АНОТАЦІЯ

Робота описує автоматизацію SDTM та ADaM у клінічних випробуваннях, яка наразі є популярною темою у фармацевтичній галузі. Багато компаній мають добре організовану команду зі стандартів, яка визначає стандарти, що відповідають вимогам CDISC та регуляторних органів у межах своєї компанії. Природно, стандартизація клінічних даних веде до ініціатив з автоматизації. У цій статті ми поділимося інноваційними зусиллями щодо стандартизації специфікацій для створення даних з метою полегшення автоматизації. Ми введемо псевдокоди як простий, але стандартний структурований спосіб написання специфікацій, щоб програміст (будь то людина чи машина) міг розуміти та інтерпретувати специфікації послідовно. Ця стаття надасть приклади псевдокоду, щоб продемонструвати, як він створює чіткі, конкретні, послідовні, недвозначні правила виведення для забезпечення автоматизації. Також будуть обговорені весь процес і рівень автоматизації. Визначення та розробка стандартних специфікацій є важливими для автоматизації, щоб покращити якість згенерованих даних, скоротити час програмування та спростити рутинні повторювані завдання. Робота містить 21 сторінку та 14 рисунків.

Ключові слова: стандарти, автоматизація, псевдокод, специфікація, виведення, трансформація, ADaM.

ABSTRACT

The abstract describes automation of SDTM and ADaM in clinical trials is a popular topic in the pharmaceutical industry nowadays. Many companies have a well-established standards team, which is defining CDISC and regulatory authority compliant standards within their company. Naturally, standardization of clinical data leads to automation initiatives. In this paper, we will share innovative effort in standardizing the derivation or transformation specifications for data creation, with the intent to facilitate machine automation. We will introduce pseudo-codes, as a simple yet standard structured way of writing specifications, so that the programmer (either person or a machine) can understand and interpret the specifications consistently. This paper will provide examples of pseudo-code to demonstrate how it creates clear, specific, consistent, unambiguous derivation rules to enable automation. The endto-end flow and the level of automation will also be discussed. Defining and developing the standard specification is essential for automation to improve the quality of generated data, reduce programming time, and simplify mundane redundant tasks. The note contains 21 pages and 14 pictures.

Keywords: standards, automation, pseudo-code, specification, derivation, transformation, ADaM.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Харківський національний університет імені В.Н.Каразіна

Факультет математики і інформатики

Кафедра теоретичної та прикладної інформатики

Кваліфікаційна робота

магістр

на тему «Оптимізація обробки даних у вигляді структур SDTM і ADaM»

Виконала: студенка 2 курсу, групи МФ-61

спеціальність 122 «Комп'ютерні науки»

освітньо-наукова програма «Інформатика»

Борзенкова А.О.

_____

(прізвище та ініціали)

Керівник Мовчун Т.О.

_____

(прізвище та ініціали)

Рецензент_____

(прізвище та ініціали)

Харків – 2024 року

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

SAS — Statistical Analysis System

SDTM — Study Data Tabulation Model

SQL — Structured Query Language

ADaM – Analysis Data Model

SAP – Statistical Analysis Plan

CDISC – Clinical Data Interchange Standards Consortium

ВСТУП

Автоматизація SDTM та ADaM у клінічних випробуваннях наразі є популярною темою у фармацевтичній галузі. Багато компаній мають добре організовану команду зі стандартів, яка визначає стандарти, що відповідають вимогам CDISC та регуляторних органів у межах своєї компанії. Природно, стандартизація клінічних даних веде до ініціатив з автоматизації. У цій статті ми поділимося інноваційними зусиллями щодо стандартизації специфікацій для створення даних з метою полегшення автоматизації. Ми введемо псевдокоди як простий, але стандартний структурований спосіб написання специфікацій, щоб програміст (будь то людина чи машина) міг розуміти та інтерпретувати специфікації послідовно. Робота надасть приклади псевдокоду, щоб продемонструвати, як він створює чіткі, конкретні, послідовні, недвозначні правила виведення для забезпечення автоматизації. Будуть обговорені весь процес і рівень автоматизації. Визначення та розробка стандартних специфікацій є важливими для автоматизації, щоб покращити якість згенерованих даних, скоротити час програмування та спростити рутинні повторювані завдання.

Також ми розглянемо стандартні правила виведення у формі псевдокоду. Це ілюструє автоматизацію створення програм на основі псевдокодів. Ми окреслимо основні концепції, наведемо приклади і розглянемо процес створення програм за допомогою генератора коду. Хоча ми зосередимося головним чином на прикладах ADaM, подібні принципи виведення застосовуються і до SDTM.

1.1 Мета і завдання дослідження

Специфікації важливі для будь-якого програмування. Клінічне програмування даних не є винятком. Під час проведення дослідження часто підкреслюється необхідність якісних програмних специфікацій.

Що робить їх такими важливими? На початку програмування бажано мати чіткі, конкретні, послідовні, однозначні правила виведення. Це забезпечує якість програми, а також ефективність програмування. Крім того, це сприяє співпраці між програмістами. По суті, правила виведення - це мова, якою програмісти спілкуються між собою, зі статистиками та зі своїми даними. Важливо, щоб всі говорили однією мовою. Інформація повинна послідовно переходити від протоколу та SAP до правил виведення і транслюватися в програми, незалежно від того, хто виконує програмування (людина чи машина). Таким чином, стандартизація специфікацій виведення є ключем до досягнення послідовності та ефективності потоку інформації.

1.2 Методи дослідження та практичне значення одержаних результатів

У цій роботі буде представлен конкретний спосіб написання специфікацій. Для проекту автоматизації ми класифікували різні типи виведень, які використовуються в нашій компанії для статистичного аналізу, що розподілилися на 9 груп. Ми розглянемо їх більш детально в розділі "Дев'ять стандартних типів виведень". Для деяких груп ми надали стандартний псевдокод і розробили схеми для збору ключових елементів кожного псевдокоду для специфікацій виведення.

Псевдокод - це простий, але стандартизований структурований спосіб написання специфікацій, щоб програміст (людина чи машина) міг зрозуміти та інтерпретувати їх послідовно. Зручно, що для кожної змінної ADaM у метаданих ADaM створюється псевдокод із стандартними правилами виведення. Дивіться приклад на Рисунку 1 нижче, де у метаданих змінних ADaM колонка Псевдокод виділена зеленим. Псевдокод також застосовується до виведень на рівні значень. Дивіться приклад на Рисунку 2

Рисунок 1. Метадані змінних

Dataset Name	Relative Order	Variable Name	Variable Label	Type	Controlled Terminology	Source	Derivation	Assigned	Pseudo Code	Core	Instructions for Programmers
ADSL	170	AAGEU	Analysis Age Units	text	AGEU			Assigned as "[YEARS, MONTHS,	assigned	Cond	Conditionally required if AAGE is in
ADSL	180	AGEGR1	Pooled Age Group 1	text	[AGEGR1]		For AGE of [xxx] to [xxx], AGEGR1 = [xxx]		recode	Cond	Define AGE ranges in accordance if AGE=. then AGEGR1=""
ADSL	190	AGEGR2	Pooled Age Group 2	text	AGEGRPE		For AGE of [xxx] to [xxx], AGEGR2 = [xxx]		condderivation	Cond	AGEGR2 is reserved for EudraCT Preterm newborn - gestational age
ADSL	210	SEX	Sex	text	SEX	DM.SEX			source	Req	
ADSL	220	RACE	Race	text	RACE	DM.RACE			source	Req	If the SDTM RACE is "MULTIPLE",
ADSL	230	RACEOR	Original Race	text	RACEC	SUPPDM.RACEOR			source	Perm	
ADSL	240	RACEGRy	Pooled Race Group y	text	[RACEGRy]		For RACE of [xxx] or [xxx], RACEGRy = [xxx]		recode	Perm	Character description of a grouping used for special case when study h

Рисунок 2. Умови where, головна мета яких – надання метаданих для vlm

Dataset Name	Variable Name	Where Variable	Comparator	Check Value	Value Label	Derivation	Pseudo Code
ADVS	AVAL	PARAMCD	EQ	WEIGHT	Weight (kg)	VS.VSSTRESN Where PARAMCD=VS.VSTESTCD	copy_stres
ADVS	AVAL	PARAMCD	EQ	DIABP	Diastolic Blood Pressure (mmHg)	VS.VSSTRESN Where PARAMCD=VS.VSTESTCD. An averag record is creted for triplicate measurements on the same date.	copy_stres, std_function(dtype_average)
ADVS	AVAL	PARAMCD	EQ	SYSBP	Systolic Blood Pressure (mmHg)	VS.VSSTRESN Where PARAMCD=VS.VSTESTCD. An averag record is creted for triplicate measurements on the same date.	copy_stres, std_function(dtype_average)

ОСНОВНА ЧАСТИНА

2.1. Дев'ять стандартних типів виведень

У цьому розділі ми розглянемо дев'ять стандартних типів виведень та відповідні стандартні псевдокодами. Ми обрали 1-3 набори даних з кожної з трьох стандартних структур даних CDISC ADaM (ADSL, BDS та OCCDS), дослідили алгоритми виведення для кожної змінної та визначили загалом 9 типових виведень. Для кожного типу виведення ми визначили псевдокод.

Ця таблиця описує дев'ять стандартних псевдокодів у порядку від простих до складних правил виведення.

Псевдокод	**Опис**
source	Вказує, що змінна має походження "Попередник", де вихідна змінна визначена у стовпці "Джерело". Використовується головним чином для змінних-попередників SDTM.
assigned	Призначення постійного значення для всіх записів у наборі даних.
derivation	Вказує, що детальні специфікації виведення доступні у стовпці "Виведення" на вкладці "Метадані змінної".
recode	Вказує, що детальний алгоритм виведення доступний на вкладці "recode".
rename	Перейменування змінної. Використовується, коли тимчасова змінна перейменовується у змінну набору даних ADaM. Найпоширеніший приклад - змінні –DT, –DTM та –TM. Змінні ISO8601 datetime у вихідній змінній SDTM, які перетворюються на тимчасові SAS змінні

Псевдокод	**Опис**
	дати, дати та часу, та часу (наприклад, EXSTDT, EXSTDTM та EXSTTM) на початку програми ADaM, та перейменовуються у відповідні імена змінних ADaM (наприклад, ADEX.ASTDT, ADSL.TRTSTDTM тощо).
condderivation	Вказує, що детальний алгоритм виведення доступний на вкладці "condderivation".
lookup*	Просте злиття proc sql двох наборів даних, із використанням підмножини умов, якщо потрібно.
std_function*	Складні виведення, які передбачають кілька наборів даних та/або кілька кроків обчислень, які не можна класифікувати до вищезазначених типів виведень.
whereClauses*	Вказує, що псевдокод для параметрів на рівні значень заповнюється у таблиці VLM на вкладці "WhereClauses".

* Вказує на псевдокоди, що не представлені у цій статті.

Наразі розробляється користувацький інтерфейс, який допоможе вводити значення, специфічні для дослідження, та інтерактивно відображати згенерований код. Однак у цій статті ми будемо використовувати Excel для ілюстрації написання псевдокоду. Коли правила виведення прості, псевдокоди можуть бути створені разом із метаданими змінної.

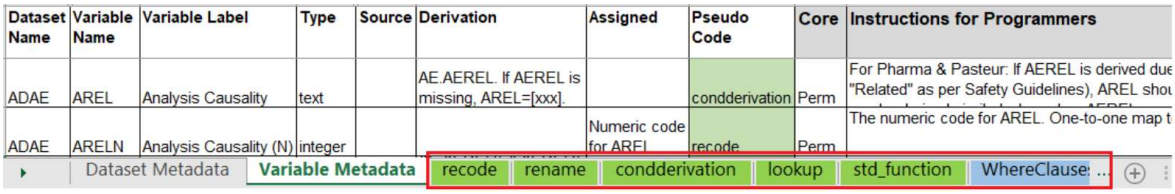
Наприклад, Рисунок 3 показує 3 прості псевдокоди "source", "derivation" та "assigned". Ці псевдокоди використовують 3 відповідні елементи метаданих змінної безпосередньо. Таким чином, стовпець, що має таку ж назву, як і псевдокод, використовується для передачі ключового елемента генератору коду.

Рисунок 3. Стовпці для додаткової інформації

Dataset Name	Variable Name	Variable Label	Type	Source	Derivation	Assigned	Pseudo Code
ADSL	STUDYID	Study Identifier	text	DM.STUDYID			source
ADSL	TRTDURD	Total Treatment Duration (Days)	integer		TRTEDT - TRTSDT + 1		derivation
ADEG	AWU	Analysis Window Unit	text			Assigned as "DAYS"	assigned

З іншого боку, коли псевдокод є "recode", "rename", "condderivation", "lookup", "std_function" або "whereclauses", додаткову інформацію потрібно ввести на іншій вкладці, яка має таку ж назву, як і псевдокод.

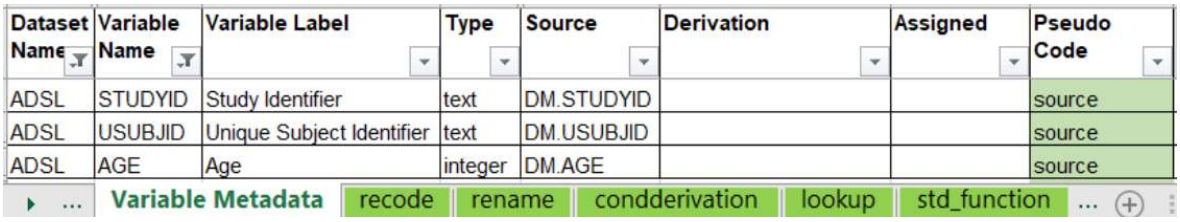
Рисунок 4. Вкладки для додаткової інформації



2.2. Детальний розгляд дев'яти типів виведення

Копіювання змінної з основного джерела даних (псевдокод = "source"). Див. Рисунок 5. Це використовується переважно для попередніх змінних з SDTM. Інформація перетворюється в програмний код для збереження змінної з основного джерела даних і використовується як джерело для define.xml для відображення імені попередньої змінної. Основне джерело даних - це набір даних, який включає більшість вихідних змінних для цього ADaM набору даних (наприклад, SDTM DM для ADSL). Приклади: ADSL.STUDYID, ADSL.USUBJID, ADSL.AGE тощо.

Рисунок 5. Псевдокод "source" у вкладці метаданих змінної



Dataset Name	Variable Name	Variable Label	Type	Source	Derivation	Assigned	Pseudo Code
ADSL	STUDYID	Study Identifier	text	DM.STUDYID			source
ADSL	USUBJID	Unique Subject Identifier	text	DM.USUBJID			source
ADSL	AGE	Age	integer	DM.AGE			source

Variable Metadata | recode | rename | condderivation | lookup | std_function

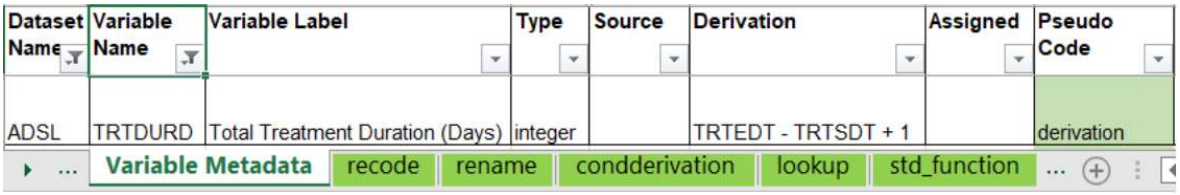
Згенерований SAS код виглядає так:

Keep STUDYID USUBJID AGE;

Просте обчислення, яке з'являється в define.xml як обчислювальний алгоритм, що безпосередньо використовується як програмний код (псевдокод = "derivation"). Див. Рисунок 6. Наприклад, чотири основні арифметичні операції: додавання, віднімання, множення та ділення. Значення вихідної змінної (-них) повинні бути доступні в тому ж записі в основному наборі даних.

Приклад: Загальна тривалість лікування (дні) (ADSL.TRTDURD).

Рисунок 6. Псевдокод "derivation" у вкладці метаданих змінної



Dataset Name	Variable Name	Variable Label	Type	Source	Derivation	Assigned	Pseudo Code
ADSL	TRTDURD	Total Treatment Duration (Days)	integer		TRTEDT - TRTSDT + 1		derivation

Variable Metadata | recode | rename | condderivation | lookup | std_function

Згенерований SAS код виглядає так:

TRTDURD = TRTEDT – TRTSDT + 1;

Призначення постійного значення для всіх записів у наборі даних (псевдокод = "assigned"). Див. Рисунок 7.

Це використовується, коли змінна має однакове значення для всіх записів у наборі даних. Постійне значення потрібно ввести у метадані змінної. Ця інформація перетворюється в програмний код для призначення постійного значення змінній і використовується як джерело для define.xml, щоб відобразити коментар до змінної як "Assigned as “xxx”"".

Приклад: Одиниця виміру вікна аналізу (ADEG.AWU).

Рисунок 7. Псевдокод "assigned" у вкладці метаданих змінної

Згенерований SAS код виглядає так:

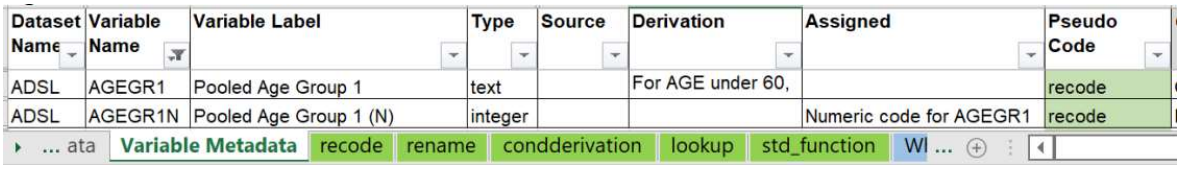
AWU = “DAYS”;

Перекодування кожного значення вихідної змінної в інше значення (псевдокод = "recode"). Див. Рисунок 8 для конвенцій метаданих змінної та Рисунок 9 для вкладки "recode".

Це зручний спосіб створення групувальних змінних або числових/символьних представлень для існуючих змінних. Користувачі можуть вводити кілька змінних у метаданих, як показано нижче. Ця конвенція може бути використана для виведення кількох змінних одночасно. У цьому прикладі ми виводимо 2 пов'язані змінні.

Приклад: Зведена вікова група 1 (ADSL.AGEGR1) та Зведена вікова група 1 (N) (ADSL.AGEGR1N).

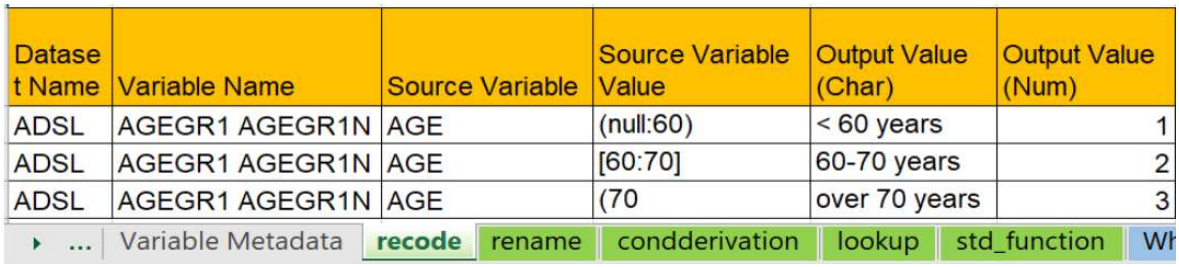
Рисунок 8. Псевдокод "recode" у вкладці метаданих змінної



Dataset Name	Variable Name	Variable Label	Type	Source	Derivation	Assigned	Pseudo Code
ADSL	AGEGR1	Pooled Age Group 1	text		For AGE under 60,		recode
ADSL	AGEGR1N	Pooled Age Group 1 (N)	integer			Numeric code for AGEGR1	recode

... ata | Variable Metadata | recode | rename | condderivation | lookup | std_function | Wl ...

Рисунок 9. Відповідна вкладка "recode"



Dataset Name	Variable Name	Source Variable	Source Variable Value	Output Value (Char)	Output Value (Num)
ADSL	AGEGR1 AGEGR1N	AGE	(null:60)	< 60 years	1
ADSL	AGEGR1 AGEGR1N	AGE	[60:70]	60-70 years	2
ADSL	AGEGR1 AGEGR1N	AGE	(70	over 70 years	3

... | Variable Metadata | recode | rename | condderivation | lookup | std_function | Wh

Згенерований SAS код виглядає так:

```
if . < AGE < 60 then do;

AGEGR1 = "< 60 years ";

AGEGR1N = 1;

end;

else if 60 <= AGE <= 70 then do;

AGEGR1 = "60-70 years";

AGEGR1N = 2;

end;
```

else if 70 < AGE then do;

AGEGR1 = "over 70 years";

AGEGR1N = 3; end;

Перейменування змінної (псевдокод = "rename"). Див. Рисунок 10 для конвенцій метаданих змінної та Рисунок 11 для вкладки "rename".

Це використовується, коли тимчасова змінна перейменовується у наборі даних ADaM. Найбільш поширений приклад - змінні --DT, --DTM та --TM. Змінні дати та часу ISO8601 (наприклад, EXSTDTC) у вихідній змінній SDTM, які конвертуються у тимчасові числові змінні дати, часу та часу SAS (наприклад, EXSTDT, EXSTDTM та EXSTTM) на ранніх етапах програми ADaM. Ці тимчасові змінні перейменовуються у відповідні назви змінних ADaM (наприклад, ADEX.ASTDT, ADSL.TRTSTDTM тощо). Створення тимчасових змінних оптимізує створення програми, коли вихідні дані відбираються у програмах ADaM, це обробляється загалом, щоб полегшити введення псевдокоду користувачем і допомогти з виконанням програми SAS і виведенням необхідних змінних без помилок.

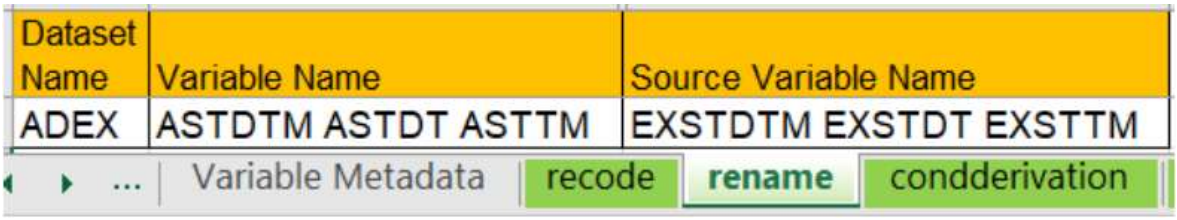
Приклад: Дата початку аналізу (ADEX.ASTDT), Дата і час початку аналізу (ADEX.ASTDTM), Час початку аналізу (ADEX.ASTTM).

Рисунок 10. Псевдокод "rename" у вкладці метаданих змінної

Dataset Name	Variable Name	Variable Label	Type	Source	Derivation	Assigned	Pseudo Code
ADEX	ASTDTM	Analysis Start Datetime	integer		or EX.EXSTDTC		dataset_derivation(DTC), rename
ADEX	ASTDT	Analysis Start Date	integer		date portion of		dataset_derivation(DTC), rename
ADEX	ASTTM	Analysis Start Time	integer		time portion of		dataset_derivation(DTC), rename

... Variable Metadata | recode | rename | condderivation | lookup | std_function | Where(...

Рисунок 11. Відповідна вкладка "rename"



Dataset Name	Variable Name	Source Variable Name
ADEX	ASTDTM ASTDT ASTTM	EXSTDTM EXSTDT EXSTTM

... | Variable Metadata | recode | rename | condderivation

Згенерований SAS код виглядає так:

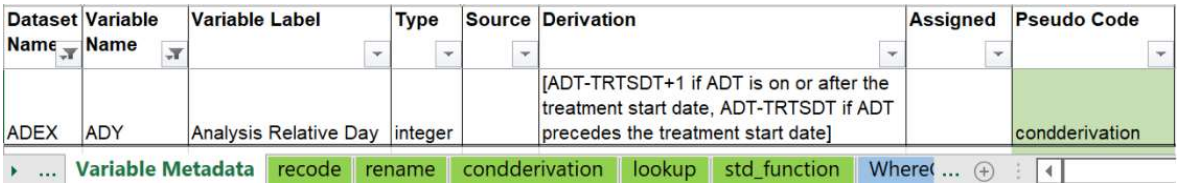
```
rename EXSTDT = ASTDT
       EXSTDTM = ASTDTM
       EXSTTM = ASTTM;
```

Виведення змінної на основі умов if-then залежних змінних (псевдокод = "condderivation"). Див. Рисунок 12 для конвенцій метаданих змінної та Рисунок 13 для вкладки "condderivation".

Значення можуть бути введені як заздалегідь визначені значення або вихідні змінні, або за допомогою формули. Залежні змінні та вихідні змінні повинні бути доступні в тому ж записі в основному наборі даних.

Приклад: Відносний день аналізу (ADY). Виведення відрізняються в залежності від того, чи ADT передує TRTSDT або збігається/перевищує TRTSDT.

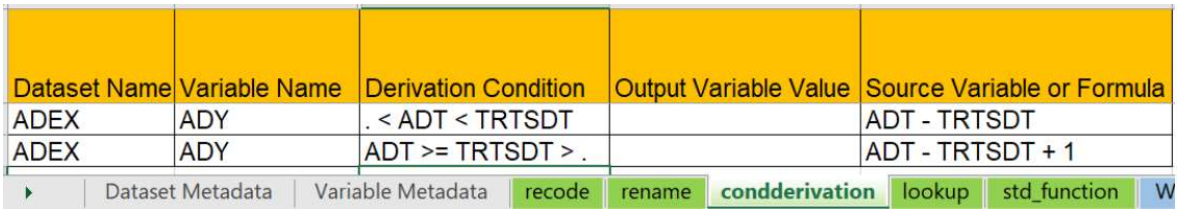
Рисунок 12. Псевдокод "condderivation" у вкладці метаданих змінної



Dataset Name	Variable Name	Variable Label	Type	Source	Derivation	Assigned	Pseudo Code
ADEX	ADY	Analysis Relative Day	integer		[ADT-TRTSDT+1 if ADT is on or after the treatment start date, ADT-TRTSDT if ADT precedes the treatment start date]		condderivation

... | Variable Metadata | recode | rename | condderivation | lookup | std_function | Where(...

Рисунок 13. Відповідна вкладка "condderivation"



Dataset Name	Variable Name	Derivation Condition	Output Variable Value	Source Variable or Formula
ADEX	ADY	. < ADT < TRTSDT		ADT - TRTSDT
ADEX	ADY	ADT >= TRTSDT > .		ADT - TRTSDT + 1

Dataset Metadata | Variable Metadata | recode | rename | condderivation | lookup | std_function | W

Згенерований SAS код виглядає так:

```
if . < ADT < TRTSDT then ADY = ADT - TRTSDT;
else if ADT >= TRTSDT >. then ADY = ADT – TRTSDT + 1;
```

Псевдокоди для кожної змінної можуть бути визначені на рівні глобального стандарту, і користувачі можуть налаштовувати псевдокод залежно від специфіки дослідження.

2.2. Автоматизація програм

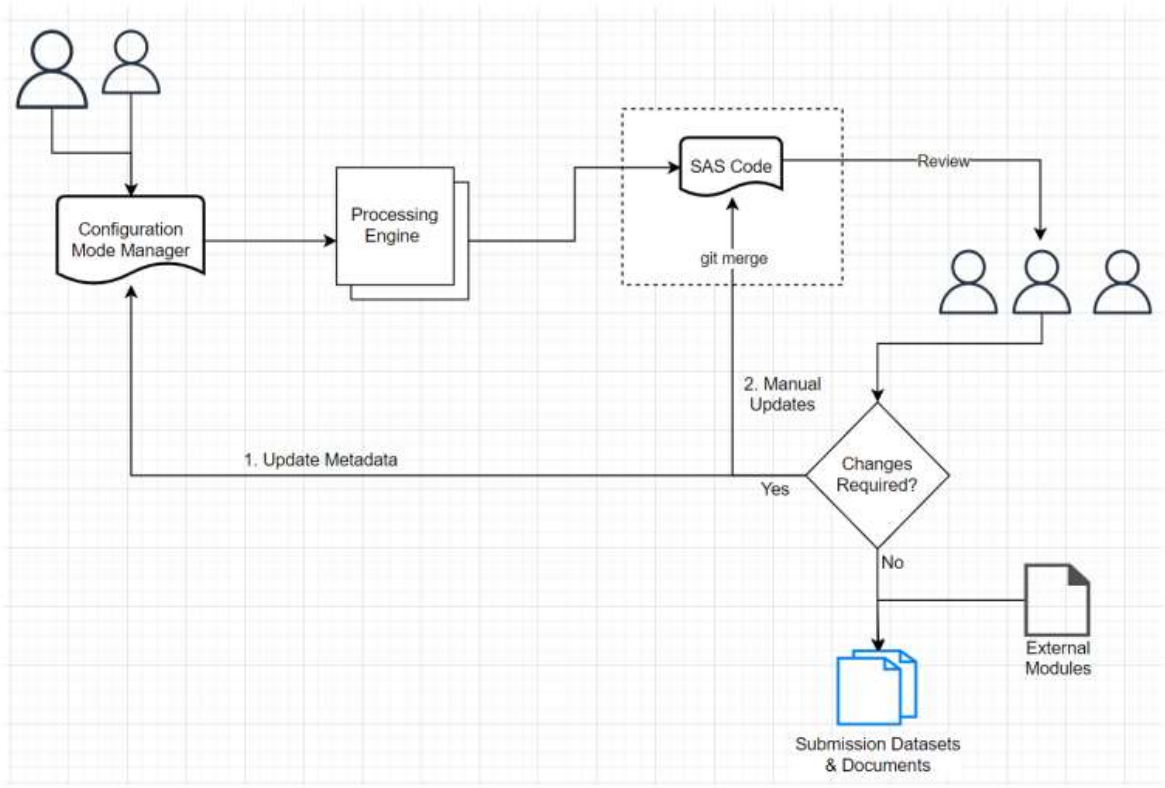
Щоб досягти автоматизації, ми застосовуємо підхід на основі конфігурації, генеруючи кінцеві програми для SDTM та ADaM. Використовуючи стандартні метадані, ми створюємо бібліотеки трансформації метаданих, які використовуються генератором коду. Ця конфігурація вводиться в двигун генератора коду, який створює кінцеві програми.

Для виведення змінної ми створили універсальну модель конфігурації, яка захоплює всю необхідну інформацію. Ця модель може бути змінена користувачами відповідно до складності вимог, створюючи гнучку модель, якою керують користувачі, а не розробник програмного забезпечення.

Генератор коду використовує цю конфігурацію разом з іншими параметрами (формати програм тощо) для створення вихідного коду. Користувачі можуть вибирати, якою мовою буде створено вихідну програму. На даний момент ми протестували SAS та R.

Генератор коду — це інструмент програмного забезпечення, який автоматично генерує код на основі вхідних специфікацій або параметрів. Згенерований код призначений для надання стартової точки для програміста, зменшуючи кількість ручного кодування, необхідного для проекту. Генератори коду можуть генерувати код для різних цілей, таких як створення коду для доступу до бази даних, створення коду користувацького інтерфейсу або створення коду для певних шаблонів проектування. Згенерований код може бути налаштований і вдосконалений відповідно до потреб конкретного проекту. Використання генераторів коду може підвищити ефективність розробки програмного забезпечення та зменшити ймовірність помилок у кодуванні. Рисунок 14 описує процес автоматизації програм.

Рисунок 14: Потік процесу автоматизації програм



2.3 Переваги та виклики псевдокоду

Навіть на стадії розробки проекту, пропонуються наступні переваги псевдокодів:

- Агностичність

Псевдокод не залежить від мови програмування. Як зазначено в розділі "Автоматизація програм", псевдокод — це не просто програмний код, а машинозчитувальний алгоритм виведення, який можна перекласти на будь-який програмний код. Схема для кожного псевдокоду може бути повторно використана будь-якою мовою програмування. Ми розробили шаблон для кожного псевдокоду в SAS і плануємо розробити інший набір шаблонів в R.

- Консистентність

Псевдокод і додатковий введення користувача в відповідних колонках і вкладках слугують джерелом для генератора коду, який створює програму, а також специфікації виведення для програмістів. Більше того, для трьох псевдокодів: “source”, “derivation”, і “assigned”, записи, надані користувачем у колонках “Source”, “Derivation” і “Assigned” слугують джерелом для define.xml. Ми розробляємо спосіб перетворення решти 6 псевдокодів у атрибути define.xml. Це в кінцевому підсумку запобігатиме програмістам від багаторазового введення тієї самої інформації в документах специфікацій, програмному коді та define.xml, забезпечуючи їхню узгодженість.

- Структурованість

Псевдокод забезпечує структуру для визначення алгоритмів виведення. Під час дослідження алгоритмів виведення змінних для визначення псевдокодів ми зрозуміли, що поточні специфікації виведення не завжди очевидні для написання програмного коду. Рівень ясності та детальності варіювався від змінної до змінної і від дослідження до дослідження. І алгоритми, визначені в специфікаціях, відрізнялися від автора до автора, від

специфікації до специфікації. У результаті одна й та сама специфікація могла бути інтерпретована по-різному. На відміну від специфікацій, написаних природною мовою, псевдокод забезпечує конкретний структурований алгоритм, який вимагає певного введення користувача, але в заздалегідь визначеному форматі з однаковим рівнем деталізації і інтерпретується однозначно.

Ми протестували наш генератор коду на кількох пілотних дослідженнях, і стикнулися з такими викликами:

- Псевдокод і функції генератора коду є новими концепціями і потребують навчання та підтримки користувачів.
- Визначення правильних/складних псевдокодів на рівні дослідження може бути складним.
- Цей процес додає нетривіальний шар метаданих, який вимагає сучасних інструментів для попереднього перегляду майбутніх SAS кодів і введення псевдокодів.

ВИСНОВОК

У цій роботі ми представили підхід до автоматизації створення програм наборів даних. Хоча цей проект ще розробляється, ми впевнені, що цей підхід зробить рутинне щоденне програмування даних клінічних випробувань легшим та більш ефективним.

Удосконалення автоматизації процесів програмування даних сприятиме швидкішому впровадженню клінічних досліджень і підвищить якість аналізу даних. Очікується, що подальші розвиток та вдосконалення цього підходу значно полегшать роботу програмістів та сприятимуть збільшенню продуктивності в галузі клінічних досліджень.

Загалом, ініціатива з автоматизації програмування даних у клінічних дослідженнях є кроком у майбутнє, який сприятиме швидкішому та ефективнішому аналізу медичної інформації, що, в свою чергу, може позитивно позначитися на розробці нових методів лікування та підвищенні якості медичної практики.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Cody, Ron. Learning SAS by example: a programmer's guide. SAS Institute, 2007.
2. Delwiche, Lora D., and Susan J. Slaughter. The Little SAS book: a primer. SAS Institute, 2012.
3. Freeman, S., Eddy, S. L., McDonough, M., Smith, M. K., Okoroafor, N., Jordt, H., & Wenderoth, M. P. (2014).
4. SAS Book, Fifth Edition. SAS Intitute, 2015.
5. 8. Stojanovic, Mirjana. "Efficient Way to Learn SAS® with Virtually No Cost."SESUG 2003.
6. 10. Wing, Jeannette M. "Computational thinking and thinking about computing."Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences" (2008).