verisian

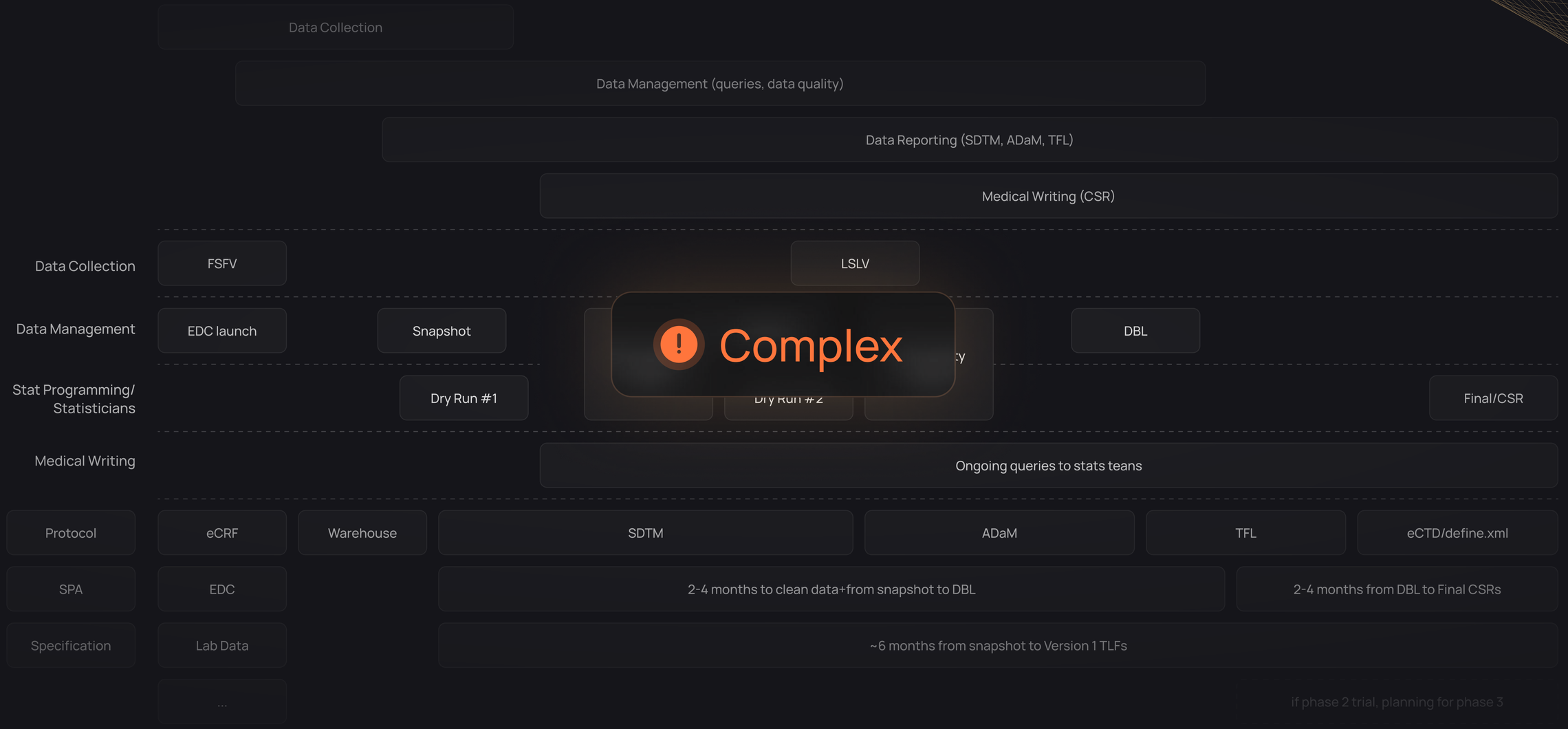
Building and Validating Clinical Trials Using AI Agents and Code Traceability

Tomás Sabat Stöfsel

CEO & Co-Founder, Verisian

Why can’t we release a drug immediately after the last pivotal trial?

Drug submission data workflow



We’re building technology to revolutionize statistical programming

We’re building technology to revolutionize statistical programming

Accelerate submission timelines

Enhance submission quality and confidence

Reduce programming effort and cost

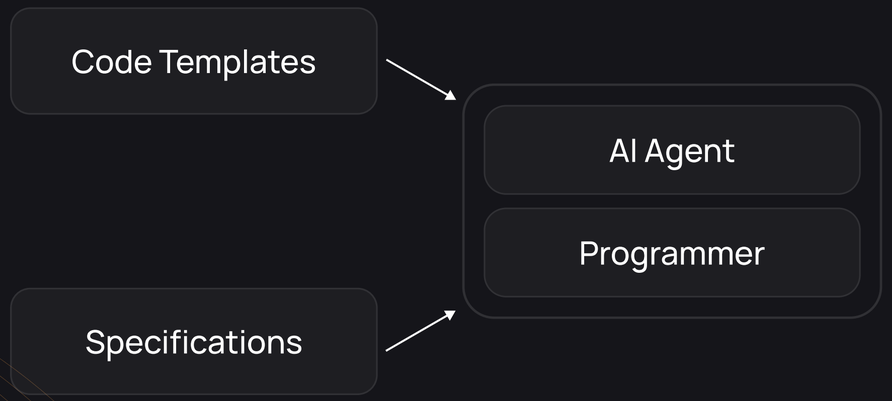
From AI-generated code to submission-ready tables

From AI-generated code to submission-ready tables

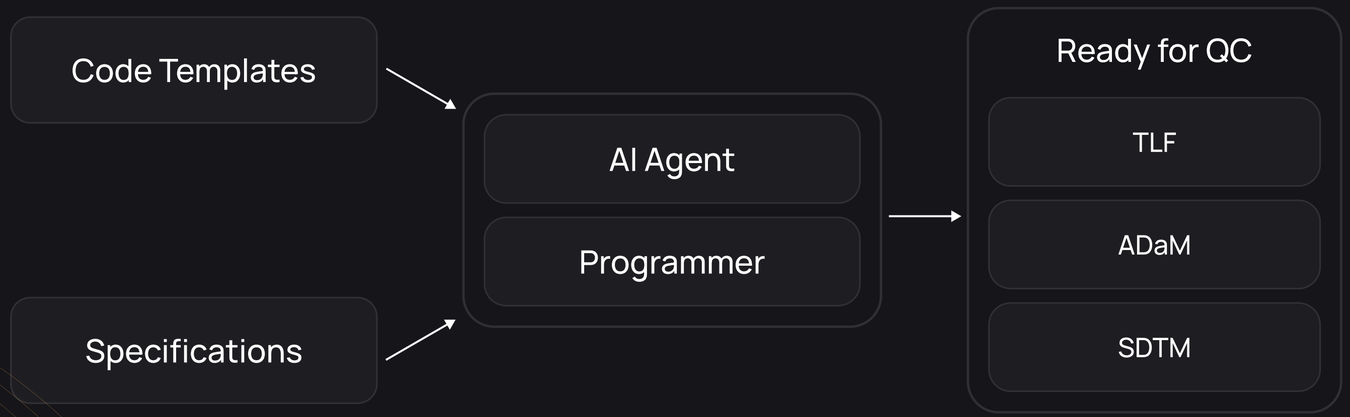
Code Templates

Specifications

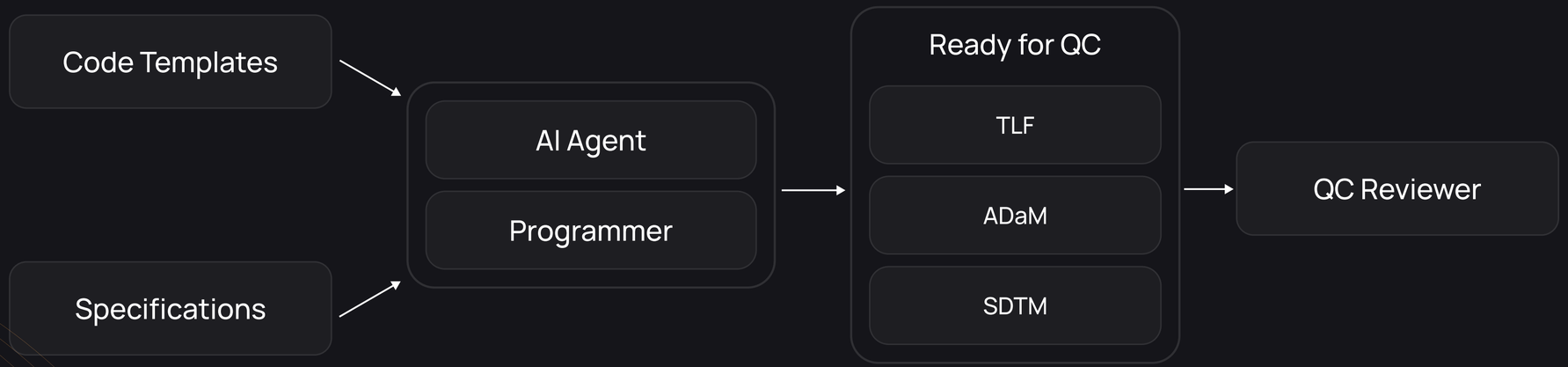
From AI-generated code to submission-ready tables



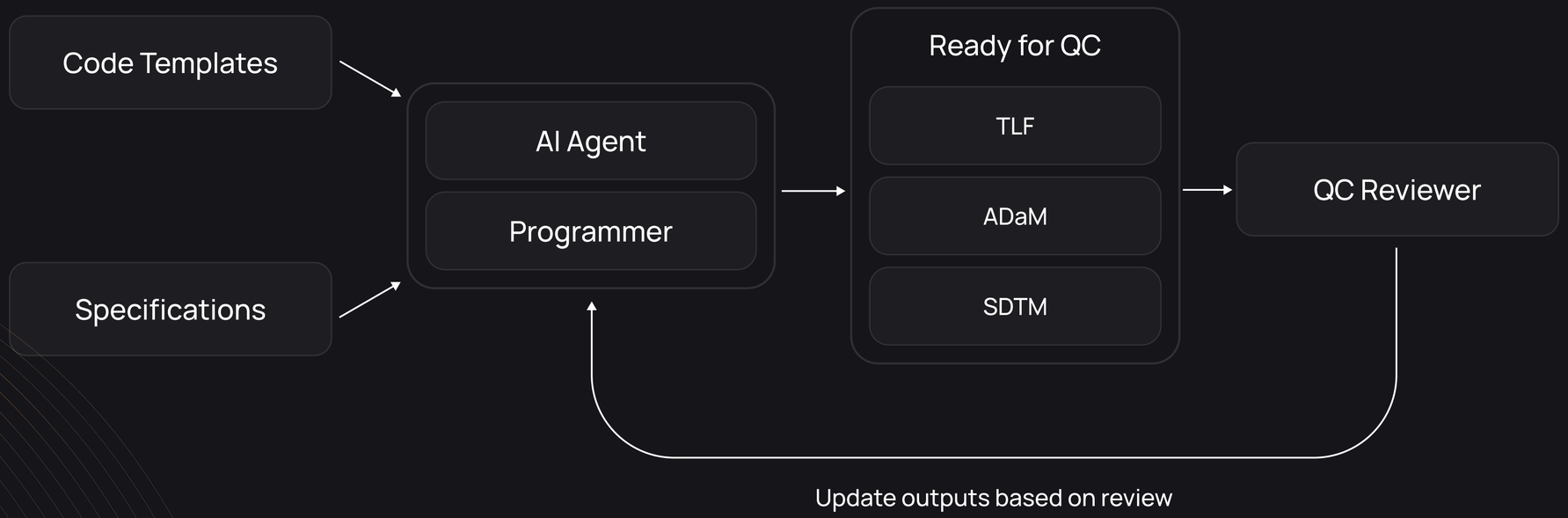
From AI-generated code to submission-ready tables



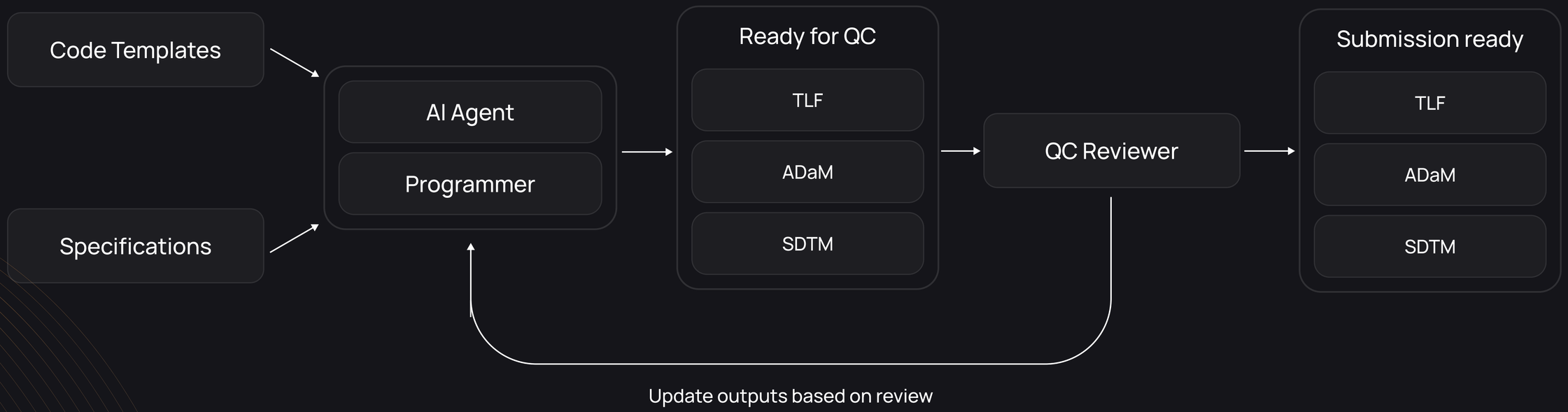
From AI-generated code to submission-ready tables



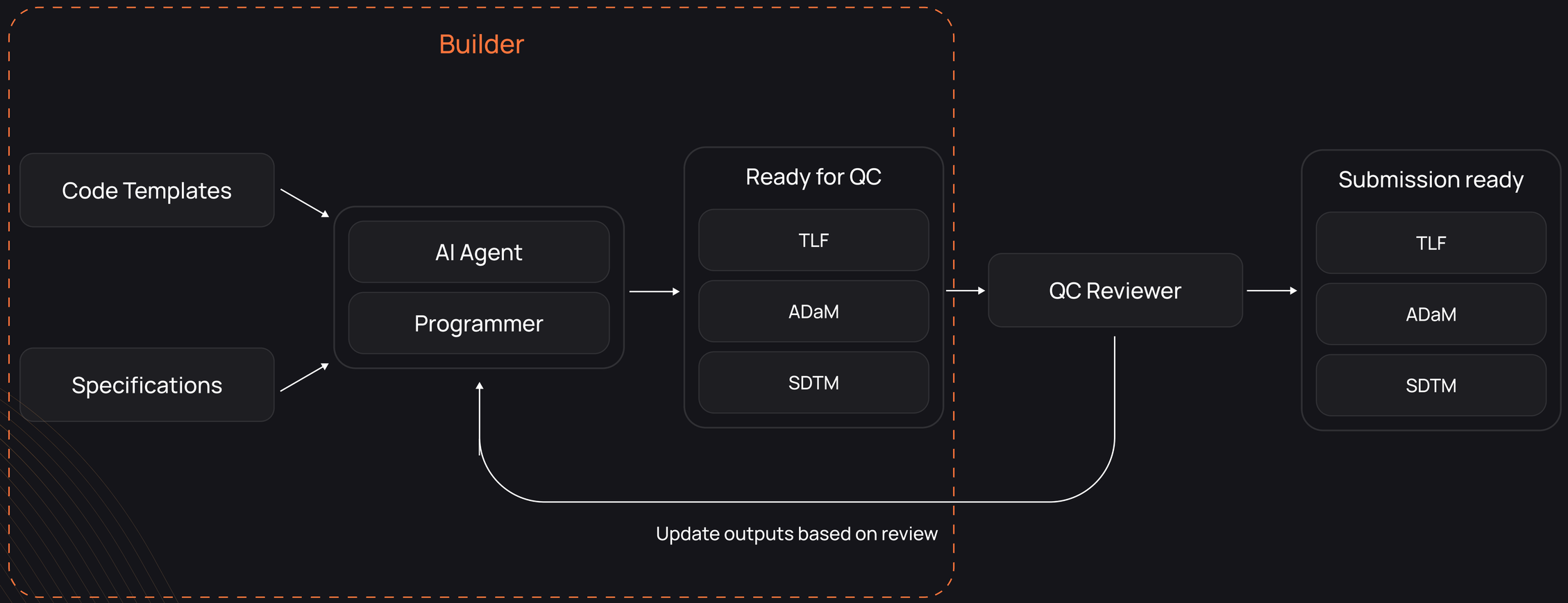
From AI-generated code to submission-ready tables



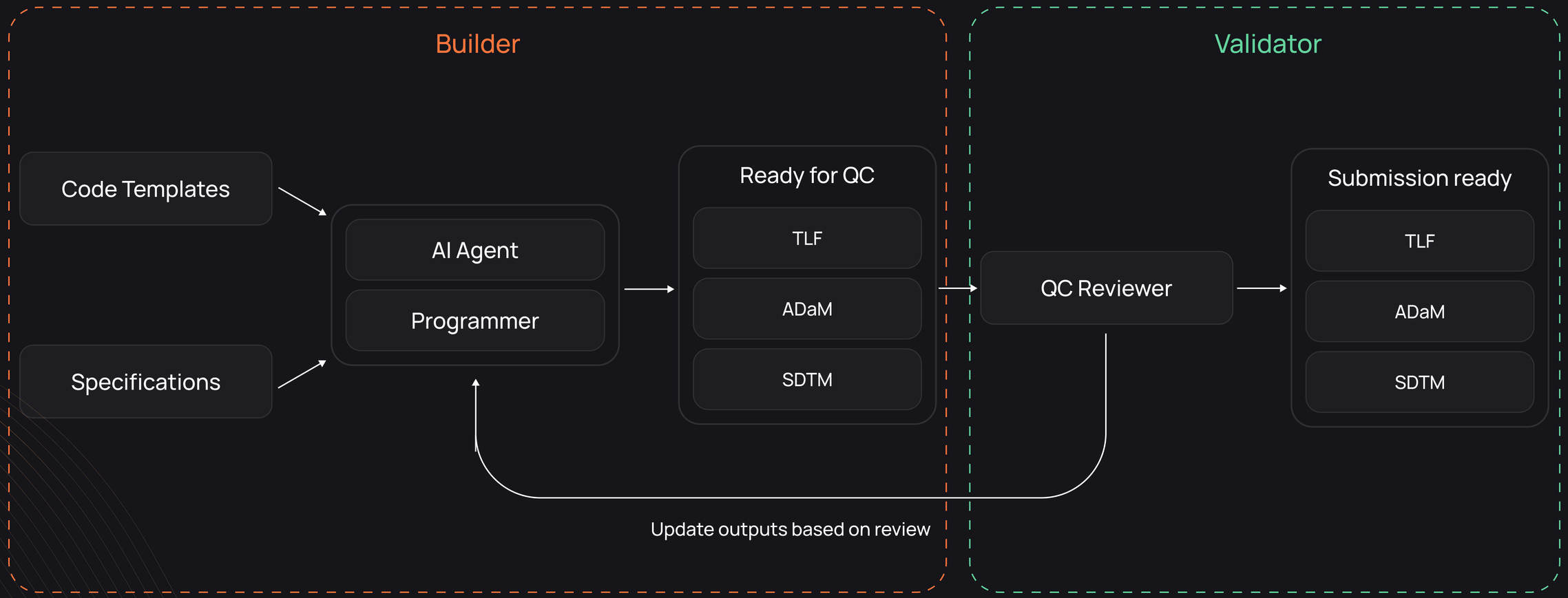
From AI-generated code to submission-ready tables



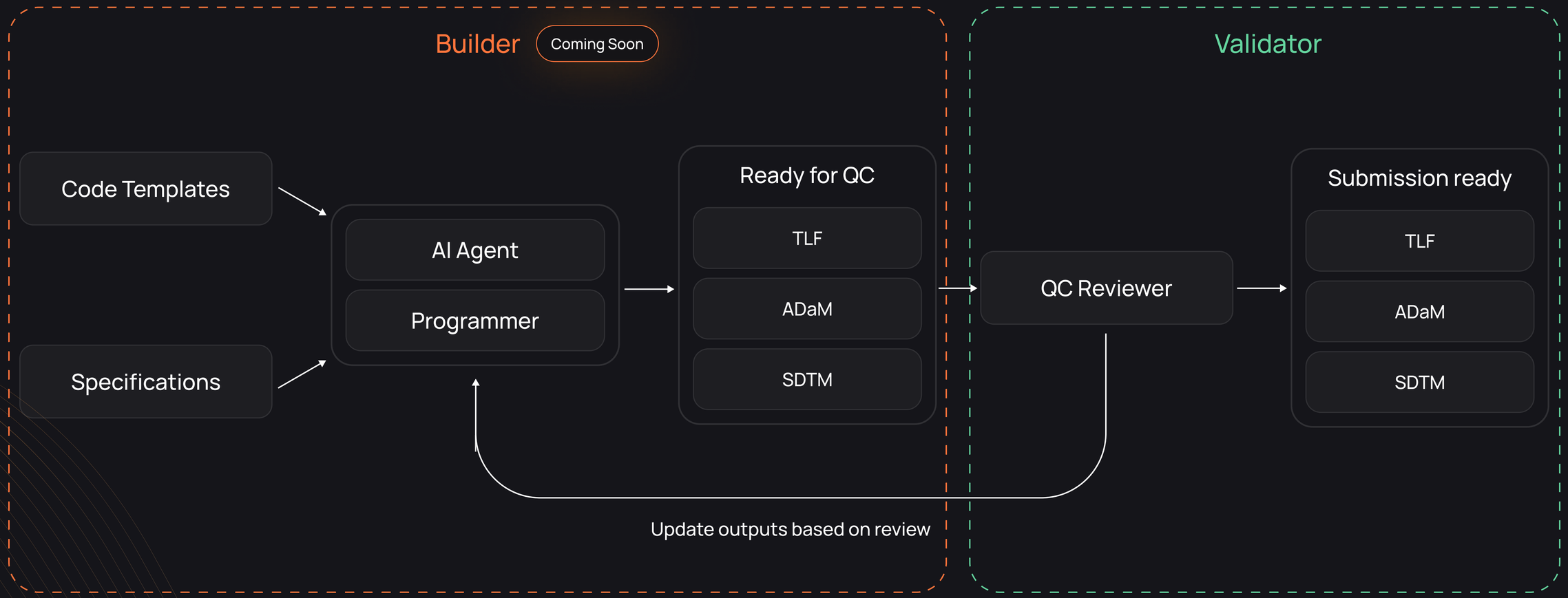
From AI-generated code to submission-ready tables



From AI-generated code to submission-ready tables



From AI-generated code to submission-ready tables



Validator

Accelerate submissions and approvals by transforming clinical trial analysis validation

What is a valid clinical trial analysis?

Valid variable derivations and data

Valid results

Double programming as the gold standard?

Double programming is flawed

Specifications are hard to write and validate

Shared misinterpretations lead to same mistakes

Code sharing undermines independence

Time pressure leads to shortcuts

Work done twice

Computational comparison fails

Two wrong programs agreeing...
Doesn't make them right

Programmers are stuck

Lack of collaboration
Lack of training

Validation or verification?

CDISC ensures format, not meaning

Tools report errors but don’t fix them

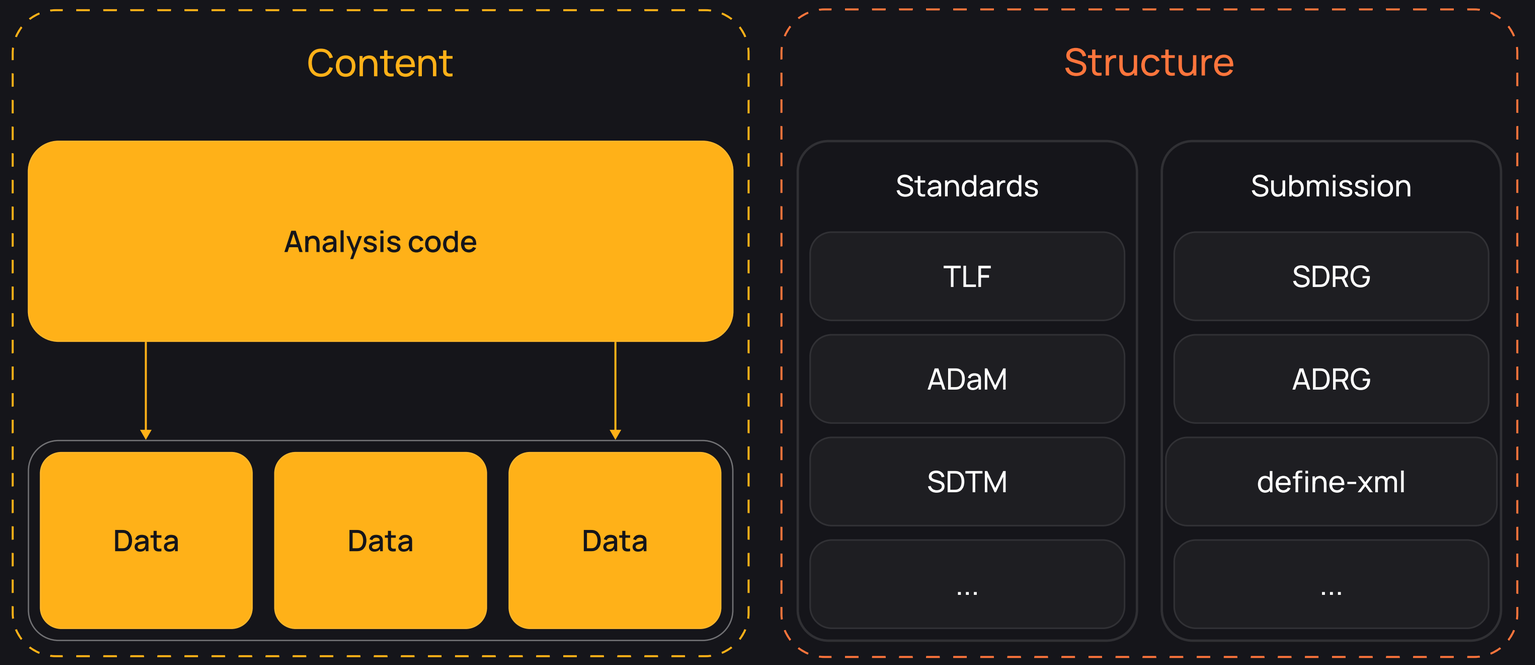
True validation requires content transparency

Checking format is not checking content

The principles of validation

Seamless integration of protocol, SAP, data, code and submission docs
Full transparency with end-to-end traceability
Constant collaboration and sharing of knowledge across teams

The source of truth of content of a trial is its code and data



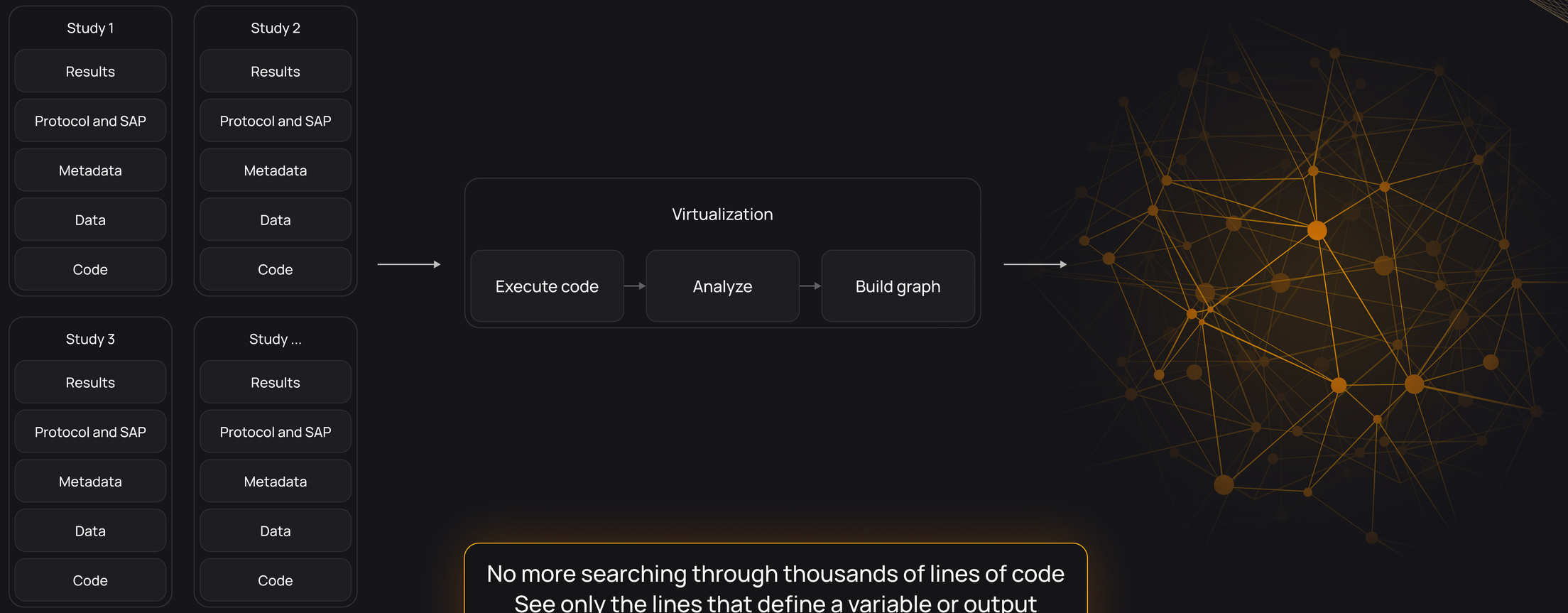
SAPs, specs, and documents describe what should happen, but the code shows what actually happens

What matters is the content of the study,
not the structure

Code Traceability changes everything

Tracing the content interactions between data, code and results
Independent of standards and languages

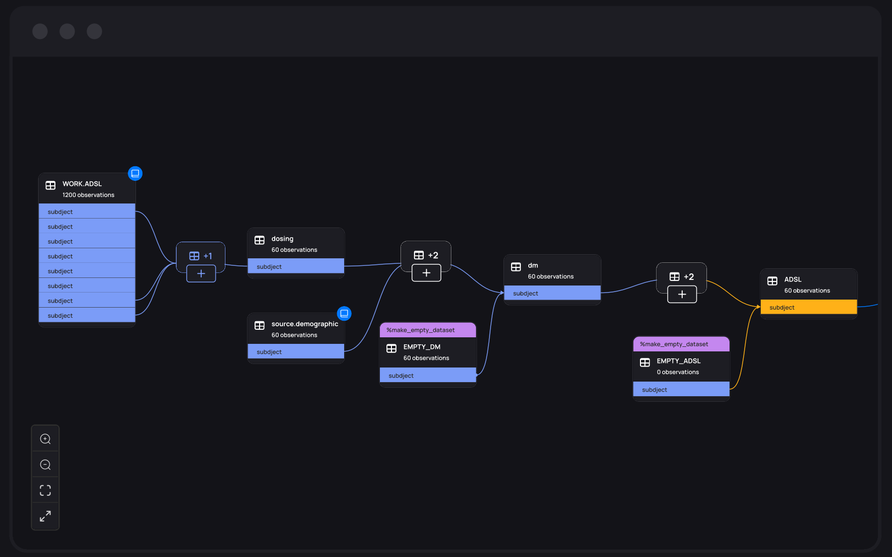
Code Traceability changes everything



No more searching through thousands of lines of code
See only the lines that define a variable or output
Instant clarity, zero noise

Code Traceability changes everything

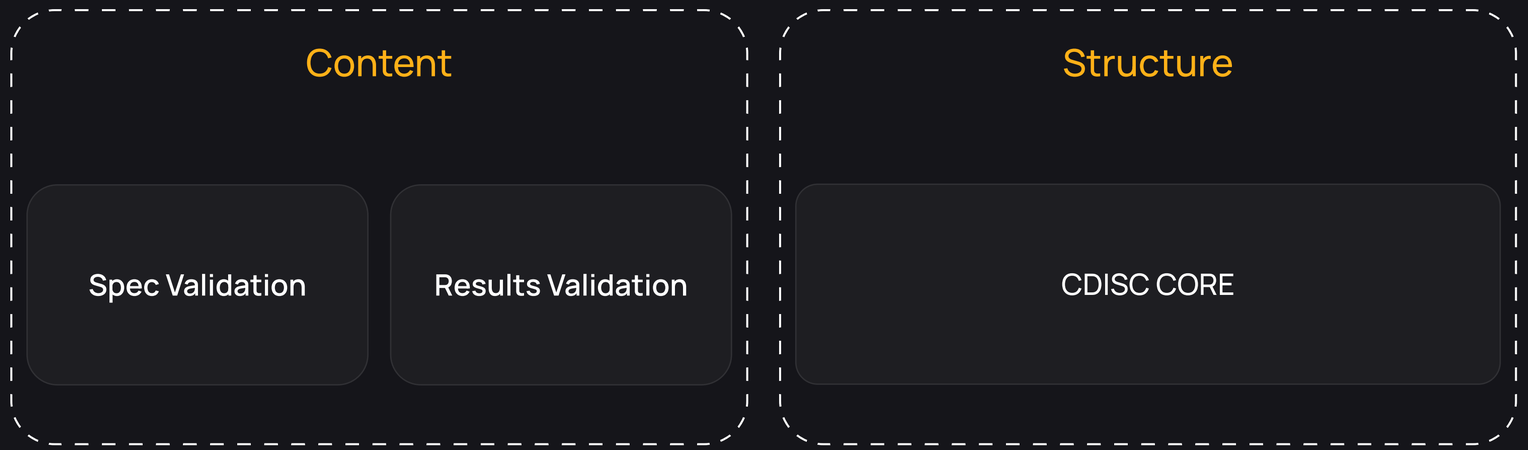
```sas
* demographic.sas;
data source.demographic;
 . label trt="Treatment";
  input trt 5;
  *...;
run;

* DM.sas;
data dm;
  *...;
  arm=put(trt, arm_demographic_trt.);
 . *...;
run;

* adsl.sas;
data ADSL;
  *..;
  trt01p=ARM;
  *...;
  trt01pn=input(put(trt01p, $trt01pn.), best.);
  *...;
run;

* adef.sas;
data adef;
  * ...;
  rename trt01pn=trtpn;
  *...;
run;
```



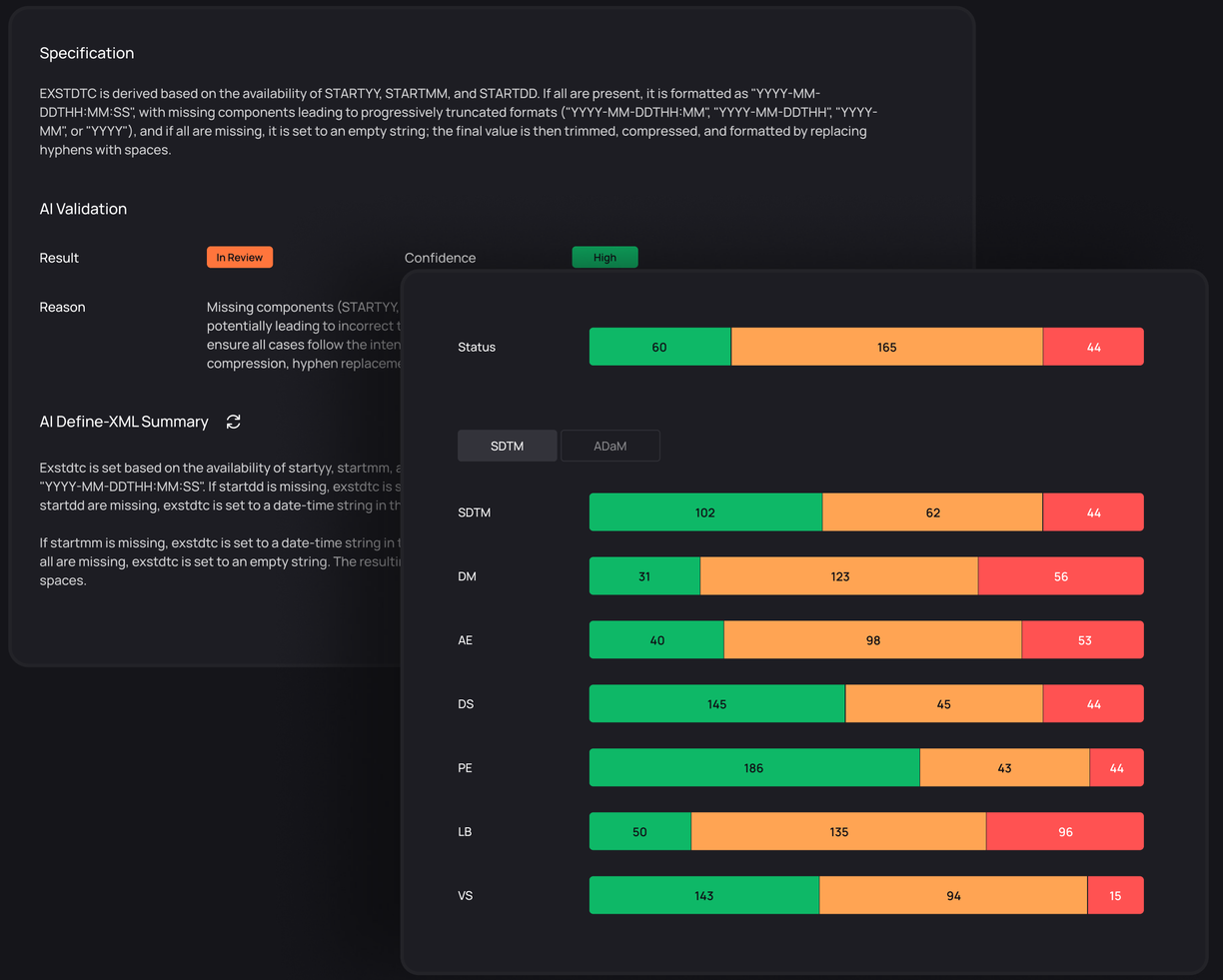
Study & patient traceability

Every transformation from raw data to final results
Find issues in seconds, not weeks
Understand why a patient is included or excluded

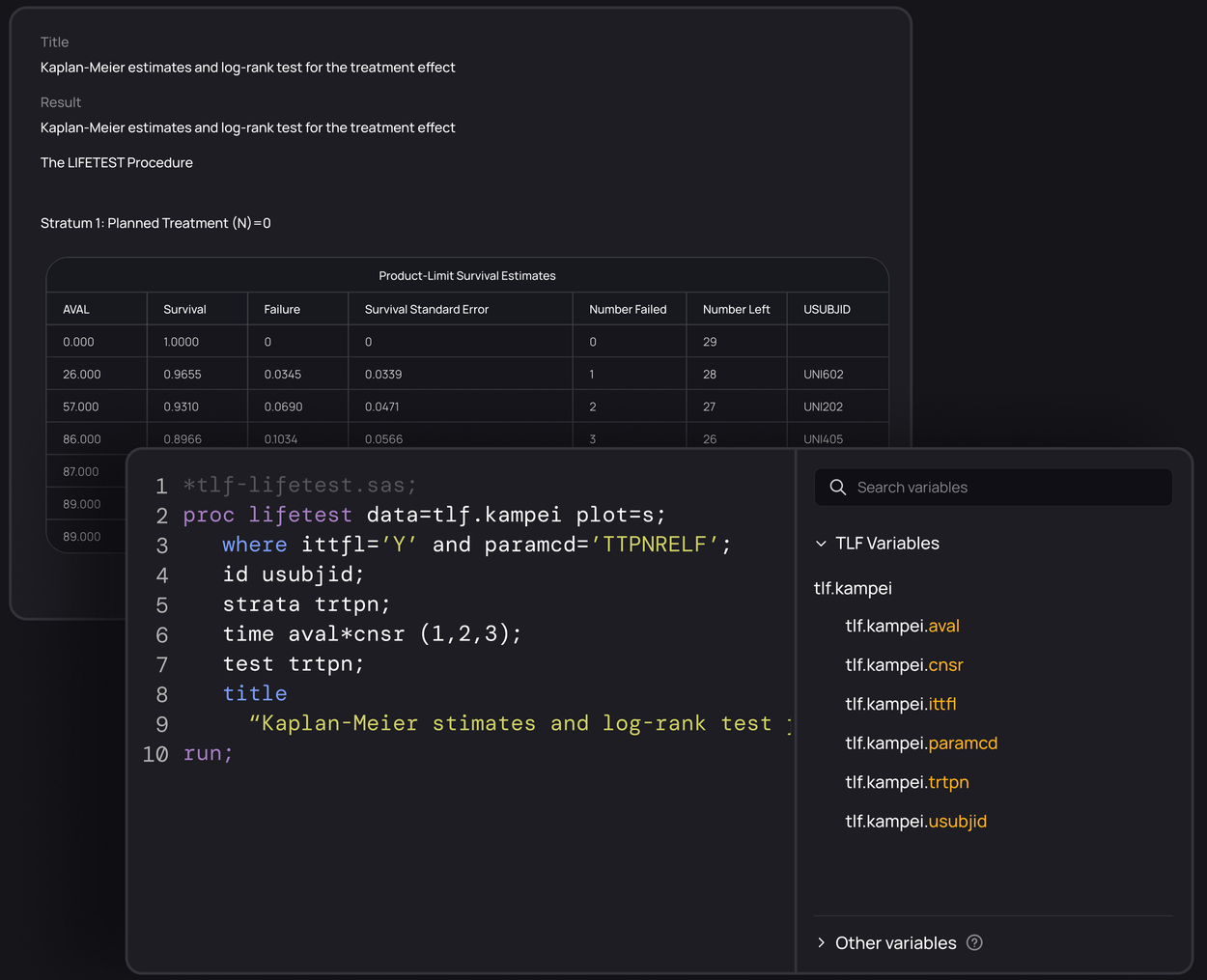
The Validator Workflow



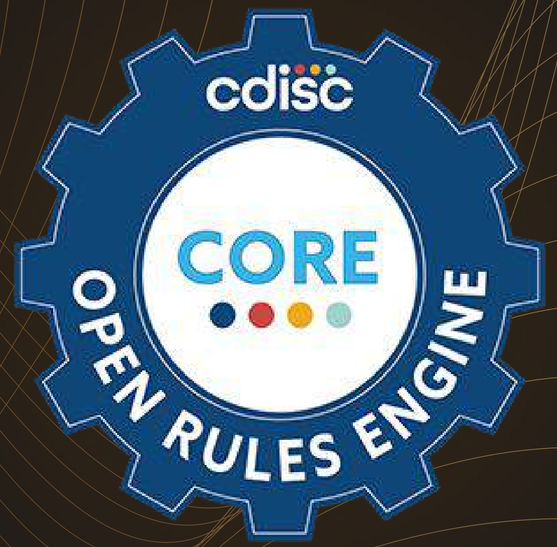
Specification Validation



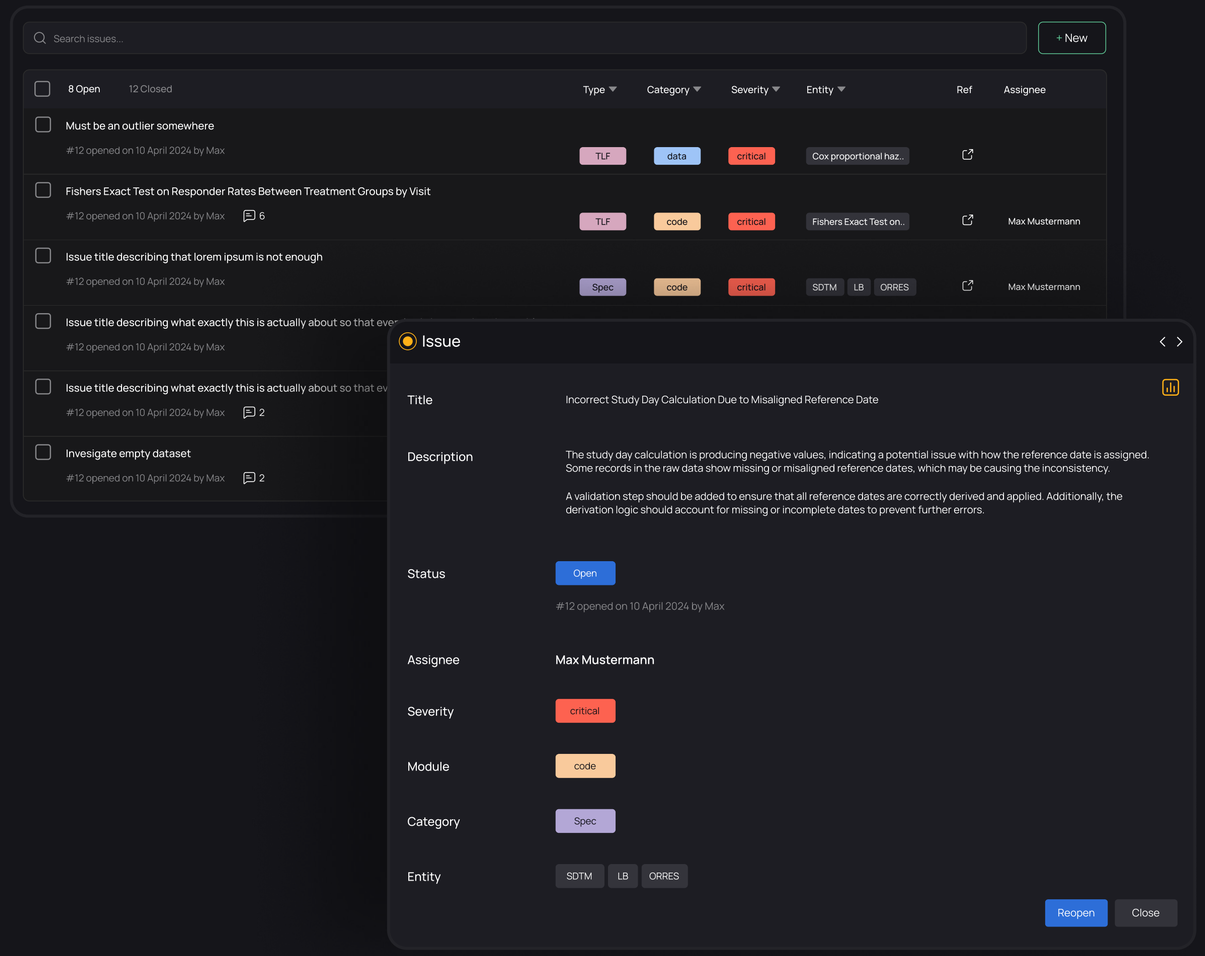
Results Validation



CDISC Core



Validation Tracker



Specification Validation
Validate your SDTM and ADaM implementations variable-by-variable using full study traceability to ensure adherence to the study's SAP and specifications
Results Validation
Validate your results, TLF-by-TLF, by reviewing the TLF, its data and relevant derivations using full study traceability to ensure correctness
Validation Tracker
Track validation issues with full study traceability. Automatically link findings to the exact variable, dataset, or result, ensuring seamless communication, assignment, and resolution across teams
Validated by Verisian

The Impact

40% reduction in programming effort and cost
80%+ faster root cause analysis
Fewer regulatory inquiries
Accelerated onboarding & knowledge transfer

Greater submission confidence

Augmented submissions

Instantly onboard regulators
Full traceability from raw data to results
Fewer inquiries, faster approvals
Legacy documentation becomes obsolete
Confidence for sponsors, regulators, and patients

Validated by Verisian

The Verisian Platform

Builder

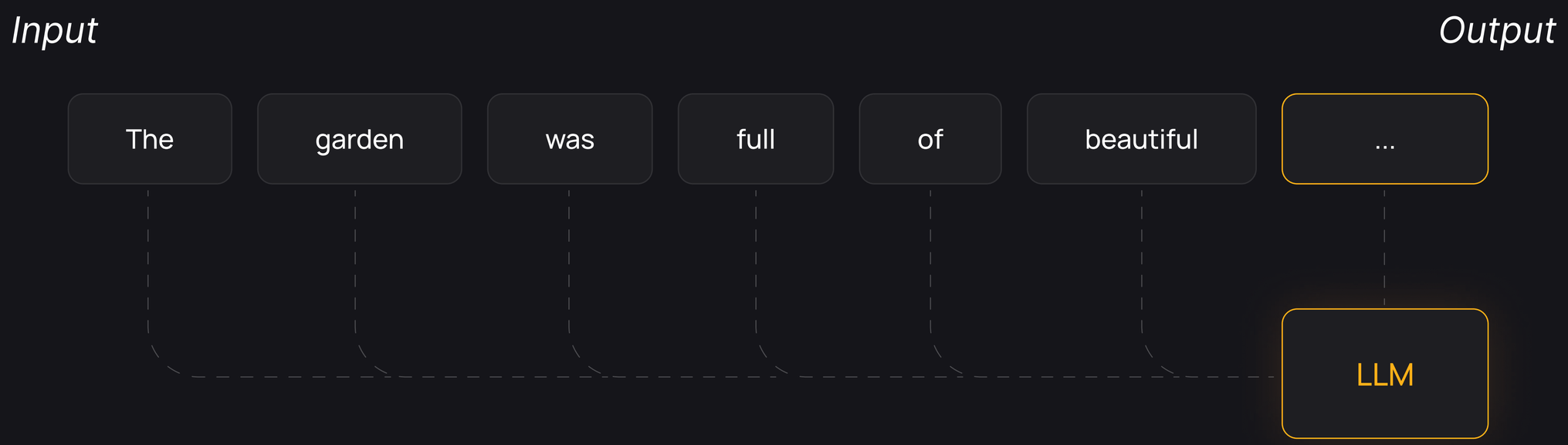
Coming Soon

Validator

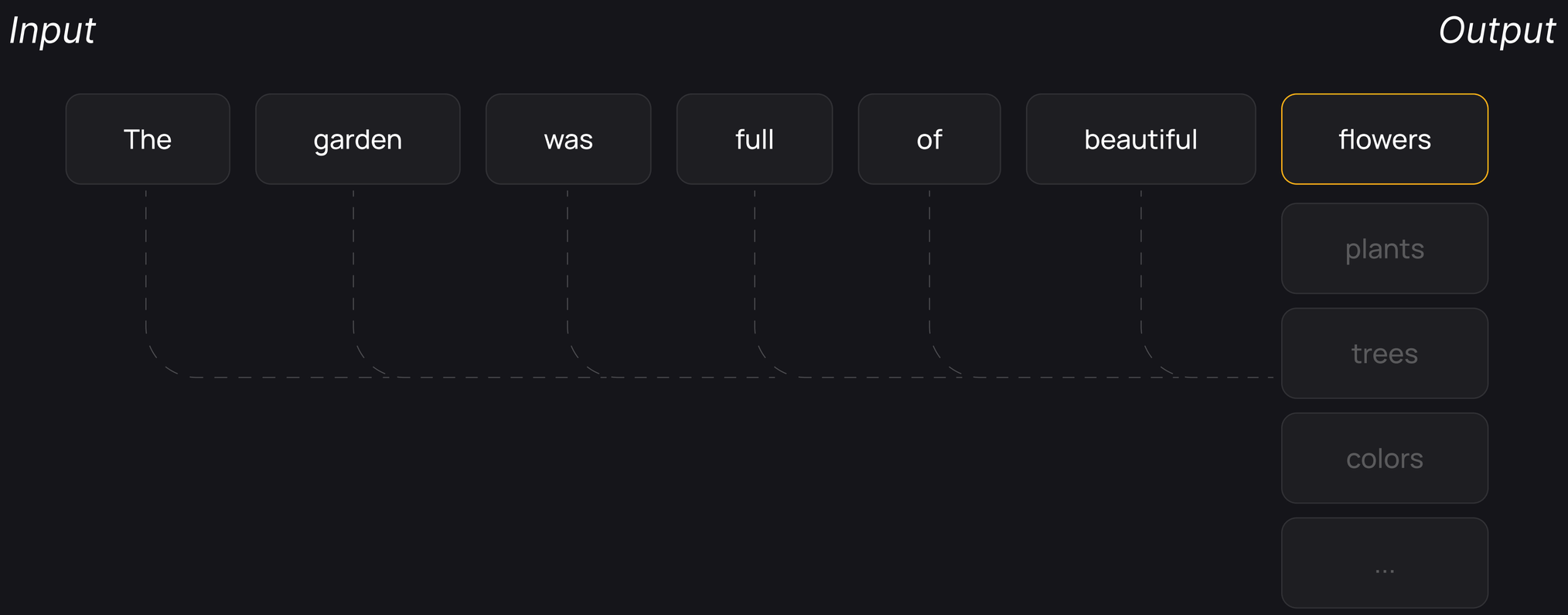
Builder Coming Soon

An Agentic AI SCE

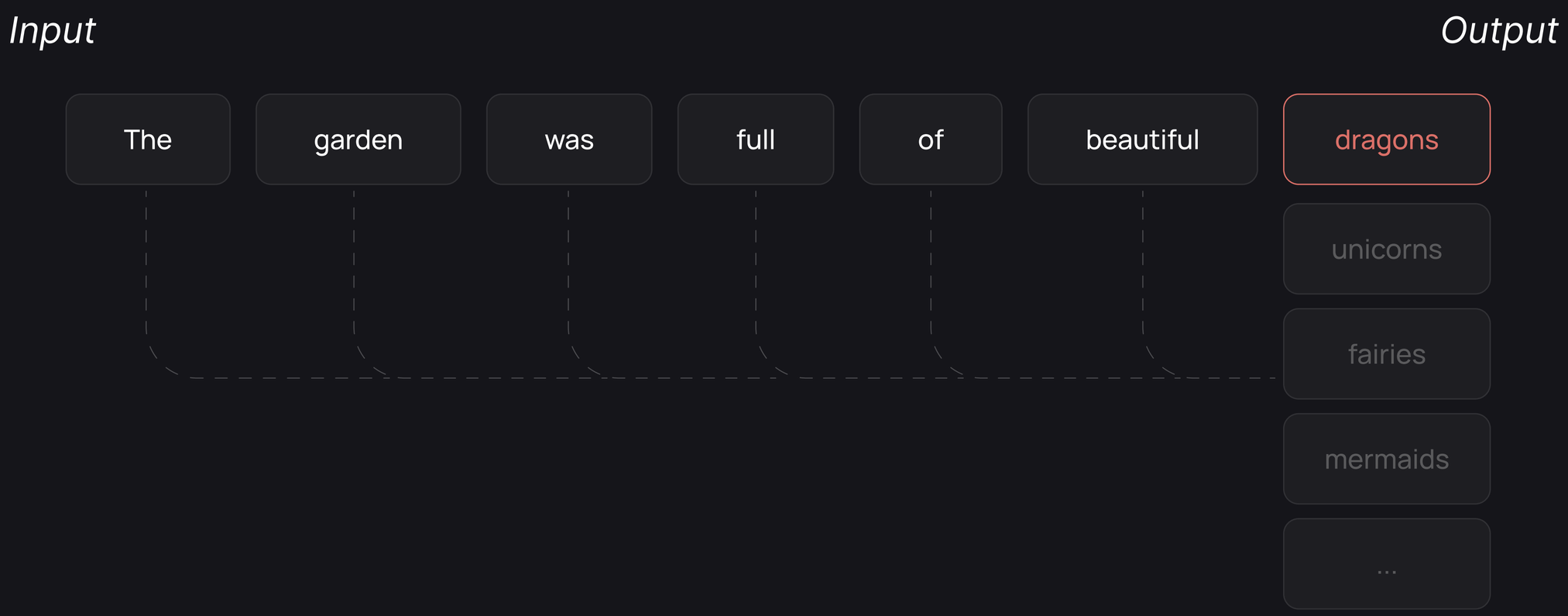
LLMs are really sophisticated autocompletes



LLMs are really sophisticated autocompletes



The bigger the question, the likelier the hallucination



Ethics and Information Technology (2024) 26:38
https://doi.org/10.1007/s10676-024-09775-5

ORIGINAL PAPER

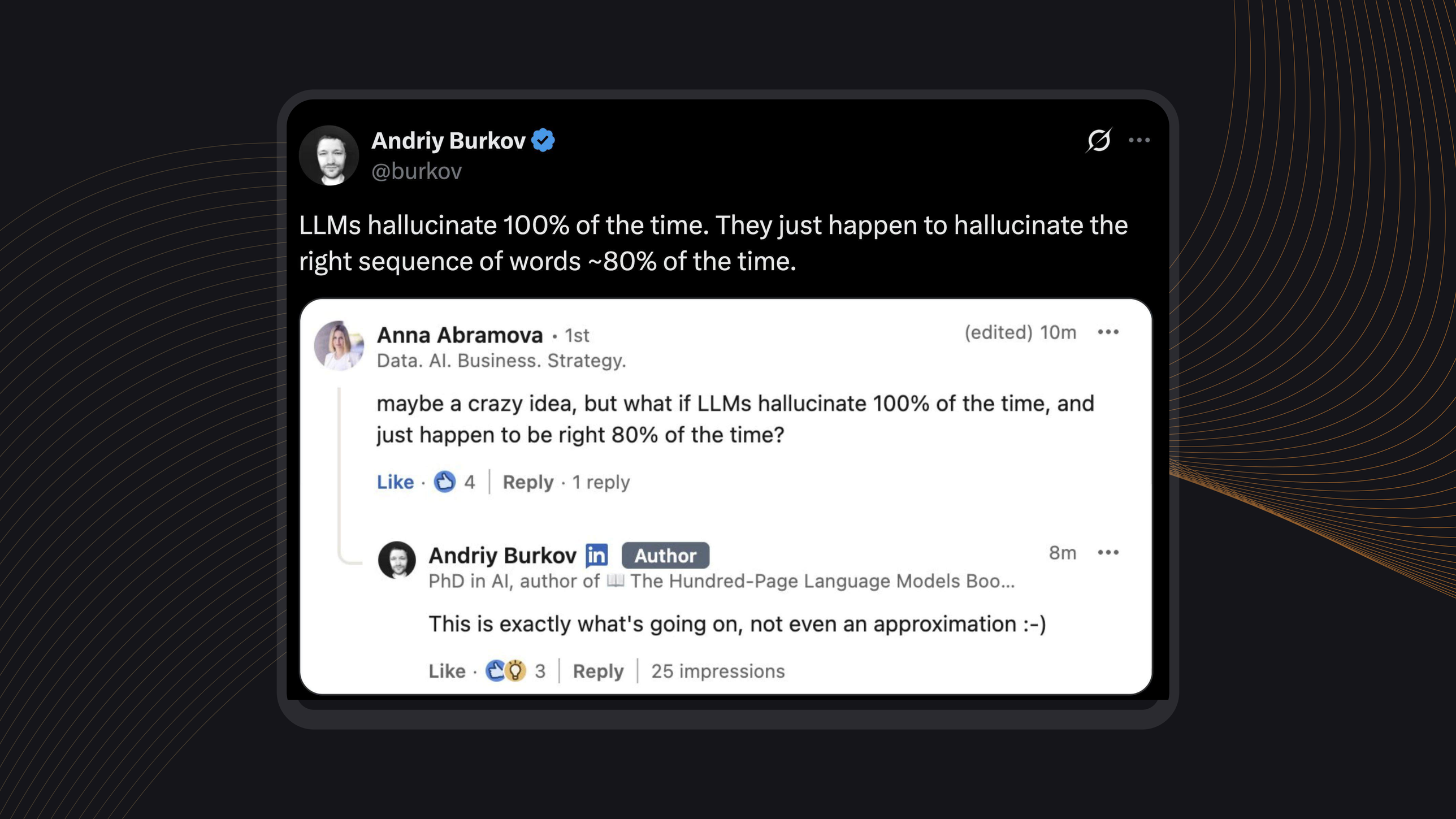
ChatGPT is bullshit

Michael Townsen Hicks[1] · James Humphries[1] · Joe Slater[1]

© The Author(s) 2024

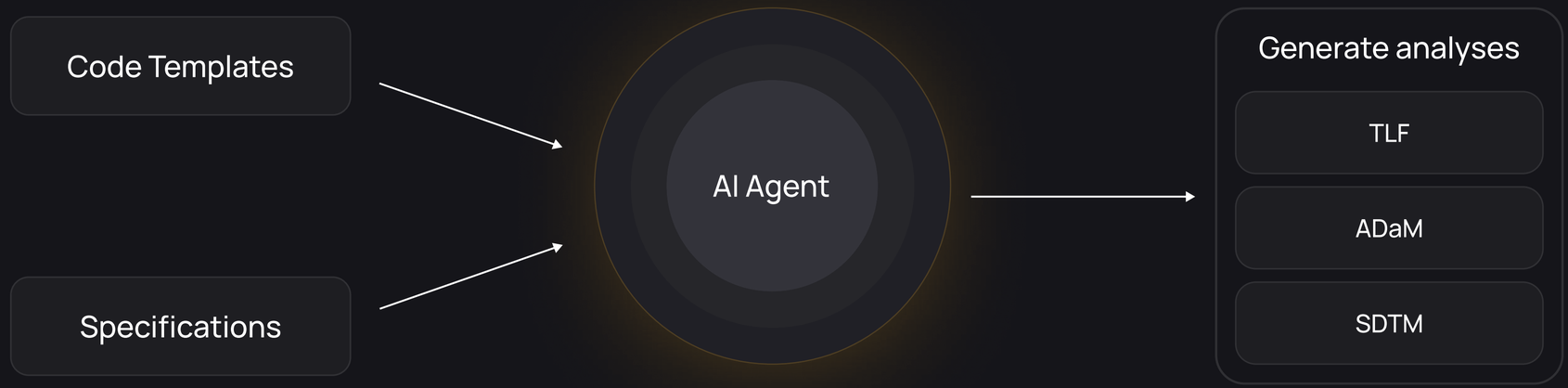
Abstract
Recently, there has been considerable interest in large language models: machine learning systems which produce human-like text and dialogue. Applications of these systems have been plagued by persistent inaccuracies in their output; these are often called "AI hallucinations". We argue that these falsehoods, and the overall activity of large language models, is better understood as *bullshit* in the sense explored by Frankfurt (On Bullshit, Princeton, 2005): the models are in an important way indifferent to the truth of their outputs. We distinguish two ways in which the models can be said to be bullshitters, and argue that they clearly meet at least one of these definitions. We further argue that describing AI misrepresentations as bullshit is both a more useful and more accurate way of predicting and discussing the behaviour of these systems.

Keywords Artificial intelligence · Large language models · LLMs · ChatGPT · Bullshit · Frankfurt · Assertion · Content


Andriy Burkov
@burkov
LLMs hallucinate 100% of the time. They just happen to hallucinate the right sequence of words ~80% of the time.
Anna Abramova · 1st
Data. AI. Business. Strategy.
(edited) 10m
maybe a crazy idea, but what if LLMs hallucinate 100% of the time, and just happen to be right 80% of the time?
Like · 4 | Reply · 1 reply
Andriy Burkov Author
PhD in AI, author of The Hundred-Page Language Models Boo...
8m
This is exactly what's going on, not even an approximation :-)
Like · 3 | Reply | 25 impressions

So how can we gain confidence in AI?

Evidence-based AI with Code Traceability

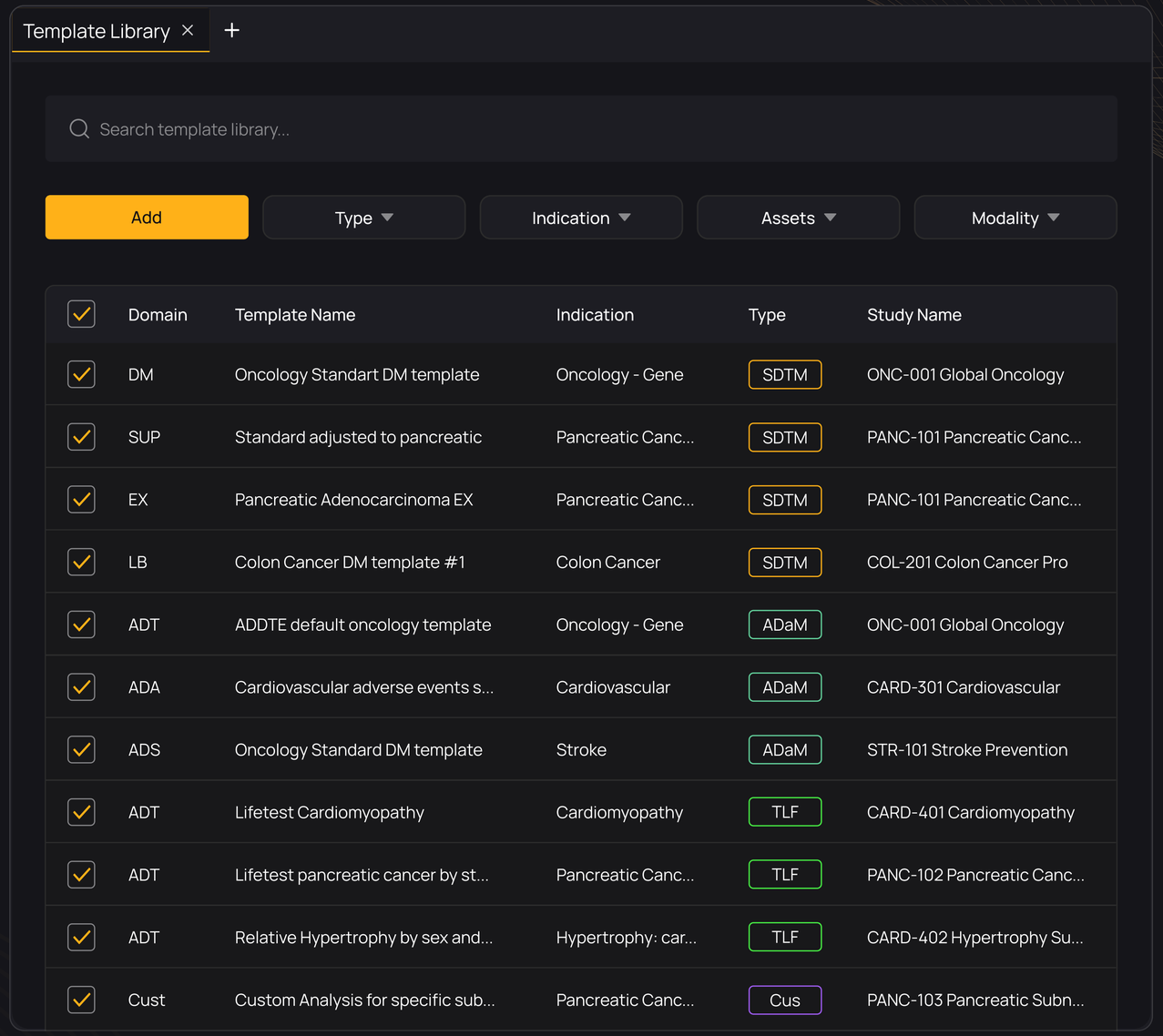


LLM context limited by code templates and specifications

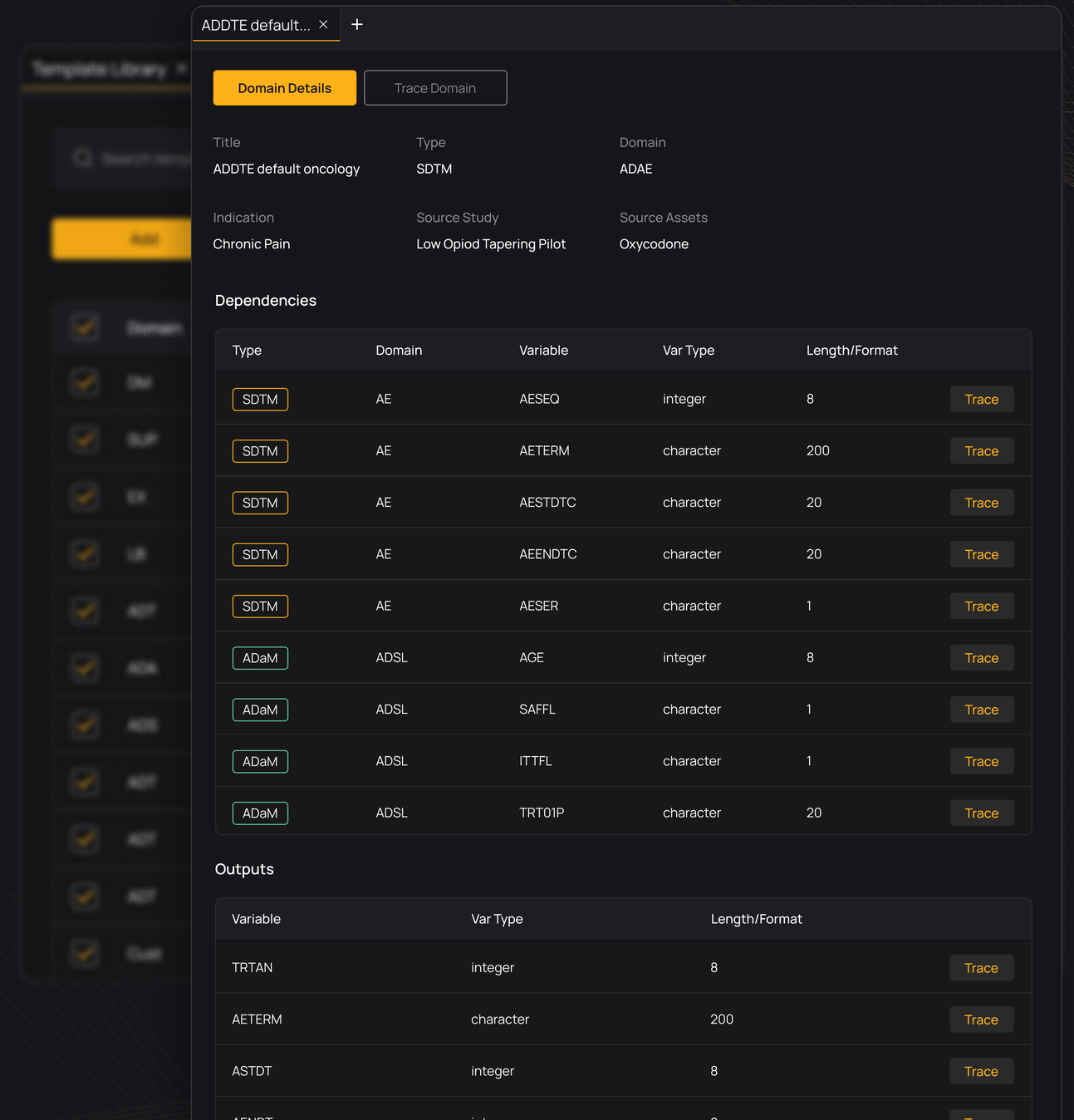
Output is fully reproducible with code traceability

Workflow to create a new analysis

Compare templates from different study templates against specifications



Compare templates from different study templates against specifications



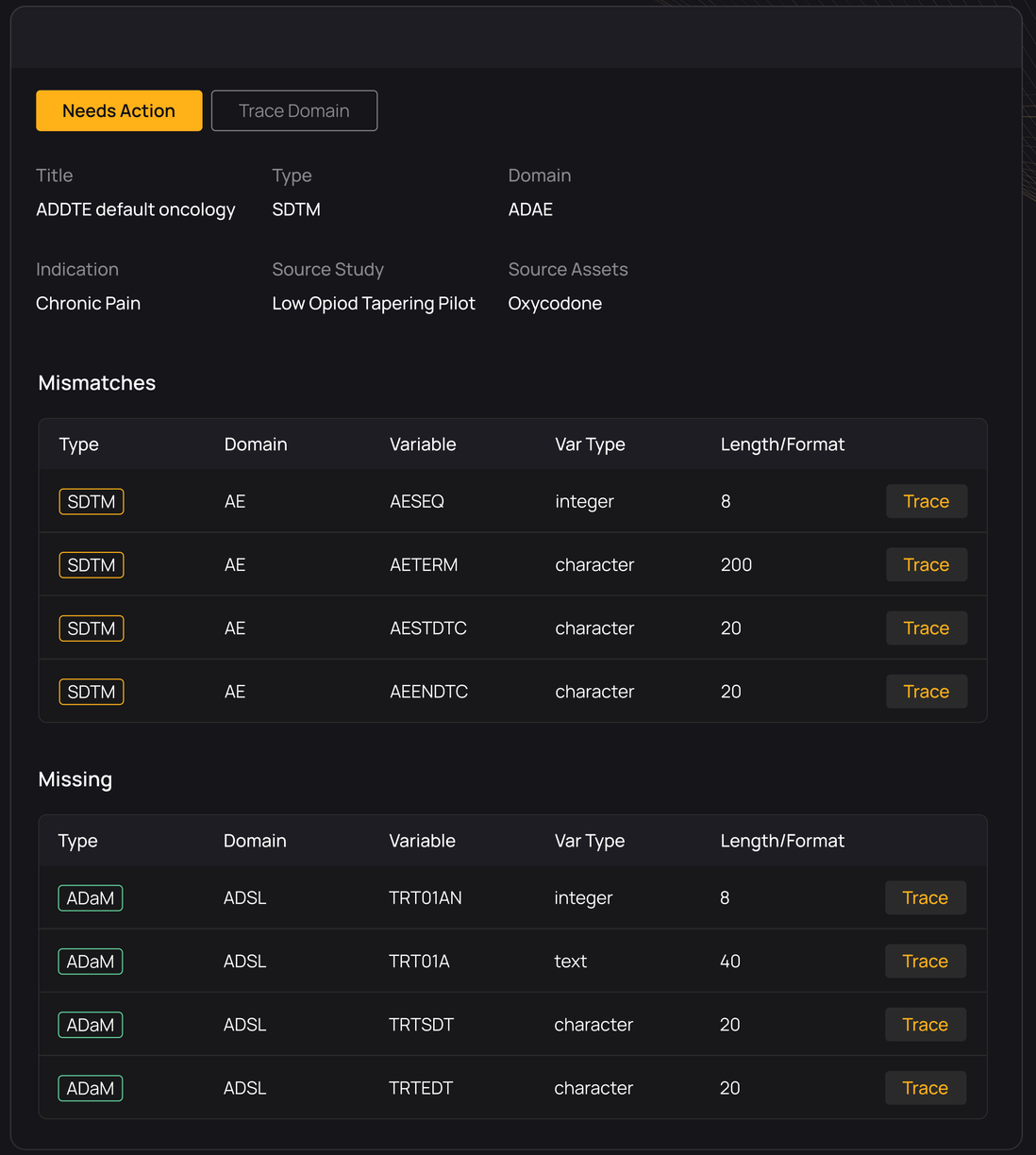
Edit in pipeline-view

ADDTE default... ×

Domain Details | Trace Domain

```
1  /* Create default ADTTE oncology dataset */;
2  data adtte;
3    set adam.adsl; /* Assume starting from ADSL */
4
5    /* Analysis parameters */
6    paramcd = "OS";
7    param = "Overall Survival";
8
9    /* Event/Censor flags */
10   if dthdt ne . then do;
11     aval = dthdt - trtsdt + 1; /* Time from treatment start to death */
12     cnsr = 0; /* Event: death */
13     adt = dthdt;
14   end;
15   else if lastdt ne . then do;
16     aval = lastdt - trtsdt + 1; /* Time from treatment start to last known alive */
17     cnsr = 1; /* Censored */
18     adt = lastdt;
19   end;
20   else do;
21     aval = .;
22     cnsr = .;
23     adt = .;
24   end;
25
26 /* Additional flags */
```

Code traceability identifies missing dependencies or mismatches



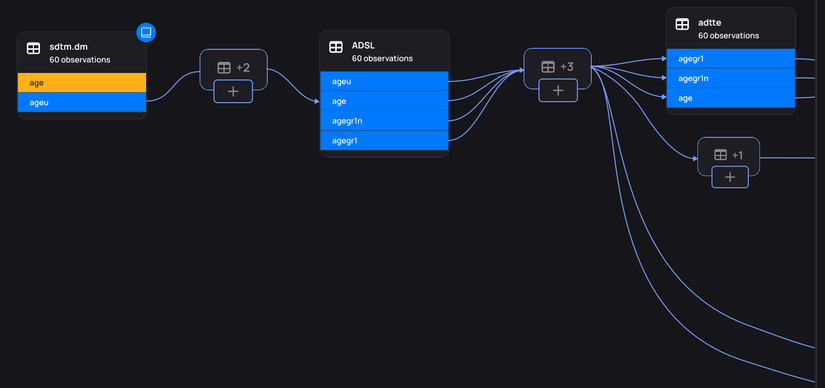
AI agent generates one variable at a time, runs the code, checks logs and outputs, validates itself, and submits to the validator

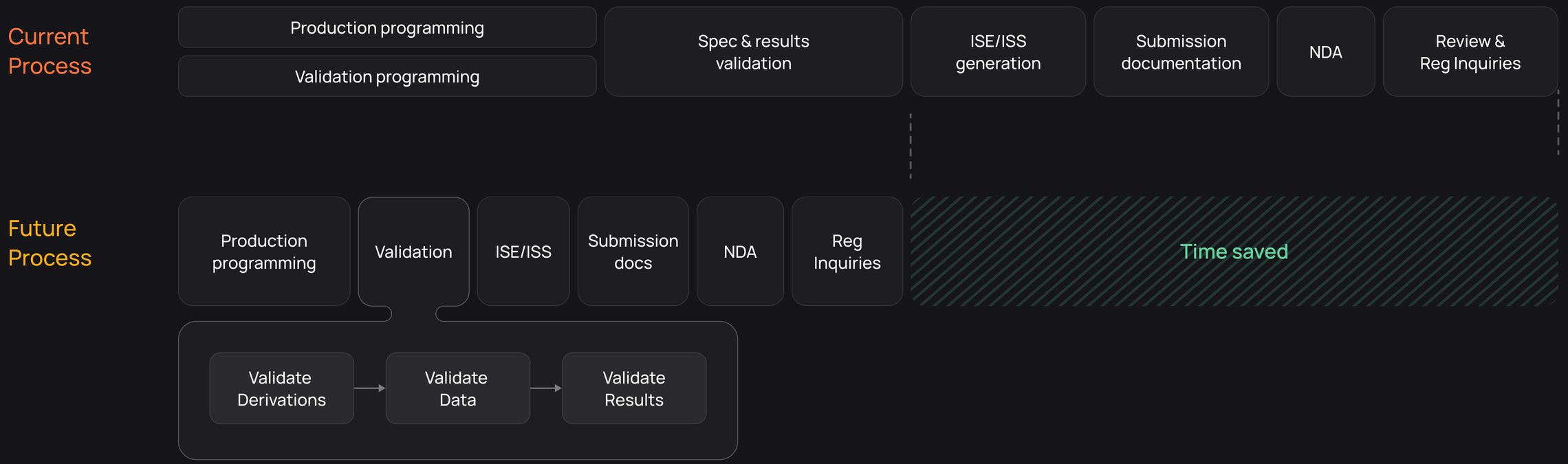
adsl.sas

adam › adsl.sas

```
/* Create ADSL dataset */
data adsl;
  set rawdm; /* Assume rawdm is the input dataset */

  /* Derive treatment arm */
  if armcd in ('PBO', 'PLACEBO') then trt01pn = 0;
  else if armcd = 'TRT' then trt01pn = 1;
+
+  /* Derive treatment label */
+  if trt01pn = 0 then trt01p = 'Placebo';
+  else if trt01pn = 1 then trt01p = 'Treatment';
+  else trt01p = '';

  /* Derive age group */                                   Accept  Reject
  if age < 65 then agegr1 = '<65';
  else if 65 <= age < 75 then agegr1 = '65-74';
  else if age >= 75 then agegr1 = '75+';

  /* Derive study population flags */
  if randt ne . then initt = 'Y'; else initt = 'N';
  if randt ne . and not missing(trt01p) then saffl = 'Y'; else saffl = 'N';
  if randt ne . and not missing(trt01p) and profiev ne 'N' then profl = 'Y'; else profl = 'N';
  if initt = 'Y' and saffl = 'Y' then efffl = 'Y'; else efffl = 'N';
+
+  /* Derive race group */
+  if race in ('WHITE', 'BLACK OR AFRICAN AMERICAN', 'ASIAN') then racegr1 = race;
+  else if race ne '' then racegr1 = 'Other';
+  else racegr1 = 'Missing';
+
+  /* Derive stratification based on gender and age group */
+  if gender = 'M' and age < 65 then strat2 = 'Male <65';
+  else if gender = 'M' and age >= 65 then strat2 = 'Male 65+';
+  else if gender = 'F' and age < 65 then strat2 = 'Female <65';
+  else if gender = 'F' and age >= 65 then strat2 = 'Female 65+';
+  else strat2 = 'Unknown';

  /* Derive treatment start and end dates */               Accept  Reject
  if length(startdt) = 10 then trtstdt = input(startdt, yymmdd10.);
  else if length(startdt) = 7 then trtstdt = input(cats(startdt,'-01'), yymmdd10.);

  if length(enddt) = 10 then trtendt = input(enddt, yymmdd10.);
  else if length(enddt) = 7 then trtendt = input(cats(enddt,'-01'), yymmdd10.);

  format trtstdt trtendt date9.;

  /* Derive baseline flag */
  if visitnum = 1 and aval ne . then basefl = 'Y';
  else basefl = 'N';                                  Reject file  Accept file

  /* Derive death flag */
  if dthdt ne . then dthfl = 'Y'; else dthfl = 'N';
```

Graph




Current Process
Production programming
Validation programming
Spec & results validation
ISE/ISS generation
Submission documentation
NDA
Review & Reg Inquiries
Future Process
Production programming
Validation
ISE/ISS
Submission docs
NDA
Reg Inquiries
Time saved
Validate Derivations
Validate Data
Validate Results

verisian