MODULE 3 — Managing AI-Driven Development in Agile Systems

Theme: Turning AI from a helper into a reliable contributor across sprints, stakeholders, and evolving systems.

◇ Module 3.1 — Where AI Fits in the Project Lifecycle

Theme: Understanding when AI accelerates work — and when it quietly increases risk.

1️⃣ Core Concept

AI is **not uniformly helpful across all phases of a project**. Its value depends on two dimensions:

1. **Boundedness of the task** (How clearly defined are inputs, outputs, and constraints?)
2. **Reversibility of mistakes** (How costly is it if the output is subtly wrong?)

You can think of this as a 2×2 grid.

2️⃣ The AI Fit Matrix (Mental Model)

Project Phase	**Bounded?**	**Mistakes Reversible?**	**AI Fit**
Ideation / Brainstorming	Low	High	▨ Very Good
Exploration / Prototyping	Medium	High	▨ Good
Implementation (scoped tasks)	High	Medium	▨ Good
Refactoring (behavior-preserving)	High	Medium	▨ Good
Architecture Definition	Low	Low	▩ Risky
Production Hotfixes	High	Low	▩ Risky
Ownership / Open-ended Maintenance	Low	Low	▩ Very Risky

3️⃣ Concrete Examples (Data Science Context)

▨ Good AI Fit

Examples

- Writing a data ingestion function with a fixed schema
- Refactoring a pandas pipeline without behavior changes
- Adding type hints or docstrings
- Writing unit tests for known behavior
- Reviewing diffs against acceptance criteria

Why

- Clear inputs/outputs
- Easy to validate
- Easy to roll back

🟨 Mixed AI Fit (Human-Led, AI-Assisted)

Examples

- Feature engineering strategy
- Model selection
- Evaluation metric choice
- Tradeoff analysis

Why

- AI can generate options
- Humans must choose
- Wrong decisions compound over time

🟥 Poor AI Fit

Examples

- Designing system architecture from scratch
- Choosing long-term data contracts
- Making production rollout decisions
- Handling live incidents
- "Own this system and improve it"

Why

- Unbounded
- High blast radius
- Errors may be silent
- Hard to fully validate

4️⃣ Why This Matters for Managing AI Projects

Most AI failures in engineering happen when teams:

- use AI in the *wrong phase*
- give AI *ownership instead of tasks*
- let AI make *irreversible decisions*
- skip human checkpoints

This module is about **placing AI correctly**, not restricting it.

5️⃣ Practical Rule You Can Use Immediately

> **If you cannot write acceptance criteria for the task, AI should not execute it.**

Or more bluntly:

> **AI should not own decisions that humans cannot easily review or undo.**

This rule alone prevents most misuse.

🧪 Mini-Exercise — Module 3.1

Task Context

You are working on a **data science project** that will eventually:

- ingest data
- clean data
- engineer features
- train models
- deploy results into an application

Below is a list of tasks that might appear during the project.

Your Task

For each task, classify it as:

- 🟢 **AI-Appropriate**
- 🟡 **AI-Assisted (Human-Led)**
- 🔴 **AI-Risky**

and give **one short reason** for each classification.

Tasks to Classify

1. Writing a CSV ingestion function with a fixed schema
2. Designing the overall system architecture for production
3. Refactoring a working data pipeline to reduce duplication
4. Choosing the target metric for a business decision
5. Debugging a failing unit test with a known error message
6. Deciding when to retrain a model in production
7. Generating a stakeholder sprint update

📝 Submission Format

```
1. Task name — 🟢 / 🟡 / 🔴
   Reason: ...

2. Task name — 🟢 / 🟡 / 🔴
   Reason: ...
```

◇ Module 3.2 — Designing Sprints for AI Work

Theme: Turning ambiguous work into AI-executable, reviewable sprint slices.

1️⃣ Core Concept

> Human sprints and AI sprints are not designed the same way.

Traditional sprint tickets often assume:

- human judgment
- implicit context
- mid-task course correction

AI cannot rely on any of those. So AI-compatible sprints must be:

- **narrow**
- **vertical**
- **reviewable**
- **stoppable**

2️⃣ The AI-Safe Sprint Definition

A sprint task is **AI-safe** if all of the following are true:

1. **Bounded inputs -** What goes in is explicitly defined.
2. **Bounded outputs -** What comes out is explicitly defined.
3. **Explicit stop condition -** AI knows when to stop.
4. **Human checkpoint -** A review or approval step exists.
5. **Low blast radius -** Mistakes affect only a small, reversible area.

If any one of these is missing, the sprint is unsafe for AI execution.

Key Takeaways

- Determining whether a task fits into a sprint in terms of size or duration is not the deciding factor of AI-safety
- AI refactors must be bounded by *single responsibility boundary*, not a sprint boundary
  - Safe boundaries: one function, one module, one pipeline step, one interface
  - Unsafe boundaries: the pipeline, the system, the architecture

- "*Could a human reviewer confidently approve this change without re-learning the system?*" -> yes = AI-safe

---

3 Vertical vs Horizontal Slicing (Critical Distinction)

❌ **Horizontal slicing (AI-dangerous)**

> "Implement data ingestion." "Add feature engineering." "Build the model."

Why this fails:

- broad scope
- implicit decisions
- unclear completion
- encourages AI overreach

Key Takeaways

- Simply: Think of horizontal slicing as entire data pipeline steps (ingestion, cleaning, modeling, etc.)
- Broadly: Horizontal slicing -> define areas
  - Spans a category of work
  - Leaves decisions open
  - Requires follow-on tasks to be "done later"
  - Does not produce a fully reviewable artifact on its own

☑ **Vertical slicing (AI-safe)**

> "Load CSV → validate schema → return DataFrame." "Refactor this function to remove duplication, no behavior change." "Add unit tests for missing-column failure case."

Why this works:

- end-to-end within a narrow slice
- behavior is testable
- completion is clear

Key Takeaways

- Simply: Think of vertical slicing as end-to-end work for one data pipeline step (ingestion: load csv -> validate schema -> return DataFrame)
- Broadly: Vertical slicing -> define behaviors
  - One narrow behavior
  - Produces something *complete* at its own scale
  - Can be reviewed, tested, and accepted independently
  - Has clear stop condition

---

4 Concrete Example (Data Science Pipeline)

❌ Vague sprint ticket

> “Clean the dataset and prepare it for modeling.”

This is a trap for AI:

- unclear definition of “clean”
- unclear success criteria
- huge design space

☑️ AI-safe sprint ticket

```
Sprint goal:
Implement missing-value handling for column price.

In scope:
- Identify missing values in price
- Replace with median valueOut of scope:
- Feature scaling
- Outlier detection
- Model trainingAcceptance criteria:
- Output DataFrame has no missing price values
- Median is computed only from training data
- Existing tests pass unchanged
```

This is AI-executable.

5️⃣ Common Sprint Design Failure Modes with AI

Watch for these red flags:

- “Improve…”
- “Make it more robust…”
- “Refactor the pipeline…”
- “Prepare for production…”
- “Handle edge cases…”

These are signals the sprint is **underspecified**.

6️⃣ Practical Heuristic (Use This Live)

> **If a sprint ticket cannot be verified in a code review, AI should not execute it.**

Or even simpler:

> **If you can’t write acceptance criteria, don’t give it to AI.**

🧪 Mini-Exercise — Module 3.2

Task Context

Below are **three poorly defined sprint tickets** from a data science project.

Sprint Tickets to Fix

1. “Clean the dataset so it’s ready for modeling.”
2. “Improve the data pipeline and make it more maintainable.”
3. “Prepare the model training step for production use.”

Your Task

For **each ticket**:

1. Rewrite it into an **AI-safe sprint ticket**
2. Include:
    - Sprint goal
    - In scope
    - Out of scope
    - Acceptance criteria

📝 Submission Format

```
### Ticket 1

Sprint goal:
...

In scope:
- ...

Out of scope:
- ...

Acceptance criteria:
- ...

### Ticket 2
...
```

◇ Module 3.3 — Managing Risk Accumulation Across Sprints

Theme: Preventing *slow, silent degradation* in AI-assisted projects.

Up to now, you've learned how to make **individual AI tasks safe**. Module 3.3 answers the harder question:

> **"What happens when lots of individually safe AI changes accumulate over time?"**

1️⃣ Core Concept: Risk Is Additive — and Sometimes Exponential

Here's the uncomfortable truth:

> **Even if every AI-assisted sprint is "safe," the project can still become unsafe.**

Why? Because risk accumulates across:

- assumptions
- abstractions
- refactors
- interfaces
- undocumented decisions

AI accelerates this accumulation because:

- it makes change cheap
- it makes refactoring tempting
- it reduces friction to "just one more tweak"

2️⃣ The Three Types of Accumulated Risk

◇ 1. Assumption Drift

Each sprint introduces small assumptions:

- column names
- data distributions
- ordering guarantees
- implicit defaults

Individually harmless. Collectively dangerous.

Example

- Sprint 1: assumes `price` exists
- Sprint 2: assumes `price` is non-null
- Sprint 3: assumes `price` is log-normal

No single sprint is wrong — but the system now depends on all three.

◇ 2. Architectural Entropy

Repeated "small refactors" can:

- fragment responsibility
- duplicate logic subtly
- erode clear boundaries

AI tends to:

- optimize locally
- reduce visible duplication
- increase invisible coupling

Symptom

> "No one change broke anything, but the system is harder to reason about."

◇ 3. Validation Gaps

Tests often lag behind behavior. AI may:

- preserve existing tests
- but introduce new behavior paths
- without adding corresponding validation

Over time:

- confidence becomes misplaced
- tests no longer cover what matters

---

3 Why AI Makes This Worse (Not Better)

Humans naturally:

- feel friction
- hesitate before refactors
- remember why things were done

AI:

- does not remember prior intent
- does not feel cost
- optimizes for immediate task success

That's why **governance is not optional** in AI-assisted projects.

---

4 The Risk Accumulation Pattern (Very Common)

This is a pattern you should memorize:

1. Sprint 1: small feature, well scoped
2. Sprint 2: refactor to "clean it up"
3. Sprint 3: another refactor for reuse

4. Sprint 4: behavior subtly diverges
5. Sprint 5: tests pass, but confidence drops

No single bad decision — just **compounding drift**.

5️⃣ Risk Control Mechanisms (What Actually Works)

🧱 1. Periodic Stabilization Sprints

Deliberately schedule sprints where:

- no new features are added
- focus is on:
  - documentation
  - invariants
  - tests
  - simplification

AI is allowed only in **review or assist mode** here.

> **If AI has been accelerating change, humans must periodically accelerate understanding.**

Stabilization sprints are how you do that.

🧭 2. Explicit Invariants

Write down things that must never change unless explicitly approved. Examples:

- "Training data schema"
- "Public pipeline interfaces"
- "Definition of target variable"

AI should not touch these without human sign-off.

Key Takeaways

- Think *Data Science Foundations for AI*, during lesson plan part of human responsibility was collapsing uncertainty at each pipeline stage by making certain assumptions that chain back to the original decision which the pipeline is supposed to be supporting.
  - Example: design feature engineering to where noise is preserved because it was decided false positives are more tolerable than false negatives.
- These things get embedded into the prompts and are documented through artifacts.

🧪 3. Trust Reset Checks

Occasionally ask:

> "If we had to explain this system to a new engineer, could we?"

If not, risk has accumulated.

6️⃣ A Practical Heuristic (Use This)

> **If you've used AI in 3 consecutive sprints on the same component, pause and review.**

This is not a hard rule — it's a *signal*.

🧪 Mini-Exercise — Module 3.3

Task Context

You're reviewing a project that had three AI-assisted sprints:

- Sprint 1: Added missing-value handling for column `price`
- Sprint 2: Refactored feature engineering for reuse
- Sprint 3: Made model training reproducible (fixed seeds + splits)

All tests pass. No incidents reported.

Your Task

1. Identify **two plausible accumulated risks**
2. For each risk:
   - explain *how* it could arise
   - suggest **one mitigation action**

📝 Submission Format

```
Risk 1:
Description:
Mitigation:

Risk 2:
Description:
Mitigation:
```

🔹 Module 3.4 — Stakeholder Communication & AI Transparency

Theme: Making AI-assisted work *boringly safe* to stakeholders.

Module 3.4 answers a different question:

> **How do you talk about AI use in a way that increases confidence instead of triggering concern?**

---

1 Core Principle

The goal of AI transparency is **not** to explain AI. The goal is to:

- preserve accountability
- demonstrate control
- prevent surprise
- avoid false confidence or fear

Bad transparency causes panic. Good transparency makes AI invisible.

---

2 The Stakeholder Anxiety Model (Why This Matters)

Stakeholders usually worry about **three things**, even if they don’t say it:

1. **Loss of control** → “Who’s actually making decisions?”
2. **Loss of accountability** → “Who’s responsible if this goes wrong?”
3. **Unbounded change“** → Is this system evolving in ways we don’t understand?”

Your communication should quietly answer all three.

---

3 What NOT to Say (Common Failure Modes)

Avoid phrases like:

- “The AI decided…”
- “The model chose…”
- “We let AI refactor…”
- “AI handled this sprint…”

These imply:

- autonomy
- delegation of responsibility
- loss of oversight

Even if technically true, they are **organizationally dangerous**.

---

4 The Safe Framing Pattern (Use This)

Here is the framing that works:

> **“AI-assisted, human-reviewed, scope-limited changes.”**

Break it down explicitly:

- **AI-assisted** → tool, not owner
- **Human-reviewed** → accountability preserved
- **Scope-limited** → blast radius controlled

This framing should appear *implicitly* in every update.

Notes

- This is consistent with *Data Science Foundations for AI*, where AI is viewed as a tool/collaborator, but development approach keep responsibility & ownership in human hands
    - AI will make recommendations, provide benefits & tradeoffs for each, but ultimately it's the humans final decision (and to provide reason why)

5️⃣ Transparency Without Over-Disclosure

You should **not**:

- explain prompt structures
- explain model internals
- justify using AI as a technology choice

You **should**:

- describe *what changed*
- describe *what didn't*
- describe *how risk was managed*

6️⃣ Example: Bad vs Good Stakeholder Update

❌ Bad

> "We used AI to refactor the data pipeline and improve maintainability."

Why this fails:

- unclear scope
- unclear control
- unclear accountability

☑️ Good

> "This sprint focused on improving maintainability of the data ingestion step. Changes were limited to readability and duplication reduction; no functional behavior changed. All existing tests passed unchanged, and the update was reviewed before merge."

Notice:

- No mention of AI mechanics
- Clear scope
- Clear validation
- Clear ownership

7️⃣ When You SHOULD Explicitly Mention AI

Explicit AI mention is appropriate when:

- stakeholders ask directly
- governance or compliance requires it
- summarizing process improvements
- documenting methodology (not results)

Even then, AI should be described as:

> **"a productivity tool under defined constraints."**

8️⃣ The Transparency Gradient (Very Important)

Different audiences need different levels of transparency:

Audience	AI Mention Level
Executives	Minimal, outcome-focused
Product managers	Process-aware
Engineers	Detailed
Compliance	Explicit and documented

One-size-fits-all transparency backfires.

🧪 Mini-Exercise — Module 3.4

Task Context

You completed a sprint with:

- AI-assisted refactoring of a data ingestion function
- No behavior changes
- All tests passed
- Human review completed

Your Task

Write **two versions** of a status update:

1. **Executive update** (3–4 sentences, no AI jargon)
2. **Engineering update** (more detail, controlled AI mention)

📝 Submission Format

```
Executive Update:
...

Engineering Update:
...
```

What I'll Review For

I'll check:

- tone appropriateness per audience
- avoidance of AI over-attribution
- clarity of scope and validation
- preservation of accountability

Take your time — this is a communication skill that separates AI leaders from AI risks.

◇ Module 3.5 — Governance Without Bureaucracy

Theme: You need rules — but not red tape.

Good AI governance is:

- lightweight
- explicit
- enforceable
- socially understood Bad AI governance is:
- heavy
- vague
- compliance-driven
- ignored in practice

This module is about designing the *minimum viable governance* that actually scales.

---

1 The Governance Paradox

If governance is:

- **too light** → AI runs wild
- **too heavy** → people route around it

So the goal is not "control everything." The goal is:

> **Make the safe path the easy path.**

2️⃣ What Governance Is Actually For

Governance is **not** about:

- slowing teams down
- approving every change
- policing AI use

Governance *is* about:

- preventing category errors
- protecting high-risk decisions
- clarifying ownership
- preserving reversibility

Think of it as **guardrails**, not gates.

3️⃣ Lightweight Governance Patterns (That Actually Work)

◇ 1. Decision Logs (Minimal, Not Bureaucratic)

A decision log answers:

- *What decision was made?*
- *Why?*
- *What assumptions does it encode?*
- *Who approved it?*

Not a document — often just:

- a Markdown file
- a PR comment template
- a short section in sprint notes

Rule of thumb: If a decision is hard to reverse → log it.

◇ 2. Approval Thresholds (Not Everything Needs Review)

Not all work carries equal risk. Example thresholds:

- 🟢 Low risk → engineer approval
- 🟡 Medium risk → peer review
- 🔴 High risk → designated owner sign-off

AI doesn’t change this — it just **raises the need for clarity**.

◇ 3. “AI Allowed / AI Forbidden” Zones

This is one of the most effective patterns. Examples: **AI Allowed**

- refactoring with no behavior change
- writing tests
- documentation
- bounded data transformations
- debugging with known errors

AI Forbidden (without explicit approval)

- defining business metrics
- changing target variables
- system architecture design
- production decision logic
- automated retraining policies

This removes ambiguity *before* it becomes a problem.

◇ 4. Ownership Is Non-Negotiable

A simple but powerful rule:

> **Every AI-assisted change must have a human owner.**

Not:

- “AI handled this”
- “the model decided”

But:

- “Owned by X, reviewed by Y”

This single rule eliminates most governance failures.

◇ 5. Scale by Rules, Not Meetings

If governance requires:

- standing meetings
- committees
- heavy approvals

...it will fail at scale.

If governance is:

- a checklist
- a template
- a shared understanding

...it will propagate naturally.

4️⃣ What “Governance Without Bureaucracy” Looks Like in Practice

You’ll know it’s working when:

- engineers don’t ask “can I use AI here?”
- risky work is naturally slowed down
- safe work flows freely
- stakeholders feel informed, not alarmed
- AI use fades into the background

That’s the goal.

🧪 Mini-Exercise — Module 3.5

Your Task

Define **5 lightweight project rules** for AI usage. Each rule should:

- be short (1 sentence)
- be enforceable
- reduce ambiguity
- not require meetings or approvals to interpret

Think of these as:

> *Rules that prevent disasters, not rules that optimize productivity.*

📝 Submission Format

```
Rule 1:
...

Rule 2:
...

Rule 3:
...

Rule 4:
...

Rule 5:
...
```

What I’ll Review For

I'll look for:

- clarity over completeness
- proportionality
- alignment with Modules 3.1–3.4
- avoidance of bureaucratic language
- realism (rules people would actually follow)

Once we complete Module 3.5:

- Module 3 will be fully complete
- you'll have a **coherent governance model**
- and you'll be operating at a level most teams never reach with AI

Whenever you're ready, go ahead and submit your **5 project rules for AI usage**.

🧠 Module 3 Capstone — AI-Assisted Project Leadership

🎯 Capstone Goal

Design a **multi-sprint AI-assisted project plan** that demonstrates:

- clear boundaries for AI usage
- explicit risk management across time
- lightweight governance (no bureaucracy)
- stakeholder-ready communication
- clear escalation points when risk increases

This is not about writing code. This is about **designing the conditions under which AI-driven work remains safe, explainable, and trusted**.

🧩 Scenario (Given)

You are leading a **data science / analytics project** that will evolve over several sprints. High-level project objective (fixed):

> Build a data-driven system that supports decision-making using structured datasets, with the expectation that the solution will eventually be extended, maintained, and reviewed by others.

Constraints:

- AI assistance is allowed, but **humans retain ownership**
- The system will grow across multiple sprints
- Stakeholders expect regular updates
- Some decisions will be hard to reverse

You are responsible for designing the **project structure**, not implementing it.

🧠 Your Deliverables

You will produce **five artifacts**, each lightweight but precise.

1️⃣ Multi-Sprint Plan (3 Sprints)

Define **three consecutive sprints**, each with:

- Sprint goal (1–2 sentences)
- Type of work (feature, refactor, stabilization, etc.)
- AI role(s) allowed
- Why this sprint is safe for AI involvement

Guidance

- At least one sprint should be **behavior-changing**
- At least one sprint should be **non-behavioral (refactor or stabilization)**
- Sprints should show *intentional sequencing*, not a random backlog

---

2️⃣ AI Boundaries Per Sprint

For each sprint, clearly specify:

- What AI **is allowed** to do
- What AI **is forbidden** from doing
- Who owns final decisions

This should mirror the **AI Sprint Scope Contract** mindset, but at a higher level.

3️⃣ Risk Accumulation Analysis

Identify **two risks** that could emerge *across* the three sprints. For each risk:

- Explain how it could accumulate
- Identify the sprint where it first appears
- Propose one mitigation (lightweight, realistic)

This should explicitly reflect **Module 3.3 thinking**.

4️⃣ Governance & Escalation Points

Define:

- What kinds of changes require **extra review**
- What signals trigger a **stabilization sprint**
- When AI usage would need to be paused or constrained further

This is about **judgment**, not policy.

5️⃣ **Stakeholder Communication Plan**

Describe:

- What stakeholders will be told **each sprint**
- How AI involvement will be framed
- How confidence and accountability will be preserved

You do **not** need to write the full updates — just outline the strategy.

📝 Submission Format

Use this structure:

```
## 1. Multi-Sprint Plan
Sprint 1:
Sprint 2:
Sprint 3:

## 2. AI Boundaries Per Sprint
Sprint 1:
Sprint 2:
Sprint 3:

## 3. Risk Accumulation Analysis
Risk 1:
Risk 2:

## 4. Governance & Escalation
...

## 5. Stakeholder Communication Plan
...
```

☑️ What I'll Be Evaluating

I'll be looking for:

- internal consistency (rules match actions)
- realistic leadership judgment
- clear ownership boundaries
- proportional risk controls
- alignment with **Modules 3.1–3.5**
- whether this plan could actually be used by a real team

There is **no single "correct" plan** — this is about *quality of reasoning*, not answers.

🧠 Why This Capstone Matters

If **Modules 0–2** were about:

- *how to work with AI*

And **Module 3** was about:

- *how to manage AI safely*

Then this capstone is about:

> **how to lead AI-assisted work responsibly.**

Completing it means you’re operating at:

- senior IC level
- tech lead level
- or early staff/leadership level

depending on context.
