

ArcKit EA

Agentic Enterprise Architecture Platform

A technical white paper on executable enterprise architecture: TOGAF ADM + Open Agile
Architecture + Loop Factory

Table of Contents

- 1. [Executive Summary](#)
 - 2. [The EA Problem](#)
 - 3. [ArcKit EA Architecture](#)
 - 4. [Plugin Reference](#)
 - 5. [Workflow Walkthrough: Cross-Border Agentic Underwriting](#)
 - 6. [Use Case Library](#)
 - 7. [Comparison: ArcKit EA vs. Traditional EA Tooling](#)
 - 8. [Implementation Roadmap](#)
 - 9. [References](#)
-

1. Executive Summary

Enterprise architecture (EA) has a delivery problem. TOGAF ADM is the de-facto standard for architecture governance, but it produces artefacts that are **documents, not executable programs**. A 200-page transition architecture sits in a repository, gets stale within one quarter, and provides no machine-readable contract that a delivery pipeline can consume.

ArcKit EA solves this. It is a plugin architecture that makes enterprise architecture **executable and AI-agentic**:

- `arckit-togaf-adm` — 12 commands that turn TOGAF ADM into structured, traceable artefacts (16 doc-types)
- `arckit-oaa` — 5 commands that turn Open Agile Architecture into sprint-level canvases (5 doc-types)
- **Loop Factory** — a LangGraph state machine that **consumes** those artefacts and **autonomously builds, tests, and ships** the system they describe

The result: architecture is no longer a planning artefact that delivery ignores. Architecture **is** the build spec. The gap between "what the architecture says" and "what actually gets deployed" is closed by a continuous execution loop.

Key numbers:

- 22 doc-types across two plugin layers (16 TOGAF + 5 OAA + 1 shared traceability)
 - 17 commands (12 TOGAF-ADM + 5 OAA), plus 4 shared skills via core (principles, requirements, stakeholders, strategy)
 - 8-phase Loop Factory execution cycle (DISCOVER → REFLECT)
 - 6–10 weeks to first shippable product (vs. 6–18 months for TOGAF-only)
 - Per-sprint compliance evidence via compliance-as-code rules in OAA `OASEC`
-

2. The EA Problem

2.1 The Artefact Gap

Traditional EA tools (Sparx, ARIS, Bizz) produce **diagrams and documents**. These artefacts are:

Property	Traditional EA	ArcKit EA
Format	UML/SysML XML, PDF	Markdown (machine-readable)
Executable	No	Yes (feeds Loop Factory)
Traceable	Manually maintained	Auto-generated (TRAC)
Update cadence	Quarterly (best case)	Per-sprint
Consumed by CI/CD	No	Yes (LangGraph nodes)
Compliance evidence	Manual audit trail	Compliance-as-code rules

The core failure mode: **the architecture document and the deployed system diverge immediately** because there is no automated path from one to the other.

2.2 The Cadence Mismatch

TOGAF boards run quarterly. Agile delivery runs on 2-week sprints. Most organisations run both in parallel and the two never meet — the architecture board approves a transition wave in Q2, and by the time it reaches a delivery team in Q3, three sprints have already moved the ground.

ArcKit EA reconciles this by treating **OAA sprints as the tactical checkpoint** and **TOGAF boards as the strategic checkpoint**, with the OAA `0AG0V` (governance cadence) artefact explicitly mapping sprint-level governance up to the quarterly board. No parallel cadences — one unified pipeline.

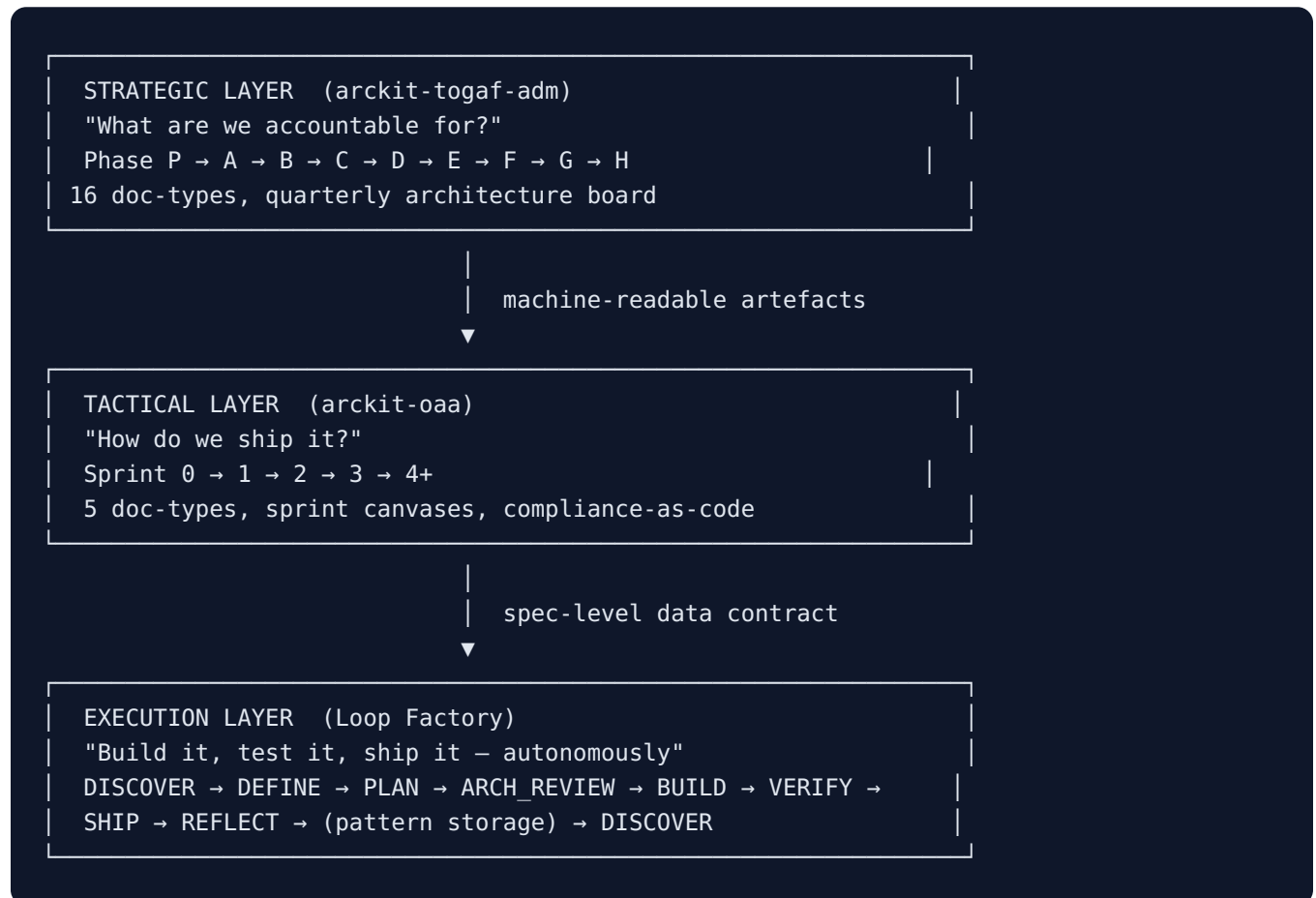
2.3 The Compliance Gap

Regulatory frameworks (APRA CPS 230, GDPR Art. 22, ISO 27001 Annex A) require **evidence**, not just policy. Traditional EA produces policy documents. ArcKit EA's OAA plugin produces `0ASEC` — **compliance-as-code rules** that are executable validation checks run on every commit, not a PDF that an auditor has to read and cross-reference.

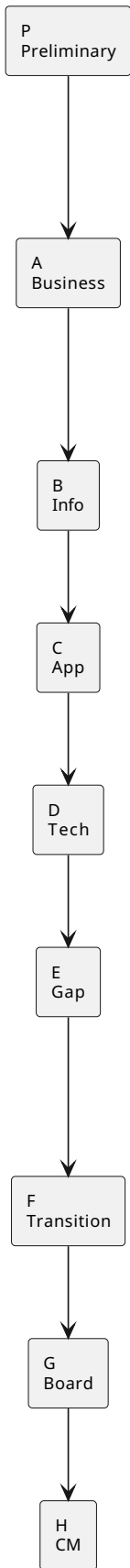
3. ArcKit EA Architecture

3.1 Three-Layer Model

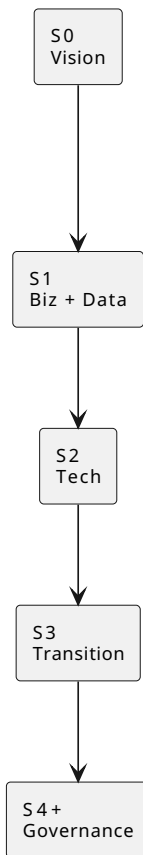
ArcKit EA is a **compiler + CPU** model:



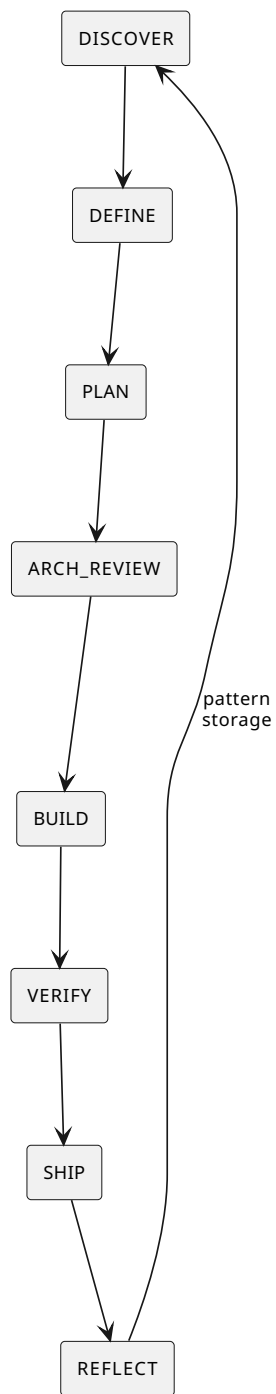
Strategy Layer (arckit-togaf-adm)



Tactical Layer (arckit-oaa)



Execution Layer (Loop Factory)



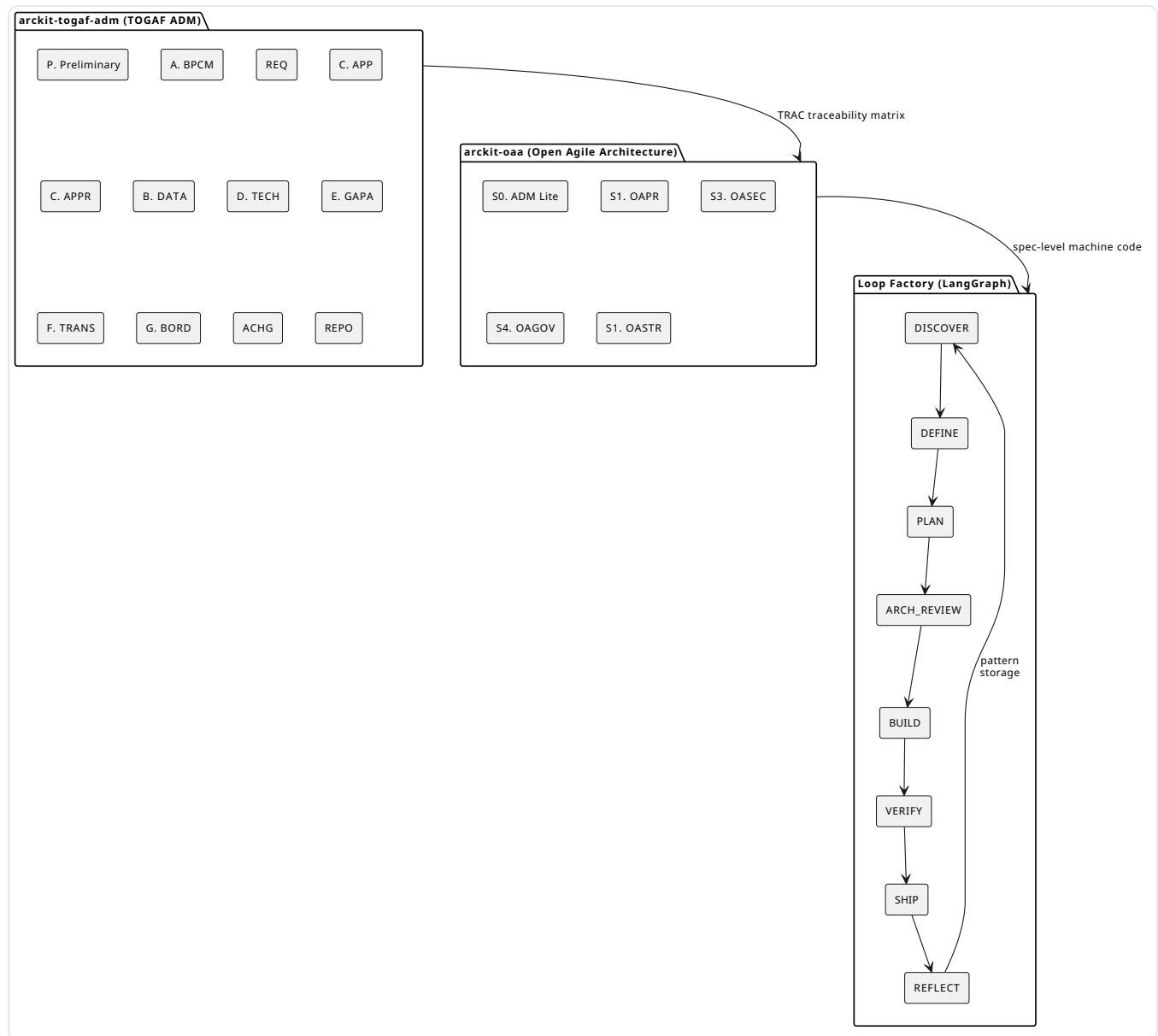
machine-readable artefacts

spec-level data contract

pattern storage

3.2 Data Flow Between Plugins

The data contract between layers is the **TRAC (Traceability) matrix** — a machine-generated cross-reference from every OAA requirement ID back to the TOGAF requirement it satisfies. This is enforced by the `arckit:traceability` post-build hook, which scans all `ARC-*.md` artefacts after each sprint.



ArcKit EA Plugin Data Flow](arckit-ea-plugin-dataflow.svg)

3.3 Artefact Naming Convention

All artefacts follow `ARC-NNN-TYPE-vN.N.md` (multi-instance: `ARC-NNN-ADR-NNN-vN.N.md`). This is critical for Loop Factory's ingestion — the state machine parses artefact filenames to determine what to consume at each phase.

Doc-type	Plugin	Description
ADMP	togaf-adm	Architecture Development Method Preliminary
BPCM	togaf-adm	Business Capability Map
REQ	togaf-adm	Requirements Register
APP	togaf-adm	Application Inventory
APPR	togaf-adm	Application Rationalization
GAPA	togaf-adm	Gap Analysis
TRANS	togaf-adm	Transition Architecture
BORD	togaf-adm	Architecture Board Minutes
ACHG	togaf-adm	Architecture Change Request
REPO	togaf-adm	Architecture Repository
DISC	togaf-adm	Discovery — current-state inventory & baseline
DATA	togaf-adm	Data model, classification, GDPR mapping
TECH	togaf-adm	Technology stack, cloud topology, GPU sizing
PRIN	shared (core)	Architectural principles
STKE	shared (core)	Stakeholder map & register
STRAT	shared (core)	Strategy (org + digital)
OAAAL	oaa	OAA Sprint Map (5-sprint plan)
OAPR	oaa	Product & Team Ownership
OASTR	oaa	Strategy Canvas (dual transformation)
OASEC	oaa	Security & Compliance (per sprint)
OAGOV	oaa	Governance Cadence
TRAC	shared	Traceability Matrix (auto-generated)

The `togaf-agent-full` recipe additionally emits agent-architecture artefact types (A-prefix codes, e.g. `AAGI` , `AAMT` , `AAOV`) — see the `arokit-agent-architecture` plugin.

4. Plugin Reference

4.1 arckit-togaf-adm

12 commands, 2 build recipes.

Command	Output	Use
arckit:adm-preliminary	ADMP	Scope, assumptions, constraints, stakeholder map
arckit:business-capability-map	BPCM	Capability map with maturity scores
arckit:discovery	DISC	Current-state inventory & baseline
arckit:application-inventory	APP	Current-state application inventory
arckit:application-rationalization	APPR	Consolidate/retire/replace decisions
arckit:data-architecture	DATA	Data model, classification, GDPR mapping
arckit:technology-architecture	TECH	Technology stack, cloud topology, GPU sizing
arckit:gap-analysis	GAPA	Current → target gap, risk register
arckit:transition-architecture	TRANS	Migration waves, strangle pattern, rollback
arckit:architecture-board	BORD	Board minutes, sign-off chain
arckit:architecture-change	ACHG	Change request (OAA → TOGAF ratification)
arckit:architecture-repository	REPO	Living repository index

Shared skills (via arckit-claude core): `arckit:principles` (PRIN), `arckit:requirements` (REQ), `arckit:stakeholders` (STKE), `arckit:strategy` (STRAT) — invoked by both plugin recipes.

Build recipes:

- `togaf-adm-full` — 9 phases, standard TOGAF
- `togaf-agent-full` — adds agent-specific artefacts: `AAGI` (agent inventory), `AAGR` (agent design), `AAOV` (agent governance), `AASE` (agent security), `AAIN` (agent integration), `AAMT` (agent maturity)

Default for EyWALink engagements: `togaf-agent-full` .

4.2 arckit-oaa

5 commands, 1 build recipe.

Command	Output	Use
arckit:oaa-adm-lite	OAAL	5-sprint execution plan (OAA ADM Lite)
arckit:product-architecture	OAPR	Product & team ownership, RACI
arckit:agile-security	OASEC	Per-sprint security stories + compliance-as-code rules
arckit:agile-governance	OAGOV	Sprint governance cadence, architecture debt ledger
arckit:agile-strategy	OASTR	Dual transformation strategy canvas

Build recipe: oaa-full (6 phases: strategy → product → ADM Lite → security → governance + optional maturity)

4.3 Decision Framework

Scenario	Recommended	Rationale
Regulator-mandated enterprise baseline	togaf-adm alone	Defensibility, full traceability matrix
Greenfield AI product, < 8-week deadline	oaa alone	Sprint velocity, embedded security
Legacy estate modernization, 12-month programme	togaf-adm alone	Application rationalization depth
Cross-border multi-jurisdiction deployment	both	TOGAF baseline + OAA execution
Agentic AI platform (agent fleets)	togaf-agent-full	AAGI/AAGR/AAOV/AASE/AAIN artefacts
Continuous delivery, self-improving	both → Loop Factory	OAA outputs are Loop Factory fuel

5. Workflow Walkthrough: Cross-Border Agentic Underwriting

Scenario: APAC insurance client. Agentic AI underwriting platform. APRA CPS 230 + GDPR dual jurisdiction. 12-week hard deadline. 6 engineers.

Week 1–2: TOGAF Baseline (condensed)

Run `arckit:adm-preliminary` → `arckit:business-capability-map` → `arckit:requirements` .

Output:

- `ARC-001-ADMP-v1.0.md` — scope: APAC agentic underwriting, dual jurisdiction, data classification "sensitive"
- `ARC-001-BPCM-v1.0.md` — capabilities: Underwriting Decisioning, Risk Assessment, Model Orchestration, Compliance Reporting
- `ARC-001-REQ-v1.0.md` — 47 requirements including `REQ-012` (APRA CPS 230 §7 AI governance), `REQ-015` (data residency: APAC-only), `REQ-023` (model explainability)

Week 3: OAA Sprint Map

Run `arckit:aaa-adm-lite` against the TOGAF baseline.

Output: `ARC-001-OAAL-v1.0.md` — 5-sprint plan:

Sprint	Focus	Cited TOGAF reqs
S0 (Vision)	AI workload type, data classification, jurisdiction, latency targets	REQ-001, REQ-015
S1 (Biz + Data)	Capability gaps, value streams with SLAs, data asset inventory	REQ-012, REQ-015, REQ-023
S2 (Tech)	Model serving, vector store, GPU cluster sizing, network topology	REQ-015, REQ-023
S3 (Transition)	Migration waves, strangle pattern, dependency sequencing, rollback	REQ-012
S4+ (Governance)	Ongoing review gates, change velocity thresholds, compliance evidence	REQ-012

Weeks 3–12: Sprint Execution

Each sprint produces:

- **OAPR** — team ownership (6 engineers, product owner, AI safety engineer)
- **OASEC** — security stories + APRA CPS 230 compliance-as-code rules
- **OAGOV** — sprint governance cadence, architecture debt ledger

Key invariant: OAA sprints **never edit TOGAF artefacts directly**. Changes flow as **ACHG** (architecture change request) → ratified by TOGAF board → version increment. This preserves the baseline's integrity.

Continuous: Loop Factory Execution

Each sprint's artefacts feed Loop Factory's LangGraph state machine:

```
DISCOVER    ← ARC-001-REQ-v1.0.md + ARC-001-OAAL-v1.0.md
DEFINE      ← ARC-001-OAPR-v1.0.md
PLAN        ← ARC-001-OAAL-v1.0.md + ARC-001-BPCM-v1.0.md
ARCH_REVIEW ← ARC-001-OASEC-v1.0.md + ARC-001-ADMP-v1.0.md [HUMAN GATE]
BUILD       ← task breakdown + OASEC compliance-as-code rules
VERIFY      ← ARC-001-OAGOV-v1.0.md [UAT + compliance]
SHIP        ← ARC-001-OAGOV-v1.0.md [rollback criteria]
REFLECT     ← arckit:health + arckit:traceability → pattern storage
```

The **ARCH_REVIEW** node is a **human gate** — architecture approval is not automated. Everything else is autonomous.

6. Use Case Library

6.1 Australian Sovereign AI (APP/APRA)

Trigger: APP Privacy Act or APRA CPS 230 compliance requirement for AI workloads.

Path: togaf-agent-full → oaa-full → Loop Factory

Key artefacts: AAGI (agent inventory mapped to APP principles), AASE (agent security per APP 11.1), OASEC (compliance-as-code rules for APP 11.2)

Differentiator: Compliance-as-code rules are **executable validation checks**, not PDF policy documents. Auditors get a machine-verifiable evidence trail.

6.2 Cross-Border APAC Deployment

Trigger: Multi-jurisdiction AI deployment (AU + SG + JP, etc.)

Path: togaf-adm-full (baseline) + oaa-full (execution) → Loop Factory

Key artefacts: REQ (dual-jurisdiction requirement IDs), OASEC (per-jurisdiction compliance rules), TRANS (strangle-pattern migration with jurisdiction-specific rollback)

Differentiator: Single source of truth for regulatory requirements (REQ artefact). Change the regulation → all downstream artefacts and compliance rules update automatically.

6.3 Legacy Estate Rationalization

Trigger: Consolidation/retire/replace programme across 50+ applications.

Path: togaf-adm-full alone (no OAA, no Loop Factory)

Key artefacts: APP (inventory), APPR (rationalization decisions), TRANS (migration waves), ACHG (change requests for in-flight work)

Differentiator: APPR is machine-readable — delivery teams can query "which applications are in wave 3?" without parsing a 200-page PDF.

6.4 Greenfield Agentic AI Product

Trigger: New AI product, < 8-week deadline, agile team.

Path: oaa-full alone (skip TOGAF baseline; add it later if scope grows)

Key artefacts: OAAL (sprint map), OAPR (ownership), OASEC (security per sprint), OAGOV (lightweight governance)

Differentiator: Time-to-first-sprint: 1–2 weeks. No quarterly board overhead. Security is embedded per sprint, not bolted on at the end.

7. Comparison: ArcKit EA vs. Traditional EA Tooling

Dimension	Sparx EA / ARIS / Bizz	ArcKit EA
Artefact format	UML/SysML XML, PDF	Markdown (git-native, machine-readable)
Executable	No	Yes (Loop Factory consumes artefacts)
AI-assisted authoring	Limited (Sparx AI)	Full (agent-driven, guided intake)
Traceability	Manual (spend hours maintaining)	Auto-generated (TRAC matrix, per-sprint)
Compliance evidence	Policy PDF + manual audit	Compliance-as-code (executable rules)
Update cadence	Quarterly	Per-sprint (OAA)
Delivery integration	None (separate tool)	Native (LangGraph state machine)
Open source	No	Yes (Apache 2.0)
Cost	\$50–200/user/month	\$0 (open source)
Learning curve	2–4 weeks (model editor)	< 1 day (markdown + CLI commands)

Where traditional tools still win:

- Formal UML/SysML diagram rendering (ArcKit uses PlantUML — strong for deployment/sequence diagrams, but not full SysML profile support)
- Built-in stakeholder management workflows
- Regulatory reporting templates (ArcKit generates raw artefacts; formatting for specific regulatory submissions is a manual step)

8. Implementation Roadmap

Phase 1: Foundation (Weeks 1–2)

- [] Install ArcKit: `arckit init` or Claude Code plugin
- [] Run `togaf-adm-full` (or `togaf-agent-full`) for baseline
- [] Output: 16 doc-types, stakeholder map, capability map, requirements register
- [] **Gate:** Architecture board sign-off on `ADMP` + `REQ`

Phase 2: Tactical Planning (Week 3)

- [] Run `oaa-adm-lite` against TOGAF baseline
- [] Output: `OAA` (5-sprint map), `OAPR` (ownership)
- [] **Gate:** Product owner approves sprint map

Phase 3: Sprint Execution (Weeks 3–12)

- [] Run `oaa-full` per sprint
- [] Output: `OASEC` (security), `OAGOV` (governance) per sprint
- [] Run `arckit:traceability` after each sprint
- [] **Gate:** Per-sprint compliance verification (`OASEC` rules pass)

Phase 4: Continuous Execution (Weeks 3+)

- [] Feed OAA artefacts into Loop Factory
- [] Loop Factory runs: DISCOVER → REFLECT autonomously
- [] `ARCH_REVIEW` = human gate (architecture approval)
- [] **Gate:** UAT + compliance verification per sprint

Phase 5: Operational Maturity (Ongoing)

- [] `REFLECT` pattern storage feeds knowledge graph
 - [] `arckit:health` monitors architecture drift
 - [] Quarterly TOGAF board reviews accumulated OAA evidence
 - [] `ACHG` flow for any out-of-cycle changes
-

9. References

- ArcKit source: github.com/terrygzhou/arc-kit
 - arckit-togaf-adm plugin: [plugins/arckit-togaf-adm/](#) (12 commands, 2 recipes)
 - arckit-oaa plugin: [plugins/arckit-oaa/](#) (5 commands, 1 recipe)
 - Loop Factory: [projects/loop_engineering_factory/](#) (LangGraph state machine)
 - TOGAF ADM: opengroup.org/togaf
 - OAA C208 standard: pubs.opengroup.org/architecture/o-aa-standard
 - Best practice paper: EYW-169 (Obsidian Vault / Eywalink / Architecture)
-

Appendix A: Artefact Quick Reference

File	Type	Plugin	Machine-consumable
ARC-001-ADMP-v1.0.md	ADMP	togaf-adm	Yes (Loop DISCOVER)
ARC-001-BPCM-v1.0.md	BPCM	togaf-adm	Yes (Loop PLAN)
ARC-001-REQ-v1.0.md	REQ	togaf-adm	Yes (Loop DISCOVER)
ARC-001-OAAL-v1.0.md	OAAL	oaa	Yes (Loop DISCOVER/PLAN)
ARC-001-OAPR-v1.0.md	OAPR	oaa	Yes (Loop DEFINE)
ARC-001-OASEC-v1.0.md	OASEC	oaa	Yes (Loop BUILD/VERIFY)
ARC-001-OAGOV-v1.0.md	OAGOV	oaa	Yes (Loop VERIFY/SHIP)
ARC-001-TRAC-v1.0.md	TRAC	shared	Yes (Loop REFLECT)

Appendix B: PlantUML Diagram Sources

- `arckit-ea-layered-architecture.puml` → `arckit-ea-layered-architecture.svg`
- `arckit-ea-plugin-dataflow.puml` → `arckit-ea-plugin-dataflow.svg`

Render (no Java required — public PlantUML server):

```
python3 whitepaper/scripts/plantuml_render.py arckit-ea-layered-architecture.puml
```

*White paper generated by the Enterprise Architecture Lead agent. Source: eywalink-site / whitepaper
(regenerate with `pnpm run whitepaper`)/*