BOHIOSYSTEM

DELIVERABLES AND ADR EXAMPLE · BOHIOSYSTEM

Architecture Deliverables

Exactly what the client receives in an architecture engagement, with a real example of a decision record.

DOCUMENT
BS-SVC-02B

VERSION
1.0

DATE
August 1, 2026

PREPARED BY
BohioSystem

WHAT IS DELIVERED

> **"Architecture design" means very different things depending on who sells it. This document defines exactly what you receive.**

All deliverables are written in Markdown and version-controlled in the project repository, alongside the code. Diagrams are generated from text, so they update through the same workflow as the software and do not fall out of date.

4	1 ADR	Markdown	Text
C4 MODEL VIEWS	PER RELEVANT DECISION	VERSION-CONTROLLED FORMAT	DIAGRAMS AS CODE

01 The four views

We use the C4 model because it separates the level of detail by audience: leadership, the product team, and the technical team do not need the same diagram.

VIEW	ANSWERS	FOR WHOM
Context	What does the system do and who does it talk to?	Leadership, business stakeholders, and anyone new to the project.
Containers	What deployable pieces is it made of?	Technical team and infrastructure owners.
Components	How is each piece organized internally?	The development team building it.
Code	How is a specific point implemented?	Only where it adds value: generated on demand, not maintained.

02 Full deliverables package

Architecture document

- Business context and constraints.
- C4 context, container, and component views.
- Quality attributes with their targets.
- Identified risks and their mitigation.

Decision log

- One numbered file per decision.
- Context, options, decision, and consequences.
- Status: proposed, accepted, superseded.
- Link to the decision that replaces it.

Data model

- Entity-relationship diagram.
- Data dictionary per entity.
- Sources of truth per domain.
- Retention and archiving policy.

Implementation plan

- Phases with their dependencies.
- Entry and exit criteria per phase.
- Effort estimate.
- Risks and contingency plan.

03 What an ADR is and why it matters

An Architecture Decision Record is a short file that records a technical decision and, above all, why it was made.

The value of an ADR is not in the decision, it is in the context. Two years from now, when someone asks, "why did we use this instead of that?", the answer will be written alongside the code, along with the alternatives that were evaluated and what was accepted in exchange. Without that record, the team ends up debating the same thing again or — worse — reverses a decision without understanding what problem it solved.

> RULE OF THUMB
>
> An ADR is written when the decision is **expensive to reverse**: database choice, service boundary, integration protocol, authentication strategy, deployment model. It is not written to choose a date library.

04 A real ADR example

This is the exact format we deliver, with a representative case.

FIELD	CONTENT
Identifier	ADR-014
Title	Modular monolith instead of microservices for the initial phase
Status	Accepted · supersedes ADR-006
Context	The platform will have four functional domains. The team has five people and currently has neither 24/7 operations nor prior experience with distributed systems. Growth is expected, but first-year load is moderate and the boundaries between domains are not yet stable.
Options	**A.** Microservices from the start. **B.** Modular monolith with explicit internal boundaries. **C.** Monolith with no module separation.
Decision	Option B. A single deployment, with four modules with explicit boundaries, internal communication through interfaces, and one database per separate schema.

FIELD	CONTENT
Reason	The boundaries between domains are still being discovered, and getting a microservice split wrong is far more expensive to fix than moving code between modules. Option A adds operational complexity the team cannot sustain today; option C would prevent separating things out later.
Consequences	Positive: simple deployment, full local debugging, transactions without distributed coordination. Negative: no module can be scaled independently, and a serious failure affects the whole process. Accepted in exchange for being able to extract a module into an independent service when its load justifies it, without rewriting the logic.
Review	Reassessed if a module exceeds 60% of total consumption or if the team grows beyond twelve people.

05 Quality attributes

Every architecture sacrifices something. What makes it defensible is having chosen what to sacrifice, and having written it down.

ATTRIBUTE	QUESTION IT ANSWERS	HOW THE TARGET IS SET
Availability	How long can it be down?	Monthly percentage and maximum downtime window.
Performance	How long can an operation take?	95th-percentile response time under a defined load.
Scalability	How far does it grow without a redesign?	Target load at 12 and 36 months.
Security	What does it protect, and from whom?	Threat model and controls by layer.
Maintainability	How much does it cost to change?	Onboarding time for a new developer.
Recoverability	How long to get back up, and how much data is lost?	Recovery time and recovery point objectives.

THE CENTRAL TRADE-OFF

> A system that maximizes every attribute does not exist. Our job is to make the trade-off explicit and have the business — not the technical team alone — approve it.

Let’s talk

NEXT STEP

> Write to us and we’ll schedule a working session at no cost to review your specific case.

CONTACT
contacto@bohiosystem.dev

WEB
bohiosystem.dev

DOCUMENT
BS-SVC-02B · v1.0