

Fachhochschule Graubünden
University of Applied Sciences

AISE502: AI in Software Engineering II

Lecture 13: Threats, the Shifted Matrix, Agent Orchestration – and Synthesis

Script: Part V, Sections 42.6–42.7, 43–45

Agenda

1. Axis B completed: a new threat class – **OWASP LLM Top 10**, prompt injection; regulation as a hard constraint – the **EU AI Act**
2. How AI **shifts the matrix**: the C10 row cell by cell; D12 across the seven patterns; five shifted cells; MLOps maturity; A6 restated
3. **Agent orchestration** – the emergent eighth pattern: the advisor workflow; agent vs. workflow; topologies and classical analogues
4. The economics of autonomy and the default rule; a capability-profile sketch
5. Synthesis: one theory, five parts – **Maxim 8**; one discipline at two binding sites
6. Exam orientation
7. This week's exercise: **M5 closes** – eval harness in CI, guard, cost observability; threat model incl. prompt injection via news + basic hardening; optional distinction work (Kür)

Recap: where we are

- Part V is the framework's stress test on two axes: **Axis A** – AI in the process (agentic tools, verification bottleneck, guardrails); **Axis B** – AI in the product (components that are non-deterministic, fallible, latency-heavy, priced per call)
- Lecture 12: Axis-A evidence – two RCTs and their resolution; the **verification bottleneck** (Maxim 7: generation is cheap, verification and architecture are binding – D7 and D9 gain weight); Axis A compact: documentation as control interface, fitness functions as operating licence, who merges answers; Axis B: sentiment call wired wrong and right; three component types; SE4AI classics; reference architecture (**LLM gateway** as single measurement point); eval-harness basics
- The C10 verdict has been computed **three times**: mini-match (L, MM, MS – Lecture 3), profile and real systems (Lecture 6), the full row with its cell rationales (Lecture 10; the 7×10 matrix itself was read in Lecture 7)
- Lecture 6 filed the outlook: agent orchestration is a *composition pattern* reusing the catalogue's topologies – workflows before agents ($15\times$ tokens). Today that claim is paid out
- Today: Axis B completed (threats, regulation) $\rightarrow$ how AI *shifts* the matrix $\rightarrow$ the eighth pattern with a profile sketch $\rightarrow$ synthesis and exam orientation. **The last lecture of new material**; week 14: one lesson synthesis and exam hints, three lessons final presentations, architecture defence and peer reviews (A3, M6)

tenth class (done) – twelfth dimension (done) – shifted cells (today) – one composition pattern (today)

A new threat class – OWASP Top 10 for LLM Applications (1/2)

AI components add an attack surface that classical threat models do not cover.

The OWASP Top 10 for LLM Applications 2025 codifies it; the script pairs each risk with its **architectural** counter-measure – deliberately, because the defence is *structural, not model-internal*.

ID	Risk	Architectural counter-measure
LLM01	Prompt injection (direct and indirect)	Defence in depth: privilege separation, output validation, human-in-the-loop for sensitive actions
LLM02	Sensitive information disclosure	Data minimisation in prompts; output filtering at the gateway
LLM03	Supply chain	Vetting of models, weights, and dependencies; registry discipline
LLM04	Data and model poisoning	Data governance and provenance for training/index data
LLM05	Improper output handling	Treat output as untrusted input: schema validation, encoding, ontology guard

A new threat class – OWASP Top 10 for LLM Applications (2/2)

ID	Risk	Architectural counter-measure
LLM06	Excessive agency	Least-privilege tool design; deterministic services own irreversible actions
LLM07	System prompt leakage	No secrets or authorisation logic in prompts
LLM08	Vector and embedding weaknesses	Access control and tenant isolation on the retrieval index
LLM09	Misinformation	Grounding with mandatory citations; domain-axiom checks
LLM10	Unbounded consumption	Rate limits, token budgets, cost circuit breakers per tenant

Most counter-measures are elements of the gateway architecture of Lecture 12.

Prompt injection – why the model cannot solve it

Important Note

- Prompt injection is **not fully solvable inside the model**, because LLMs process instructions and data in the *same channel*
- Any document, e-mail, or web page the system reads can carry instructions (“ignore your previous rules and . . . ”) – and no reliable in-model separator exists
- The defence is therefore **defence in depth at the system level**: least-privilege tools (LLM06), output validation (LLM05), human approval for consequential actions, consumption limits (LLM10)
- This is the security-flavoured restatement of the section’s design rule: **the architecture, not the model, is the trust boundary**
- For the course project, concretely: no LLM output may reach the booking path without passing the **ontology guard**; no agent tool may perform an **irreversible action**

Regulation as a hard constraint: the EU AI Act

Regulation closes the quality-attribute loop with legal force – **Regulation (EU) 2024/1689**, the AI Act, entered into force on 1 August 2024; a risk-based approach with four classes:

Class	Examples / duties
Unacceptable risk	prohibited practices, e.g. social scoring
High risk	Annex III use cases: creditworthiness assessment, employment, critical infrastructure – duties: risk management, data governance, documentation, logging, oversight, accuracy/robustness/cybersecurity
Limited risk	transparency duties: labelling AI interaction and generated content
Minimal risk	–

Date	What applies
2 Feb 2025	prohibitions; AI-literacy duties
2 Aug 2025	governance; GPAI duties
2 Aug 2026	general applicability incl. Annex III high-risk
2 Aug 2027	high-risk AI in regulated products

AI Act obligations – K(a), not weights

- For this theory the AI Act has a precise, limited role: its obligations are **quality attributes with legal force** that enter the requirements profile as **hard constraints** $K(a)$**, not as weights** – Part I: constraints are knock-out filters, never averaged away
- A finance-related advisory platform – class C10, particularly with any *creditworthiness* bearing – can fall into the **high-risk** class
- Then logging of agent steps, technical documentation, human oversight, and demonstrated robustness stop being engineering preferences and become **conditions of legal operation**

Obligation	**Architectural element it lands on**
logging	gateway telemetry + audit journal
human oversight	human-in-the-loop interfaces at the determinism boundary
robustness	fallback chains + eval harness

Key Concept

Compliance, correctly designed, is not a parallel work stream – **it is the same architecture, documented**.

How AI shifts the matrix

You have computed the C10 verdict three times – what were those computations doing to the rest of the matrix?

- The verdict itself needs **no fourth derivation** – the three computations of the recap (Lectures 3, 6, 10)
- Two of the four absorption forms on the recap's map – the **tenth class** and the **twelfth dimension** – are exactly the artefacts those computations used
- This section supplies the generalisation in three steps:
  - (i) the *supply-side* reading of the row you own
  - (ii) the full **D12 column** it exercised
  - (iii) the **cells that moved** – cell by cell, with stated and measurable reasons
- Then: when the pipeline promise is real (MLOps maturity) – and A6 restated as a falsifiable claim

The C10 row – which D12 mechanism each cell exercises

Start from the row you own (Lecture 10) – the one reading the three computations used but never stated in one place: *which D12 mechanism each cell exercises*.

Cells	Fit	D12 mechanism exercised
MM, HX	++	**boundary and port**: a CI-verifiable module boundary and an anti-corruption adapter on a port contain a fallible, entangled component
EDA, PF	+	**queue**: asynchronous absorption of latency, rate limits, and outages; pipeline-shaped ingestion and evals
L	–	none of the three – no queue, no port, no measurement point
MS	○	seconds-scale, fallible calls in synchronous chains (the missing queue)
SL	○	minutes-long LLM and solver work against platform timeout ceilings

Key Concept

Recommendation (Part IV, ADR-007): a hexagonal modular monolith plus pipelines and an orchestrated agent workflow, with EDA as the secondary job/audit spine – governed by token and latency budgets and the determinism boundary: *agents propose; deterministic services decide and book*.

D12 across the seven patterns

The D12 row of the Lecture-6 consolidated table, now with tactic-level rationales – D12 (AI integrability) measures whether the pattern naturally provides **the queue, the port, and the measurement point** a slow, fallible, per-call-priced component requires.

Pattern	D12	Rationale
L – Layered	○	technical layers give the non-deterministic component no boundary, no queue, and no measurement point of its own
MM – Modular monolith	+	a dedicated AI module with a hard, CI-verifiable interface contains the component cheaply
HX – Hexagonal	++	the LLM is an adapter on a port: swappable, mockable, contract-guarded – the ACL discipline structurally built in
MS – Microservices	○	per-service isolation helps; synchronous chains through seconds-scale calls hurt – net neutral
EDA – Event-driven	++	queues absorb exactly what LLMs are worst at: latency, rate limits, outages; natural batching points
PF – Pipes-and-filters	++	ingestion, training, and eval pipelines are pipes-and-filters by construction
SL – Serverless	○	event-glue around batch AI APIs fits; platform timeout ceilings collide with minutes-long LLM/solver runs

Which existing cells shift, and why (1/2)

Beyond the new row and column, AI as a runtime component moves *existing* evaluations in stated directions – all five visible in the D12 ratings, each carrying a measurable reason:

1. **Asynchronous patterns gain (EDA, PF ↑).** Queues and pipelines absorb what LLMs are worst at – latency, rate limits, outage – and ingestion and eval pipelines are pipes-and-filters by construction
2. **Hexagonal gains most (HX ↑).** The ACL/port discipline is exactly what the CACE problem demands; Assumption A1's cost-of-change criterion bites hardest at *model replacement*; testing against deterministic fakes is the only way to keep the deterministic 95 % of the system deterministic
3. **Synchronous distributed chains lose (MS ↓ where LLM calls sit in the request path).** Seconds-scale latency and per-hop failure probability multiply along the chain; without constitutive stability patterns this is a cascade design

Which existing cells shift, and why (2/2)

4. **Serverless is conditional (SL $\sim$).** Platform timeout ceilings against minutes-long LLM and solver runs cap it; event-glue around batch APIs remains a fit
5. **A cost dimension becomes load-bearing everywhere.** Cost per request, feature, and tenant is a runtime quality attribute with no counterpart in classical profiles; it belongs in the gateway and in CI budgets. Routing across cheap and expensive models is the new incarnation of a classical resource-arbitration tactic: **cascades** up to $\sim$98 % cost reduction at comparable quality; **learned routers** more than 2$\times$ cheaper without quality loss

Key Concept

The matrix does not get *rewritten* by AI; it gets *shifted* – in **five stated directions, for five stated and measurable reasons**. A student who can name, for any cell movement, the quality-attribute mechanism behind it (which tactic the pattern bundles or impedes for a slow, fallible, per-call-priced component) has understood both Part IV and Part V.

MLOps maturity: the three-level ladder

For type-(b) components the PF cells' promise (the D12 $++$) is realised only at sufficient process maturity – the canonical three-level ladder:

Level	Characteristics	Key trait
Level 0	manual, script-driven, interactive; data science and operations separated; releases rare, no CI/CD, minimal monitoring – *the documented reality of many teams*	model handed "over the fence" as an artefact
Level 1	automated ML pipeline with continuous training; automated data and model validation, triggers, metadata store, feature store	*the pipeline, not the model, is the deployment artefact*
Level 2	CI/CD automation of the pipeline components themselves	fast, reliable experiment-to-production cycles

Nine consolidated MLOps principles: CI/CD automation, workflow orchestration, reproducibility, versioning of data/model/code, collaboration, continuous training and evaluation, metadata tracking, monitoring, feedback loops.

When the pipeline promise is real: the fit-theoretical reading

Key Concept

Fit-theoretical reading: the MLOps level describes how much of **D9** (testability/deployability) and **D12** the organisation can actually *cash in*.

- A Level-0 team holding a $++$ pattern rating realises little of it
- This is the **Axis-B echo of DORA's Axis-A finding** (Lecture 12): *guardrail maturity, not tool adoption*, converts potential into performance

Assumption A6 restated as a falsifiable claim

Three of the four absorption forms are now on the table, each *computed rather than asserted*: the tenth class, the twelfth dimension, the shifted cells – the fourth, the emergent composition pattern, follows next.

Key Concept

Assumption A6 restated as a falsifiable claim.

- Runtime AI components are non-deterministic, fallible, latency-heavy, per-call-priced, and subject to drift and vendor deprecation
- They *stretch* existing quality dimensions by orders of magnitude and add sub-attributes: token cost per request, eval pass rate, provider deprecation risk, prompt-injection resistance
- What does *not* change is the method: scenarios with response measures, tactics, trade-off analysis, ADRs, fitness functions
- The theory absorbs AI – as a tenth application class, a twelfth profile dimension, shifted cell values, and one emergent composition pattern – instead of being reinvented for it
- The quality gate is carried by the one new test-artefact class A6 named from the start: the **eval harness**

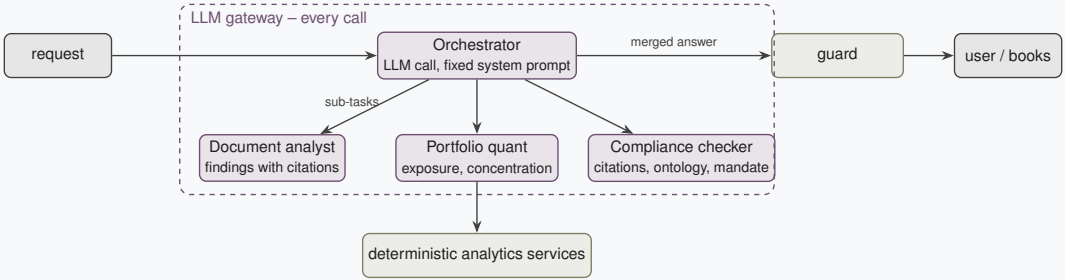
Agent orchestration – the emergent eighth pattern

“Should I reduce my exposure to European industrials?” – no single model call can answer that responsibly, so what structure can?

- The final structural novelty of the AI era is the **orchestration of agents**
- As everywhere in this script, **the concrete system comes before the catalogue**: first the advisory workflow the course project actually builds, then the name of what it is an instance of

Rhythm of this block: case → what an agent is and is not → topologies and classical analogues → choosing a topology → economics and the default rule → capability-profile sketch → project link

Case: the course project's advisor workflow



Orchestrator (an LLM call with a fixed system prompt) splits the request into sub-tasks, delegates, then merges the results into one grounded answer. **Document analyst**: extracted findings from the ingestion corpus *with citations*; **portfolio quant**: exposure and concentration numbers by calling only deterministic analytics services – *arithmetic is not a job for a language model.* **Compliance checker**: verifies the draft – every cited passage exists, every entity resolves against the ontology, every mandate constraint holds.

Strip the labels: a shape you know cell by cell

- Every call by every participant flows through the **LLM gateway** – routed, cached, budgeted, logged – and nothing any agent produces reaches the user or the books without passing the **guard**: *agents propose; deterministic services decide and book*
- Now strip the labels: a coordinator decomposing work for specialised workers is the **mediator topology of event-driven architecture** (Lecture 5); the fixed retrieve–extract–check sequence inside each specialist is a **pipeline** (Lecture 6); peers coordinating over shared context would be the **broker topology**
- This is the claim the Part II outlook (Lecture 6) filed in advance; this section pays it out – the full six-row mapping follows

Key Concept

The section's deliberately *deflationary* claim: agent orchestration is **not a new architectural style** but a **composition pattern for non-deterministic runtime components** that reuses the topologies of the seven patterns you already know – which is why it can be evaluated with the **rating grid you already have**.

What an agent is – and is not

Definition: Agent; workflow vs. agent

An *agent* is an LLM running in a loop with tools and state: at each iteration the model observes the current state (conversation, tool results, working memory), selects an action (a tool call or an answer), and the loop executes it and feeds the result back – until a termination condition holds. Schematically: **agent** = **loop** + **tools** + **state**. The load-bearing distinction: a **workflow** orchestrates LLM calls and tools along *predefined code paths* – deterministic structure, non-deterministic building blocks; an **agent** lets the model *steer its own process and tool use* – the control flow itself becomes model output. Autonomy is therefore not a binary but a **dial**, and every notch on it costs latency, tokens, and testability.

Engineering guidance of the source that defined this vocabulary – matching the module's philosophy verbatim: **find the simplest solution possible** · **prefer simple, composable patterns over frameworks** · **escalate to agents only when the task genuinely requires open decision paths**.

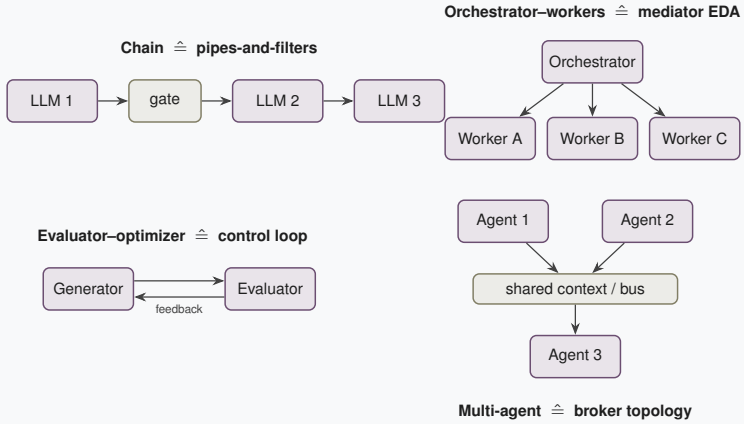
Orchestration frameworks that model workflows as *explicit graphs* make the topology **inspectable** – an architectural virtue for the same reason a C4 diagram is (Lecture 3).

Topologies and their classical analogues

The Lecture-6 outlook table, now with its third column: every workflow topology maps exactly onto a structure from Part II that you know cell by cell – and every property the classical pattern is known for (and every weakness) *transfers*.

Topology	Classical analogue	Structure and use
Prompt chaining (chain)	Pipes-and-filters (PF)	each call processes the previous output; programmatic gates between stages; fixed decomposition
Routing	Routing layer / mediator	a classification step directs inputs to specialised prompts or models; the runtime sibling of model routing
Parallelisation (sectioning, voting)	Broker-style fan-out	independent subtasks in parallel, or repeated runs with majority vote
Orchestrator–workers (tree/-graph)	Mediator EDA	a lead model decomposes the task dynamically and delegates to workers; central workflow control
Evaluator–optimizer	Feedback control loop	generator and evaluator model iterate until a quality criterion holds
Autonomous multi-agent	Broker topology	peer agents coordinate over shared context; maximal flexibility, minimal central control

Four topologies, drawn – gates in code, judgement in the model



Deterministic gates (green) between non-deterministic stages (violet) are the **workflow discipline**: *structure stays in code, judgement stays in the model.*

The evaluation logic applies unchanged – sharpened along three axes

Because the topologies are the old topologies, the evaluation logic of this theory applies **unchanged** – sharpened along exactly three axes:

- **Non-determinism** – testability is read as *eval coverage* (the eval harness of Lecture 12)
- **Token economics** – cost per *request*, not per infrastructure
- **Fallibility** – fault tolerance is read as guardrails, evaluator loops, and deterministic fallbacks behind ports

Consequence: the choice of topology can be compressed into the same style of decision aid the matrix provides – next frame.

Choosing a topology from the task signature

Read top-down and **stop at the first matching row** – the ordering encodes “simplest structure first”.

Task signature	Topology	Cost/risk note
Fixed decomposition; every intermediate result verifiable	Prompt chain (workflow)	cheapest and most testable; deterministic gates between stages
Heterogeneous input categories with specialised handling	Routing	cheap; the router itself needs its own eval
Broad, parallelisable subtasks; breadth-first search	Parallelisation or orchestrator–workers	token cost multiplies with worker count ($\sim$15$\times$ class)
Output must clear a measurable quality bar	Evaluator–optimizer	latency and cost grow per iteration; needs a reliable evaluator
Path genuinely unknown; open-ended tool use	Agent	highest cost and risk; guardrails, budgets, and oversight mandatory

Project Link: Portfolio Intelligence Platform

Your Axis-B ADR must justify your topology *against this table*.

The economics of autonomy – and the default rule

- The rigour case for restraint, quantified by the best-documented production account (previewed in the Lecture-6 outlook): Anthropic's multi-agent research system – an orchestrator–workers design – beat a single agent by **90.2 %** at roughly **15× the tokens** of a chat interaction (simple agents ∼4×); token use alone explains **80 %** of the variance
- A textbook **CBAM decision** in the sense of Part IV: autonomy is bought with cost, latency, and error accumulation – justified only where the task's utility-response curve clears the price: broad, parallelisable research questions do; a form-filling workflow does not
- Regulation closes the loop from the other side: the AI Act's logging, documentation, human-oversight, and robustness duties attach to *exactly the autonomy this section prices*

Key Concept

The default rule for agent architecture. Workflows before agents; the simplest structure first; autonomy only on demonstrated need; every escalation an ADR with a measurement contract. *An orchestration decision without a token budget and an eval threshold is an opinion – Maxim 6 applies to agents without modification.*

Agent orchestration – capability-profile sketch

Dimension	Rating	Ground
D1 Read scalability	○	state small and replicable; throughput bounded by provider rate limits
D2 Write scal. & elasticity	○	fan-out cheap in structure, expensive in tokens; volume quota-bound
D3 Latency & predictability	−−	seconds per step, steps multiply; open-ended loops have no tail bound
D4 Consistency & integrity	−−	non-deterministic by construction; integrity only outside, in deterministic validation
D5 Availability & isolation	−	each step adds provider failure modes and error accumulation; evaluator loops and fallbacks mitigate
D6 Security & auditability	○	logging natural (+), legally required in high-risk uses; injection, excessive agency widen the surface (−)
D7 Evolvability	+	prompts, tools, models swap behind contracts; churn claws part back
D8 Simplicity & TTM	−	a chain workflow is genuinely simple; every notch of autonomy is not
D9 Testability & deployability	−−	exhaustive testing impossible; evals with statistical thresholds replace assertions
D10 Operating cost	−−	token cost per request orders of magnitude up; multi-agent $\sim 15\times$ chat
D11 Team scaling	○	sub-agent and tool ownership parallelises teams moderately (PF analogy)
D12 AI integrability	++	it *is* the composition pattern for AI components – its own best host
Status	provisional (state 2025/26) – every cell carries its own temporal fitness function; default hypotheses, re-verified on every model generation	

Agent orchestration – reading the sketch

- **Why only a *sketch* – three caveats:** no Richards–Ford star ratings; no decade of production case studies; every cell is a **default hypothesis** in the sense of the theory's own limits discussion – to be replaced by measurement and **re-verified on every model generation**
- The cells read, as always, "as the dominant structure of the subsystem it governs" – here the **AI subsystem**, not the whole platform
- **The reading:** the profile explains at a glance why agent orchestration can never be the dominant structure of a whole platform of class C1–C9 – it is **vetoed by every High weight on D3, D4, D9, or D10** (stage 2 of the three-stage match, Lecture 7)

Key Concept

Agent orchestration is, and remains, an **edge pattern** – hosted behind the ports of a deterministic core, exactly where the C10 recommendation places it.

Project link: Axis B is what you build – and what you build it as

Project Link: Portfolio Intelligence Platform

Axis B is *what* you build; the advisor workflow – an orchestrator with two to three specialised sub-agents (document analyst, portfolio quant, compliance checker – in the exercise sheet: `ResearchAgent`, `RiskAgent` / `OptimizationAgent`, and the ontology guard), all behind the LLM gateway – is what you build it *as*. The graded Axis-B deliverables are the **containment artefacts**:

(i) the **gateway** with fallback chain and per-request **cost observability** – cost per request, per feature, reported on a dashboard and enforced as a CI budget (model routing itself is distinction work)

(ii) the **ontology guard** – every extracted entity resolves against the deterministic data store, every cited passage exists, portfolio axioms hold

(iii) the **eval harness** (Lecture 12 listing) wired as a **CI gate**

(iv) one **ADR** that justifies your chosen orchestration topology against the task-signature table, with its **token budget and eval threshold** as the measurement contract

Sub-agents propose; your deterministic services decide and book.

What did AI change? Two temptations, one position

What, in the end, did AI change? – the bet of the opening can now be settled. Architecture teaching in the AI era faces **two symmetric temptations**:

Denial

- generative AI as a passing tool fashion that leaves software engineering untouched
- **falsified by the adoption data alone**: **90 %** of practitioners report using AI at work by 2025

Exceptionalism

- AI systems as a new discipline with new vocabulary, new roles, new decision logic
- **rejected on the assembled evidence**: nothing AI does, on either axis, required a decision no ADR can record, a correctness no response measure can capture, or a structure no tactic vocabulary describes

Key Concept

Between the temptations lies the position defended since Part I: **absorption**.

The script read backwards as a single argument (1/2): Parts I–III

Part	What it established	Artefact
Part I	architecture is the set of significant, hard-to-reverse decisions (**A1**); everything is a trade-off (**A2**); quality attributes – not features – drive structure (**A3**); requirements decide anything only as measurable scenarios (**A4**)	the **twelve dimensions** on which all later judgements run
Part II	turned seven patterns into **capability profiles** by explaining every rating through the tactics a pattern bundles or impedes	the consolidated **capability table**
Part III	turned ten application classes into **requirements profiles** – recurring bundles of architecturally significant requirements with response measures and hard constraints	the **requirements table**

The script read backwards as a single argument (2/2): Parts IV–V

Part	What it established	Artefact
Part IV	matched them: knock-out screening, veto rule, holistic ordinal reading; **hybrids are the normal case**; the decision procedure ends in an **ADR**, and every ADR ends in a **measurement contract** – because a decision is a hypothesis tested over the life cycle (**A5**)	the 7×10 **fit matrix**
Part V	subjected the whole construction to its hardest contemporary stress test – **and the construction held** (**A6**)	Axis A / Axis B outcomes

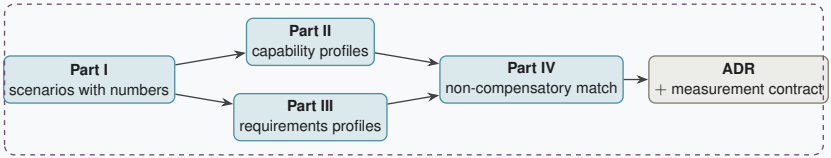
- **Axis A** raised the value of the theory's artefacts – documentation as agent context, fitness functions as operating licence – rather than obsoleting them
- **Axis B** was absorbed as one class, one dimension, five cell shifts, and one composition pattern whose profile the theory's own grid can express

Maxim 8 – the pipeline of this script in one sentence

Key Concept

Maxim 8. The theory absorbs AI: a tenth application class, a twelfth dimension, shifted cells, one emergent composition pattern – *same assumptions, same procedure, same contract*. The pipeline of this script is one sentence long:

scenarios with numbers (Part I) meet capability profiles (Part II) and requirements profiles (Part III) in a non-compensatory match (Part IV) whose result is an ADR with a measurement contract – and nothing about AI, on either axis, changes a single step of it (Part V)



Part V: nothing about AI, on either axis, changes a single step

The deepest result: one discipline at two binding sites

One symmetry deserves to be made explicit – the deepest result of Part V:

Axis A concluded

- *generation* is cheap
- *verification* is the bottleneck
- so the **SDLC** must be rebuilt around machine-checkable specifications, guardrails, and human accountability

Axis B concluded

- *model output* is cheap
- *validation* is the bottleneck
- so the **runtime** must be rebuilt around contracts, ontology guards, eval harnesses, and a human-owned determinism boundary

Key Concept

These are the **same conclusion at two different binding sites**: whether the LLM sits in the toolchain or in the product, the discipline it demands is **containing cheap, fallible generation behind expensive, explicit verification** – which is, on reflection, what this module has called *architecture* all along.

Discussion: is it one discipline?

Discussion

Close the module with the symmetry. Consider the artefact pairs across the two axes:

Axis A artefact	Axis B artefact
`AGENTS.md`	the system prompt and ontology
the CI fitness function	the runtime ontology guard
the code-review obligation	the human-oversight duty of the AI Act
the test suite as the agent's operating licence	the eval harness as the model's operating licence

- For each pair: is this genuinely *one* engineering discipline observed at two binding sites – or are there differences of kind, in **reversibility**, in **accountability**, in **failure semantics**, that the symmetry hides?
- And if it is one discipline: which parts of this script would survive the next order-of-magnitude improvement in model capability – and which would you expect to rewrite?

The written examination – the facts

Written examination (50 %)

- **50 %** of the module grade – the project is the other 50 %
- **End of semester, 60 minutes**
- **Open book:** script and own notes, *on paper*
- **Closed internet**
- Focus: **architecture reasoning – profiles, matching, trade-offs**
- Language: all materials, the script, and the exam in English

Six learning objectives, five parts

Learning objective (Lecture 1)	Where the script settles it
1. explain why architecture selection is a **matching problem** – no pattern is good or bad in itself	Part I: A2 everything is a trade-off, A3 quality attributes drive structure; Part II: every rating explained through the tactics a pattern bundles or impedes
2. construct a **requirements profile** $R(a)$: scenarios with response measures, utility tree, weights	Part I: A4 requirements decide only as measurable scenarios; Part III: ten classes as ASR bundles with response measures and hard constraints $K(a)$
3. derive the **capability profile** $C(p)$ of seven patterns from their tactics	Part II: capability profiles, every rating explained by tactics (the consolidated capability table)
4. run the **three-stage, non-compensatory match** and defend the result	Part IV: knock-out screening, veto rule, holistic ordinal reading; hybrids are the normal case
5. record decisions as **ADRs** and attach a **measurement contract**	Part IV: the procedure ends in an ADR, every ADR in a measurement contract – A5, a decision is a hypothesis tested over the life cycle
6. engineer **AI components** into a system (Axis B) and use **AI tools** in the process (Axis A) with the same discipline	Part V: A6 held – one class, one dimension, five cell shifts, one composition pattern; Axis A raised the value of the artefacts

Open book: the map and the four tables

Your map: Maxim 8's one sentence



Part V: AI changes no step of it

The four tables the argument runs on

- the **twelve dimensions** (Part I)
- the **consolidated capability table** (Part II)
- the **requirements table** of the ten classes (Part III)
- the 7×10 **fit matrix** (Part IV)

Part V's additions

- the **D12 row with rationales**; the **agent capability-profile sketch**; the **task-signature table**; the **OWASP table**

The spine: the six load-bearing assumptions **A1–A6** – and the Maxims, up to Maxim 8

Focus (Lecture 1): **architecture reasoning – profiles, matching, trade-offs**; in the words of the Section 43 key concept: *name, for any cell movement, the quality-attribute mechanism behind it.*

This week's exercise: M5 closes – harness, guard, cost observability

Project Link: Portfolio Intelligence Platform

Week 13 of **M5 – Multi-Agent Orchestration, Evaluation, and Hardening**; coaching session (1 lesson):

1. **Evaluation harness as a CI gate**; report accuracy and failure modes
2. **Observability** of token cost and latency per request
3. **Threat model** incl. prompt injection via news (today's Section 42.6: the OWASP table as your checklist; the same-channel rule) + **basic hardening**
4. **Scaling/optimisation**: caching, batching

Optional distinction work (semester plan: *Kür*): autonomous planning, self-repair, model routing, CI/CD, tracing – every topology escalation justified against the task-signature table and priced per the default rule.

Milestone M5: eval harness in CI + guard + cost observability. Keep the deterministic core free of LLM calls – this is the line that is graded.

Summary

1. **OWASP LLM Top 10**: prompt injection cannot be solved in the model (same channel) – defence in depth at the system level; *the architecture, not the model, is the trust boundary*
2. **EU AI Act**: quality attributes with legal force enter $R(a)$ as hard constraints $K(a)$, never weights – compliance is the same architecture, documented
3. **C10 row and five shifts**: boundary/port, queue and capped cells; EDA/PF ↑, HX ↑ most, MS ↓ in synchronous chains, SL conditional, cost load-bearing – the matrix is *shifted*, not rewritten; MLOps level = how much of D9/D12 a team can cash in
4. **Agent orchestration** is a composition pattern, not a style: agent = loop + tools + state; workflows vs. agents; topologies map onto PF, mediator EDA, broker, control loop
5. **Economics and profile**: 90.2 % at $\sim 15\times$ tokens is a CBAM decision; $--$ on D3/D4/D9/D10 makes it an edge pattern – workflows before agents, every escalation an ADR with token budget and eval threshold (Maxim 6)
6. **Maxim 8**: the theory absorbs AI – same assumptions, same procedure, same contract; one discipline at two binding sites: containing cheap, fallible generation behind expensive, explicit verification

Next week

Week 14 – synthesis, presentations, defence

- one lesson synthesis and exam hints: a **recap of today's synthesis and exam orientation plus Q&A** – *no new material*
- three lessons final presentations, architecture defence and peer reviews (deliverable A3, milestone M6)
- present the system and **defend your architectural trade-offs**; reflect on where AI helped and where it hurt – in building (A) and in the system (B)
- then the written examination (open book, 60 min)

Reading

- this week: Part V, Sections 42.6–42.7, 43–45
- ahead: **the whole script, Parts I–V** – the four tables and Maxim 8's one sentence as the map

Exercise / deliverable

- M5 closes this week: eval harness in CI + guard + cost observability; threat model + hardening; optional distinction work
- **A3: final presentation with architecture defence, week 14**



Thank you!

Dr. Florian Herzog

Fachhochschule Graubünden, Chur

AISE502 – AI in Software Engineering II