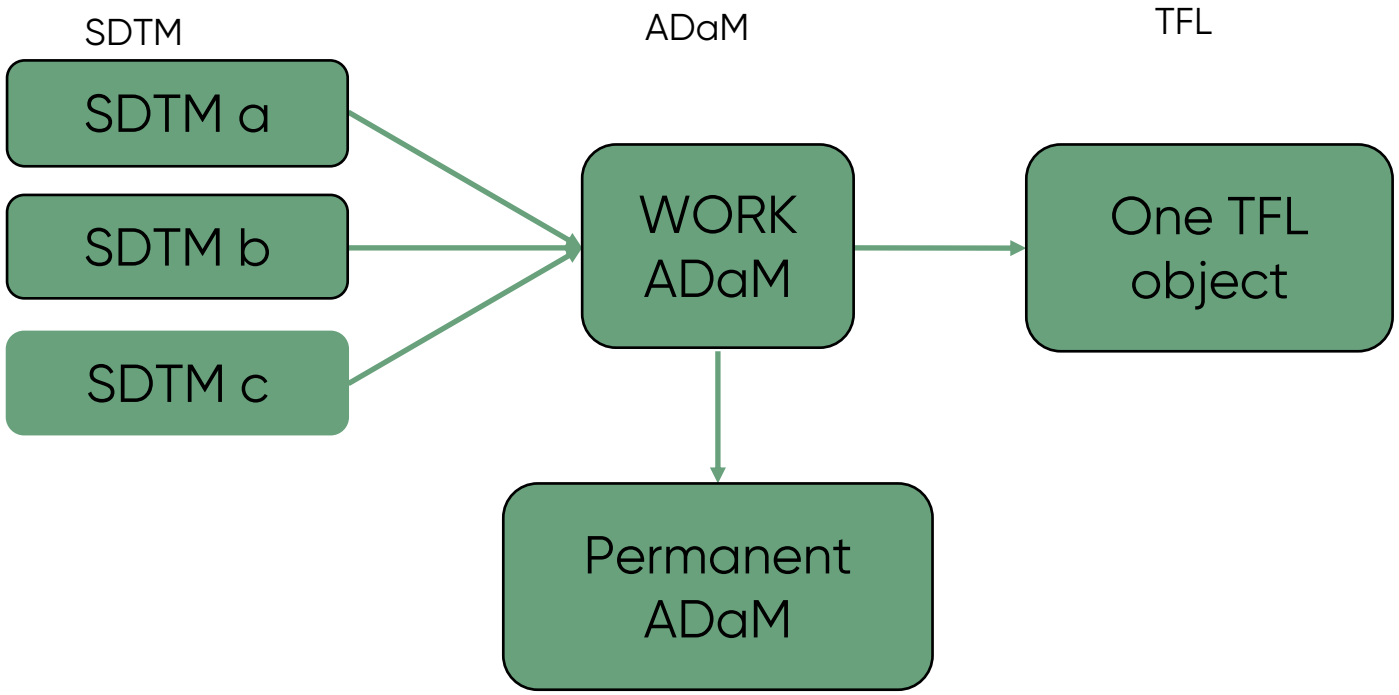
- AdaM1->ADaM2->ADaM3->ADaM1
- It will always be random which one is the youngest
- How many times must you run the circle to propagate all changes to all ADaMs?
- Validated systems exists that will not capture this
- Create all ADaMs from a common origin independently

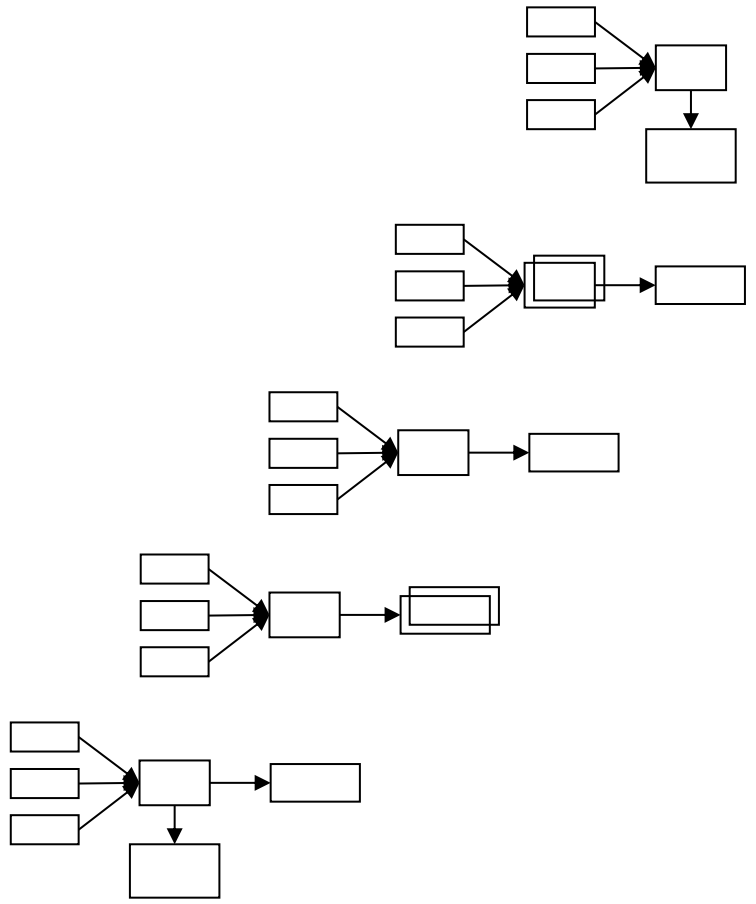




Proposed ADaM architecture

General example





Many such blocks in parallel. Extension of common, implemented as macros or mappings

Proposed TFL architecture

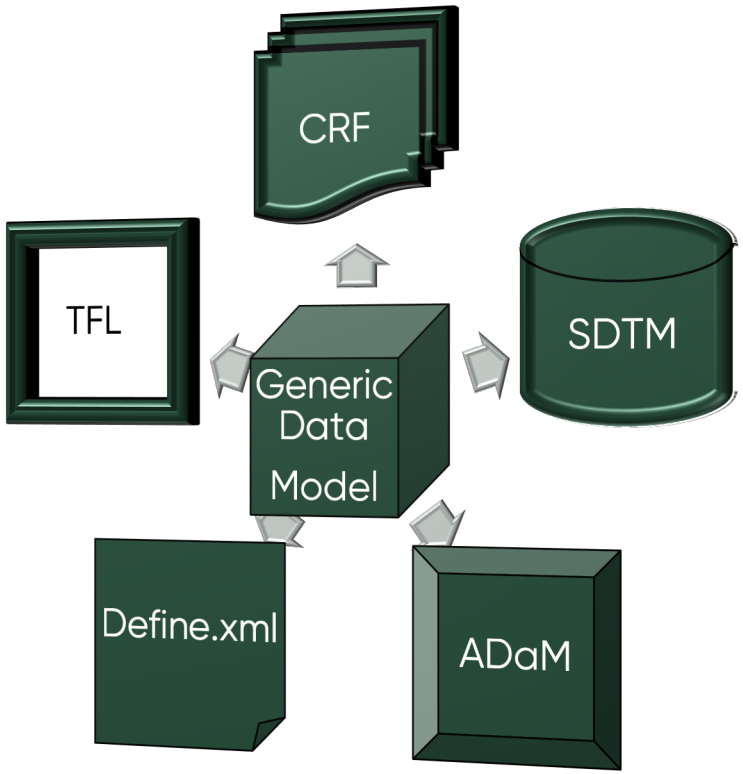
Example table program code

```
%WorkADSL;
%WorkADAE;

%TableShell(shell="...");
%TableGet(tableno=...);
%TableTitle(byvars="...");
%TableFootnote(addendum="...");
%TableHeaders(continued=...);
%TableRows(box="...");
%TableStats(pval=..., cl=...);
%TableODS(destination="...");
%TableOutput(colwidths=...);
%TableCleanUp;
```

Next Generation

Model/Viewer



- One central store of a generic data model
- Everything else is views or perspectives of the same data
- How to QC the data?
- How to QC the views?

SRC--- variables

Ensuring traceability

SRCDOM Contains either the originating SDTM domain, or the Common dataset

SRCVAR Contains either the originating SDTM variable, or the variable in the Common dataset

SRCSEQ Contains either the SDTM sequence number, or an introduced ASEQ variable

This way, the Common dataset is transparent, ready for submission, yet traceability is complete

Conclusion

- By applying a little structure to ADaM programs up front, dependencies can be avoided
  - Put common variables in a common program or dataset
  - Pick from the common source when producing ADaMs
  - Update dataset specific variables independently
  - Save re-QC'ing resources
- Make sure all data points can be traced to SDTM and it's sources

Questions

Contact details

Dermatology
beyond the skin

Jørgen Mangor Iversen
Principal SAS Programmer
Statistical Programming, Biometrics
+45 20 60 61 76
jmidk@leo-pharma.com