

Balancing Automation and Flexibility in Programming Processes

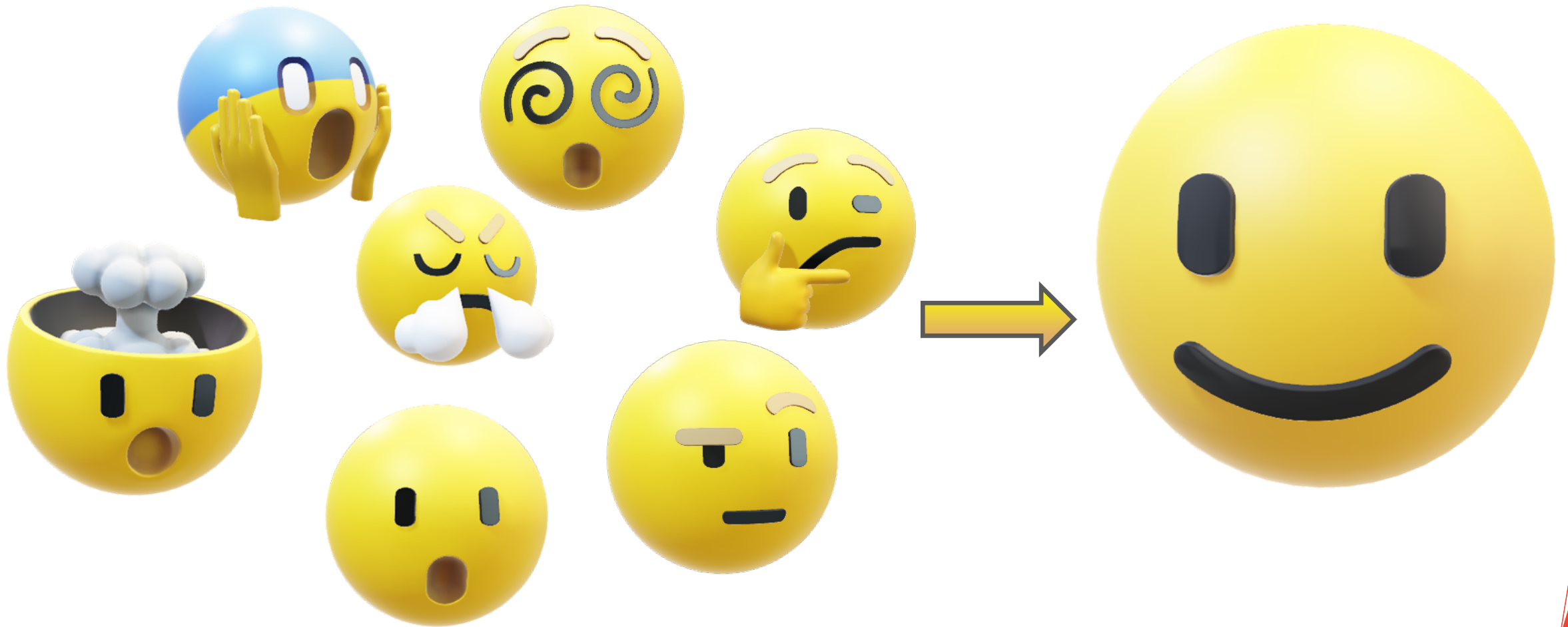
Helen Nicholson
Punitha Philip



The intelligent choice
for clinical trials



Striking the balance between automation and flexibility can be challenging!

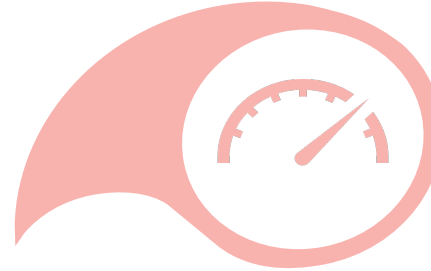


Many technical solutions and systems are built around end-to-end standardisation to get new and better drugs to market safely in the most efficient and timely manner

Achieve target timelines



Faster program creation



End-to-end
standardisation



Quality

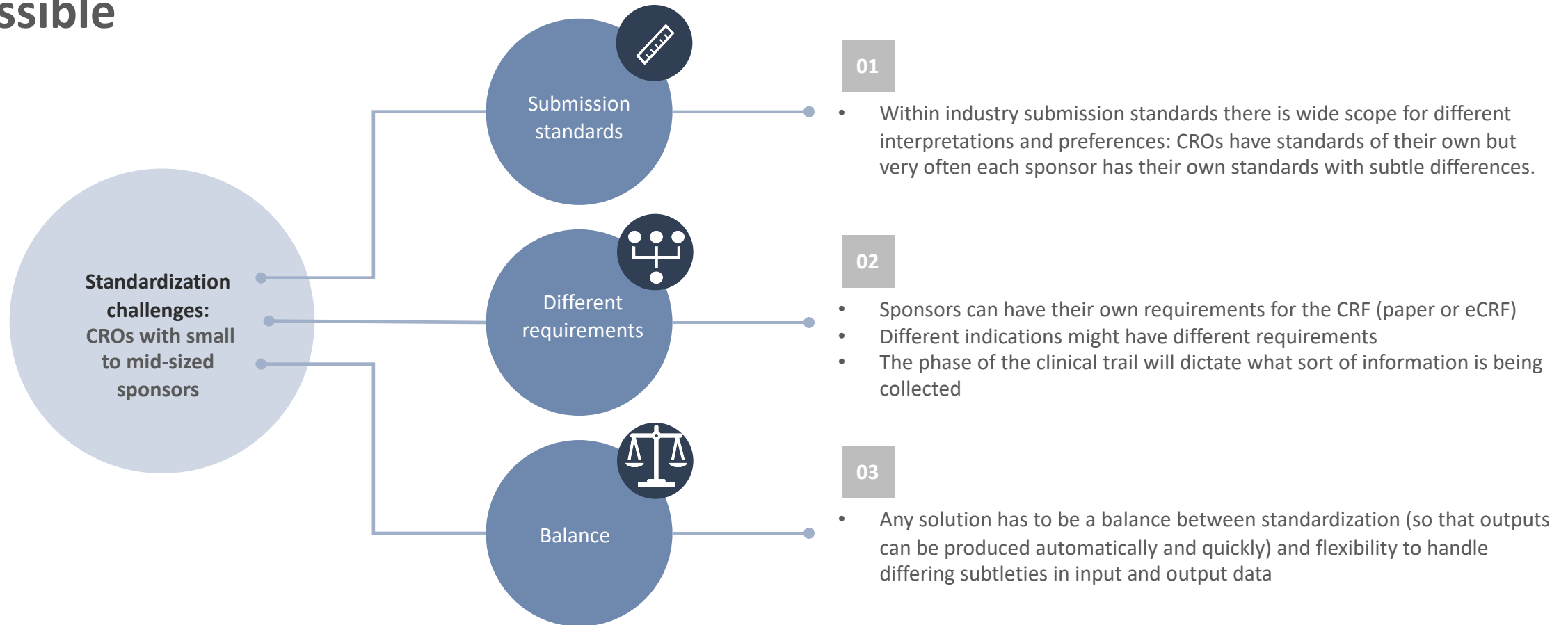
Quick validation



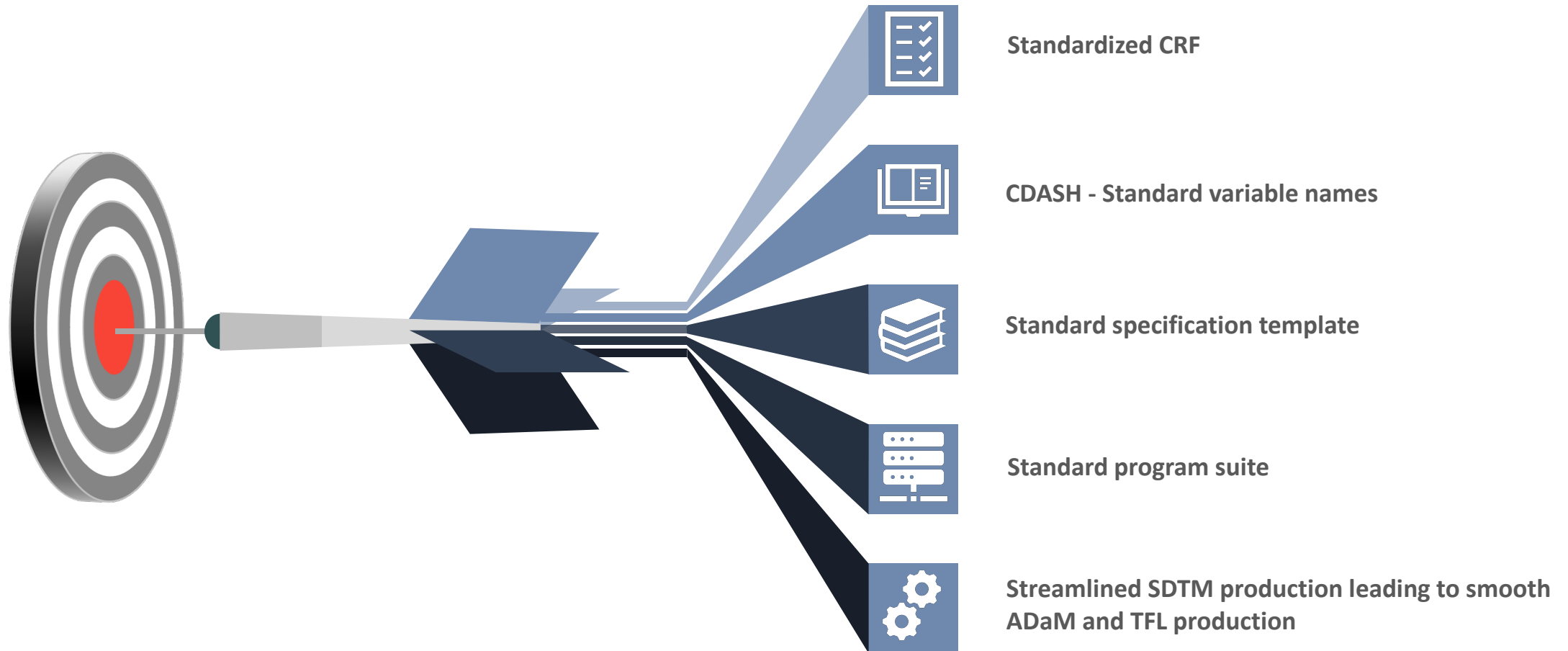
Efficiency



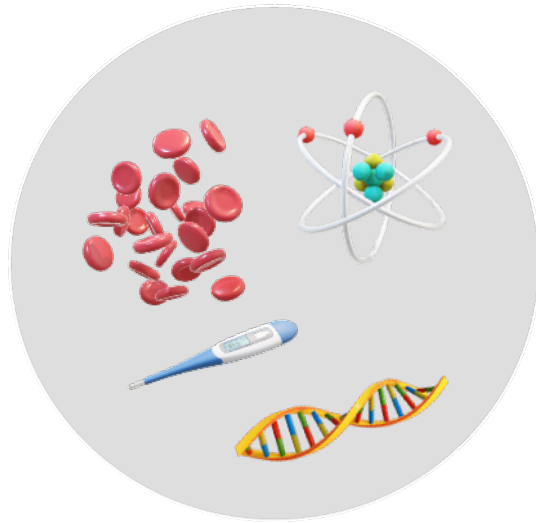
As a CRO with several different sponsors, all with their own requirements the target is still the same - to get the drug to markets as quickly as possible



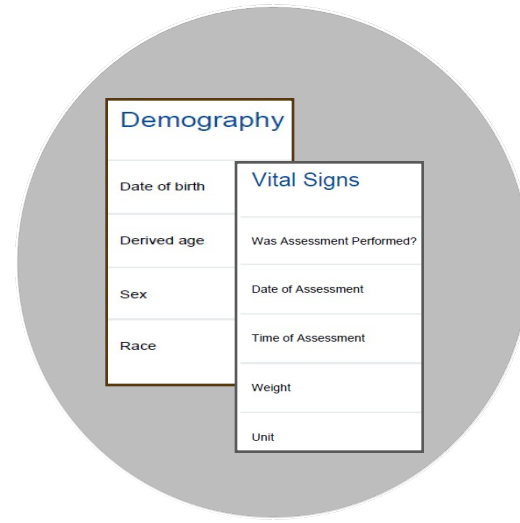
The ideal solution would be to have standardised trial pages/forms



In reality, each study has very different data requirements so you need the ability to collect many different sorts of data.



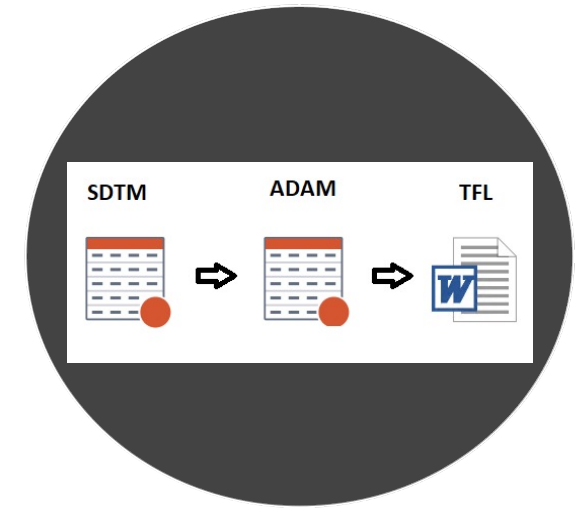
Different study data requirements



CDASH - but modified as per sponsor requirements



Study/sponsor specific specification and non standard programs



SDTM production with increased programmer involvement

Balancing standardization with adaptability gives flexible solution



=

**Flexible
Solutions**

Our solution allows the majority of the programming work to be done automatically, and then completed by a programmer

Dataset					
Domain	Name	Dataset type	Label	Sorted by	Data included/excluded
AE	AE	SDTM	Adverse Events	STUDYID, USUBJID, AETERM, AESTDTC, AEENDTC	Keep records where AETERM is not missing Bring records from: RAW.DM where CMTR
CM	CM	SDTM	Concomitant Medications	STUDYID, CMSTDTC	
DA	DA	SDTM	Drug Accountability	STUDYID, DATEST	
DD	DD	SDTM	Death Details	STUDYID	
DM	DM	SDTM	Demographics	STUDYID	
DS				STUDYID, DSDECO	

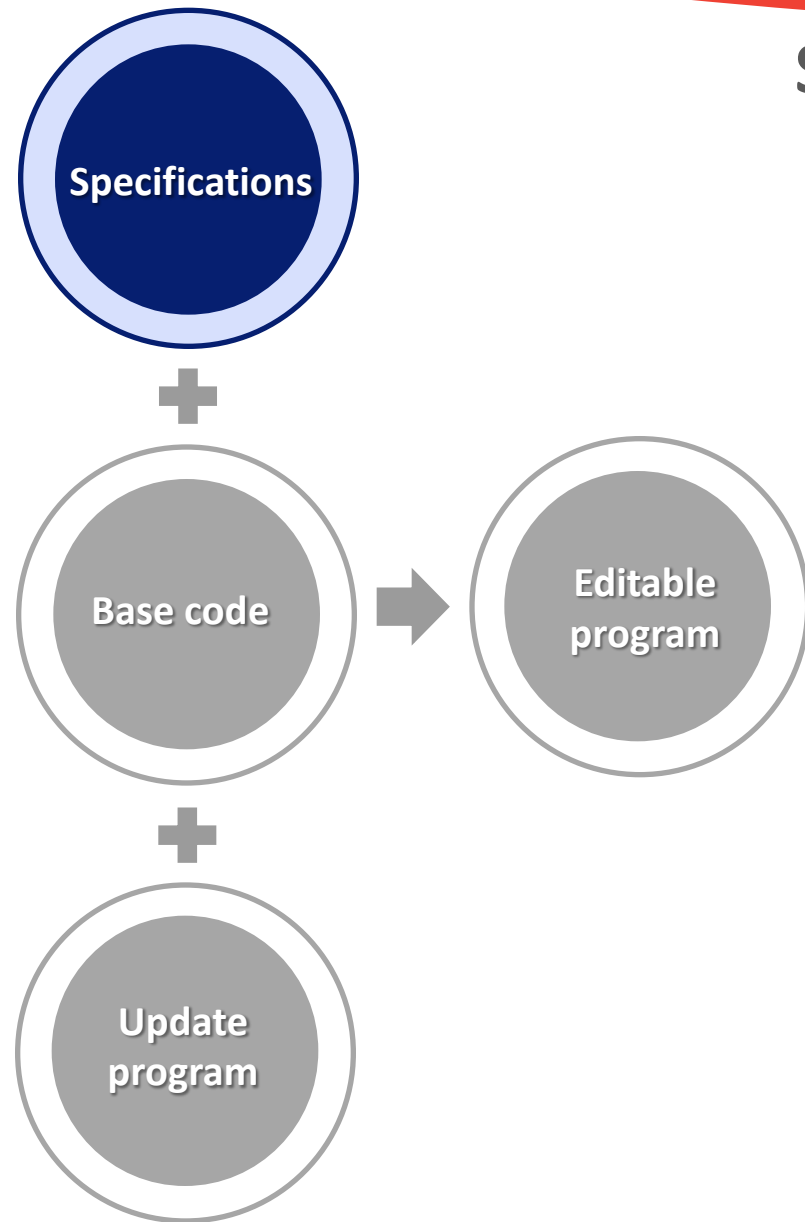
Base code

```
Code  Log
1  /*****
2  *
3  *   Study: generic
4  *
5  *   Program name:  dm_qc.sas
6  *
7  *   Location:  ...&ProjPath/programs/sdtm_qc
8  *
9  *   Description:  To qc SDTM.DM and SUPPDM
10 *
11 *   Input files:  raw.dm, ex_c1, dd, ds_eos, ds_scr
12 *
13 *   Output files:  the compare
14 *
15 *   Macros:  divertlg01 scanlog01 day01 qcstatus01 qc_label qc_compare
16 *
17 *   Assumptions:
18 *
19 *   Program History:
20 *   =====
21 *
22 *   Date       |      Name      | Reason/Comments
23 *   -----
24 *   ddmmyyyy   | _user_name_    | Original
25 *   -----
26 *   ****
27 *
28 *
29 *   for debugging and sorting out subject ;
30 *   %let debug= ;*and subject eq '01-001-A' ;
31 *
32 *   *** divert the log and bring in the labels for the variables ***;
33 *
34 *   %let ds=dm;
35 *   %divertlg01(logname=sdtm_qc/&ds._qc);
36 *
37 *   %qc_labels(ds=dm) ;
```



Edit updated program and finalisation

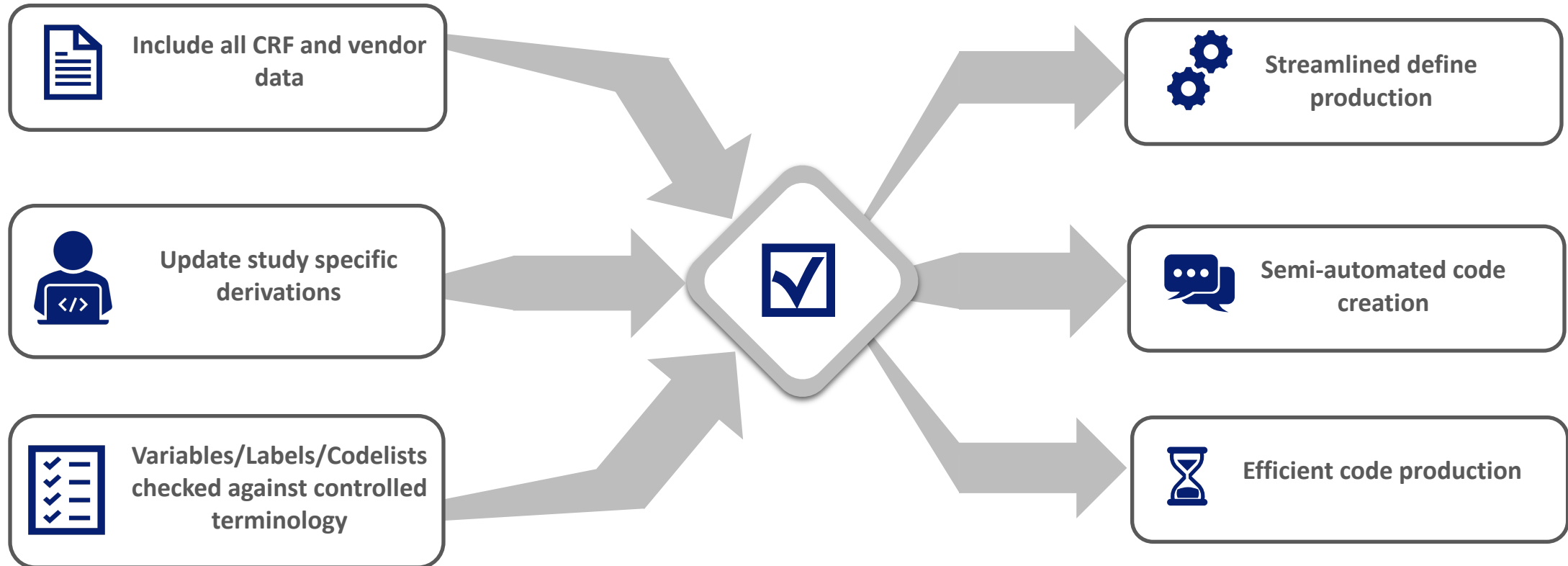
Specifications are based on a template



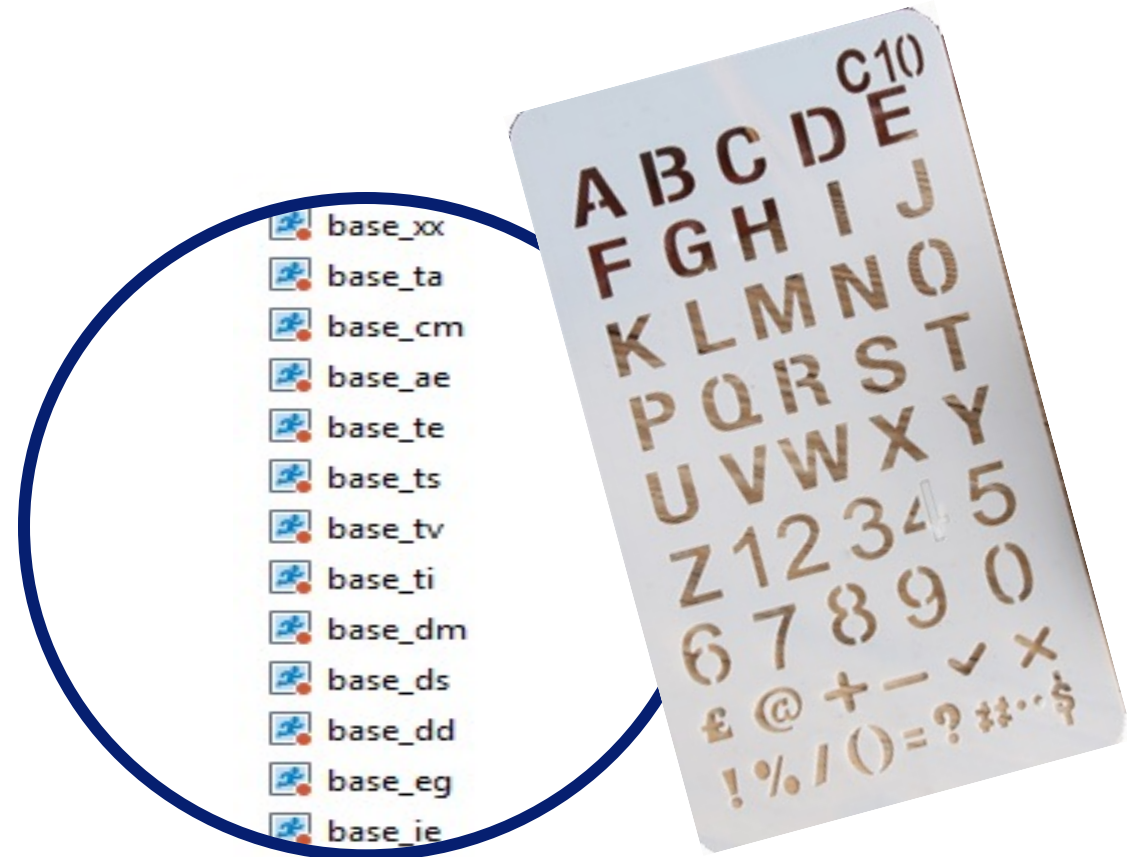
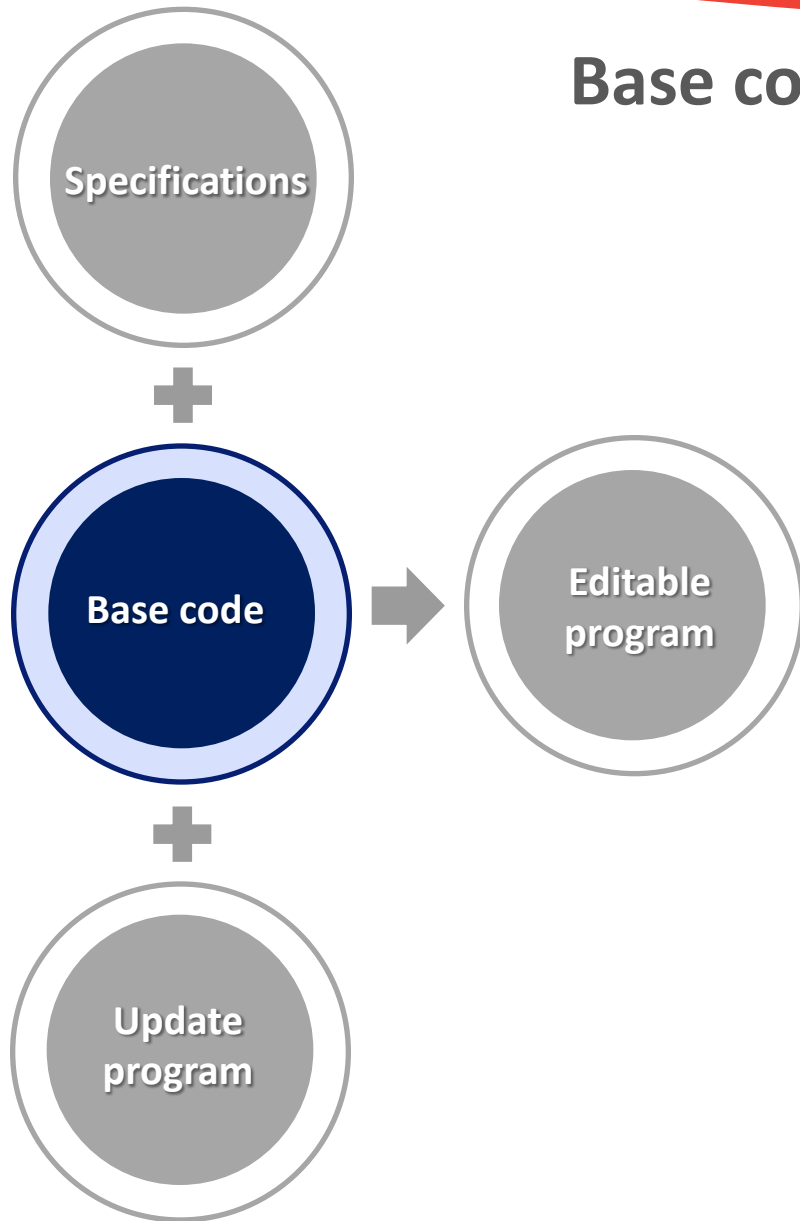
Dataset	Variable	Label	Select	Type	Core Status (Req / Perm / Cond)	Code List
ADSL	STUDYID	Study Identifier	x	text	Req	
ADSL	USUBJID	Unique Subject Identifier	x	text	Req	
ADSL	SUBJID	Subject Identifier for the Study	x	text	Req	
ADSL	SITEID	Study Site Identifier	x	text	Req	
ADSL	SITEGRy	Pooled Site Group y		text	Perm	
ADSL	SITEGRyN	Pooled Site Group y (N)		integer	Perm	
ADSL	REGIONy	Geographic Region y		text	Perm	REGION
ADSL	REGIONyN	Geographic Region y (N)		integer	Perm	REGION
ADSL	COUNTRY	Country		text		
ADSL	AGE	Age	x	integer	Req	
ADSL	AGEU	Age Units	x	text	Req	AGEU
ADSL	AGEGRy					
ADSL	AGEGRyN					
ADSL	AAGE					
ADSL	SEX					
ADSL	ASEX					

Domain	Variables	Label	Type	Length	Source	Derivation/Mapping
AE	STUDYID	Study Identifier	Char	15	Assigned	Set to 'xxxxxxxxxxxxx' as per protocol.
AE	DOMAIN	Domain Abbreviation	Char	8	Assigned	Set to 'AE'
AE	USUBJID	Unique Subject Identifier	Char	30	Derived	Derive from concatenation of STUDYID and R
AE	AESEQ	Sequence Number	Num	8	Derived	Sequential number identifying records within sequence of the domain. Use the sort identifi
AE	AESPID	Sponsor-Defined Identifier	Char	10	CRF	RAW.AE.AESPID
AE	AEREFID	Reference ID	Char	20	CRF	RAW.AE.AEREFID
AE	AETERM	Reported Term for the Adverse Event	Char	200	CRF	RAW.AE.AETERM capitalised
AE	AELLT	Lowest Level Term	Char	200	CRF	RAW.AE.AE_LLT
AE	AELLTCD	Lowest Level Term Code	Num	8	CRF	RAW.AE.AE_CODE
AE	AEDECOD	Dictionary-Derived Term	Char	200	CRF	RAW.AE.AE_PTNM
AE	AEPTCD	Preferred Term Code	Num	8	CRF	RAW.AE.AE_PT
AE	AEHLT	High Level Term	Char	200	CRF	RAW.AE.AE_HLTN
AE	AEHLTCD	High Level Term Code	Num	8	CRF	RAW.AE.AE_HLT
AE	AEHLGT	High Level Group Term	Char	200	CRF	RAW.AE.AE_HLGN
AE	AEHLGTC	High Level Group Term Code	Num	8	CRF	RAW.AE.AE_HLGT
AE	AECAT	Category for Adverse Event	Char	60	CRF	RAW.AE.AECAT
AE	AESCAT	Subcategory for Adverse Event	Char	60	CRF	RAW.AE.AESCAT
AE	AEBODSYS	Body System or Organ Class	Char	200	CRF	RAW.AE.AE_CLAS
AE	AEBDSYCD	Body System or Organ Class Code	Num	8	CRF	RAW.AE.AE_SOC
AE	AESOC	Primary System Organ Class	Char	200	CRF	RAW.AE.AE_CLAS

The specifications form the foundation of the process and hence need to be completed as per the guidelines



Base code is the template code for each domain



Commonly used domains have a base program; a general program is available to cover non standard domains



Standard derivations for the domain

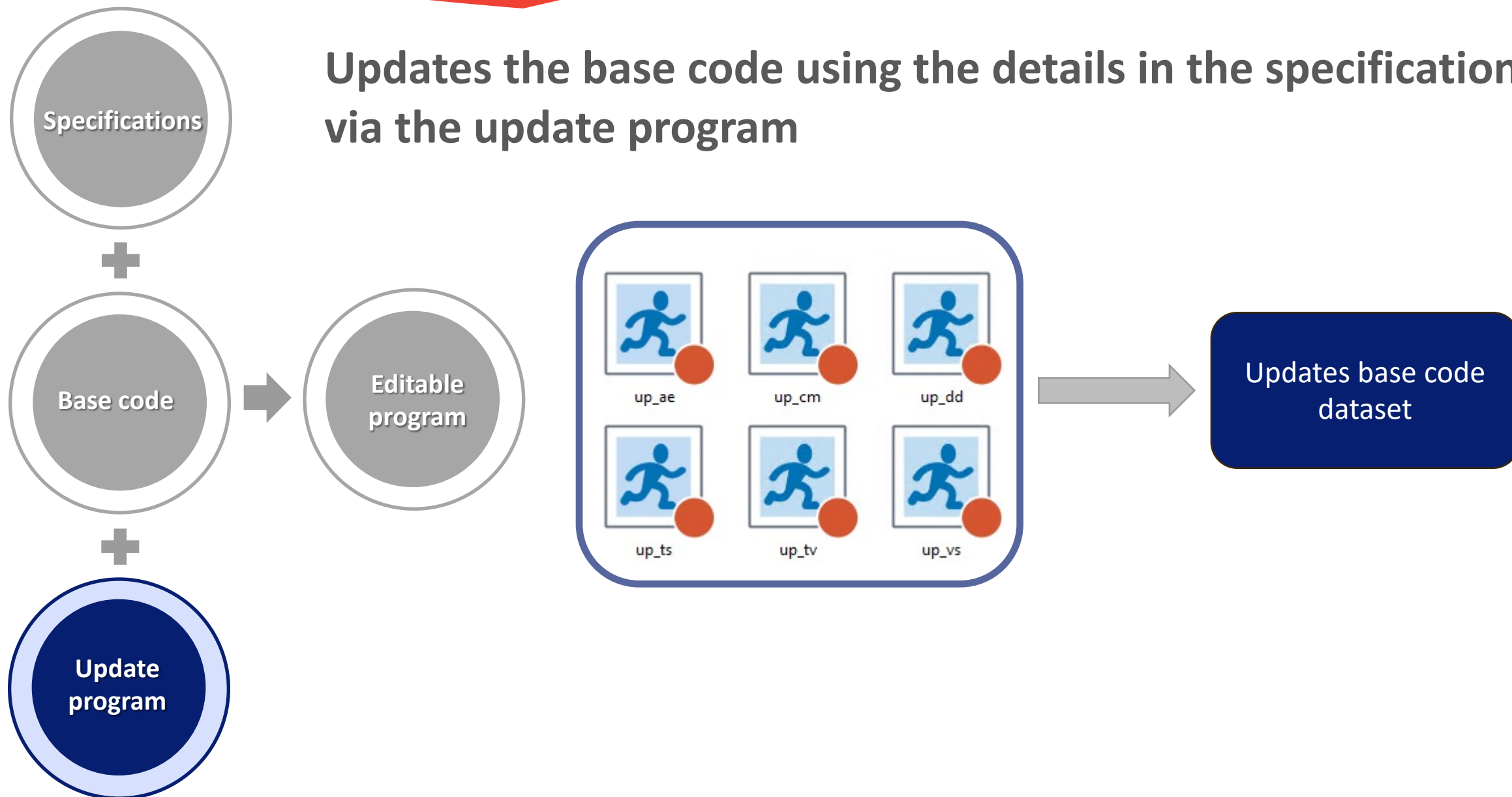


Calls to standard macros used in program structure

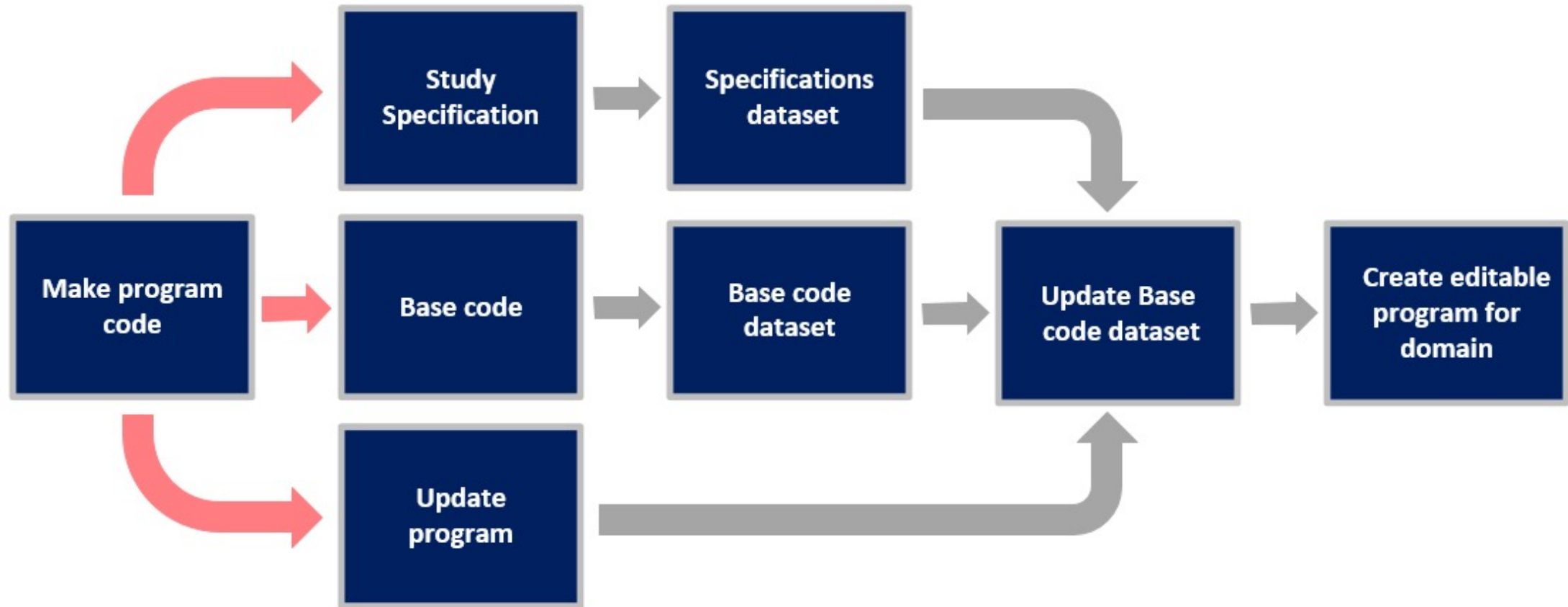


Calls to macros to populate variable attributes

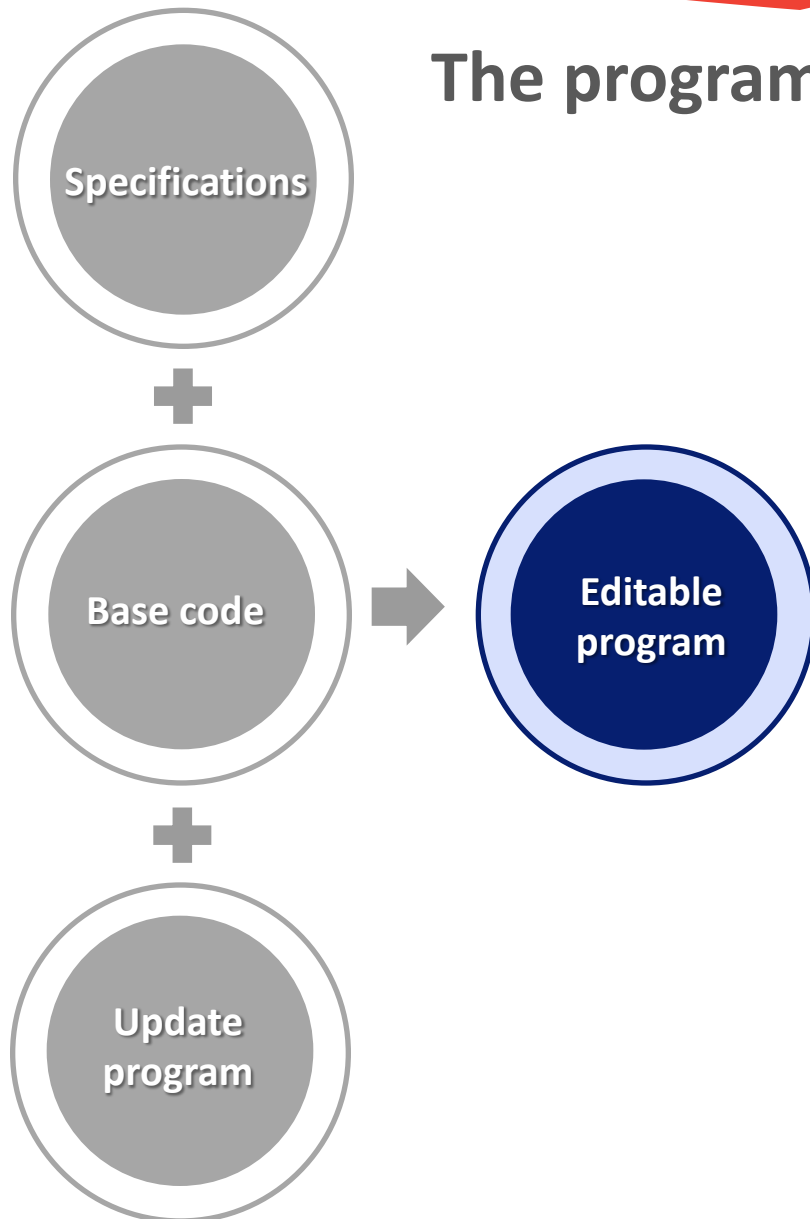
Updates the base code using the details in the specifications via the update program



The specifications along with the base and update programs are used to programmatically create the editable program for a domain

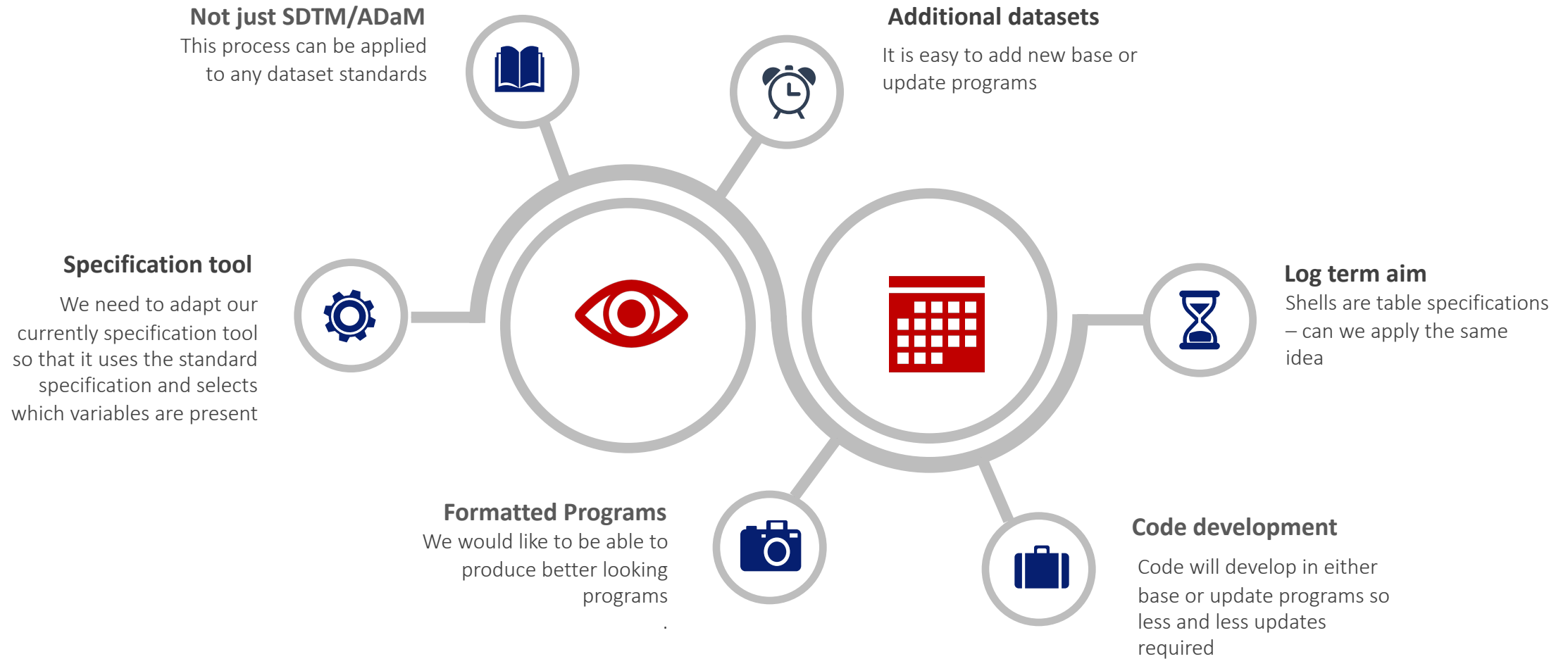


The program saved is edited and finalised by the programmer

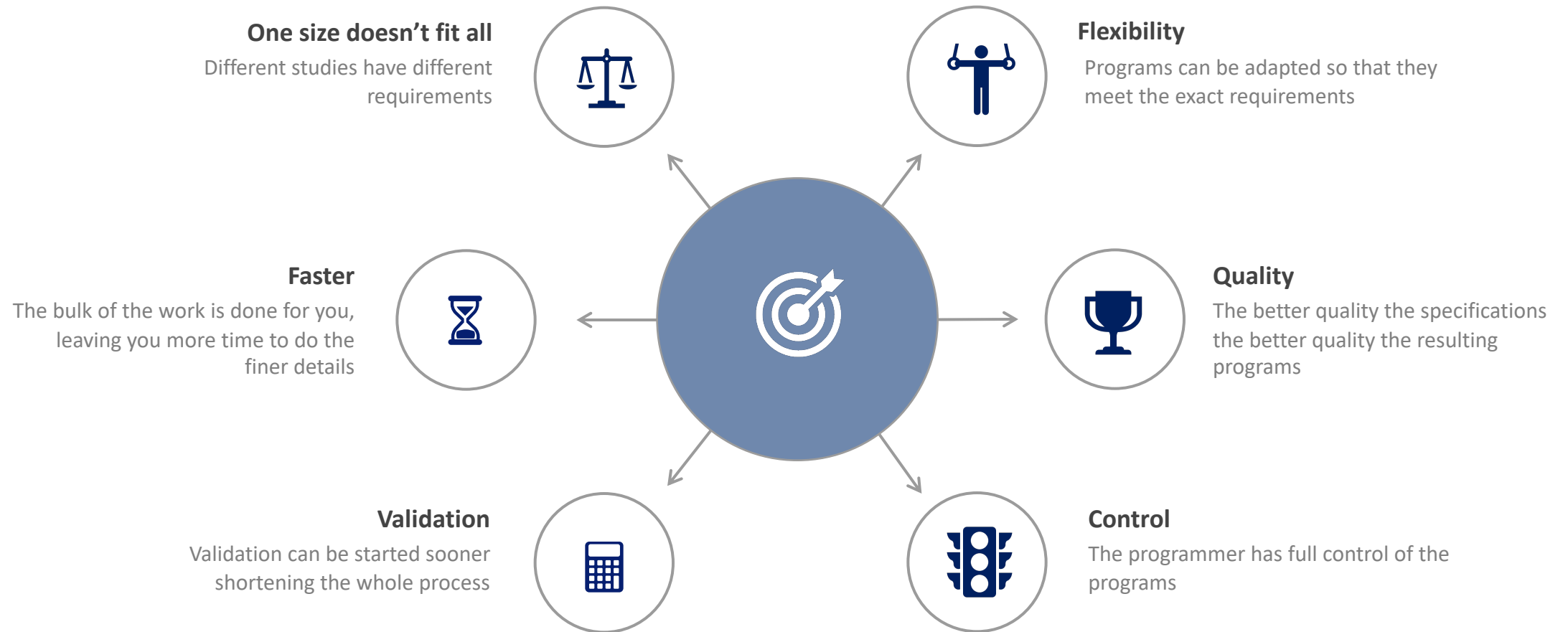


```
19 *
20 *   Program History:
21 *   =====
22 *
23 *   Date      |      Name      |   Reason/Comments
24 *   -----+-----+-----
25 *   04FEB2021 | <Programmer>   | Original
26 *   -----+-----+-----
27 *   *****/
28 *
29 *
30 * for debugging and sorting out subject ;
31 %let debug= ;*and subject eq '01-001-A' ;
32
33 *** divert the log and bring in the labels for the variables
34
35 %let ds=ae;
36 %divertlg01(logname=sdm_qc/&ds._qc);
37
38 %qc_labels(ds=ae) ;
39
40 proc datasets lib=work kill nolist;
41 run;
42 quit;
43
44 *** Bring in raw data ***;
45
46 data ae;
47 set raw.ae (rename=(aeterm=aeterm1 aeacn=aeacn1 aeser=aeser1 aere1=aere11
48 astdat=astdtdc aeendat=aeendtc
49 aeout=aeout1
50 aeacnoth=aeacnoth1
51 aesdisab=aesdisab1
52 aesdth=aesdth1
53 aeslife=aeslife1
54 aescong=aescong1
55 aesmie=aesmie1
56 /*add extra rename in here */
57 ));
58 length usubjid aeterm aellt aedecod aehlt aehlgt aebodsys aesoc aeser aeacn aere1
59 aeout aescong aesdisab aesdth aeshosp aeslife aesmie
```

Adaptable solutions can be applied in all aspects of the programming process



In conclusion



Helen Nicholson - hnicholson@cmedresearch.com

Punitha Philip – pphilip@cmedresearch.com

Thank you!

