

Model Harness

Models as a foundation
for development and execution

Domain-level descriptions
for AI development

Software as models

Structure, rules, state machines

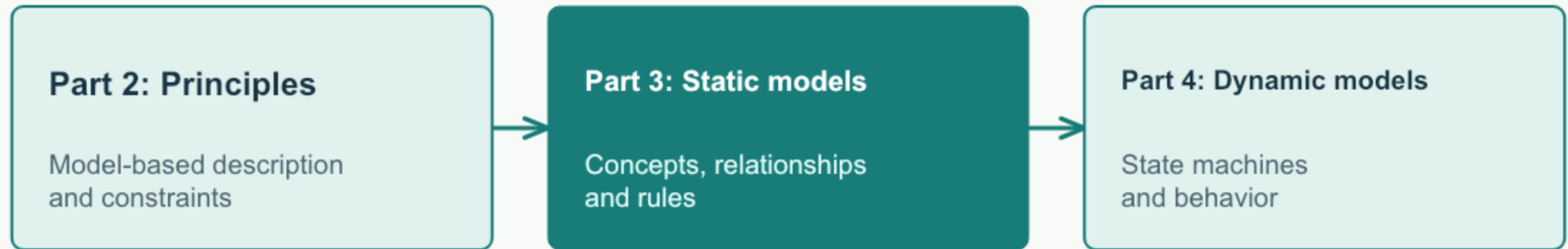


Checking and execution

Constraints on AI changes

Place in the Series

Parts 2–4 cover the principles, static structure and dynamic behavior of the Model Harness.



Parts 5–7 cover architecture, programming and runtime. Parts 8–10 cover knowledge, observation and verification.

Models Linked by a Glossary

Model-driven development from the preceding series supports AI development.

Literate Model

Requirements, context
and decisions

Object Model

Formal structure
and behavior

Knowledge Model

Semantic relationships,
grounds and sources

The glossary manages names, meanings and identities to maintain correspondence among models.

Software Close to Domain Knowledge

Models reduce the distance between business concepts and program representations.

Description in CML

Concepts, relationships
and rules

Execution in Textus

Run executable
models directly

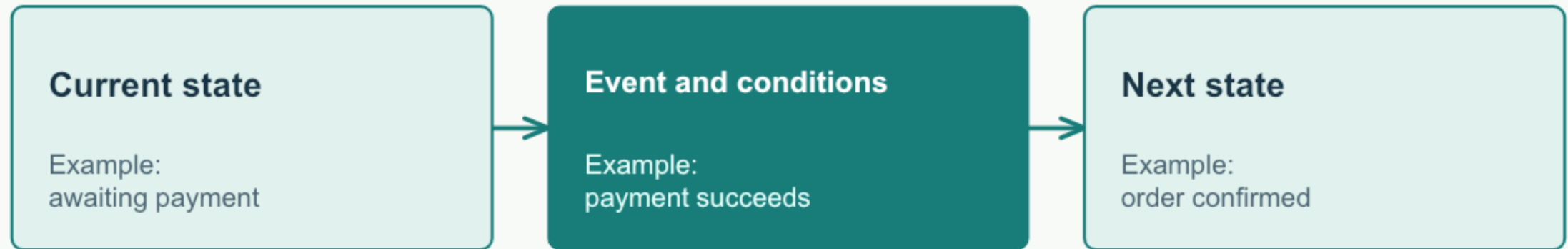
AI implementation

Supply what the model
alone does not provide

Cozy transforms models. Complementary implementation must reflect their structure, rules and behavior.

Behavior as State Machines

Explicit states and permitted transitions give people, AI and processors a shared structure.



Business rules also describe failure transitions and the handling of stock reservations.

Mechanisms of the Model Harness

Review, static checking and direct execution constrain generated models and changes.

Semantic review

Glossary and
textus-cbd-support

Build-time checks

Syntax, references
and constraints

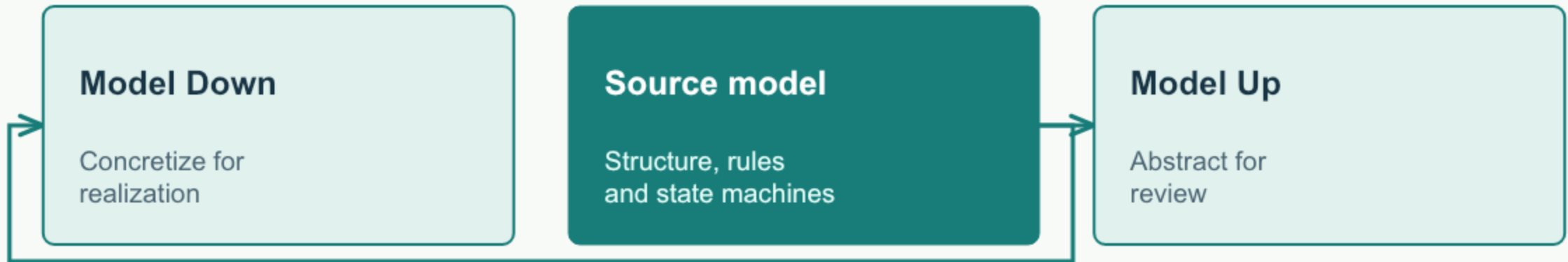
Direct execution

Textus applies
the model's rules

Review covers executable models and Use Cases. People judge meaning and adoption.

Model Up/Down

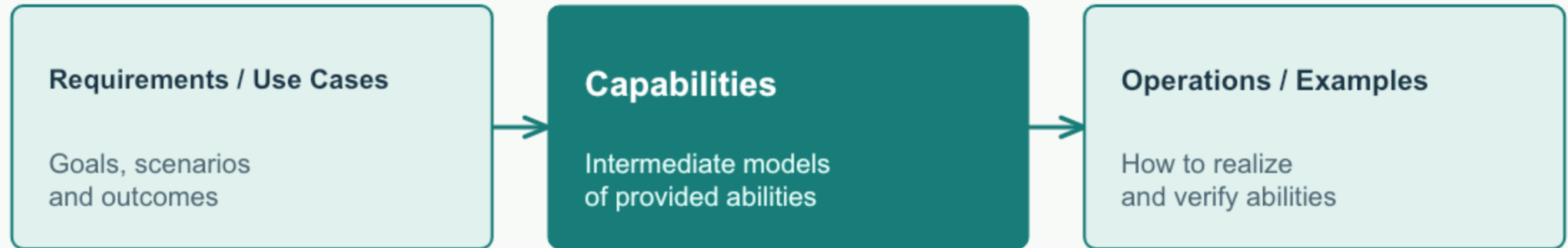
Models become concrete for realization and abstract for a specific review purpose.



SimpleModeling concept. Model Up: state machine to flow diagram, or Use Case Model to actor-goal list.

Requirements, Realization and Verification

Retained mappings support checks of both formal consistency and fitness for business goals.



Separate shared domain rules from application-specific flows to assess where changes belong and their impact.

Models and Execution Results

Structure checks and business judgment assess clarity, consistency, validity and accuracy.



Performance, reliability and security require checks across requirements, models and execution.

Describing Software as Models

Domain-oriented models for development and execution

Model structure and behavior

Static: concepts, relationships, rules

Dynamic: states, events, transitions

Constrain AI-generated changes

Model reviews and static checks

Textus executes models directly

Check purpose through requirements

Link requirements to realization

Compare rules, examples, and results

Model Up/Down

Down: concretize toward realization

Up: abstract models for review

Example: state machine to flow diagram

Next: concepts, relationships, and rules

Model Up/Down is specific to SimpleModeling