Fachhochschule Graubünden
University of Applied Sciences

AISE502: AI in Software Engineering II

Lecture 3: The Supply Side, the Match, and the Decision Record

Script: Part I, Sections 4–6

Agenda

1. Recap: the demand side is built
2. The seven candidates: a first look
3. Tactics: the causal mechanism from structure to quality
4. Deriving a capability rating $c_i(p)$
5. The match: what the construction dictates
6. The three-stage fit procedure
7. The mechanics on a real case: matching the advisory platform (C10)
8. Why the weighted sum fails
9. Recording the decision: ADR and MADR
10. The measurement contract

Recap: where we are

demand $\rightarrow$ **supply** $\rightarrow$ **match** $\rightarrow$ **record** $\rightarrow$ test

Done (weeks 1–2): twelve dimensions; scenarios with response measures; utility tree; weights, shape, constraints – $R(a)$, worked for C10 and the back-office tool.

Today – the remaining three framework elements:

- the **supply side** $C(p)$: seven candidate patterns, rated *by derivation*, not by taste
- the **match** $\mathrm{fit}(a, p)$: three stages, non-compensatory – run live on your project class C10
- the **record**: ADR/MADR and the measurement contract

Important Note

A1 (requirements dossier) is due this week – scenarios, utility tree, $R(\text{platform})$ with weights, shape, and constraints.

The seven candidates (1/2)

The set P of the framework – each in one sentence, with its signature strength and cost:

Pattern (code)	Structure in one sentence	Signature strength	Signature cost
Layered / 3-tier (L)	one deployable, cut into horizontal *technical* layers	simple and cheap: one build, one database	a typical feature cuts across all layers
Modular monolith (MM)	one deployable, cut into *domain* modules with machine-enforced boundaries	monolith economics with modular evolvability	boundaries erode without governance
Hexagonal (HX)	a domain core behind technology-neutral ports with swappable adapters	hermetic testability; swappable technology	indirection that pays off only under change
Microservices (MS)	many independently deployable services, database per service	team scaling; independent deployment	no ACID across services; highest operating cost

You have met four in production dress: Stack Overflow ran L-style, Monzo runs MS, Prime Video's failed design was SL, Segment's rollback was an MS cut.

The seven candidates (2/2)

Pattern (code)	Structure in one sentence	Signature strength	Signature cost
Event-driven (EDA)	components communicate through asynchronous events via a broker	decoupling, elasticity, fault isolation	eventual consistency; end-to-end testing hard
Pipes-and-filters (PF)	data flows through a chain or DAG of processing steps	throughput and reproducibility for batch work	not built for interactive latency
Serverless / FaaS (SL)	functions on managed infrastructure: scale-to-zero, per-execution billing	elasticity; zero idle cost	cold starts; cost inverts under sustained load

Important Note

HX is not a competitor in the same race. It organises dependencies *inside* whichever host it is applied to and **composes** with the other six – a hexagonal modular monolith is *one* coherent design, not two rival ones.

Three levels of design vocabulary

Definition: Architectural pattern (style)

A named, documented **macro-structure**: a topology of elements, permitted relations between them, and known consequences.

Definition: Architectural tactic

An **elementary design decision** that directly influences the response of *one* quality attribute – the atomic unit of architectural design. Examples: *heartbeat*, *redundancy* (availability); *use an intermediary*, *defer binding* (modifiability); *introduce concurrency*, *maintain multiple copies* (performance).

Students routinely conflate the levels – the framework keeps them apart, because its **explanatory mechanism lives exactly in the gap** between them.

Patterns are bundles of tactics

Key Concept

A pattern structurally **privileges** some tactics while **impeding** others – so every pattern helps some quality attributes and hurts others *by construction*, not by accident. The causal chain **topology** $\rightarrow$ **tactics** $\rightarrow$ **quality attribute responses** makes every $++$ and every $--$ explainable and testable.

Example: EDA as a tactics bundle

An event-driven architecture structurally packages *use an intermediary* (modifiability) and *introduce concurrency* (performance) – while structurally *impeding transactions* (consistency). It helps D7 and hurts D4 by construction.

And one demarcation for everything that follows: patterns are evaluated **per subsystem**, not per company logo – hybrids are the normal case.

Deriving a rating $c_i(p)$: three sources, applied in order

A rating $c_i(p) \in \{++, +, \circ, -, --\}$ is an *ordinal* claim: how well does the topology structurally support dimension D_i?

1. **Tactics analysis** – which tactics does the topology make cheap, which does it impede? The causal core: a $--$ that cannot be traced to a structural impediment is an *assertion*, not a rating.
2. **Published ratings** – Richards & Ford's star ratings calibrate ours, five-step to five-step ($5\star \to ++, \ldots, 1\star \to --$); **every deviation is footnoted**. Where no stars exist (HX, SL): primary sources, flagged.
3. **Documented production systems** – calibration points: a rating that contradicts a documented production result must either explain the context difference or **yield**.

Key Concept

A capability rating is a **testable prediction**, not a preference. Supply is constructed with the same rigour as demand.

Micro-derivation 1: why EDA rates ++ on D7 (evolvability)

- **Tactics analysis:** the broker topology *is* the tactic "use an intermediary", built into the macro-structure. A new consumer – a recommendation engine, an audit feed – attaches to the event stream **without touching a single producer**. D7's response measure, *change dispersion*, is structurally minimised: adding functionality is additive, not invasive.
- **Published rating:** Richards & Ford rate the style's evolvability at five stars → ++, no deviation to footnote.
- **Production evidence:** LinkedIn's Kafka ecosystem grew for a decade by attaching new consumers to the same durable log.

Important Note

The rating is a **trade**, exactly as A2 predicts: the same intermediary that decouples producers from consumers also *impedes transactions and hides the workflow* – the costs land on D4 and D9.

Micro-derivation 2: why microservices rate −− on D4 (consistency)

- **Tactics analysis:** the weakness is *structural*. Database-per-service is constitutive of the pattern – so **no ACID transaction spans a service boundary**. An invariant crossing services (order → stock → ledger) must be maintained by a **saga**: a sequence of local transactions with *compensating actions*. Intermediate states are visible; compensation is *not* rollback.
- Against D4's response measures (invariant violations: target 0 for ledgers), the topology impedes the transaction tactic **by construction**. Hence −−.
- **Production evidence:** a documented mitigation exists but is conditional – Monzo makes sagas work under *extreme technological homogeneity*.

The matching procedure handles exactly this case: a **veto** on a High-weight dimension that only a *documented* mitigation can lift.

What ++ and −− look like (selection)

Dimension	What ++ looks like structurally	What −− (or the weak end) looks like
D2 Write scalability	partitioned ingest with elastic consumers (EDA, SL)	one relational database receiving every write (L)
D4 Consistency	one ACID transaction boundary around all state (L, MM)	sagas across service databases (MS); eventual consistency through a broker (EDA)
D5 Availability	bulkheaded services: blast radius one service (MS, EDA)	one process: blast radius 100 % by construction (L, MM: −)
D8 Simplicity & TTM	one repository, one pipeline, shipping this week (L, PF)	platform engineering before the first feature (MS, EDA)
D9 Testability	hermetic domain tests behind ports (HX)	verification only against an integrated environment (L: −); event flows resisting end-to-end tests (EDA: −)
D12 AI integrability	a natural queue, port, and measurement point (EDA, HX, PF)	an honest middle: platform timeouts vs. minutes-long AI runs (SL: ○)

Note the refused −− entries (D6, D12 in the full table): where a weakness is genuinely two-faced or workload-dependent, the honest rating is a **middle** one – the supply-side counterpart of A4's ban on vague requirements.

Native workload shape – and $C(p)$ assembled

Every pattern has a **native workload shape** $S(p)$ – a *type*, not a rating:

L, MM, MS	interactive request/response
EDA	streams and asynchronous flows
PF	scheduled batch
SL	event-triggered, short-lived work
HX	inherits its host's shape (composition pattern)

Definition: Capability profile $C(p)$

$C(p) = \big(c_1(p), \ldots, c_{12}(p);\ S(p)\big)$, $c_i(p) \in \{++, +, \circ, -, --\}$: ordinal ratings of structural support for each dimension, justified through **tactics**, anchored in published star ratings, deviations footnoted – plus the native shape, which feeds the workload-shape **gate** of the match.

Part II derives all seven profiles in full – starting next week.

AI Lens: an LLM component stresses D3, D10, and D12

AI Lens: Axis B preview

Suppose one component is an LLM call. Nothing new is needed – the coordinate system already carries it:

- **D3**: LLM calls cost *seconds* where classical calls cost milliseconds – time behaviour becomes a *structural* concern (queues, asynchronous integration, caching)
- **D10**: per-token pricing makes operating cost a *per-request* attribute – a bad prompt chain is a cost regression the way an $n+1$ query is a latency regression
- **D12**: the pattern either provides the *queue, port, and measurement point* – or it does not; the seven patterns differ sharply on exactly that

The supply side absorbs AI as **ratings on existing dimensions** – assumption A6 at work.

What the construction dictates

Three properties of the profiles are *results of their construction* – and each imposes a requirement on any admissible aggregation:

1. **Constraints are knock-outs.** A violated obligation cannot be traded against merit → eliminate *before* any scoring. The **workload shape** belongs to the same family: a pattern whose native $S(p)$ contradicts the dominant $S(a)$ cannot carry the class's core.
2. **High weights carry vetoes.** A High weight *was defined* as the presence of (H, H) leaves – scenarios whose failure is existential. A pattern structurally weak exactly there fails those scenarios; excellence elsewhere does not un-fail them → **non-compensatory** on High-weight dimensions.
3. **Everything is ordinal.** Ordinal inputs license ordinal outputs – *rankings and exclusions, never percentages* – and every result must be checked for stability under plausible re-weighting.

These three requirements admit **essentially one procedure** . . .

The three-stage fit procedure

Definition: Architecture–application fit $\mathrm{fit}(a, p)$

An ordinal aggregate on the scale $\{++, +, \circ, -, --\}$, determined by a deliberately **non-compensatory, three-stage procedure**:

1. **Knock-out screening.** Hard constraints $K(a)$ eliminate patterns before any scoring. The *workload-shape gate*: if $S(p)$ does not match the dominant $S(a)$, the cell is capped at $\circ$ (subsystem role) – $+$ only for a *constitutive* subsystem of a shape-hybrid class; $-/--$ where it would harm binding scenarios.
2. **Veto rule on High-weight dimensions.** $c_i(p) = --$ on a High dimension caps the fit at $-$; $c_i(p) = -$ caps it at $\circ$ – *unless a documented mitigation exists* (a tactic or hybrid with production evidence), in which case the cell says so and the cap is lifted.
3. **Holistic ordinal reading with mandatory sensitivity analysis.** Survivors are ranked by support of the High set; clustering Medium conflicts can downgrade one step. Result: a *ranking with exclusions*. If it flips under plausible re-weighting, that instability is a genuine **trade-off point** – escalate to scenario-based analysis (ATAM).

Why non-compensatory – and why the veto can be lifted

Non-compensatory mirrors how architectural failure actually works: Prime Video's serverless design was *excellent* on elasticity and deployability – none of that compensated for the cost structure its workload shape imposed.

The mitigation clause keeps the rule honest:

- a veto can be lifted – but only by a *documented* tactic or hybrid with production evidence
- "pod-sharded replication of the monolith" (Shopify) or "event-driven edges around an ACID core" – **never by optimism**

Example: Same class – opposite structures: LMAX and Monzo

LMAX runs core trading on *one* deterministic, event-sourced JVM thread (6M orders/s); Monzo runs banking on $\sim$ 2,800 microservices. **Both** satisfy the class's binding scenarios. Lesson: $R(a)$ alone defines a *feasible set*; the constraints and context in $K(a)$ decide *within* it. A theory pretending to compute a single winner would be falsified by this pair.

The mini-match: R(C10) against L, MM, MS – stage 1

The demand side (from week 2): High on {D6, D7, D9, D10, D12}; shape *hybrid*; EU AI Act and GDPR in K.

Stage 1 – knock-out and shape gate:

- S(C10) is hybrid: an interactive advisory dialogue at the core, batch/asynchronous pipelines (indexing, evaluation) beside it
- all three candidates are natively *interactive* $\rightarrow$ all pass the gate **for the core** – the pipelines will be carried by PF- and EDA-shaped *subsystems* in any design
- no hard constraint eliminates a candidate outright – but note for stage 3: the AI Act's logging and oversight duties *favour* structures where every model call flows through **one auditable path**

The mini-match – stage 2: vetoes on the High dimensions

- **L (layered):** rates − on High-weighted **D7** → cap ○ unless mitigated – and at C10's extreme change rate of models, prompts, and frameworks *there is none*: technical layers give the non-deterministic component no boundary, no queue, no measurement point. **D9** is a second − on a High dimension; D12 offers only ○. Two unmitigated vetoes that *cluster* push the stage-3 reading one step below the cap: −.
- **MS (microservices):** rates −− on High-weighted **D10** → would cap at − – but here the *mitigation clause earns its keep*: C10's cost concern is **per-request AI cost** (tokens, GPU), governed at a *gateway* – orthogonal to distribution. The platform-cost veto relaxes to ○. What stops MS from rising further: synchronous service chains *multiply* seconds-scale LLM latency and failure rates, and the pattern's signature payoff (team scaling) sits on **Low-weighted** D11 – for a student-sized team the microservice premium buys nothing.
- **MM (modular monolith): no veto fires at all** – + or better on every High dimension; its one structural − (D5, blast radius) is Medium-weighted and mitigated by replicated instances.

The mini-match – stage 3 and the verdict

MM leads the High set outright; **HX inside** raises D9/D12 to $++$ – the LLM becomes a mockable, swappable adapter on a port.

High dimension of C10	**L**	**MM ($+$ HX inside)**	**MS**
D6 Security & auditability	$+$	$+$	$\circ$
D7 Evolvability	$-$ *(veto)*	$+$	$++$
D9 Testability & deployability	$-$ *(veto)*	$+$ ($++$ with HX)	$+$
D10 Operating cost	$++$	$++$	$--$ *(veto, relaxed)*
D12 AI integrability	$\circ$	$+$ ($++$ with HX)	$\circ$
Verdict $\mathrm{fit}(\mathbf{C10}, p)$	$-$	$++$	$\circ$

Key Concept

The **hexagonal modular monolith** is the primary recommendation – and it is exactly the architecture of your course project. One profile, three candidates, three stages: a *ranking with exclusions*.

Discussion

Discussion

Run the same three candidates against R(back-office) from week 2 – High on D4, D6, D7, D8, D10. Which veto fires first, and does the verdict change?

Check yourselves: MS now hits **two unmitigated** $--$ **vetoes at once** (D8 and D10, both High for this class); L's strengths sit exactly on the class's High set (D4, D8, D10), and only D7 caps it – liftable by scoping to a small, stable domain. The verdicts become **L** $+$**, MM** $++$**, MS** $--$: the same winner, but a radically re-ordered field.

Why the obvious alternative fails

Why not score 1–5, multiply by weights, add up ($V = \sum_i w_i \cdot v_i$)? The additive form is licensed only under three conditions – **all three fail here**:

1. **Cardinal (interval) scales** – but the ratings are ordinal by construction: what would "microservices score 4.3 on consistency" *mean*?
2. **Preferential independence** – violated *by definition*: A2 says the value of "scalability $= ++$" depends on what happens to consistency; trade-offs *are* preferential dependence
3. **Weights as trade-off rates** – nobody can state, or defend to an auditor, the rate at which audit-trail quality is exchangeable for deployment frequency

The AHP repair inherits documented defects: adding an alternative (even a *copy*) can **reverse the ranking**; Dyer's verdict: "flawed as a procedure for ranking alternatives".

Kept from multi-criteria analysis: the *explication discipline* – criteria on the table, weights argued, options compared. **Dropped**: the arithmetic pretensions.

Beware pseudo-precision

Important Note

A weighted-sum matrix over ordinal ratings produces numbers – “pattern A: 3.87; pattern B: 3.79” – whose significant digits are **artefacts of the procedure**, not measurements of anything. Such numbers end discussions that should be had (the 0.08 gap will not survive any plausible re-weighting) and lend false authority to buried assumptions.

The professional habit this module trains is the opposite:

- report **rankings with exclusions**, and state the **veto** behind each exclusion
- run the **sensitivity analysis**
- if the recommendation flips under plausible weights, you have found a genuine **trade-off point** – a *finding* to escalate to stakeholders, not an error to hide with more decimals

The status of the matrix

Applying the procedure to all pairs yields the 7×10 matrix of Part IV. One point governs how every cell is read:

Key Concept

The matrix is an **explication and communication instrument** – a compressed, teachable heuristic that forces criteria, weights, and assumptions into the open – *not* a computation that determines decisions. The rigorous method behind every contested cell is the **ATAM**: scenario walk-throughs, sensitivity points, trade-off points, risks.

Three honest limits, holding throughout:

1. the scales are **ordinal** – no percentages, ever
2. the ratings are **context-dependent** – the serverless cost rating literally *inverts* with load shape
3. **hybrids are the normal case** – the matrix is read *per subsystem*

The decision is the primary artefact

A1 says: what we choose is a **decision** with high reversal cost – and undocumented decisions *evaporate*:

- the code shows *what* was built, never *why*
- within a few staff rotations the rationale is gone; later changes violate constraints nobody remembers – which is what teams experience as **"legacy"**

ISO/IEC/IEEE 42010:2022 draws the normative conclusion: a conformant architecture description **must** record architecture decisions *and their rationale* (Clause 6.10).

Definition: Architecture Decision Record (ADR)

A short text document – one to two pages, **versioned in the code repository**, one decision per file – with Nygard's structure: **Title** · **Status** (proposed / accepted / deprecated / superseded) · **Context** (the forces at play, value-neutral) · **Decision** (active voice: "We will . . . ") · **Consequences** (positive *and* negative).

Three design choices make the format work – and MADR extends it

1. **Co-location** – ADRs live in the repository, next to the code they govern, not in a wiki that dies with the project office
2. **Brevity** – one decision, one page: a format cheap enough to be used beats a comprehensive one that is not
3. **Immutability** – a superseded ADR is never edited or deleted; a new ADR supersedes it. The record of *why we changed our minds* is often more valuable than the current answer.

MADR adds explicit **decision drivers** and **considered options** (each with pros and cons), plus an optional **confirmation** section.

Key Concept

MADR is this module's format for a structural reason: *decision drivers* + *considered options* is exactly **one row of the matching matrix in narrative long form** – a completed matrix is the tabular compression of many MADRs.

ADR-011: the LLM gateway decision (your project, week 6)

```
ADR-011:  Route all LLM calls through one gateway port

Status:  accepted | supersedes ADR-004

Context:  providers deprecate models on 6-12-month cycles; token costs must be
attributable per request; domain logic must stay testable without paid, non-deterministic
API calls.

Drivers:  D7 evolvability - provider change stays cheap · D9 testability - hermetic tests
· D12 - per-request cost observability

Options:  (1) direct provider-SDK calls from domain services · (2) one gateway behind a
domain-owned port ← chosen · (3) per-feature adapters without a shared gateway

Consequences:  + provider migration is an adapter task; domain tests run against fakes.
− one more runtime component; ∼20-50 ms latency; the gateway needs its own SLO.
```

Confirmation (measurement contract, in embryonic form):

- static rule: no domain module imports the provider SDK (ArchUnit, CI gate, 0 violations)
- eval-harness pass rate $\geq 95\,\%$ on every model/prompt change
- token cost per request $\leq$ budget (p95), monitored continually

Diagrams and the measurement contract

C4 – diagrams for decisions, sparingly: four zoom levels (system context, container, component, code). For this module, **container level dominates**: a modular monolith is *one* container with enforced internal boundaries, a microservice system is *many*, an event-driven system inserts a *broker* container between them.

The measurement contract – the fifth framework element:

- every ADR ends with the **fitness functions and thresholds that would tell us the decision has failed**
- static rules as CI gates · scenario response measures as automated tests · cost and latency budgets as telemetry alarms
- in operation, the four **DORA metrics** test whether the delivery-relevant promises hold
- on breach: a documented evolution path (Strangler Fig) recorded as a *superseding ADR* – never a silent rewrite

AI Lens: an agent drafts the ADR – a human owns the decision

AI Lens: Axis A

ADRs are an ideal task for AI assistance and a **hard boundary for AI authority**. An agent with access to the repository, the utility tree, and the matrix can *draft* a competent MADR in minutes: enumerate options, fill pros and cons from the capability profiles, propose fitness functions. Use that. But the decision itself is an **act of accountability**: a nameable person weighs the drivers, accepts the negative consequences, and answers for them later.

Second-order effect: agents *read* ADRs and convention files on every run – documentation quality is reproduced **at machine speed, in whichever direction it points**. A precise ADR corpus is leverage; a stale one is automated misdirection.

Summary: Part I is complete

1. **Supply:** seven candidates; ratings derived from *tactics*, calibrated against published stars, checked against production – never taste
2. **Match:** knock-outs and shape gate $\rightarrow$ vetoes on High dimensions (liftable only by documented mitigations) $\rightarrow$ ordinal reading with sensitivity analysis; weighted sums fail three preconditions – the matrix is an *explication instrument*, not a computation
3. Worked end to end: C10 vs. L/MM/MS $\rightarrow$ **hexagonal modular monolith** $(++)$ – your project architecture
4. **Record:** MADR – drivers $+$ options $=$ one matrix row; co-located, brief, immutable
5. **Measure:** every ADR ends in a measurement contract; breach $\rightarrow$ superseding ADR

Key Concept

The chain to remember: class $\rightarrow$ scenarios $\rightarrow$ utility tree $\rightarrow$ $R(a)$ $\rightarrow$ non-compensatory match against $C(p)$ $\rightarrow$ ADR $\rightarrow$ measurement contract $\rightarrow$ operation – and, because profiles drift, back around.

This week and next week

Exercise this week

- Requirements workshop II: finalise R(platform) – weights, workload shape, knock-out constraints
- ontology as a contract
- **A1 due: requirements dossier** (scenarios + utility tree + $R(a)$)

Lecture 4

- Part II begins: **Layered, Modular Monolith, Hexagonal**
- problem $\rightarrow$ profile $\rightarrow$ engineering $\rightarrow$ “build it and study it”

Reading

- this week: Part I, Sections 4–6
- ahead: Part II, L/MM/HX

Thank you!

Dr. Florian Herzog

Fachhochschule Graubünden, Chur

AISE502 – AI in Software Engineering II