Pierre-Yves
PAMART

JOURNÉE FRANÇAISE DES TESTS LOGICIELS

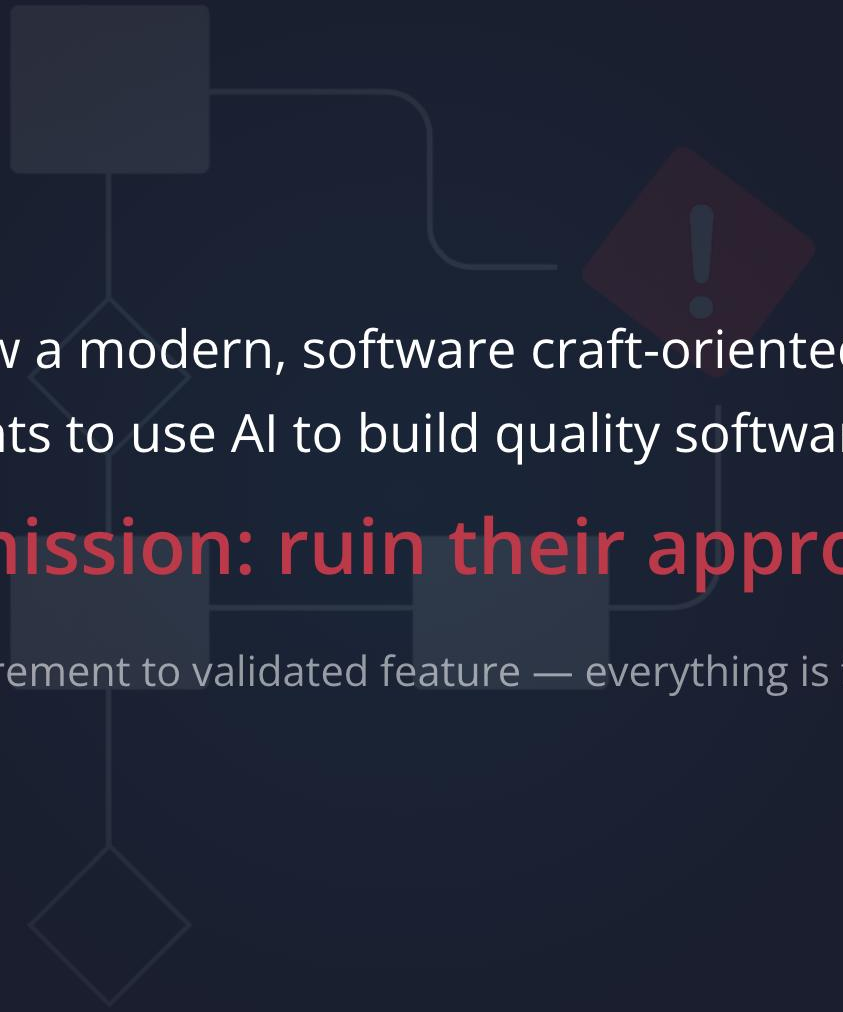
Un développeur infiltré chez les testeurs : comment faire échouer le BDD, l'ATDD & le TDD surtout à l'ère du Spec-Driven Development

9 juin 2026

BEFFROI DE MONTROUGE

SOCIETE GENERALE

CFTL
Comité Français des Tests Logiciels



We will follow a modern, software craft-oriented team that wants to use AI to build quality software.

Our mission: ruin their approach.

From requirement to validated feature — everything is fair game.

CAUTION

The opinions expressed here are solely those of the author.

- This talk offers a partial, biased, and passionate perspective
- Heavy use of irony — take it at least at second degree
- If it feels familiar, you are definitely not alone

PIERRE-YVES
PAMART

DISTINGUISHED ENGINEER

MAJOR SYSTEMIC BANK

SPEAKER

An Expert at Your Service

1,000+
SOFTWARE PRODUCTS & SYSTEMS

1,300+
SOFTWARE ENGINEERS ACROSS 200+ FEATURE TEAMS

3,000+
PEOPLE IN CORPORATE FUNCTIONS TECHNOLOGIES SCOPE

45+
EXPERTS ACROSS 6 DOMAINS

Craft & AI
RAISING THE BAR — MINDSET & ENGINEERING PRACTICES

👨‍👩‍👧‍👦 & 🎺
FATHER OF 2, PASSIONATE BRASS MUSICIAN

SESSION LOGISTICS

📋 QR code

— slides PDF available at the end

💬 Questions

— we'll take them at the end of the talk

☕ Want more discussion?

— coffee or a drink afterward

SABOTAGE TECHNIQUE

#0

Distort the meaning

GOAL	create chaos
ZONE	requirements in backlog → feature implemented & verified
FOUNDATION	misunderstanding

SABOTAGE TECHNIQUE

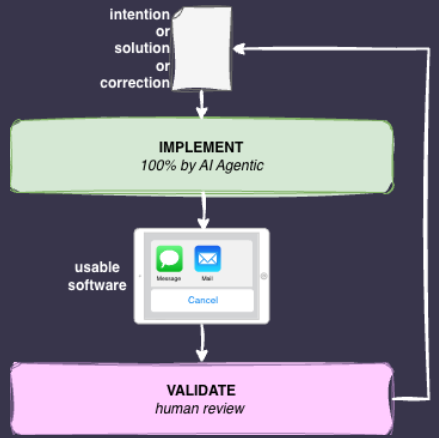
#1

Confuse Vibe Coding & SDD

Two very different approaches

Vibe Coding

- **Intention** → implementation → correction
- Delegate as much as possible
- Fast iteration on small scope
- Great for **prototyping**
- Limits: large systems, legacy, architecture



VS

Spec-Driven Development

- Differents levels of AI delegation: spec-first, spec-anchored, spec-as-source, etc.
- Structured, traceable progression
- AI exploits well-formed material
- Built for **complex systems**

Vibe coding works for prototyping.

SDD is for **systems that need to survive reality**.

Context	Recommended Approach	Level of Specification	Example
Prototype / exploration	Vibe coding	Minimal (direct prompt)	Internal POC, hackathon
Feature on existing codebase	Lightweight SDD	Spec.md + tests	API endpoint addition
Production / team	Full SDD	Architecture + Spec + Plan + Tasks	Payment system
Critical system	Full SDD + formal verification	↑ + Proofs + Mutation Testing	Financial infrastructure

SABOTAGE TECHNIQUE

#2

Confuse Requirement & Specification

Requirement ≠ Specification

A user story expresses an **intention**.

A specification contains **exploitable detail**.

There is real formulation work in between — and if it’s missing, what we delegate is **incomplete by construction**.

OFFICIAL IREB DEFINITIONS

Requirement

1. A **need** perceived by a stakeholder.
2. A **capability or property** that a system shall have.
3. A **documented representation** of a need, capability or property.

> *"Customers want to earn a free coffee after 10 purchases."*
> → *An intention. No rule, no edge case, no measurable criterion.*

Specification

AS A WORK PRODUCT

A systematically represented description of the *properties* of an item (a system, a device, etc.) that satisfies given criteria.

AS A PROCESS

Eliciting, documenting and validating the *properties* of an item.

> *Given a customer has 10 stamps, when they order a coffee, then it is offered free of charge and the counter resets to 0.*
> → *Testable. Unambiguous. Ready to delegate.*

Source: IREB Glossary of Requirements Engineering — ireb.org

SABOTAGE TECHNIQUE

#3

Don’t know what BDD is

"BDD? You mean... Base De Données?"

The joke reveals a real issue:

many teams manipulate terms they only **half understand**.

SABOTAGE TECHNIQUE

#4

Reduce BDD to testing

"We do BDD!" — really?

🥒 Cucumber

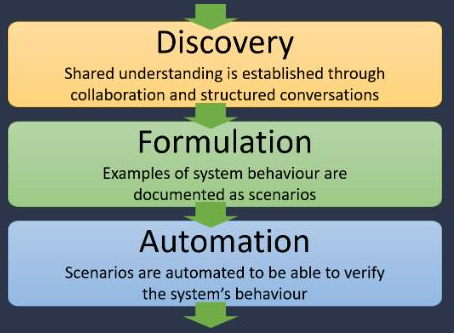
+ Gherkin syntax

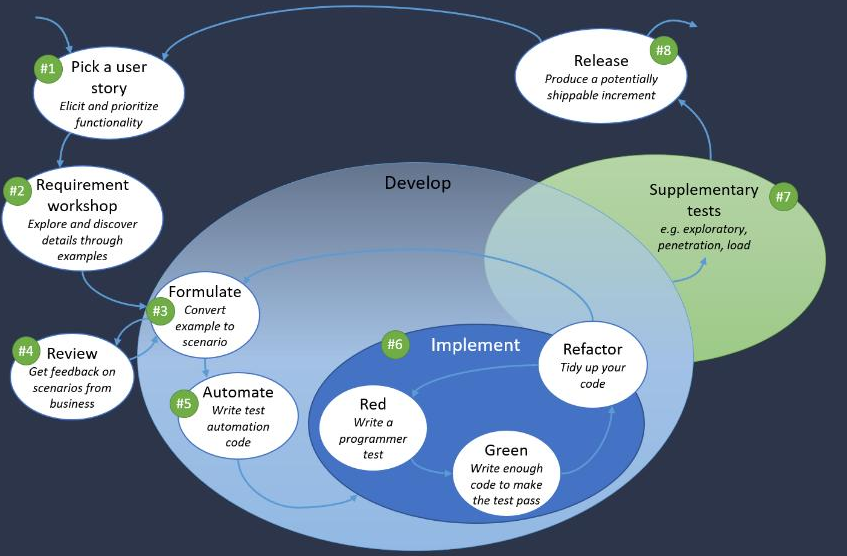
+ Step definitions

+ CI automation

❌ This is not BDD. This is the last step of BDD or ATDD in the best case...

What BDD really is





Rose, S. & Nagy, G. (2020). ***Discovery: Explore Behaviour Using Examples****. Leanpub. Fig. 22 — Tasks and activities in the BDD approach.*

SABOTAGE PROGRESS

5 / 11 techniques deployed

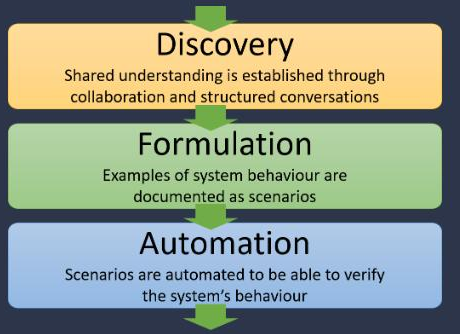
- ✓ **#0** Distort the meaning
- ✓ **#1** Confuse Vibe Coding & SDD
- ✓ **#2** Confuse Req. & Specification
- ✓ **#3** Don't know what BDD is
- ✓ **#4** Reduce BDD to testing
- ○ #5 Sabotage Example Mapping
- ○ #6 Write anemic Gherkin
- ○ #7 Skip design
- ○ #8 Flatten BDD / ATDD / TDD
- ○ #9 Sell fake TDD
- ○ #10 Feed garbage to agents

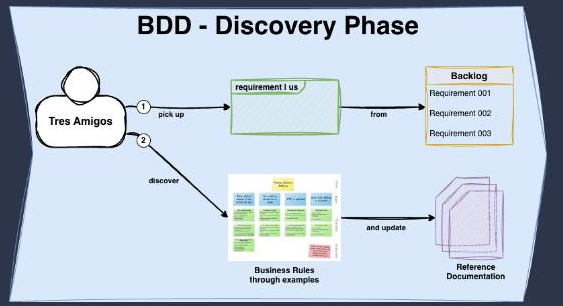
SABOTAGE TECHNIQUE

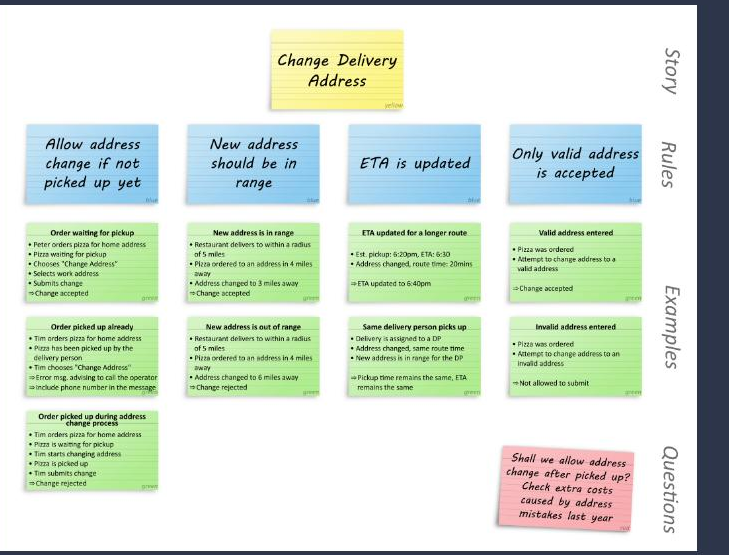
#5

Sabotage Example Mapping

What is Example Mapping?





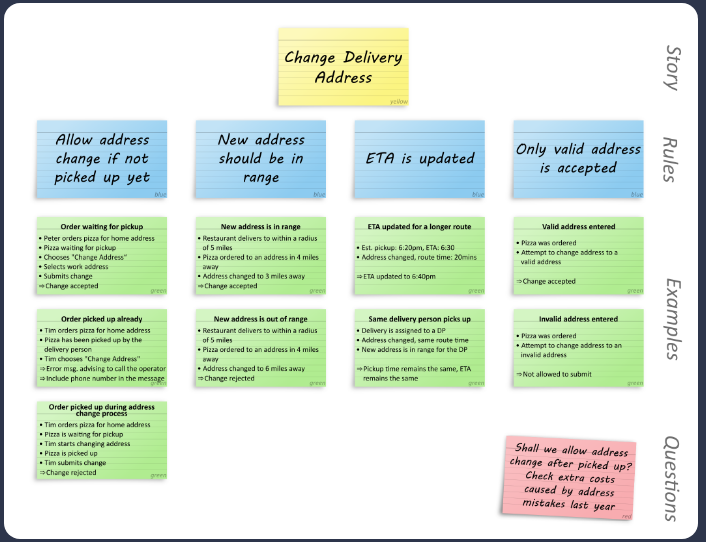


Rose, S. & Nagy, G. (2020). Discovery: Explore Behaviour Using Examples*. Leanpub. Figure 14 — The final example map.*

How to ruin an Example Mapping session

- 🎭 **Destroy the Three Amigos**
  - 🏢 Without business experts (BA or domain expert) — only QA & devs
  - 📥 QA + BA only — requirements dumped straight to the backlog board, no devs
  - 👨‍💻 Developers only — no QA, no domain expertise at all
- 🔮 Forget critical business rules
- 😊 Cover happy paths only — ignore all edge cases
- 💬 Write vague, unrealistic examples
- 🗣️ Abandon business language — go full technical
- 🤖 Use AI alone — with zero domain context
- 💣 Let the machine fill gaps with *plausible guesses*
- ⚙️ Express scenarios as **technical solutions**, not business behaviours

⚠️ Plausible-but-wrong material is **excellent fuel for failure**.



Rose, S. & Nagy, G. (2020). ***Discovery: Explore Behaviour Using Examples****. Leanpub.*

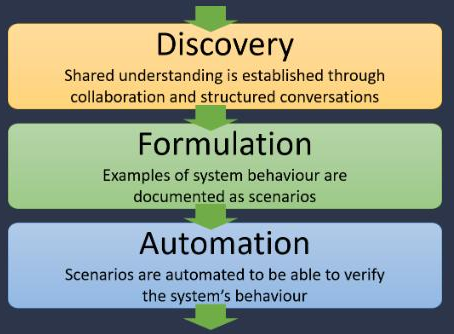
Figure 14 — The final example map.

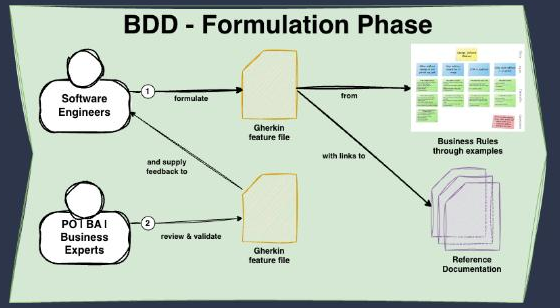
SABOTAGE TECHNIQUE

#6

Write anemic Gherkin

What is Formulation?






```gherkin
@REQ-1234 @bounded_context(pricing) @story(loyalty-42)
Feature: Loyalty discount on coffee orders
  In order to reward customer loyalty and increase retention
  As a returning customer with a loyalty card
  I want to receive a 10% discount on eligible orders once I have completed my 10th purchase

  """markdown
  This feature explains the loyalty discount rules for customers who have completed
  at least 10 previous orders.

  The discount is applied at checkout and provides a 10% reduction on the total price
  of eligible items. The feature also covers edge cases such as reaching the threshold
  for the first time, interactions with promotions, and exclusions for certain item
  types like gift vouchers.

  ## References
  - [Loyalty discount rules](docs/reference/pricing/loyalty-discount.md)
  - [Promotions stacking policy](docs/reference/pricing/promotions.md#stacking-policy)
  - [ADR-021 — Loyalty tiers](docs/adr/ADR-021-loyalty-tiers.md)
  - [Domain model — LoyaltyDiscount](docs/domain-model/pricing/LoyaltyDiscount.md)
  """

  Background:
    Given the loyalty discount rate is configured at 10%
    And the loyalty eligibility threshold is 10 completed orders

  @REQ-1234-R01 @implemented
  Rule: A customer with 10 or more completed orders receives a 10% discount

    @happy
    Example: Loyal customer gets a discount on a standard order
      Given a customer "Alice" who has completed 10 previous orders
      When she orders a "Caffè Latte" priced at €4.50
      Then the total should be €4.05
      And the receipt shows "Loyalty discount (10%): -€0.45"
      And a "Loyal Customer" badge is displayed on her account

    @happy
    Scenario Outline: Discount is consistent across coffee types
      Given a loyal customer with <orders> completed orders
      When they order a "<coffee>" priced at €<price>
      Then the total should be €<expected>

      Examples: Standard menu items — docs/reference/menu/pricing.md
        | orders | coffee         | price | expected |
```

Gherkin: anemic vs. rich

Anemic Gherkin

```gherkin
Feature: loyalty discount

  Scenario: test discount
    Given I open the browser and navigate to http://localhost:3000/shop
    And I click on the login button
    And I type "alice@test.com" in the email input field
    And I type "P@ssw0rd123" in the password input field
    And I click the submit button
    And the database table "customers" has a row where id=42 and loyalty_count=10
    And I add item with SKU "LATTE-001" to the cart by calling POST /api/cart with body {"sku":"LATTE-001'
    When I click the checkout button and the PricingService.calculateTotal() method is invoked
    Then the DOM element "#order-total" should contain the string "4.05"
    And the field discount_applied in the orders table is set to true
    And the HTTP response body contains {"discount":0.45,"discountType":"LOYALTY_10PCT"}

  Scenario: test discount with more orders
    Given I open the browser and navigate to http://localhost:3000/shop
    And I click on the login button
    And I type "bob@test.com" in the email input field
    And I type "P@ssw0rd456" in the password input field
    And I click the submit button
    And the database table "customers" has a row where id=99 and loyalty_count=20
    And I add item with SKU "BREW-003" to the cart by calling POST /api/cart with body {"sku":"BREW-003",'
    When I click the checkout button and the PricingService.calculateTotal() method is invoked
    Then the DOM element "#order-total" should contain the string "4.50"
    And the field discount_applied in the orders table is set to true
    And the HTTP response body contains {"discount":0.50,"discountType":"LOYALTY_10PCT"}
```

No context, no rules, no tags, solution-oriented wording

vs

Rich Gherkin using Gherkin V6+ syntax

```gherkin
@REQ-1234 @bounded_context(pricing) @story(loyalty-42)
Feature: Loyalty discount on coffee orders
  In order to reward customer loyalty and increase retention
  As a returning customer with a loyalty card
  I want to receive a 10% discount on eligible orders once I have completed my 10th purchase

  """markdown
  This feature explains the loyalty discount rules for customers who have completed
  at least 10 previous orders.

  The discount is applied at checkout and provides a 10% reduction on the total price
  of eligible items. The feature also covers edge cases such as reaching the threshold
  for the first time, interactions with promotions, and exclusions for certain item
  types like gift vouchers.

  ## References
  - [Loyalty discount rules](docs/reference/pricing/loyalty-discount.md)
  - [Promotions stacking policy](docs/reference/pricing/promotions.md#stacking-policy)
  - [ADR-021 — Loyalty tiers](docs/adr/ADR-021-loyalty-tiers.md)
  - [Domain model — LoyaltyDiscount](docs/domain-model/pricing/LoyaltyDiscount.md)
  """

  Background:
    Given the loyalty discount rate is configured at 10%
    And the loyalty eligibility threshold is 10 completed orders

  @REQ-1234-R01 @implemented
  Rule: A customer with 10 or more completed orders receives a 10% discount
```

Tags, refs, business language, business rules, meaningful scenarios

SABOTAGE TECHNIQUE

#7

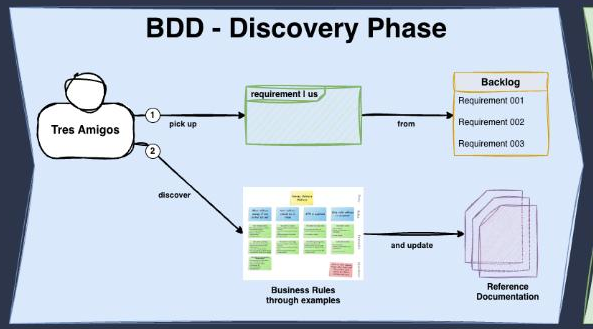
Skip design entirely

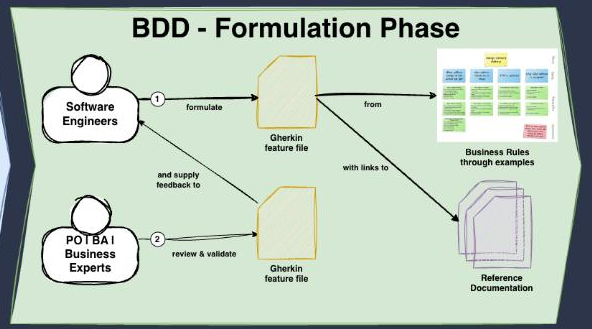
Implement vs. Plan + Construct

PROBLEM SPACE

SOLUTION SPACE

SPECIFY





IMPLEMENT

BDD - Automate Phase
start with an acceptance test

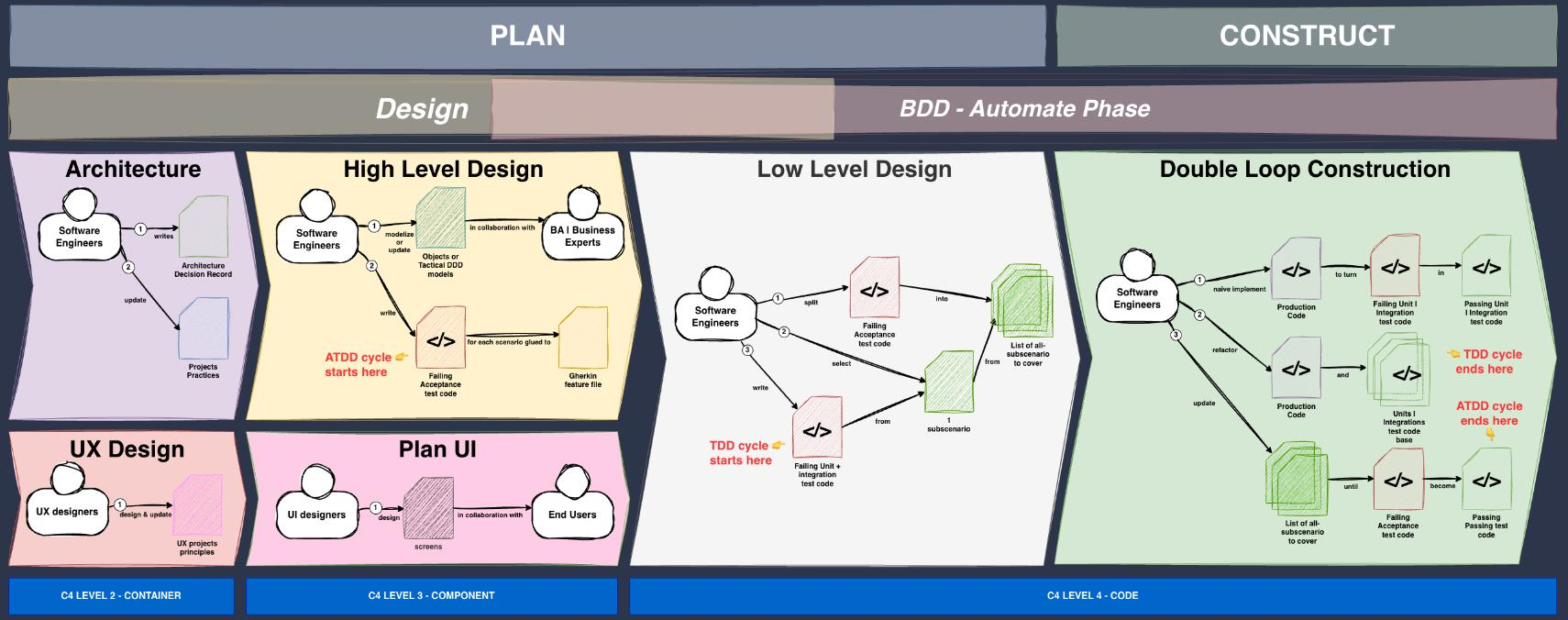
PLAN

CONSTRUCT

Design

BDD - Automate Phase
start with an acceptance test

SOLUTION SPACE



SABOTAGE TECHNIQUE

#8

Flatten BDD / ATDD / TDD

BDD · ATDD · TDD

▼ flatten everything ▼

"Just start with a test, right?"

The moment you skip design, you erase discovery, formulation, architecture, trade-offs, and **every C4 design layer we just saw**.

SABOTAGE TECHNIQUE

#9

Sell fake TDD

Fake TDD vs. Real TDD

✗

Fake TDD

- 🔴 One red test
- 🟢 One green test
- 🔵 One refactor
- **Done.**

Missing:

- No decomposition
- No micro-goals
- No emergent design
- Tests never refactored

vs

✓

Real TDD

- 🎯 Start with an **objective**
- 📋 Break it down into micro-goals
- 🔴🟢 Move in **small steps**
- 📝 Update the plan as you learn
- 🌱 Let design **emerge**
- 🔵 Refactor **code AND tests**

Otherwise, it's choreography — not design.

SABOTAGE TECHNIQUE

#10

Feed garbage to agents

How agents actually fail

INPUT QUALITY GAPS

01 Vague requirements — no specifications

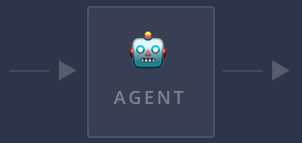
02 Missing design decisions

03 No shared understanding

04 Implicit interfaces

05 Anemic specifications & examples

06 Confused test levels



OUTPUT RISK PROFILE

CHARACTERISTIC

Plausible appearance

RISK

Structural fragility

Agents do their best with what we give them.
The failure is upstream.

The real leverage is not the prompt.

It’s the **quality of the material** we hand over.

SABOTAGE PROGRESS

11 / 11 techniques deployed

- ✓ **#0** Distort the meaning
- ✓ **#1** Confuse Vibe Coding & SDD
- ✓ **#2** Confuse Req. & Specification
- ✓ **#3** Don't know what BDD is
- ✓ **#4** Reduce BDD to testing
- ✓ **#5** Sabotage Example Mapping
- ✓ **#6** Write anemic Gherkin
- ✓ **#7** Skip design
- ✓ **#8** Flatten BDD / ATDD / TDD
- ✓ **#9** Sell fake TDD
- ✓ **#10** Feed garbage to agents

AI DELEGATION FRAMEWORK

What do we delegate, on what basis, and with which guardrails?

01

Context engineering

Enrich what AI receives before it acts

02

Harness engineering

Design control structures around AI outputs

03

Automatable checks

Let machines verify machine-generated outputs

04

Human choices

Reserve judgment where accountability cannot be delegated

> The question is not *"should we use AI?"* — it's already here.
> The question is: **what do we choose to preserve?**

Our role tomorrow

AI IS AN AMPLIFIER — NOT A CORRECTOR

Strong specs · shared understanding · clear design	× AI →	Accelerated value
Vague requirements · no design · confusion	× AI →	Accelerated failure

01 Specification integrity
The raw material AI executes — quality here multiplies downstream

02 Collective intelligence
Business · Test · Dev aligned — shared understanding no machine can replace

03 Design authority
Trade-offs, architecture, test strategy — decisions that shape everything downstream

04 System comprehension
You must understand what you build — driving blind is dangerous at any speed

The teams that win with AI are those who bring **better thinking** — not less.

In a spec-driven world, your human judgment is the only variable left.

Thank You

The sabotage was ironic. **The challenge is deeply collective.**

SLIDES & SOURCE

github.com/pypamart/jftl2026

LET'S CONNECT

Pierre-Yves Pamart

Questions?