AEA by Ivory & Co.

Domain Architecture Standard

Version: v1.0.0 — Published Anchor: Scope

1. Definition

> Domain Architecture defines the business's value logic and the stable capabilities required to realise it.

Domain Architecture is the architecture of Scope.

Its governing question is:

> What business are we architecting, how does it create economic value, and what stable business abilities must exist for that value logic to operate?

Its core structure is:

> Business Model !’ Value Chain !’ Capability

Where the enterprise operates more than one Business Model, Enterprise Domain is formed from the combined Business Models, their Value Chains and the consolidated Capability Model required across them.

2. Why Domain Architecture Exists

An enterprise needs a stable definition of what business it is actually capable of being.

Strategic direction can change.

Processes can be redesigned.

People can change.

Technology can be replaced.

Products and offers can evolve.

But beneath those changes the enterprise requires a coherent description of:

- the mechanisms through which it creates economic value;
- the value paths through which those mechanisms operate;
- and the stable business abilities required to make them possible.

Domain Architecture establishes that scope.

It prevents the enterprise from confusing current processes, organisation structures, systems, products or projects with the enduring abilities upon which the business depends.

> Domain Architecture establishes the stable business scope against which the rest of the enterprise can be designed.

3. Business Model

A Business Model is a distinct mechanism through which the business creates economic value and generates revenue and profit.

A Business Model is not merely a customer segment, product line, legal entity or sales channel.

It represents a materially distinct economic mechanism.

A useful architectural approximation is:

Business Model "H Revenue Mechanism × Customer Relationship Type × Recurrence

This is a reasoning aid rather than a mandatory mathematical formula.

At minimum, a Business Model should establish:

· Revenue Mechanism;
· Customer Relationship Type;
· Recurrence;
· and its Associated Value Chain.

The detailed information specification belongs downstream.

4. Revenue Mechanism

Revenue Mechanism describes how economic value is monetised.

Established examples include:

· Subscription;
· One-off Sale;
· Retainer;
· Transaction;
· Usage;
· Licensing;
· Gainshare;
· Commission;
· Marketplace;
· Hybrid.

This list is not necessarily closed.

The purpose of Revenue Mechanism is to distinguish materially different economic models, not to create labels for every pricing variation.

A change in price does not automatically create a different Business Model.

A materially different mechanism through which revenue and profit are generated may.

5. Customer Relationship Type

Customer Relationship Type describes the broad commercial relationship through which the Business Model participates with its customer or beneficiary.

Examples may include:

· B2B;
· B2C;
· DTC;
· B2B2C;
· or another materially distinct relationship type.

No closed taxonomy is imposed by this Published Standard.

Customer Relationship Type contributes to Business Model identity where the relationship materially changes the economic mechanism or associated value logic.

The detailed Actor and Relationship architecture remains Ecosystem-owned.

Business Model may classify the commercial relationship type without acquiring ownership of the Ecosystem Relationship.

6. Recurrence

Recurrence describes the pattern through which the economic exchange repeats or persists.

It may distinguish recurring, non-recurring, transactional, usage-based or other materially relevant patterns.

Recurrence contributes to understanding Business Model identity.

It should not be confused with the recurrence of operational work, which belongs to Execution.

7. Value Chain

A Value Chain is the connected path through which a Business Model sources, creates, exchanges, delivers, captures and retains value.

Every material Business Model has exactly one Value Chain.

The Value Chain expresses the economic and value logic of that Business Model rather than a detailed operational process sequence.

A useful derivation lens is:

Source / Supply / Inputs !’ Value Creation !’ Offer / Exchange !’ Pricing / Commercialisation !’ Conversion !’ Delivery !’ Revenue Capture !’ Customer / Beneficiary Outcome !’ Retention / Continuity

This is not a mandatory process flow.

Not every Business Model will express every element in the same way or at the same resolution.

The Value Chain exists to expose the connected value path, not to prescribe how work is executed.

8. One Business Model, One Value Chain

Each material Business Model has one associated Value Chain.

This establishes a stable architectural relationship between the economic mechanism and the value path through which it operates.

Where two apparently separate offerings operate through the same economic mechanism and connected value logic, they may remain within one Business Model.

Where materially different economic mechanisms require distinct value logic, separate Business Models may be required.

The test is architectural rather than organisational.

Different product does not automatically mean different Business Model. Different team does not automatically mean different Business Model.

9. Enterprise Domain

The Enterprise Domain is the combined Domain Architecture of the enterprise.

It is formed from:

- all material Business Models;
- the Value Chain associated with each Business Model;
- and the consolidated Capability Model required across them.

Where several Business Models require the same Capability, the Capability is represented once in the consolidated Capability Model with appropriate traceability.

Enterprise Domain is not a duplication of capabilities by Business Model. It is the coherent consolidation of the stable abilities required across the enterprise's value logic.

10. Capability

A Capability is a stable business ability required for the enterprise to realise its Business Models and value logic.

A Capability describes what the enterprise must be able to do.

It does not describe how that ability is currently performed.

A Capability should remain meaningful if:

- the process changes;
- the technology changes;
- the people change;
- the organisation structure changes;
- the sourcing model changes;
- or the particular product manifestation changes.

This stability is central to Capability identity.

11. What a Capability Is Not

A Capability is not:

- a Process;
- Workflow;
- Task;
- Procedure;
- SOP;
- Role;
- Skill;
- Job;
- Team;
- Organisation Unit;
- System;
- Application;
- AI Agent;
- Product;
- Service;
- Offer;
- Project;
- Initiative;
- or Business Model.

Those objects may enable, fulfil, express, change or depend upon a Capability.

They do not become the Capability itself.

> Capability identity must survive changes in fulfilment and implementation.

12. Capability Decomposition

Capabilities may be decomposed to expose the stable abilities required within a broader Capability.

There is no mandatory universal hierarchy depth.

Decomposition stops when the object ceases to describe a stable business ability and begins describing how work is performed or another architecture's object.

The governing test is:

> Is this still a stable business ability, or have we crossed into process, task, role, skill, system, offer or another architectural proposition?

If the answer has crossed into another proposition, Capability decomposition should stop.

13. Capability Role

A Capability may be classified by its role in the enterprise's value logic.

The established Capability Roles are:

- Value-Chain Capability
- Enabling Capability

A Value-Chain Capability participates directly in the connected value path of one or more Business Models.

An Enabling Capability enables the enterprise and its Value-Chain Capabilities to operate effectively without itself forming the primary value path for a Business Model.

This distinction is architectural, not a judgement of importance.

An Enabling Capability may be critical to enterprise survival.

14. Capability Posture

Capability Posture expresses the intended strategic posture of a Capability.

The established postures are:

- Differentiating
- Standard

A Differentiating Capability is one whose distinctive performance or configuration is intended to contribute materially to the enterprise's Competitive Advantage.

A Standard Capability is one that must perform effectively but is not intended to be a primary source of Competitive Advantage.

Directional Architecture owns the intended basis of Competitive Advantage.

Domain determines which Capabilities must differentiate in order to realise that intended advantage.

> Directional defines the intended basis of advantage. Domain identifies the Capabilities that must differentiate to support it.

Older classifications such as Compete, Maintain and Exit concern investment, evolution or change posture rather than stable Capability identity and are not treated as the canonical Capability Posture taxonomy in this Standard.

15. Business Value Anchors

Capabilities exist because they contribute to enterprise value.

A Business Value Anchor identifies the form of business value to which a Capability materially contributes.

Relevant value may include:

- revenue;
- profit;
- margin;
- cash flow;
- customer outcome;
- goodwill;
- Trust;
- resilience;
- strategic option value;
- long-term enterprise value;
- or another materially established form of value.

This is not necessarily a closed taxonomy.

Business Value Anchors prevent Capability Architecture from becoming detached from the economic and enterprise value logic it exists to support.

16. Capability Outcome

Each Capability exists to produce or sustain an outcome appropriate to that stable ability.

The governing performance question is:

> Is the Capability producing the outcome for which it exists?

This is different from asking whether the underlying work is being performed correctly.

Execution Operational Metrics ask:

> Is the work being performed as it is intended to be performed?

Domain Capability measures ask whether the stable business ability is producing its intended outcome.

> Operational performance and Capability outcome are related but different propositions.

17. Capability Measures

Domain Architecture owns the meaning of measures used to assess whether a Capability is producing the outcome for which it exists.

A Capability measure may require information, evidence or derivation.

Intelligence may architect that fulfilment or own a derivation where applicable.

Reality captures the actual observed or derived value.

The governing separation is:

> Domain defines the Capability measure !’ Intelligence may establish the derivation !’ Reality captures the actual value.

Measure definition and measured Actuality are different architectural propositions.

A Capability does not become Reality-owned because its state or performance is measured.

18. Capability State and Maturity

Domain owns the identity and meaning of any Capability state or maturity model established for the Capability.

Reality captures the actual state or maturity value at a point in time.

Directional may establish the state the Capability needs to reach.

Possibility may explore what the Capability could become.

Other architectures may contribute evidence or conditions relevant to assessment.

The governing rule is:

> Capability identity and state model belong to Domain. Actual Capability state belongs to Reality.

Capability maturity is therefore not a separate architecture.

It emerges from assessment of the Domain Capability using relevant architectural evidence.

19. Capability Policy

A Capability Policy is a durable standing rule governing how a specific Capability may or may not be exercised.

Capability Policy belongs to Domain where the rule remains valid despite redesign of the operational mechanism.

The governing test is:

> Would the rule still be valid if the operational process, tooling, people and sequence were redesigned tomorrow?

If yes, the rule may be a Capability Policy.

If no, it is likely an Execution Operational Policy or another architecture's rule.

> Durable rule about the ability belongs to Domain. Rule about how current work is performed belongs to Execution.

20. Capability Charter

A Capability Charter is a Domain convergence point through which the core architectural proposition of a Capability can be brought together with relevant cross-architecture references.

It may provide a governed view of matters such as:

- Capability identity;
- purpose;
- outcome;
- Business Model and Value Chain traceability;

- Capability Role;
- Capability Posture;
- Business Value Anchors;
- Capability Policies;
- measures;
- and references to relevant propositions from other AEA architectures.

The Capability Charter does not transfer ownership of those referenced propositions to Domain.

Its detailed schema belongs in the downstream Domain Architecture package.

21. Capability Change

The introduction, retirement or material redefinition of a business Capability constitutes a Domain Architecture change.

Because Capabilities are foundational to the enterprise's ability to realise its value logic, a material Capability change triggers impact review across all eleven AEA architectures.

> Capability change triggers review across the Architecture Set. Review does not presume that every architecture must change. It establishes whether change is required.

A process redesign does not automatically constitute Capability change.

A technology replacement does not automatically constitute Capability change.

A Role reassignment does not automatically constitute Capability change.

The test is whether the stable business ability itself has been introduced, retired or materially redefined.

22. Capability Sourcing and Fulfilment

Capability identity remains Domain-owned regardless of how the Capability is fulfilled.

Workforce Architecture may determine that a Capability is sourced through a Build, Buy or Partner route.

Where the sourcing unit is Capability, the enterprise is sourcing the stable business ability from another organisation.

Ecosystem may establish the provider Actor and Relationship.

Execution may define retained or interfacing work.

Environment may define relevant enablement.

Reality captures actual performance.

Trust may determine warranted Reliance upon the provider.

> External fulfilment does not transfer Capability identity out of Domain.

The old concept of Capability Deployment is retired as a single mixed classification because it conflated sourcing, fulfilment, automation and orchestration propositions now owned across Workforce, Environment, Execution and Ecosystem.

23. Capability and Workforce Sourcing

The current Workforce sourcing model separates:

Sourcing Route

- Build
- Buy
- Partner

from:

Sourcing Unit

- Capability
- Execution Outcome
- Labour
- Skill

Domain participates when the sourced unit is a Capability because the Capability identity remains Domain-owned.

Domain does not own the sourcing route merely because a Capability is being sourced.

Domain defines the Capability. Workforce determines how participation or the relevant sourcing unit is obtained.

This separation allows the enterprise to source the same Capability through different fulfilment arrangements without redefining the Capability.

24. Domain and Directional Architecture

Directional Architecture defines the strategic scope and choices the business must be designed to realise.

Domain Architecture defines the value logic and stable Capabilities required to realise those choices.

Directional may establish:

- strategic scope;
- commercial choices;
- Competitive Advantage;
- growth choices;
- Route-to-Market choices;
- strategic trade-offs;
- and other directional commitments.

Domain translates the implications of those choices into Business Model, Value Chain and Capability architecture where appropriate.

Directional establishes strategic choice. Domain establishes the stable business scope required to realise it.

Directional does not own the Capability Model.

Domain does not independently choose enterprise strategy merely because it reveals Capability implications.

25. Domain and Competitive Advantage

Directional owns the intended basis of Competitive Advantage.

Domain owns Capability Posture.

The interlock is:

> Directional — What basis of advantage are we choosing?
>
> Domain — Which Capabilities must differentiate for that advantage to be realisable?

A Capability should not be labelled Differentiating merely because it performs well, is expensive, is strategically important or is unusual.

The posture must trace to the enterprise's intended basis of Competitive Advantage.

26. Domain and Execution Architecture

Domain owns stable business ability.

Execution owns instructions for how work is performed.

> Capability "` Process.

One Capability may require many Process Modules.

One Process Module may support several Capabilities.

A Capability Model must not be created by simply renaming current processes.

A process redesign does not automatically redefine the Capability.

Execution uses the Capability landscape as a primary coverage lens for determining where explicit work architecture is required.

> Domain establishes what the enterprise must be able to do. Execution establishes how relevant work is performed.

27. Domain and Workforce Architecture

Domain owns Capabilities.

Workforce owns Roles, Skills Required, Sourcing, Participant Types, Workforce Structures, Assigned Entities and Participants.

> Capability "` Role.
>
> Capability "` Skill.
>
> Capability "` Team.

Where Workforce sources a Capability, Domain retains Capability identity.

Where Workforce sources Labour or Skill, those sourcing propositions must not be represented as Capability change.

A Capability can remain stable while its fulfilment moves between human, agent, organisational or hybrid arrangements.

28. Domain and Ecosystem Architecture

Domain provides a primary completeness basis for Ecosystem.

For every material Capability, Ecosystem asks:

> What Actors and Relationships are required for that Capability to be delivered, accessed, exchanged or sustained?

Ecosystem then establishes the Actor, Relationship, Exchange and Dependency architecture.

Domain does not own those Actors merely because a Capability depends upon them.

The reverse traceability test is:

> For every Ecosystem Actor, which Capability or Capabilities establish why that Actor is architecturally material?

This creates a many-to-many relationship between Capability and Ecosystem structure.

29. Domain and Environment Architecture

Domain owns Capabilities.

Environment owns the physical and technological enablement required for enterprise instructions to be executable as intended.

A system, platform, building, tool or agent runtime is not a Capability.

A Capability may depend upon work that requires those Environments.

> Capability describes stable business ability. Environment provides physical and technological enablement for the work through which that ability is realised.

A technology replacement does not automatically create a Capability change.

30. Domain and Intelligence Architecture

Domain originates Requirements of Understanding where it needs to understand a Business Model, Value Chain, Capability, Capability measure, Capability state or another Domain proposition.

Intelligence architects fulfilment of those requirements.

Information about a Capability is not the Capability.

Where a Capability measure requires a derivation, Intelligence may own the derivation.

Reality captures the resulting actual value where it represents Actuality.

> Domain defines the proposition. Intelligence enables Understanding. Reality captures Actuality.

31. Domain and Reality Architecture

Domain owns:

- Business Model identity;
- Value Chain identity;
- Capability identity;
- Capability Role;
- Capability Posture;
- Capability Policy;
- Capability measure meaning;
- and Capability state or maturity model definitions.

Reality captures actual:

- Capability performance;
- Capability state;
- maturity value;
- measure value;

· or other material Domain Actuality.

State is not identity.

A Capability does not move to Reality because its actual condition is captured there.

Domain defines what the Capability is and what its states mean. Reality captures the state it is actually in.

32. Domain and Possibility Architecture

Domain defines the established Business Models, Value Chains and Capabilities of the enterprise.

Possibility explores what they could become.

Possible:

· Business Models;
· Value Chains;
· Capabilities;
· Capability combinations;
· sourcing arrangements;
· or other Domain changes

remain Possibility propositions until adopted.

Possibility may reveal a future Capability requirement.

It does not create an established Domain Capability merely by hypothesising it.

Possibility explores future business scope. Domain defines adopted business scope.

33. Domain and Expression Architecture

Domain owns Business Models, Value Chains and Capabilities.

Expression owns how enterprise value becomes packaged, presented and experienced through Offers and other manifestations.

Capability "` Offer.

A Capability may support many Offers.

An Offer may draw upon many Capabilities.

A packaged product or service may change without changing the underlying Capability.

Domain establishes the value logic and stable abilities. Expression establishes the encounterable manifestation of value.

34. Domain and Trust Architecture

Trust may be a Business Value Anchor where warranted Reliance contributes materially to enterprise value.

This does not make Trust a Domain-owned proposition.

Trust Architecture owns the conditions under which Reliance between parties is warranted.

Domain may define a Capability whose outcome includes creating conditions that support Trust, but the actual Reliance proposition remains Trust-owned.

Likewise, Capability sourcing may create Dependencies or reliance on external providers.

Ecosystem owns the Relationship and Dependency.

Trust owns warranted Reliance.

Domain retains Capability identity.

35. Domain and AEA in Action

AEA in Action may combine Domain with other architectures to construct:

- Capability Charters;
- operating models;
- BOAS;
- strategic plans;
- transformation views;
- capability investment views;
- sourcing views;
- change impact assessments;
- and other purpose-specific renders.

Domain contributes the Business Model, Value Chain and Capability architecture.

Other architectures contribute their own propositions.

Cross-architecture composition does not transfer ownership.

Detailed proprietary methods for combining Domain with the rest of AEA belong in AEA in Action rather than the Published Standard.

36. Governing Principles

Domain Architecture is the architecture of Scope.

Domain Architecture defines the business's value logic and the stable capabilities required to realise it.

Business Model !’ Value Chain !’ Capability.

A Business Model is a distinct mechanism through which the business creates economic value and generates revenue and profit.

Business Model "H Revenue Mechanism × Customer Relationship Type × Recurrence.

Every material Business Model has exactly one Value Chain.

A Value Chain is the connected path through which a Business Model sources, creates, exchanges, delivers, captures and retains value.

A Capability is a stable business ability required for the enterprise to realise its Business Models and value logic.

Capability identity must survive changes in process, technology, people, organisation and fulfilment.

Capability decomposition stops when the object ceases to be a stable business ability.

Capability Role = Value-Chain Capability or Enabling Capability.

Capability Posture = Differentiating or Standard.

Directional defines the intended basis of Competitive Advantage. Domain identifies the Capabilities that must differentiate to support it.

The governing Capability performance question is: Is the Capability producing the outcome for which it exists?

Operational performance "` Capability outcome.

Domain defines the Capability measure !’ Intelligence may establish the derivation !’ Reality captures the actual value.

Capability identity and state model belong to Domain. Actual Capability state belongs to Reality.

State is not identity.

Capability Policy remains valid despite redesign of process, tooling, people and sequence.

External fulfilment does not transfer Capability identity out of Domain.

Capability Deployment is retired as a mixed classification.

Capability change triggers impact review across all eleven AEA architectures; review does not presume change.

Capability "` Process "` Role "` Skill "` System "` Offer.

Possibility explores future business scope. Domain defines adopted business scope.

37. AI Protocol

AI may assist Domain Architecture by:

· identifying candidate Business Models;
· testing whether economic mechanisms are materially distinct;
· identifying Revenue Mechanisms;
· identifying Customer Relationship Types;
· identifying Recurrence patterns;
· constructing candidate Value Chains;
· identifying candidate Capabilities;
· testing Capability stability;
· identifying where a proposed Capability is actually a Process, Role, Skill, System, Offer, Project or other object;
· identifying shared Capabilities across Business Models;
· testing Capability Role;
· testing Capability Posture against Directional Competitive Advantage;
· identifying candidate Business Value Anchors;
· identifying Capability measure requirements;
· and identifying potential cross-architecture impacts of Capability change.

AI must not:

· create a new Business Model merely because a new product exists;
· create a new Business Model merely because a new channel exists;
· treat a legal entity as a Business Model without establishing the economic mechanism;
· treat the Value Chain as a mandatory process sequence;
· create a Capability by renaming a Process;
· create a Capability by renaming a team, system, Role, Skill, product or project;
· decompose Capability beyond the stable-business-ability boundary;
· label a Capability Differentiating merely because it is important or performs well;
· infer Competitive Advantage independently of Directional Architecture;
· treat actual Capability state as Domain-owned Actuality;
· treat a Capability measure definition as its actual value;
· treat an external provider as owning the Capability identity;
· revive Capability Deployment as a mixed Build/Buy/Partner/Automate/Orchestrate taxonomy;

· turn a Possibility hypothesis into an established Capability without adoption;
· or expose proprietary AEA in Action composition methods as Published Standard doctrine.

AI may assist the definition of business scope. It must not manufacture stable business ability from current implementation.

38. Conformance

A conformant application of Domain Architecture must preserve, where applicable:

1. Business Model; 2. Revenue Mechanism; 3. Customer Relationship Type; 4. Recurrence; 5. one Associated Value Chain for each material Business Model; 6. Value Chain; 7. Capability; 8. Capability traceability to relevant Business Models and Value Chains; 9. consolidated enterprise representation of shared Capabilities; 10. Capability Role; 11. Capability Posture where material; 12. Business Value Anchors where material; 13. Capability Outcome; 14. Capability measure definition where material; 15. Capability state or maturity model where established; 16. Capability Policy where established; 17. distinction between Capability identity and actual Capability state; 18. distinction between Capability measure meaning and actual measure value; 19. distinction between Capability and Process; 20. distinction between Capability and Role or Skill; 21. distinction between Capability and system or technology; 22. distinction between Capability and Offer; 23. current Workforce sourcing boundary where Capability is externally fulfilled; 24. Requirements of Understanding originating in Domain where applicable; 25. and impact review where a Capability is introduced, retired or materially redefined.

39. Non-Conformance

An application is non-conformant where it:

· treats a product line as a Business Model without establishing a distinct economic mechanism;
· treats a channel as a Business Model merely because sales occur through it;
· treats a Value Chain as a detailed process map;
· assigns more than one Value Chain to a single material Business Model without redefining the Business Model boundary;
· duplicates shared Capabilities solely because several Business Models use them;
· defines a Process as a Capability;
· defines a Task as a Capability;
· defines a Role as a Capability;
· defines a Skill as a Capability;
· defines a team or organisation unit as a Capability;
· defines a system, application or AI agent as a Capability;
· defines a product, service or Offer as a Capability;
· defines a project or initiative as a Capability;
· continues Capability decomposition after the object has become implementation detail;
· treats Capability Posture as an investment lifecycle taxonomy;
· assigns Competitive Advantage ownership to Domain;
· treats actual Capability maturity or state as Domain-owned Actuality;
· treats a Capability measure definition as the measured value;
· transfers Capability identity to an external provider because the Capability is sourced;
· uses the retired Capability Deployment taxonomy as canonical Domain doctrine;
· or treats a hypothetical future Capability as established Domain Architecture.

40. Published Gaps

GAPS — None currently identified that prevent publication of this Standard.

Detailed Business Model fields, Value Chain fields, Capability information schemas, Capability Charter structure, Capability measure specifications, Capability state and maturity models, Business Value Anchor classifications, traceability structures and native Capability Map construction protocols belong in the downstream Domain Architecture package.

Their absence from this Published Standard does not constitute an architecture doctrine gap.

41. Publication Position

Domain Architecture provides AEA with its architecture of Scope.

It establishes the stable business logic beneath changing strategies, processes, people, technologies and offers.

Its core architecture is:

Business Model !’ Value Chain !’ Capability

Business Model establishes the economic mechanism.

Value Chain establishes the connected path through which that mechanism sources, creates, exchanges, delivers, captures and retains value.

Capability establishes the stable business abilities required to make that value logic realisable.

The architecture then preserves the distinctions that keep business scope stable:

Capability "` Process

Capability "` Role

Capability "` Skill

Capability "` System

Capability "` Offer

Capability identity "` actual Capability state

Capability measure definition "` measured Actuality

Capability sourcing "` Capability identity

and:

future Capability possibility "` adopted Capability.

This allows the enterprise to redesign execution, change technology, alter workforce composition, source externally, create new offers and evolve its operating model without accidentally redefining the underlying business every time an implementation choice changes.

The governing proposition remains:

Domain Architecture defines the business's value logic and the stable capabilities required to realise it.