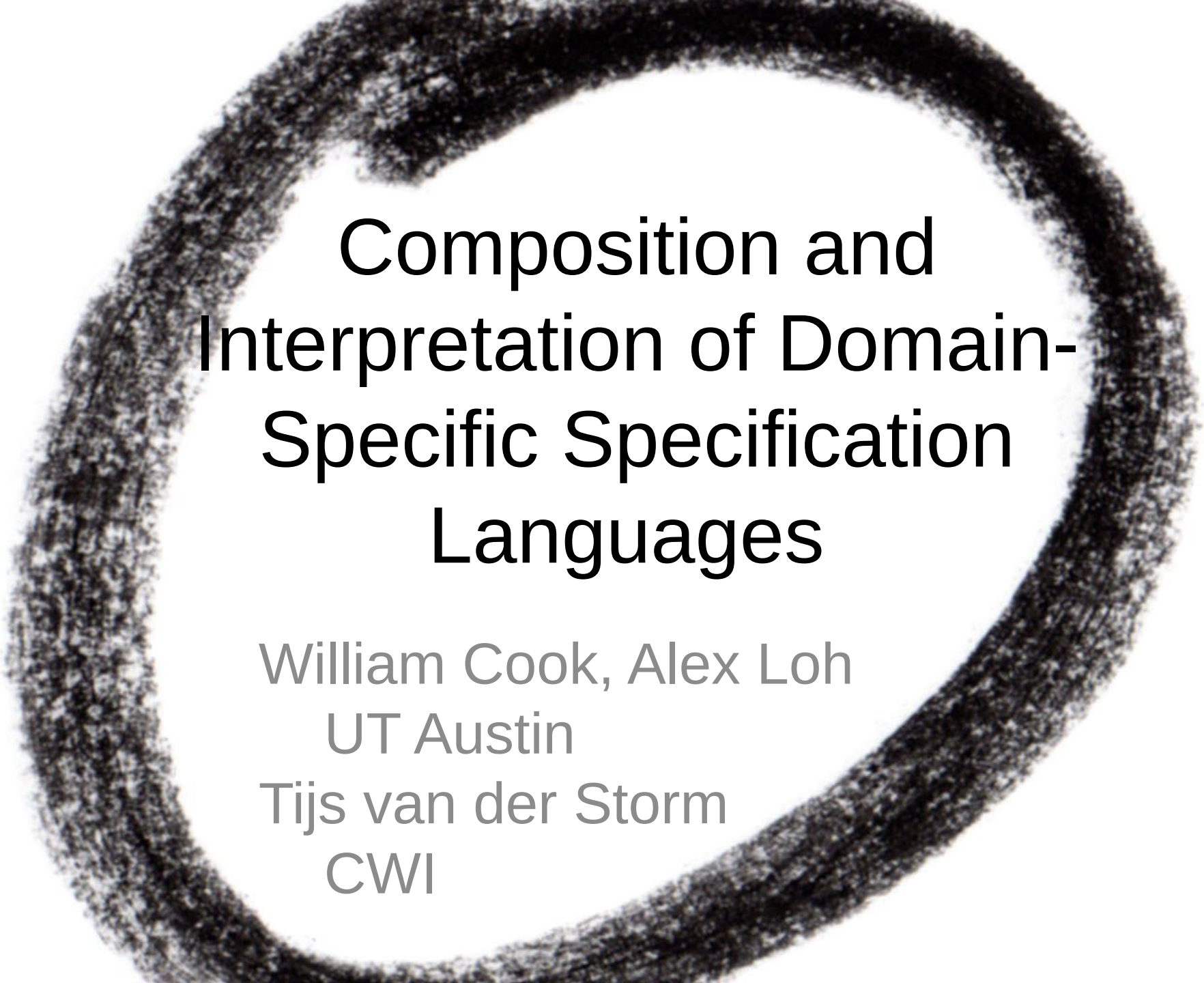


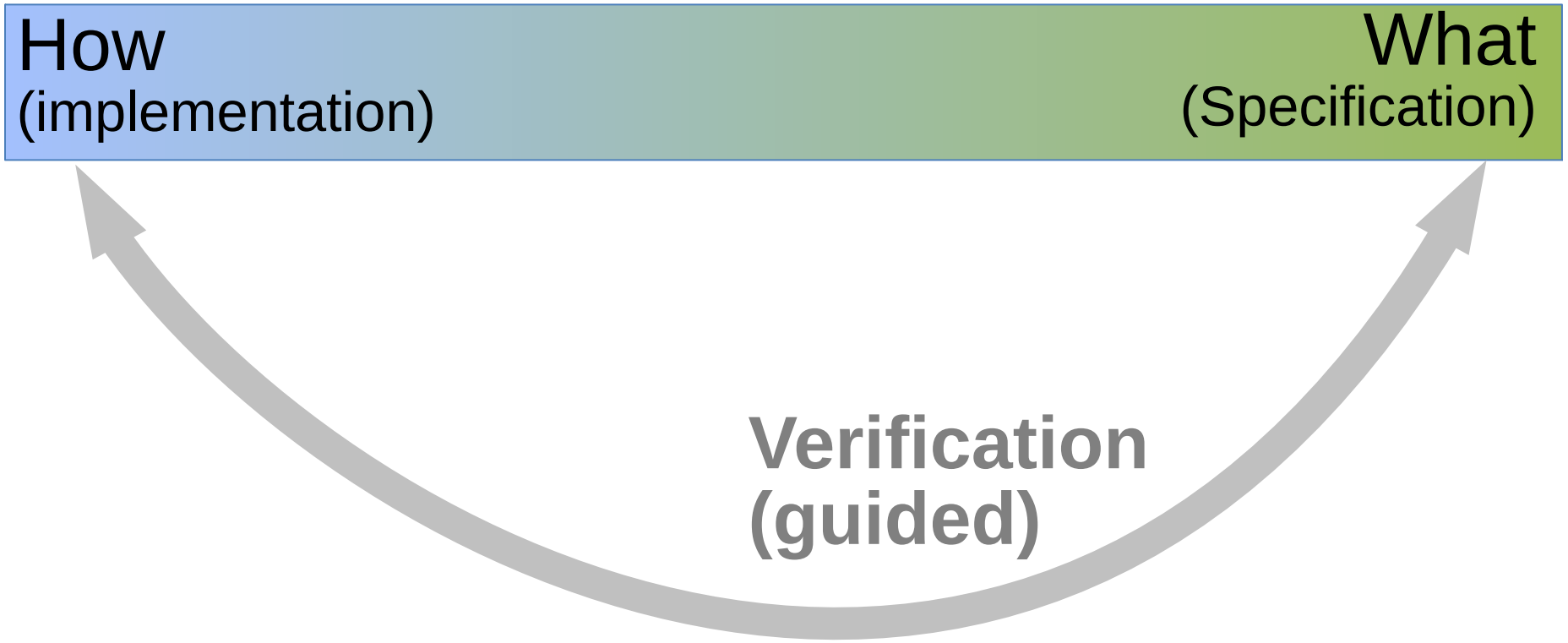
Composition and Interpretation of Domain-Specific Specification Languages

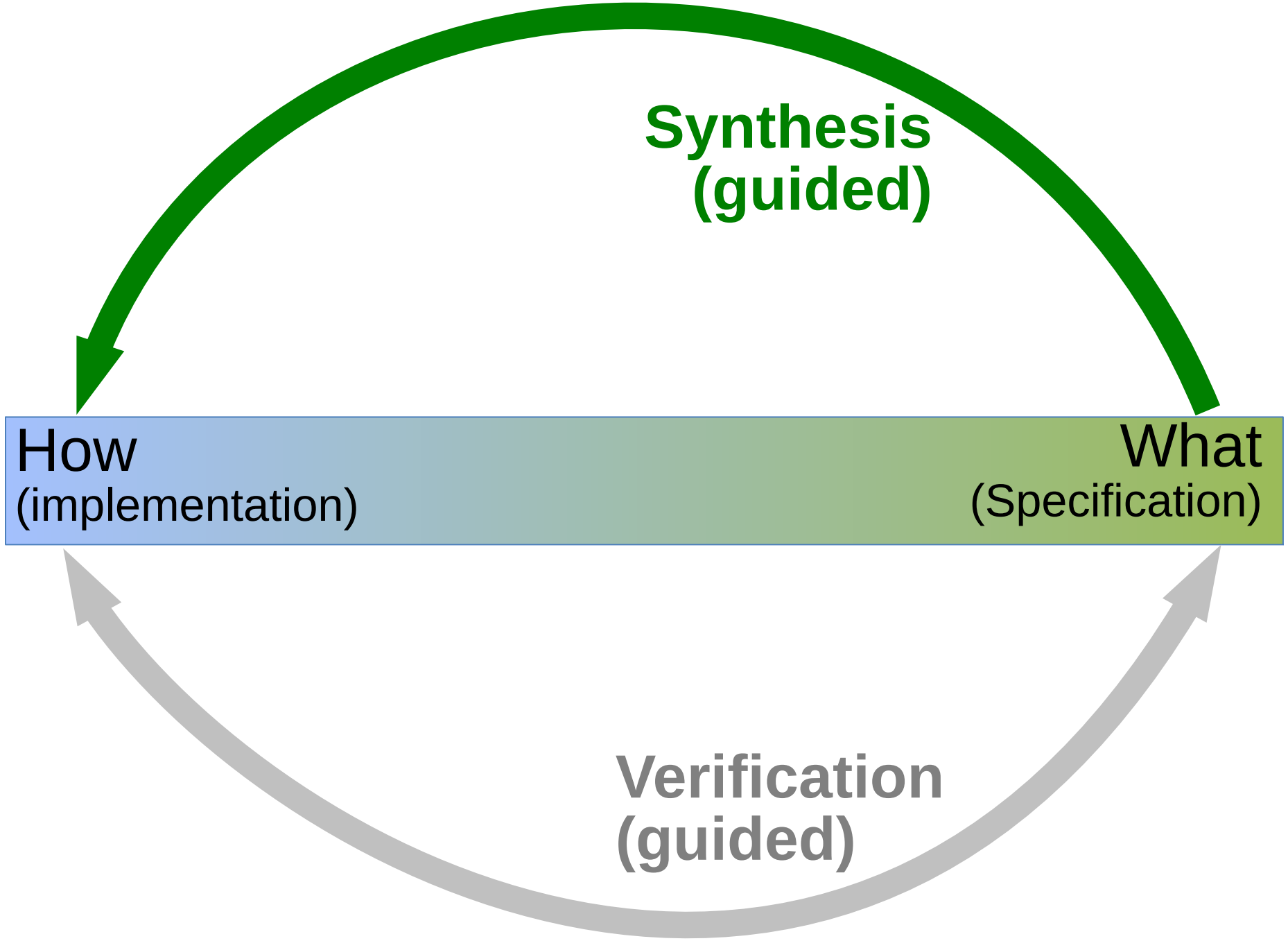
William Cook, Alex Loh
UT Austin
Tijs van der Storm
CWI

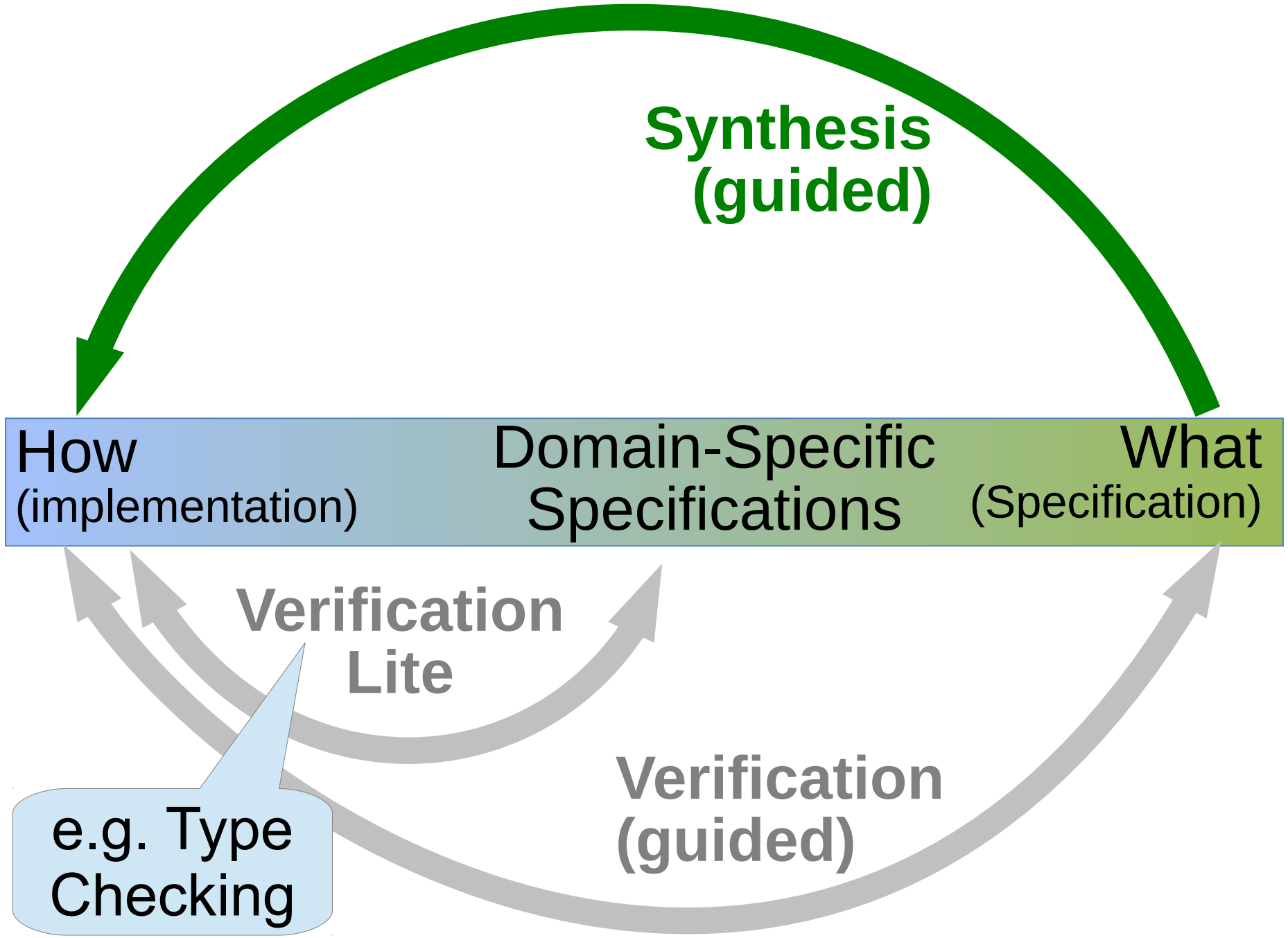
Why?

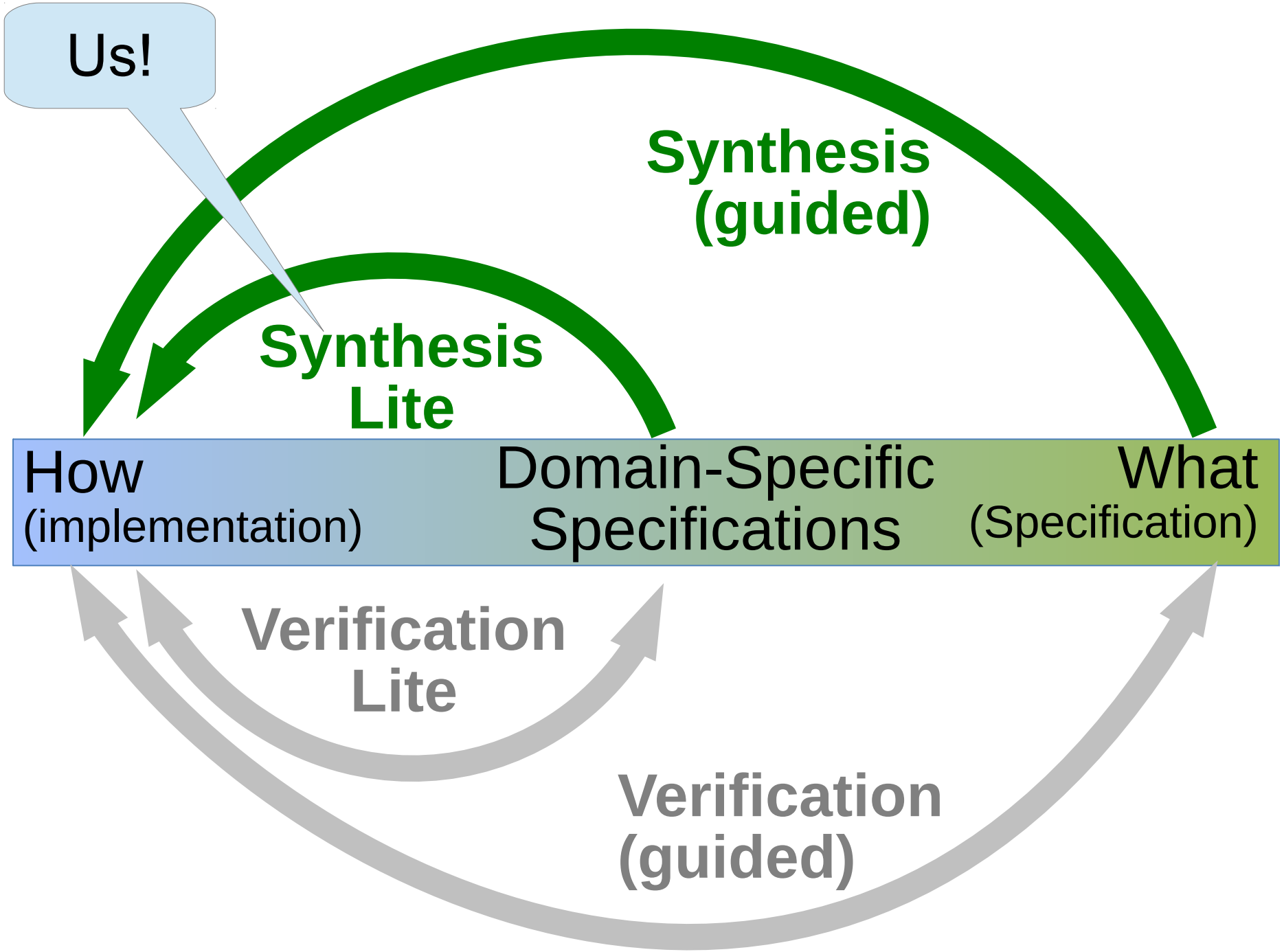
How?

Spectrum of programming


How
(implementation)
What
(Specification)
Verification
(guided)


Synthesis
(guided)
How
(implementation)
What
(Specification)
Verification
(guided)


Synthesis
(guided)
How
(implementation)
Domain-Specific
Specifications
What
(Specification)
Verification
Lite
Verification
(guided)
e.g. Type
Checking


Us!
Synthesis
(guided)
Synthesis
Lite
How
(implementation)
Domain-Specific
Specifications
What
(Specification)
Verification
Lite
Verification
(guided)

Examples

BNF

SQL

ERD

Datalog

Make

Statecharts

Excel

Ndlog

HTML

XUL

XACML

WebDSL

Facets of a system

Data
User Interface
Security
Workflow
Persistence
Grammars

Similar Everywhere

Not the same anywhere

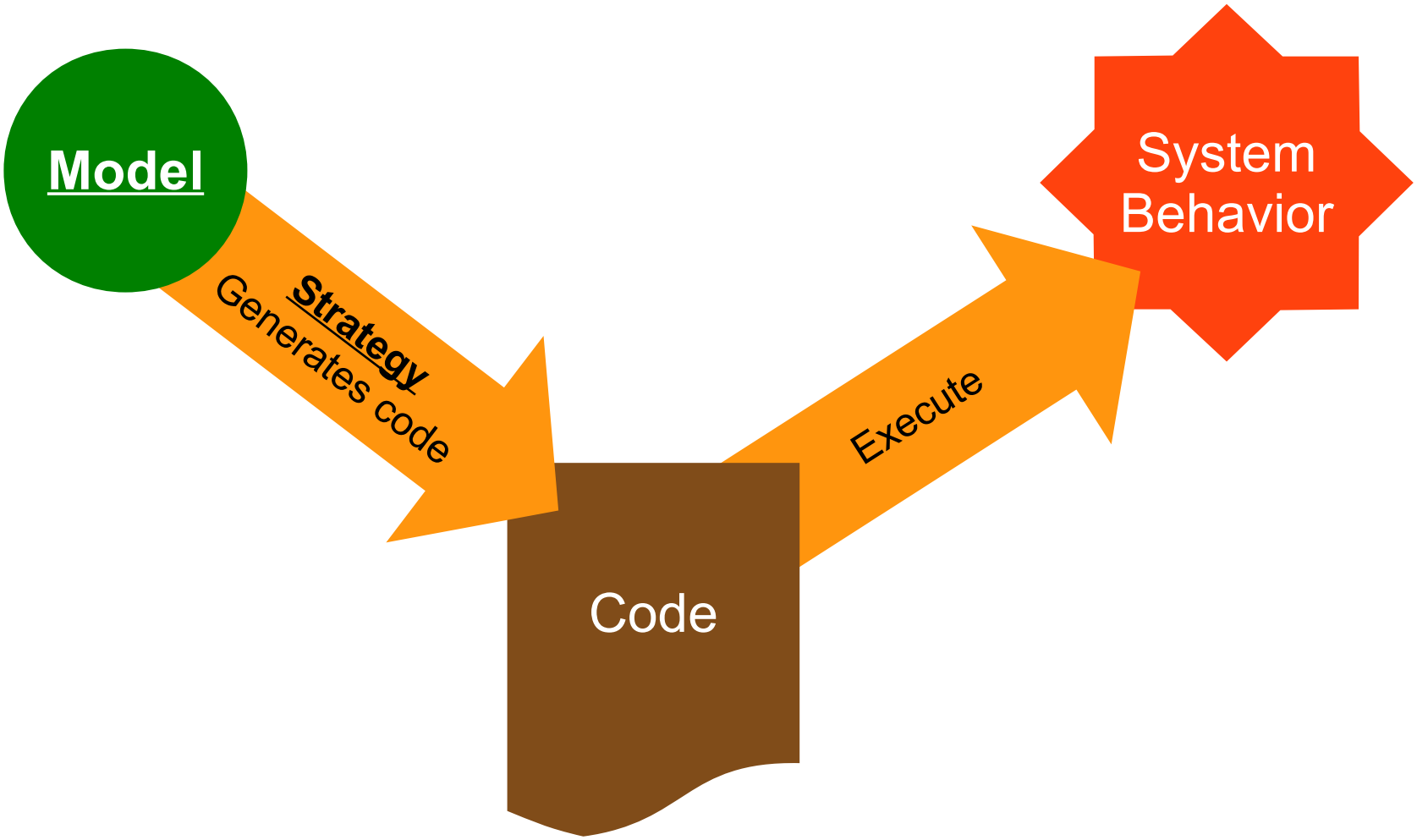
Another Dimension of Modularity

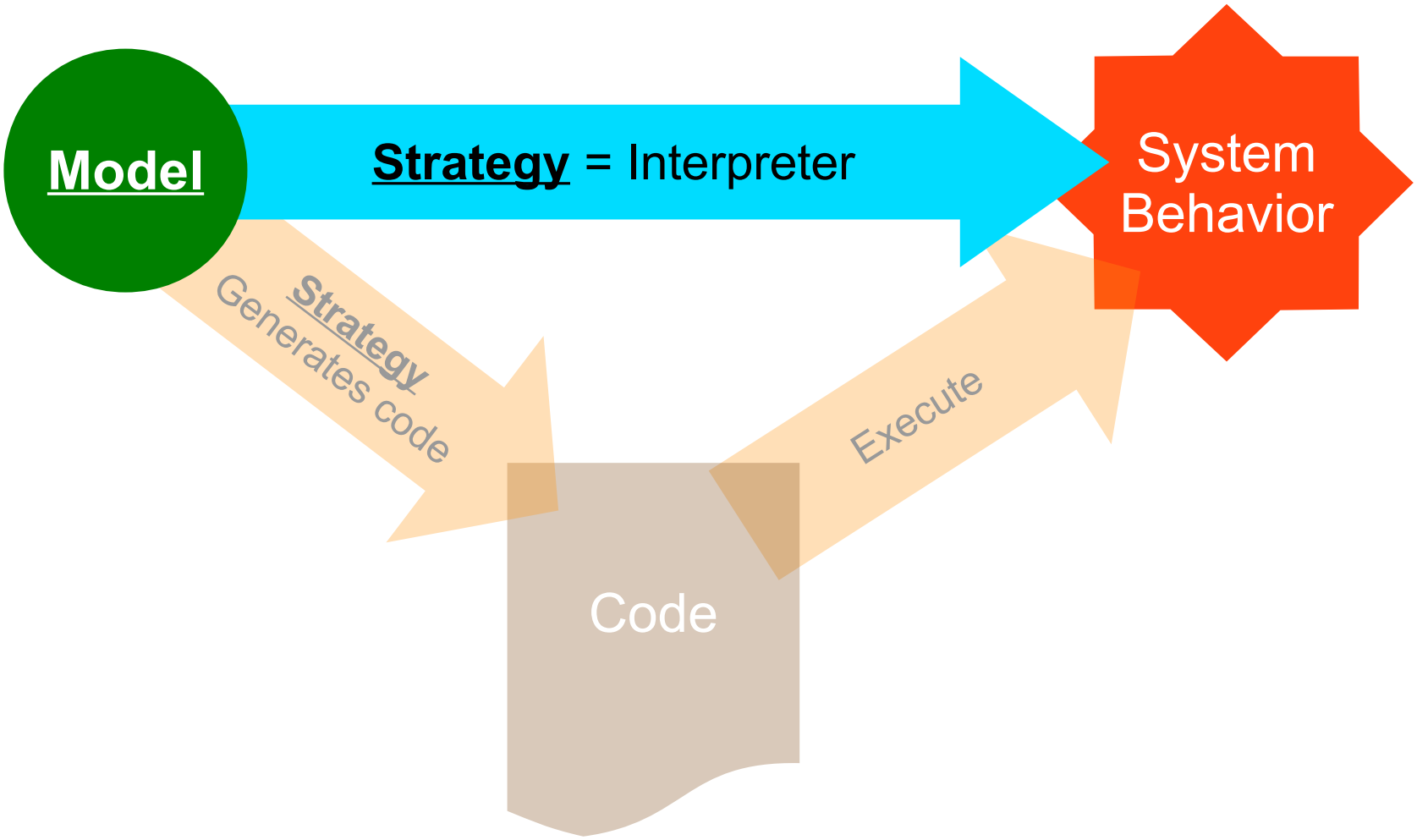
- Traditional:
  - Divide problem into sub-problems
  - Repeat until subproblem is solvable directly
- Model-based:
  - Divide problem into
    - What: specific details for particular problem (i.e. model)
    - How: general strategy
  - Strategy applies to any instances of similar problems

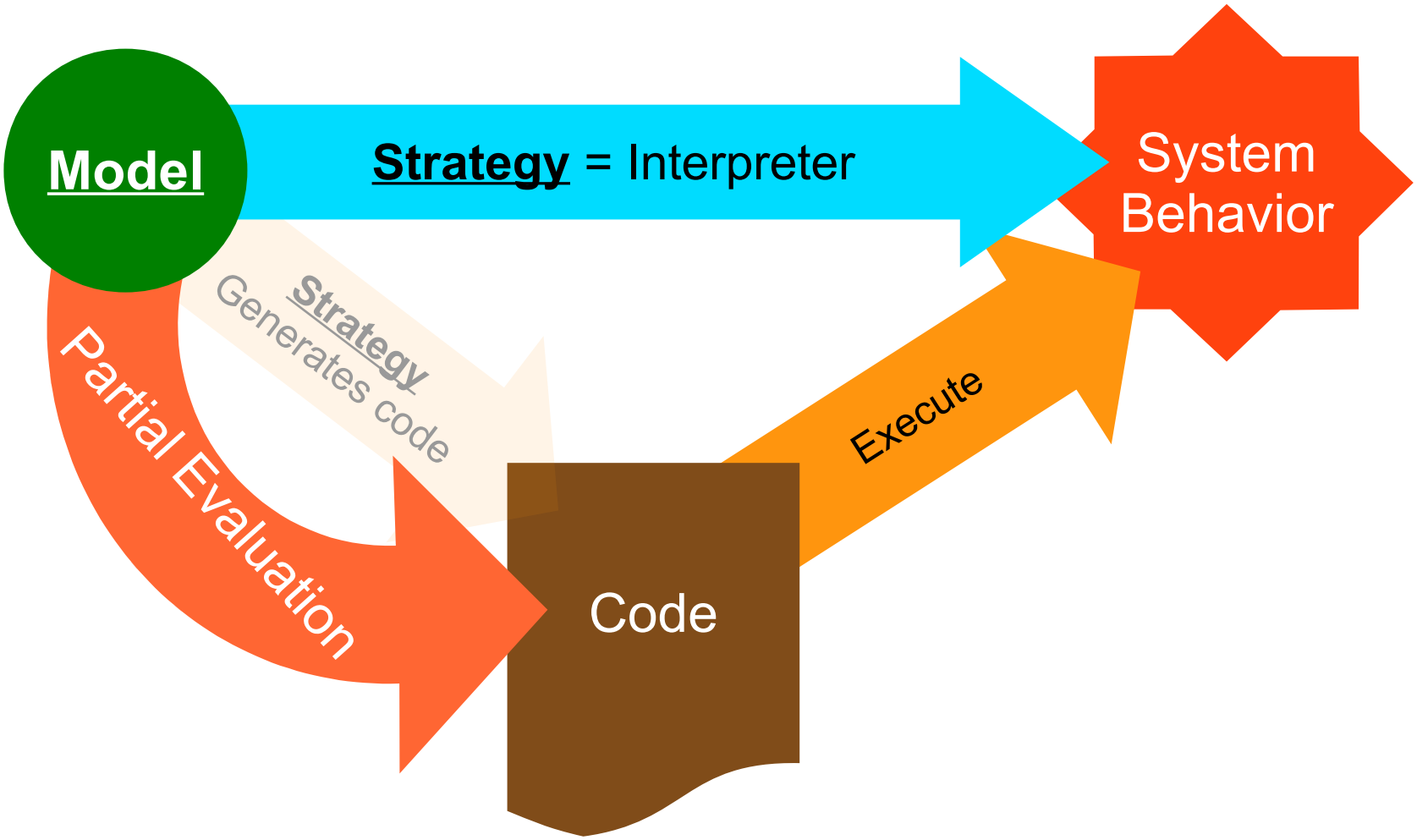
Behavior = Model + Strategy

(different strategies for each facet of system)

System = Behavior^{n}


Model
Strategy
Generates code
Code
Execute
System
Behavior


Model
Strategy = Interpreter
System Behavior
Strategy
Generates code
Code
Execute


Model
Strategy = Interpreter
System Behavior
Strategy
Generates code
Partial Evaluation
Execute
Code

Multiple DSLs

- Combining Behaviors
  - Combine interpreters
    - Call another interpreter through API
  - Combine generated code
    - Generated code must invoke other generated code?

Modifying Strategies

- Interpreter Inheritance
  - If interpreter is OO, easy!
- Generic interpreter wrapper
  - Interpreter *mixin*: Debugging

- Aspect = interpreter modification

- Partial evaluation *weaves* the interpreter fragments together

Embedded versus External

- Embedded: Reuse the host language
  - e.g. expressions
  - External: include expression sub-language
- Other issues
  - Debugging
  - Optimization
  - Embedding that ends up being “external”

Summary

- Executable Specification Languages
  - Data, grammar, GUI, Web, Security, Queries,...
- External DSLs (not embedded)
- Interpreters (not compilers/model transform)
- Composition of Languages/Interpreters
  - Reuse, extension, derivation (inheritance)

Don't Design
Your Programs

Program
Your Designs

Ensō
enso-lang.org