Fachhochschule Graubünden
University of Applied Sciences

AISE502: AI in Software Engineering II

Lecture 7: The Fit, Formally – Three Cases, the Procedure, the Matrix

Script: Part IV, Sections 30–32, 34; Section 37 (introduction)

Agenda

1. Recap: the match, seen once at small scale – and run by you last week
2. Three matches, three stages: **C6** (gate), **C1** (veto), **C2** (holistic reading and its alarm)
3. The procedure in general – and what a weighted sum would have destroyed
4. The matching matrix: cell semantics and the 7×10 grid
5. Reading the matrix as a whole: columns, rows, five support points
6. The measurement contract, introduced: fitness functions and the DORA metrics
7. This week's exercise: **the design-review gate and Deliverable A2**

Recap: where we are

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

Done (weeks 1–6): Part I – the framework and the C10 mini-match (L $-$, MM $++$, MS $\circ$); Part II – seven capability profiles $C(p)$, the consolidated table, Maxims 3 and 4, ordinal reading only; Part III opened – the C10 profile and the C1/C2 mirror pair (*weights, not dimensions, define a class*).

Last week's exercise: you ran the three stages on your own platform (knock-out $\rightarrow$ veto $\rightarrow$ ordinal reading) and began the ADR – today you will recognise every step.

Today:

- three cases from Part IV, **one per stage**
- the general statement and the **seventy-cell grid**
- the **measurement contract**, introduced

Important Note

A2 is due this week – the architecture dossier (ADR + C4 + measurement contract); the design-review gate closes the design phase; production code only after the gate.

Part IV opens: three matches, three stages

What does the three-stage procedure actually do when it runs? – usually far less than students expect: **most of the work happens before anything is scored**.

- Every Part III class ends with a **verdict and a promise** – a one-sentence primary and secondary recommendation, and the assurance that Part IV computes it through the three-stage procedure. You have seen one (C10) and the C1/C2 sketch; the other rows follow in weeks 8–9
- You own both operands: $R(a) = (w_1(a), \ldots, w_{12}(a);\ S(a);\ K(a))$, $w_i \in \{\mathrm{H}, \mathrm{M}, \mathrm{L}\}$ – from A1 and Part III; $C(p) = (c_1(p), \ldots, c_{12}(p);\ S(p))$, $c_i \in \{{+}{+}, +, \circ, -, {-}{-}\}$ – the seven columns of Part II. You have watched $\mathrm{fit}(a, p)$ run once at small scale (the C10 mini-match); not yet seen: **the machine at full load**
- Cases first, generalisation after (as Parts II and III worked) – three matches computed end to end, each exposing one stage: **C6** – knock-out screening and shape gate (Stage 1) · **C1** – veto rule with documented mitigations (Stage 2) · **C2** – holistic ordinal reading with its built-in sensitivity alarm (Stage 3). Each lands on exactly the verdict its Part III section states

Today:	§30 three cases	§31 the procedure	§32 the matrix	§34 reading the matrix	§37 the contract (introduction)
Lecture 10:	§33 rationales	§35 hybrids and evolution paths	§36 eight-step decision procedure		
Lecture 11:	§37 the contract in depth	§38 organisational complement	§39 limits		

Case 1 – C6 against all seven: the shape gate

The nightly risk run must finish by 06:00, reproduce to the bit, and cost as little as possible: which of the seven patterns can even apply for the job?

- The C6 profile (given here as the operand; Part III, Lecture 9): High on **D2** (makespan reading), **D9** (reproducibility reading), **D10**; shape **scheduled batch**; deterministic seeds as a hard constraint
- Before comparing a single rating, hold $S(\text{C6}) =$ scheduled batch against the native-shape row $S(p)$ of the capability table (deck 6): **only PF proceeds to Stages 2–3**; the table carries the gate and its harm clause, and its verdict column is identical to the C6 row of the matching matrix

Pattern	Native shape $S(p)$	Stage-1 outcome	Verdict
L	interactive	gate caps at ○; harm clause – no answer to makespan or checkpointing, −− against High D2	−−
MM	interactive	gate caps at ○: orchestration codebase around monolithic kernels	○
HX	(host's)	gate caps at ○: ports touch no binding dimension	○
MS	interactive	gate caps; harm clause – communication cost multiplied against High D10	−−
EDA	stream / async	gate caps at ○: job-status glue beside scheduler and DAG	○
PF	scheduled batch	**gate passed** – proceeds to Stages 2–3; no veto on {D2, D9, D10}	++
SL	event-trig., short-lived	gate caps at ○: burst fan-out for communication-light sections only	○

Case 1 – one survivor, and the didactic point

- **Stages 2 and 3 – one survivor.** Only PF reaches the veto stage, and no veto fires: against the High set {D2, D9, D10} it rates $+$, $+$, $++$ (PF column, deck 6) – throughput from data-parallel frameworks, reproducibility by construction, utilisation-driven cost
- Its $--$ on D3 sits on a Low weight and is *inert*; the holistic reading ranks a field of one
- **Result:** $\mathrm{fit}(\textbf{C6}, \textbf{PF}) = ++$**, every other pattern at** $\circ$ **or below** – the C6 row of the matrix, computed almost entirely at Stage 1
- The didactic point generalises: **run the cheapest test first** – six of seven candidates died before a single rating was weighed
- The knock-outs of $K(a)$ belong to the same stage and work the same way: $K(\mathrm{C1})$ eliminates any structure that cannot guarantee an ACID booking core, an immutable audit journal, and ten-plus-year retention – *before* scoring, however well it scales
- The verdict the class's Part III section states (Lecture 9): **pipes-and-filters on HPC/batch infrastructure as primary**, serverless fan-out for bursty, communication-light parallel sections as secondary – with the honest subsystem roles (the $\circ$ cells) stated, not hidden

Case 2 – C1: the gate passes both, the veto rule decides

Two candidates pass the gate, both natively interactive – and one of them cannot commit a transaction across its own internal boundaries: how does the procedure decide core banking?

- From the deck-6 mirror pair: C1 High on **D4, D5, D6, D7, D9**; $K(\text{C1})$: ACID booking core, BCBS 239 / FINMA auditability; shape interactive with batch edges. Contested pair: **MM against MS** – the two columns side by side
- **Stage 1.** Both natively interactive $\rightarrow$ the gate passes both; no hard constraint in $K(\text{C1})$ eliminates either – *a constraint names an obligation, not a pattern*; both can in principle be operated under FINMA-grade audit obligations
- **Stage 2 for MS.** $c_4(\text{MS}) = --$: no ACID transactions across service boundaries; sagas trade atomicity for choreography complexity. D4 is High in C1 $\rightarrow$ **the veto fires and caps the cell at** $-$
- The fact is *structural*: a mitigation can only show that living without the property is *survivable* – yes, but conditional: Monzo (deck 5), roughly 2,800 microservices in production banking, under **organisational scale plus extreme technological homogeneity** (one language, one monorepo, central migration automation)
- Cap lifted **only to** $\circ$, the condition recorded in the cell rationale; MS's $--$ on D8 and D10 sit on Low weights – no further veto fires. **Result:** $\text{fit}(\textbf{C1}, \textbf{MS}) = \circ$

Case 2 – stage 2 for MM, and the stage-3 reading

- **Stage 2 – veto rule for MM.** $c_4(\mathrm{MM}) = ++$ (cross-module ACID transactions) – no veto
- But $c_5(\mathrm{MM}) = -$ on High-weight D5 $\rightarrow$ caps the cell at $\circ$ – *unless a documented mitigation exists*
- It does: **hot-standby replication of whole monolith instances** – the classical banking high-availability tactic, in production at Fineract-class core-banking systems. The cap is lifted.
- **Stage 3 – holistic reading.** MM now stands at $++$ on D4, $+$ on D6, $+$ on D7, $+$ on D9 – support on every High-weight dimension of the class, with the one structural weakness mitigated
- **Result:** $\mathrm{fit}(\textbf{C1}, \textbf{MM}) = ++$
- The ranking MM $\succ$ MS for the C1 core is **stable under plausible weight variation**: it would flip only if D11 (team scaling) rose to High *and* the Monzo homogeneity condition held – exactly what the C1 cell rationale (Section 33, Lecture 10) records as the escalation condition

Case 2 – two kinds of mitigation, and the C1 verdict

Operational weakness

- MM's one-process blast radius (D5)
- *repaired outright* by a standard tactic – hot standby, pod replication

Structural weakness

- atomicity surrendered at the boundary (MS, D4): no mitigation *restores* ACID across service boundaries – sagas buy coordination with compensating actions, not atomicity
- can only be *made survivable*, under a condition most organisations do not meet (Monzo, deck 5) – the premium paid in platform staffing with no gain for typical team sizes

- The division of labour generalises: the veto rule does the heavy lifting, and the **"documented mitigation" clause is where engineering knowledge – not arithmetic – enters the computation**
- The rest of the row follows the same mechanics (Section 33, Lecture 10); in particular **HX – a delta discipline, not a competitor – joins MM at** $++$ by isolating the long-lived booking core from volatile channels and providers

The verdict the class's Part III section states (Lecture 8): a hexagonal modular monolith for the booking core (MM and HX at $++$), EDA at the edges and PF for the batch runs as secondary, and microservices only when organisation size forces D11 to High – the Monzo condition – exactly the C1 row of the matrix.

Case 3 – C2: when scoring cannot separate the survivors

Two finalists carry ++ where it matters and neither dominates: what does the procedure return when scoring cannot separate the survivors?

- C2 (from the deck-6 mirror pair): the **widest High set in the catalogue** – D1, D3, D5, D7, D9, D11 – and a constraint set that knocks out almost nothing; the discrimination work is done by the *weights*, not the constraints
- **Stage 1** requires one honest observation about shape: the class core has **two constitutive paths** – the interactive read path that serves the feed, and the asynchronous fan-out path that delivers posts (the five-second delivery scenario, given here from the C2 profile of Part III, is binding for the class)
- Neither MS (natively interactive) nor EDA (natively stream/async) is shape-foreign to the path it would carry → **the gate passes both**
- **Stage 2** fires one veto against each, and documented practice lifts both: MS's − on High-weighted D3 (mitigation: edge caching, precomputed timelines); EDA's − on High-weighted D9 (mitigation: schema/contract tests, progressive delivery)
- Both reach Stage 3 intact

Case 3 – stage 3: the comparison refuses to close

High dimension (C2)	MS	EDA
D1 Read scalability	++	++
D3 Latency & predictability	(mitigated)	+
D5 Availability & isolation	++	++
D7 Evolvability	++	++
D9 Testability & deployability	+	(mitigated)
D11 Team scaling	++	+

- **Neither dominates**: MS leads where *teams* multiply (D9, D11 – independent deployments), EDA where *consumers* multiply (D3 on the asynchronous path; D7 in its attach-new-consumers reading)
- The **mandatory sensitivity analysis flips the ordering under entirely plausible variation**: weight D11 the way a several-hundred-team organisation must, and MS wins; frame the feed as what it technically is – an eventually consistent, precomputed product of an event flow – and EDA wins
- By Stage 3's own rule, that instability is **not noise**: it marks a genuine trade-off point in the ATAM sense, to be escalated to scenario-based analysis rather than smoothed over

Case 3 – the record refuses the either/or

- Twitter's timeline architecture is **both patterns at once**: fan-out-on-write *is* publish/subscribe – an event flow whose product, the precomputed timeline, is served by independently scaled services
- The honest reading of the instability is not "the procedure failed to pick a winner" but "**the class genuinely needs both patterns, placed**" – the bridge to hybrids (Section 35, Lecture 10), where hybrids turn out to be the normal case, not the exception
- The verdict the class's Part III section states (Lecture 8): an **EDA + microservices hybrid at organisational scale** (MS and EDA at ++), a modular monolith as secondary until that scale is *measured*, not assumed – Mastodon runs the full fan-out mechanics in a Rails monolith – exactly the C2 row

Key Concept

Three cases, three stages, one division of labour. The knock-out screening and shape gate kill most candidates before any scoring (C6: the cheapest test runs first); the veto rule disciplines the High set and prices every mitigation as documented engineering rather than optimism (C1); the holistic ordinal reading ranks the survivors while flagging its own instability as a finding, not an error (C2). *Every one of the seventy cells was produced by exactly this division of labour.*

The formal statement – the deck-3 box, plus one clause

What rule were the three matches following? Stated briefly – every element has already done visible work: $\mathrm{fit}(a, p)$ is the ordinal aggregate of the dimension-wise comparison of $R(a)$ and $C(p)$, same five-step scale, **non-compensatory, three stages**.

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

1. **Knock-out screening and workload-shape gate.** $K(a)$ eliminates before any scoring; $S(p) \neq S(a)$ caps at $\circ$ (subsystem role), $+$ only for a *constitutive* subsystem of a shape-hybrid class, $-/--$ where it would harm the binding scenarios. **Not shown in deck 3 – the gate is read per constitutive path**: a pattern is not shape-foreign to a class one of whose binding scenarios constitutes a path of its native shape (Case 3: C2's fan-out delivery scenario).
2. **Veto rule on High-weight dimensions.** $c_i(p) = --$ on a High dimension caps at $-$; $c_i(p) = -$ caps at $\circ$ – unless a documented mitigation exists (a tactic or hybrid composition with production evidence): the cell says so, the cap is lifted.
3. **Holistic ordinal reading with mandatory sensitivity analysis.** Survivors ranked by support of the High set; clustered Medium conflicts *can* downgrade one step; a *ranking with exclusions*, never "12 % better"; a flip under plausible weight variation marks a trade-off point (ATAM) $\rightarrow$ scenario-based analysis.

Three stages, decreasing hardness – each with a worked face

The three stages are ordered by **decreasing hardness**, and each now has a worked face:

Stage	What it encodes	Worked face
1 Knock-out and shape gate	facts no merit elsewhere can compensate – a violated BCBS 239 obligation, an interactive pattern asked to carry a scheduled-batch core	Case 1 (C6): this stage running the show, emptying six of the row's seven cells on shape alone
2 Veto rule	**Assumption A4**: the High weights come from the (H, H) leaves of a utility tree (deck 2), so a structural failure on such a dimension fails precisely the scenarios that define the class – unless engineering practice has produced a documented way around it	Case 2 (C1): both halves of the rule – a mitigation that *repairs* (MM's hot standby) and one that merely makes *survivable under condition* (MS's Monzo condition)
3 Holistic ordinal reading	deliberately the softest: produces an ordering, and carries a built-in alarm for its own instability	Case 3 (C2): the alarm fired and returned a hybrid rather than a false winner

Why the fit is not a weighted sum

- Deck 3 (Part I): $V(p) = \sum_i w_i \cdot v_i(p)$ presupposes cardinal scales, preferential independence, and weights as trade-off rates – all three violated by ordinal profiles (A2); AHP inherits rank reversal
- **What the three cases add – a demonstration of what the formula would have destroyed: C6** – it would have averaged the shape gate away under good scores elsewhere · **C1** – it would have let MS's missing cross-service ACID be compensated by team scaling · **C2** – it would have manufactured a decimal-point winner exactly where the honest output is a flagged trade-off point
- Kept from multi-criteria decision analysis: the *explication discipline* (criteria, weights, assumptions forced into the open); dropped: its arithmetic pretensions – the matrix is an **explication and communication instrument**, not a computation that determines decisions; behind every contested cell stands **ATAM**, and, where money decides, **CBAM** (utility-response curves, return on investment)

Key Concept

The fit computation is **non-compensatory by design**: constraints knock out before anything is scored, structural failures on High-weight dimensions veto unless a documented mitigation exists, and only then does a holistic ordinal ranking follow – with mandatory sensitivity analysis. *A weighted sum over ordinal profiles would be formally illegitimate and would average away exactly the failures that matter most.*

Cell semantics: what one cell of the grid actually claims

What does one cell of a seventy-cell grid actually claim? A matrix cell answers one precisely delimited question – and misreading that question is the **most common student error** with this instrument.

Definition: Cell semantics of the matching matrix

A cell $\mathrm{fit}(a, p)$ states the fit of pattern *p as the dominant structure of the core* of application class *a* – the pattern that owns the class's binding quality attribute scenarios. It does *not* state whether *p* is useful anywhere in a system of class *a*: hybrid roles at the edges (an event journal beside an ACID core, a batch pipeline beside an interactive product) are stated in the cell rationale, not in the cell value.

- **Consequence 1: a − cell is not a prohibition.** EDA rates − as the dominant structure of a banking core, yet the same rationale names the immutable event journal as the natural regulatory audit trail at that core's edges
- **Consequence 2: the hexagonal column needs a special reading.** HX is a delta pattern of dependency organisation, not a distribution style; it composes with a host (typically MM), and its cells read "as the internal discipline of the class's core"

The 7 × 10 matching matrix

Application class	L	MM	HX†	MS	EDA	PF	SL
C1 Core banking / transactions	○	**++**	**++**	○	−	○	−
C2 Social media / content platform	○	+	○	**++**	**++**	○	○
C3 Back-office / workflow	+	**++**	+	−−	−	○	○
C4 ERP / enterprise core system	○	**++**	+	−−	−	○	−−
C5 E-commerce platform	−	**++**	+	+	+	○	+
C6 Simulation / batch compute	−−	○	○	−−	○	**++**	○
C7 Decision support / BI analytics	+	+	○	−	○	**++**	+
C8 Real-time / IoT streaming	−−	−	○	+	**++**	+	−
C9 Collaboration / messaging	○	+	○	+	**++**	−	−−
C10 AI-native analysis / advisory	−	**++**	**++**	○	+	+	○

Ratings ++ (excellent fit) to −− (structural misfit); **bold** = cells underlying the primary recommendation of each class; shaded rows = the rows you computed today. † HX is a delta pattern – it composes with a host (typically MM), its cells read “as the internal discipline of the class’s core”. L = layered/3-tier, MM = modular monolith, MS = microservices, EDA = event-driven, PF = pipes-and-filters/batch pipeline, SL = serverless/FaaS.

Reading the grid: the rows you computed

- The matrix is the three cases of Section 30, **done seventy times**
- Rows C6, C1, C2: the rows you have just computed; the remaining seven were produced by exactly the same three stages – knock-out and shape gate, then the H-dimension veto with documented mitigations, then the holistic ordinal reading
- Every cell is traceable to $R(a) \times C(p)$ through the per-class rationales (Section 33 – Lecture 10)
- † HX: a delta pattern – it composes with a host (typically MM), and its cells read "as the internal discipline of the class's core"

Key Concept

Read a matrix cell as the answer to one question only: *how well does this pattern serve as the dominant structure of this class's core?* The edges of the same system routinely use patterns whose cell reads ○ or −; the hybrid roles are stated in the rationales, and Section 35 (Lecture 10) shows that **hybrids are the normal case, not the exception**.

Column patterns (1/2): the unfashionable default

What does the grid say as a whole that no single cell can? The matrix rewards a second reading – not cell by cell but by columns, rows, and boundaries.

- Column-wise, the **modular monolith is primary or secondary in seven of ten classes** – not because it is fashionable (it is conspicuously unfashionable) but because most requirements profiles weight consistency, evolvability, cost, and time-to-market higher than independent scaling, and **MM is the only pattern rated + or better on all four** of those dimensions (capability table, deck 6)
- The matrix-level restatement of Fowler's **MonolithFirst**: do not start with microservices even if you expect to need them – stable service boundaries cannot be cut before the domain is understood, and refactoring *between* services is far costlier than *within* a monolith
- **Microservices earn their premium in exactly two situations**, both visible in the grid: where High-weight read scalability, fault isolation, and team scaling coincide (C2, and conditionally C5 and C8) – and nowhere else
- The premium is real and quantified: −− on cost and simplicity; the run-cost side materialises as platform staffing – self-managed Kubernetes TCO **roughly three times** managed offerings, dominated by personnel (deck 5); the two −− cells in the MS column (C3, C4) mark the classes that **pay the premium and collect nothing**

Column patterns (2/2): workload-shaped columns, HX never negative

- **PF and EDA are workload-shaped columns.** Their $++$ cells sit precisely where the class's dominant workload shape matches the pattern's native shape – scheduled batch for PF (C6, C7), continuous stream or fan-out for EDA (C8, C9, and the C2 fan-out)
- The shape gate caps them at $\circ$ – or, by the harm clause, below – everywhere the class core is interactive
- The clearest demonstration that the gate of Stage 1 does real work: **no amount of merit on other dimensions lets a batch pipeline carry an interactive core**
- **The HX column is never negative** – not a free lunch but a property of orthogonality: as a delta pattern, hexagonal architecture composes with the host rather than competing with it, and its cost ($c_8 = -$) surfaces only as capped cells where the change rate is low (C6, C7)

Class	MM	MS	HX	EDA	PF
C1	$++$	$\circ$	$++$	$-$	$\circ$
C2	$+$	$++$	$\circ$	$++$	$\circ$
C3	$++$	$--$	$+$	$-$	$\circ$
C4	$++$	$--$	$+$	$-$	$\circ$
C5	$++$	$+$	$+$	$+$	$\circ$
C6	$\circ$	$--$	$\circ$	$\circ$	$++$
C7	$+$	$-$	$\circ$	$\circ$	$++$
C8	$-$	$+$	$\circ$	$++$	$+$
C9	$+$	$+$	$\circ$	$++$	$-$
C10	$++$	$\circ$	$++$	$+$	$+$

Five columns of the matching matrix

Row patterns and the five empirical support points

- Row-wise: **no class is served above ○ by every pattern, and no pattern serves every class above ○** – Assumption A2 made visible in a single glance at the grid
- If a dominant pattern existed, its column would be uniformly positive, and this part of the module would be one page long
- Cell-wise: the five case-study systems of the module each sit **exactly on a cell boundary** – the empirical support points at which fit and misfit have been *measured in money*

System	Cell it sits on	What was measured
Prime Video (deck 6)	the split serverless cost cell	over 90 % infrastructure cost reduction after consolidating a data-intensive flow into one process
Segment	the MS evolvability cell, read against a wrongly cut boundary	services per configuration instance, not per domain seam
Shopify	the MM write-scalability mitigation	pod-sharded replication
Uber (DOMA)	the MS team-scaling cell	the point where service count itself became the problem
Stack Overflow	the layered read-scalability deviation	cache-friendly read dominance served by roughly nine web servers

Discussion: the C5 row under two variations

The C5 row (e-commerce) for reference: L $-$ · MM $++$ · HX $+$ · MS $+$ · EDA $+$ · PF $\circ$ · SL $+$

Discussion

1. Take the C5 row (e-commerce) and **increase the organisation from 3 teams to 30** while holding traffic constant. Which cells change, through which dimension, and at which stage of the three-stage procedure?
2. Now hold the organisation at 3 teams and **multiply traffic by 50**.
3. Why does the second variation move the row so much less than the first – and what does that say about the popular claim that “we need microservices to scale”?

Key Concept

The modular monolith dominates the matrix as default *not despite but because of* its unfashionableness: most requirements profiles weight consistency, evolvability, cost, and time-to-market above independent scaling. Microservices earn their documented premium only where read scalability, fault isolation, and team scaling are simultaneously High – and the premium is paid in platform staffing either way.

How a decision made this year stays honest in year five

How does a decision made this year stay honest in year five? One case motivates the apparatus.

- What actually triggered the Prime Video re-architecture (deck 6) was **not an architecture review but a telemetry signal**: infrastructure cost per stream, measured continuously, crossed what the team was willing to pay – and that measurement, not an opinion, first forced and then vindicated the redesign
- The cost dashboard was a **fitness function in everything but name**: an objective, continuously evaluated check on an architectural characteristic whose breach converted a running structure from "accepted" into "falsified"
- Empirical anchor for building such checks systematically: DORA's finding that **loosely coupled architectures and teams are the strongest predictor of continuous delivery** – coupling, this theory's leading dimension, is a *measurable* property
- The seventh step of the eight-step decision procedure (Section 36, Lecture 10) generalises the observation into a concept – where this course differs from a classical architecture lecture: the chosen fit is codified as a **measurement contract** – the set of executable invariants under which the architecture is allowed to keep evolving. **"The architecture may change freely as long as the contract stays green."**

Fitness functions – the definition

Deck 3 named the contract as the fifth framework element – today its definition and instruments; depth (taxonomy in CI/CD, the four-layer cascade, Boehm vs. Menzies) in Lecture 11.

Definition: Architectural fitness function

"Any mechanism that provides an objective integrity assessment of some architectural characteristic." Fitness functions turn quality attributes into **executable, objective checks** and move architecture governance from review meetings into the CI/CD pipeline.

Classified along two primary dimensions:

Scope	*atomic* – one characteristic in isolation (e.g. a dependency rule as a unit test) *holistic* – combined characteristics in interplay (e.g. security and data freshness under load)
Cadence	*triggered* – event-based, on every build or deployment *continual* – running permanently in operation (e.g. chaos experiments) *temporal* – time-scheduled (e.g. dependency-freshness time bombs)

Three instrument families

Three worked instrument families recur throughout the module:

Family	Instruments	Scope / cadence	What it makes testable
1 Dependency checks as CI gates (decks 3–4)	ArchUnit (analogues: NetArchTest, dependency-cruiser, import-linter): "the domain layer imports no framework", "no cycles between modules" as unit tests that fail the build; Spring Modulith for declared module boundaries	atomic, triggered	what makes the MM ratings of the capability table *enforceable* rather than aspirational – without automated boundary verification, boundary erosion is the documented failure mode of the pattern
2 Performance and cost budgets as pipeline gates	latency thresholds, bundle sizes, Lighthouse scores declared in a budget file	(pipeline gate)	**Axis B transfer is direct: token-cost budgets and p95 latency budgets per AI use case are the same mechanism with new units**
3 Chaos experiments	Netflix's Chaos Monkey terminates production instances to test resilience assumptions permanently; formalised as the principles of chaos engineering	holistic, continual	verifies the **D5 cells**: a claimed blast radius is a hypothesis until an instance has actually been killed under load

DORA metrics: the delivery layer – and the coupling finding

- The four DORA metrics measure whether the delivery-relevant promises of a structure are being kept: **deployment frequency** and **lead time for changes** (tempo); **change failure rate** and **failed-deployment recovery time** (stability)
- Central empirical finding: elite performers lead on *all four* – **tempo and stability are not a trade-off**
- The strongest single result in the field supports coupling as the leading dimension of this entire theory: "loosely coupled architectures and teams are the strongest predictor of continuous delivery" – in the 2017 analysis, testability and deployability contributed more to continuous delivery than test and deployment automation itself
- High performance is possible with all kinds of systems – including mainframes – provided systems and teams are loosely coupled: the label "microservices" is *neither necessary nor sufficient*
- Second follow-on finding (deck 5, depth in Lecture 11): as team count grows, deployments per developer per day *rise* for high performers and *fall* for low performers

Important Note

DORA's evidence is survey-based and analysed with structural equation models – **prediction, not experimental causal proof**. The theory treats it as the best available large-*n* evidence, to be triangulated against case studies and your own measurements, not as settled law.

This week's exercise: the design-review gate and Deliverable A2

Project Link: Portfolio Intelligence Platform

The design phase closes this week. Finalise the solution design:

1. **Service cut and contracts** – bounded contexts → service decomposition and the contracts between the deterministic core, the edges, and the AI subsystem
2. **Walking-skeleton plan** – the thin end-to-end slice you will build first in Sprint 1
3. **Measurement contract** – in today's vocabulary: a *token budget* (family 2: cost budget as a pipeline gate), an *eval threshold* (pass rate on the golden set before rollout), and *module-boundary checks* (family 1: dependency rules as CI gates, ArchUnit / import-linter style)
4. **Design-review gate** – defend the ADR: which vetoes fired, which mitigations are *documented*, what the sensitivity check showed

Deliverable A2 (end of week 7): architecture dossier – ADR + C4 diagram + measurement contract. Production code starts only after the gate (exploratory spikes are allowed).

From week 8 the exercise slot becomes a one-lesson (one-hour) standup/coaching session and the lecture grows to three lessons; Sprint 1 (M3, walking skeleton) begins.

Summary

1. **Three cases, three stages**: C6 – the gate empties six of seven cells before scoring; C1 – the veto rule decides, mitigations priced as documented engineering; C2 – the reading flags its own instability and returns a hybrid
2. **Two kinds of mitigation**: an operational weakness is repaired outright (hot standby); a structural one is only made survivable under a condition most organisations do not meet (Monzo)
3. **The formal statement**: $\mathrm{fit}(a, p)$ – ordinal, non-compensatory, three stages of decreasing hardness; the gate is read per constitutive path; a weighted sum would have destroyed exactly the three decisive facts
4. **Cell semantics**: a cell rates the pattern as the dominant structure of the core only; $-$ is not a prohibition; HX reads as the internal discipline of the core
5. **The grid**: MM primary or secondary in seven of ten; MS premium in two situations only; PF/EDA workload-shaped; HX never negative; no row or column uniformly positive (A2)
6. **Five support points**: Prime Video, Segment, Shopify, Uber, Stack Overflow each sit on a cell boundary – fit and misfit measured in money
7. **Measurement contract**: fitness functions (scope $\times$ cadence; dependency checks, budgets, chaos) and the four DORA metrics; *the architecture may change freely as long as the contract stays green*

Next week

Lecture 8 (week 8, 3 lessons) – Part III: application classes C1–C5

- challenges → requirements profiles → what real systems chose
- the rows C1–C5 of today's matrix, derived from the demand side

Reading

- this week: Part IV, Sections 30–32, 34; Section 37 (introduction)
- ahead: Part III, Sections 18–23

Exercise / deliverable

- **A2 + design-review gate: this week**
- from week 8: Sprint 1 / M3 – walking skeleton (`MarketDataService` + minimal `ResearchAgent` + stable API); the exercise slot becomes a one-lesson standup/coaching

Thank you!

Dr. Florian Herzog

Fachhochschule Graubünden, Chur

AISE502 – AI in Software Engineering II