THE WHOLE FRAMEWORK IN PLAIN LANGUAGE

TOGAF,
Simply Explained

Learn it. Say it simply. Find the gap.
Explain it again.

One story • Nine ADM phases • Four domains • Zero unnecessary jargon

An independent learning guide aligned with the TOGAF Standard, 10th Edition

What is inside

1. How to use the Feynman Technique
2. What TOGAF is—and is not
3. The whole architecture journey
4. Every ADM phase in simple words
5. The four BDAT domains
6. Core terms and architecture content
7. Reuse, repositories, and governance
8. Practical techniques and modern delivery
9. A complete Sunny Toys example
10. Exam thinking and the SCOPE method
11. A one-page memory sheet and tiny glossary
12. Fifteen questions that prove understanding

How to use this book

Richard Feynman was famous for explaining hard ideas in simple words. His learning method is easy:

1. **Learn one small idea.**
2. **Close the page.**
3. **Explain it as if you are teaching a child.**
4. **Notice where your explanation becomes fuzzy.**
5. **Study that small gap, then explain it again.**

This book follows that pattern. Each section has:

- a plain explanation;
- one example;
- a memory hook; and
- a **say it back** challenge.

Do not try to memorize every sentence. Build the picture in your head.

> **Your finish line:** You can explain TOGAF without using the word “TOGAF” in the explanation.

Our one example: Sunny Toys

Sunny Toys has ten shops. It wants customers to:

- see which toys are available;
- buy online;
- collect from any shop; and
- receive the same good service everywhere.

Today, every shop works differently. Stock numbers are often wrong. There are three customer lists and five old applications. Sunny Toys needs coordinated change—not just a new website.

We will use Sunny Toys for the whole book.

FEYNMAN CHECK

Say it back: What are the five steps in the Feynman learning loop? Explain them without looking.

1. What TOGAF is

Imagine a town wants to build homes, roads, schools, pipes, and parks.

If every builder works alone:

- one road may end at a wall;
- pipes may not reach the homes;
- two schools may be built where only one is needed; and
- the town may spend money without becoming a better place to live.

The town needs a shared plan. It also needs a way to create, check, and update that plan.

An enterprise has the same problem. Its “roads and buildings” are:

- people and teams;
- business capabilities and processes;
- information;
- applications;
- technology; and
- projects and products changing all of them.

Enterprise Architecture is the shared picture and decision discipline that keeps those parts moving toward the same outcomes.

TOGAF is a framework for doing Enterprise Architecture. It gives you:

- a repeatable method;
- a shared vocabulary;
- ways to describe architecture;
- analysis techniques;
- governance practices; and
- advice for building an architecture capability.

> **Shortest correct explanation:** TOGAF helps an organization understand where it is, decide where it wants to go, plan the changes, and keep delivery moving in that direction.

What TOGAF is not

TOGAF is not:

- a finished architecture you can copy;
- a software product;
- a diagramming language;
- a project management method;
- a command to create every possible document; or
- a rigid waterfall.

It is a **method and toolbox**. You choose the parts that help with the real decision.

Three words that sound bigger than they are

Word	Simple meaning
Enterprise	The organization or part of it that we are studying
Architecture	The important structure, relationships, and rules
Framework	An organized set of ideas and tools for doing the work

FEYNMAN CHECK

Say it back: Explain TOGAF using the town analogy. Then explain why TOGAF is not a finished town plan.

2. The whole TOGAF story

Every useful architecture journey answers the same questions:

1. **Why?**
   Why must we change?
2. **Who?**
   Who cares, and what worries them?
3. **Now?**
   What matters about today?
4. **Future?**
   What should tomorrow look like?

5 **Gap?**
What must change?

6 **Order?**
What should happen first?

7 **Build?**
Are we following the plan?

8 **Learn?**
Did it work, and what changed?

For Sunny Toys:

1. **Why?** Customers leave because stock information is wrong.
2. **Who?** Customers, shop staff, managers, finance, security, and delivery teams all care about different things.
3. **Now?** Each shop has separate processes, data, and applications.
4. **Future?** One joined-up buying and collection experience.
5. **Gap?** Shared stock data, clearer ownership, connected applications, staff training, and reliable technology are missing.
6. **Order?** Clean stock data before promising real-time availability.
7. **Build?** Check each release against the agreed requirements.
8. **Learn?** Measure collection success, stock accuracy, customer happiness, cost, and new risks.

> **Memory hook:** Why → Who → Now → Future → Gap → Order → Build → Learn.

FEYNMAN CHECK

Say it back: Close this page and draw the eight questions as a circle. If one question is hard to explain, that is your next study gap.

3. The ADM: TOGAF’s main method

The **Architecture Development Method**, or **ADM**, is the heart of TOGAF.

It organizes architecture work into:

- a setup phase;
- eight lettered phases, A through H; and
- Requirements Management in the middle.

Use this mnemonic:

Prepare A Business; Connect Digital Execution For Governed Horizons.

That gives you **P-A-B-C-D-E-F-G-H**.

Preliminary Phase — Prepare

Simple question: How will we do architecture here?

Before planning Sunny Toys, the organization agrees:

- who makes which decisions;
- which method and documents are useful;
- which principles guide choices;
- where architecture knowledge is stored; and
- how architects work with leaders and delivery teams.

Main result: a working architecture capability—not merely an architecture team.

Phase A — Agree the Architecture Vision

Simple question: Why are we doing this, for whom, and how big is the work?

Sunny Toys agrees:

- the desired customer outcome;
- the sponsor;
- stakeholders and concerns;
- what is inside and outside scope;
- high-level baseline and target;
- success measures and risks; and
- permission to continue.

Main results: the **Architecture Vision** and **Statement of Architecture Work**.

Phase B — Business Architecture

Simple question: How should the business work?

Sunny Toys designs:

- the buy-and-collect value stream;

- shop and online capabilities;
- staff responsibilities;
- customer service;
- business rules; and
- ownership of stock accuracy.

Main result: business baseline, target, and gaps.

Phase C — Information Systems Architectures

Phase C contains two domains.

Data Architecture

Simple question: What information do we need, and who looks after it?

Sunny Toys defines:

- one meaning for Product, Shop, Stock, Customer, Order, and Collection;
- an owner for each important kind of data;
- where trusted data comes from;
- how it moves; and
- privacy, quality, retention, and access rules.

Application Architecture

Simple question: What applications and services use that information?

Sunny Toys decides how the website, stock service, payment service, shop application, notification service, and reporting tools work together.

Main result: data and application baselines, targets, and gaps.

Phase D — Technology Architecture

Simple question: What technology lets the target work safely and reliably?

Sunny Toys chooses the needed:

- platforms and hosting;
- networks and shop connectivity;
- identity and access services;
- monitoring and backup;
- integration technology; and
- technology standards and lifecycle rules.

Main result: technology baseline, target, and gaps.

Phase E — Opportunities & Solutions

Simple question: How do we turn all the gaps into sensible pieces of work?

Sunny Toys groups changes into work packages:

1. create trusted stock data;
2. connect shop systems;
3. launch online ordering;
4. add collection notifications; and
5. improve reporting and operations.

It may define **Transition Architectures**—safe intermediate states on the way to the target.

Main result: work packages and candidate transition architectures.

Phase F — Migration Planning

Simple question: What should happen first, and can we really do it?

Sunny Toys compares:

- value;
- cost;
- risk;
- dependencies;
- people and skills;
- readiness; and
- time.

Main results: a prioritized **Architecture Roadmap** and an executable **Implementation and Migration Plan**.

Phase G — Implementation Governance

Simple question: Is delivery building what was agreed?

Sunny Toys:

- agrees responsibilities and review points;
- checks important requirements;
- reviews evidence;
- handles justified exceptions; and
- avoids turning governance into a slow approval queue.

Main results: Architecture Contracts or equivalent agreements, compliance evidence, decisions, and managed deviations.

Phase H — Architecture Change Management

Simple question: Has the world changed enough to update the architecture?

Sunny Toys watches:

- business outcomes;
- customer behavior;
- operating problems;
- new laws and risks;
- technology changes; and
- architecture debt.

A small change may update one decision. A big change may start a new ADM cycle.

Main result: a living architecture instead of an old document.

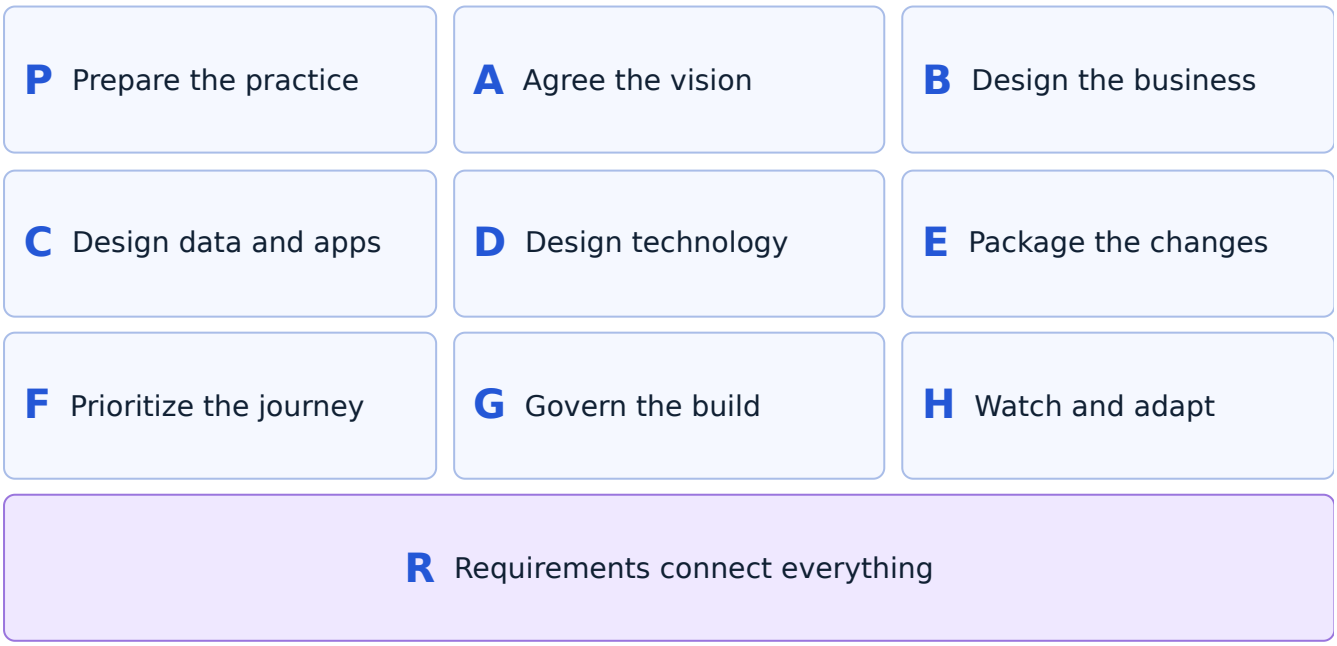
Requirements Management — the center

A **requirement** says what must be true.

Example: “A customer must only see stock that can be reserved.”

Requirements Management stays in the center because every phase:

- discovers requirements;
- uses requirements;
- tests requirements;
- changes requirements; or
- is changed by requirements.



Do not treat the ADM as a waterfall. You may revisit earlier work, combine phases, run work in parallel, or use small cycles. Tailor it to the decision, risk, and delivery speed.

FEYNMAN CHECK

Say it back: Draw P through H. For each phase, say its question in one sentence. Put Requirements Management in the center and explain why.

4. The four architecture domains: BDAT

Use **BDAT**:

Domain	Child-simple question	Sunny Toys example
B — Business	How do people create value?	Buy online and collect in a shop
D — Data	What information do they need?	Product, stock, customer, order
A — Application	What software behavior helps them?	Website, stock, payment, shop app
T — Technology	What machinery lets the software run?	Cloud, network, identity, monitoring

The domains are different, but they must fit together.

If the business promises one-hour collection, then:

- Data must show trustworthy stock.
- Applications must reserve the toy.
- Technology must keep the service available.
- People and processes must prepare the collection.

Security, risk, privacy, integration, customer experience, and sustainability cross all four domains. They are not decorations added at the end.

Memory hook: Business asks for value. Data carries meaning. Applications provide behavior. Technology provides the platform.

FEYNMAN CHECK

Say it back: Pick a school, shop, or app. Give one Business, Data, Application, and Technology example. Then name one cross-cutting concern.

5. The core language

Stakeholder, concern, view, viewpoint

These four words form one chain:

1. A **stakeholder** is a person or group that cares.
2. A **concern** is what they care about.
3. A **viewpoint** is the rule for creating a useful picture.
4. A **view** is the actual picture made for this architecture.

Example:

- Stakeholder: shop manager
- Concern: “Will collection create long queues?”
- Viewpoint: show the customer and staff journey with timings
- View: Sunny Toys’ actual collection journey diagram

Baseline, target, gap, transition, roadmap

Term	Simple meaning
Baseline	The important truth about today
Target	The future we intend to create
Gap	A difference that needs action
Transition Architecture	A safe intermediate version of the enterprise
Roadmap	The ordered path through the changes

Driver, goal, outcome, requirement, measure

Term	Sunny Toys example
Driver	Customers expect online ordering
Goal	Make buying and collection easy
Outcome	More completed sales with fewer stock mistakes
Requirement	Reserved stock must not be sold to someone else
Measure	Collection success rate and stock accuracy

Principle, constraint, assumption, risk

Term	Simple meaning
Principle	A lasting rule that guides decisions
Constraint	A limit we must work within
Assumption	Something we currently believe is true
Risk	Uncertainty that may affect an outcome

Example principle: **Stock has one trusted owner.**

Example constraint: **The first release must use existing shop scanners.**

Example assumption: **Every shop has stable internet access.**

Example risk: **Old stock records may be too inaccurate for online promises.**

> **FEYNMAN CHECK**
>
> **Say it back:** Explain each chain without reading: stakeholder → concern → viewpoint → view, and baseline → target → gap → transition → roadmap.

6. What architects produce

TOGAF distinguishes three things.

Deliverable

A **deliverable** is a formally reviewed package of work.

Think: a school project handed to the teacher.

Examples:

- Architecture Vision;
- Architecture Definition Document;
- Architecture Requirements Specification;
- Architecture Roadmap; and
- Implementation and Migration Plan.

Artifact

An **artifact** is a representation of architecture information inside or alongside the work.

There are three simple forms:

Form	It answers	Example
Catalog	What exists?	Application list
Matrix	How are things related?	Capability-to-application matrix
Diagram	How should we see it?	Application interaction diagram

Building block

A **building block** is a reusable unit of capability or solution.

- An **Architecture Building Block (ABB)** says what capability is needed.
- A **Solution Building Block (SBB)** says how it will be implemented.

Sunny Toys example:

- ABB: “Secure customer identity capability”
- SBB: “The selected identity platform and its configured services”

> **Memory hook:** A deliverable is the reviewed package. An artifact is a picture or structured list. A building block is a reusable piece of the architecture or solution.

The minimum useful architecture pack

Do not create documents because a framework has a template. For many engagements, a small useful pack is:

1. outcome, scope, sponsor, and stakeholders;
2. principles, assumptions, constraints, and risks;
3. relevant baseline;

4. target choices across the necessary domains;
5. traceable requirements and decisions;
6. gaps, work packages, and roadmap;
7. governance and exception approach; and
8. measures of success.

FEYNMAN CHECK

Say it back: Explain deliverable, artifact, ABB, and SBB using building a tree house as your example.

7. Reuse and the Architecture Repository

Imagine a kitchen.

The **Enterprise Continuum** is the labeling idea that sorts recipes and ingredients from very general to very specific.

The **Architecture Repository** is the actual pantry, cookbook shelf, fridge, and list that store and control them.

```
<strong>Enterprise Continuum</strong>
<span>Classifies knowledge from generic to organization-specific.</span>
```

```
<strong>Architecture Repository</strong>
<span>Stores, controls, connects, and helps people reuse knowledge.</span>
```

The Continuum has two related sides:

- **Architecture Continuum:** reusable architecture ideas and descriptions.
- **Solutions Continuum:** products, services, and implemented solutions.

Common Repository areas are:

Area	Simple contents
Architecture Metamodel	Rules about the kinds of information and relationships
Architecture Capability	Roles, method, governance, skills, and tools
Architecture Landscape	Baseline, transition, and target descriptions
Standards Information Base	Approved standards, products, versions, and lifecycle
Reference Library	Patterns, templates, reference models, and lessons
Governance Log	Decisions, reviews, exceptions, risks, and debt

Good repository content has an owner, status, version, and review date.

FEYNMAN CHECK

Say it back: Use the kitchen analogy to explain the difference between the Enterprise Continuum and Architecture Repository.

8. Governance without becoming the architecture police

Governance means important decisions are:

- made by the right people;
- based on visible criteria and evidence;
- recorded;
- followed during delivery;
- allowed a controlled exception when justified; and
- improved using feedback.

It does **not** mean one central committee designs everything.

Think of road rules:

- rules make movement safer;
- signs make expectations visible;
- drivers still choose their route;
- unusual situations have an exception process; and

- accident evidence helps improve the rules.

The governance loop

1. Agree principles and standards.
2. Define decision rights.
3. Make and record decisions.
4. Agree implementation responsibilities.
5. Review evidence of conformance.
6. Approve, correct, or manage an exception.
7. Measure the outcome.
8. Improve the architecture and rules.

An **Architecture Board** is one governance body. It needs a clear mandate, authority, decision criteria, membership, and escalation rules. Low-risk decisions should stay close to delivery.

An **Architecture Contract** is an agreement about responsibilities, requirements, evidence, reviews, and exceptions. It does not have to be a giant legal document.

> **FEYNMAN CHECK**
>
> **Say it back:** Explain why good governance is like road rules, not like one person driving every car.

9. The small toolbox of techniques

Use a technique only when it helps answer a question.

Technique	Simple purpose
Stakeholder analysis	Find who cares, why, how much influence they have, and how to involve them
Business scenario	Turn a fuzzy problem into actors, steps, outcomes, measures, and requirements
Capability-Based Planning	Plan around abilities the enterprise needs, not only projects or systems
Value-stream mapping	Follow how value moves from a need to an outcome
Gap analysis	Compare baseline and target to identify change
Readiness assessment	Check whether people, skills, culture, funding, and governance can support change
Risk management	Find uncertainty, understand exposure, choose treatment, and keep watching
Trade-off analysis	Compare options using explicit criteria and consequences
Dependency analysis	Find what relies on what before creating the sequence
Transition Architectures	Define safe intermediate states

A simple trade-off method

When Sunny Toys must choose a solution:

1. State the decision.
2. Name the stakeholders and concerns.
3. Define criteria and constraints.
4. Create more than one credible option, including “do nothing” where useful.
5. Gather enough evidence—not endless evidence.
6. Compare value, cost, risk, uncertainty, and reversibility.
7. Let the person with the decision right decide.
8. Record the choice, consequences, and revisit trigger.

FEYNMAN CHECK

Say it back: Pick one everyday choice. Compare two options using outcome, cost, risk, and reversibility.

10. TOGAF with Agile, products, DevOps, cloud, and security

TOGAF and modern delivery solve different parts of the problem.

- TOGAF helps decide the important direction and coordinate change.
- Product management keeps attention on persistent customer value.
- Agile creates short learning loops.
- DevOps makes delivery and operations fast and repeatable.
- Cloud provides on-demand technology services.
- Security and risk protect outcomes across every domain and phase.

They can work together.

Good modern architecture:

- works in small increments;
- gives teams clear guardrails and decision rights;
- records important decisions briefly;
- automates standards and evidence where possible;
- keeps architects close to products and delivery;
- uses operational data to update architecture; and
- escalates only choices with material risk or cross-enterprise impact.

Bad architecture creates a document mountain and arrives after all decisions have already been made.

> **Useful rule:** Centralize the decisions that require enterprise coherence. Delegate the decisions that teams can safely make locally.

FEYNMAN CHECK

Say it back: Explain how TOGAF and Agile can work together. Your answer must include direction, guardrails, short feedback, and learning.

11. How to start TOGAF in a real

organization

Do not announce, “We will implement the whole framework.”

Start with a real business problem.

Days 1–30: establish the minimum

- Confirm the sponsor and outcome.
- Identify stakeholders and decision rights.
- Agree a small set of principles.
- Tailor the ADM and minimum content.
- Create a simple repository structure.
- Choose one valuable pilot.

Days 31–60: run the pilot

- Agree the vision and scope.
- Develop only the useful baseline and target.
- Trace requirements and decisions.
- Identify gaps and work packages.
- Build a practical roadmap.
- Work beside delivery teams.

Days 61–90: connect and improve

- Run proportionate governance and exception handling.
- Measure value, speed, risk, and conformance.
- Reuse patterns and lessons.
- Fix unclear decision rights.
- Expand only what has proved useful.

Common failure patterns

Failure	Why it fails	Better move
Framework-first rollout	People see process, not value	Start with one outcome
Artifact factory	Documents grow; decisions do not improve	Produce only decision-useful content
Review-board bottleneck	Architecture becomes a queue	Delegate low-risk decisions
Ivory-tower target	Delivery evidence arrives too late	Work with product and delivery teams
Tool-led metamodel	The repository becomes the goal	Start with questions and users
Architecture without authority	Good advice can be ignored	Make decision rights explicit
Governance without feedback	Rules become stale	Measure outcomes and learn

FEYNMAN CHECK

Say it back: Explain how you would begin TOGAF without saying, “First, create every TOGAF document.”

12. One full Sunny Toys walkthrough

Here is the entire method in one short story.

Prepare

Sunny Toys gives the transformation sponsor authority. A small Architecture Board handles only cross-shop decisions. Product teams keep local delivery choices within shared guardrails.

Agree

The vision is: “Customers can confidently buy online and collect from any shop within two hours.” Scope includes stock, ordering, payment, collection, staff workflow, and enabling technology. It excludes home delivery for now.

Understand the business

The target value stream is:

Find toy → reserve → pay → prepare → notify → collect → support

Shop staff own physical stock accuracy. Customer service owns collection support. Product leaders own the end-to-end outcome.

Design information and applications

Product, Shop, Stock, Customer, Order, Payment, and Collection get clear definitions and owners. A stock service becomes the trusted source. The website and shop application use the same order and stock services.

Design technology

The target uses secure identity, managed hosting, reliable shop connectivity, monitoring, backup, and standard integration. Privacy and security requirements apply throughout.

Package and plan

Sunny Toys creates three transition states:

1. accurate shared stock;
2. online reservation with payment at collection; and
3. full online payment and two-hour collection.

The roadmap follows data dependency and business readiness, not excitement about the website.

Govern and learn

Each release supplies automated test, security, reliability, and stock-accuracy evidence. Exceptions have an owner and expiry. Sunny Toys measures customer completion, stock accuracy, collection time, support calls, cost, and incidents.

Those results update the architecture.

FEYNMAN CHECK

Say it back: Retell the Sunny Toys story without phase letters. Then add the phase letters to each step.

13. The exam idea, simply

The current Enterprise Architecture learning path has two main levels:

Foundation

Foundation checks whether you understand:

- core concepts and vocabulary;
- the ADM structure and purpose of each phase;
- foundational techniques;
- applying the ADM;
- governance; and
- architecture content.

The Part 1 examination uses 40 multiple-choice questions in 60 minutes, with a 60% passing score.

Practitioner

Practitioner checks whether you can apply the framework to a situation.

The Part 2 examination uses eight scenario-based questions in 90 minutes. Each question has four choices scored 5, 3, 1, or 0. It is open book using the provided reference material.

Use SCOPE for scenarios

1. **S — Stakeholders:** Who cares and who decides?
2. **C — Context:** What phase, scope, evidence, and constraints exist?
3. **O — Outcome:** What result or decision is needed?
4. **P — Proportion:** What is the smallest sufficient response?
5. **E — Evidence:** Which choice preserves traceability, trade-offs, and feedback?

Prefer answers that clarify outcomes and authority, address concerns, tailor the work, maintain traceability, and enable governed delivery.

Be suspicious of answers that:

- create every artifact;
- skip straight to technology;
- centralize every choice;
- ignore stakeholder concerns; or
- treat the ADM as a fixed waterfall.

FEYNMAN CHECK

Say it back: Explain the difference between Foundation and Practitioner. Then use SCOPE on a decision from the Sunny Toys story.

14. One-page memory sheet

TOGAF in one sentence

TOGAF helps an organization understand where it is, agree where it wants to go, plan the changes, govern delivery, and keep learning.

The journey

Why → Who → Now → Future → Gap → Order → Build → Learn

The ADM

Phase	Remember
Preliminary	Prepare the architecture practice
A	Agree vision, value, scope, and authority
B	Design how the business creates value
C	Design data and applications
D	Design enabling technology
E	Package gaps into work
F	Prioritize the migration journey
G	Govern implementation
H	Watch change and adapt
Requirements	Connect and trace everything

The domains

BDAT = Business, Data, Application, Technology

The content

- Deliverable = reviewed package
- Artifact = catalog, matrix, or diagram
- ABB = required capability
- SBB = implementation

Reuse

- Continuum = classifies
- Repository = stores and controls

Governance

Decision rights → decision → agreement → evidence → compliance or exception → outcome → feedback

Scenario thinking

SCOPE = Stakeholders, Context, Outcome, Proportion, Evidence

FEYNMAN CHECK

Final say-it-back: Close the book. Recreate this memory sheet on one blank page. Teach it to another person in ten minutes.

15. Tiny glossary

Term	Plain meaning
ABB	What architecture capability is required
ADM	TOGAF’s main iterative method
Architecture Board	Body accountable for defined architecture decisions
Architecture Contract	Agreement about implementation and conformance
Architecture Repository	Managed home for architecture knowledge
Artifact	Catalog, matrix, diagram, or other representation
Baseline	Relevant current state
Building block	Reusable architecture or solution component
Business capability	What the enterprise is able to do
Concern	What matters to a stakeholder
Deliverable	Formally reviewed work product
Enterprise	The organization or scope being studied
Enterprise Continuum	Way to classify assets from generic to specific
Gap	Difference between baseline and target that needs action
Governance	How decisions, controls, exceptions, and feedback work
Principle	Durable rule for making decisions
Requirement	Condition that must be satisfied
SBB	Specific implementation of required capability
Stakeholder	Person or group with an interest
Target	Intended future state
TOGAF	Framework for developing and governing Enterprise Architecture

Term	Plain meaning
Transition Architecture	Coherent intermediate state
Value stream	Stages that create value for a stakeholder
View	Actual representation for stakeholder concerns
Viewpoint	Rules for constructing and using a view
Work package	Bounded set of change actions

16. Prove that you understand

Answer these aloud in simple words:

1. What problem does Enterprise Architecture solve?
2. What does TOGAF give you?
3. Why is the ADM not a waterfall?
4. What question does each ADM phase answer?
5. Why is Requirements Management in the center?
6. How do Business, Data, Application, and Technology fit together?
7. What is the difference between a view and a viewpoint?
8. What is the difference between baseline, target, gap, transition, and roadmap?
9. What is the difference between a deliverable, artifact, ABB, and SBB?
10. Why is the Enterprise Continuum not the Architecture Repository?
11. What makes governance useful instead of slow?
12. How can TOGAF work with Agile and DevOps?
13. How would you start TOGAF in an organization?
14. How does SCOPE help with a scenario?
15. Can you retell the Sunny Toys story from problem to learning?

If an answer needs lots of jargon, simplify it. If you cannot simplify it, find the small gap and study it again. That is the Feynman Technique.

You understand TOGAF when you can:

- draw the whole journey from memory;
- explain each phase with a simple question;
- apply it to a real change;
- choose only the useful content and techniques; and
- show how decisions lead to outcomes and learning.

Continue learning

- Read the full TOGAF Handbook for detail.
- Use the fast-learning mind map for recall.
- Open the editable draw.io workbook to redraw the relationships.
- Take the timed mock exam and review mistakes by concept.
- Use the comprehensive glossary whenever a word becomes fuzzy.

Official sources

This is an independent educational summary. It does not reproduce or replace the official standard.

- TOGAF overview
- TOGAF Standard publications catalog
- TOGAF Library
- TOGAF certification portfolio