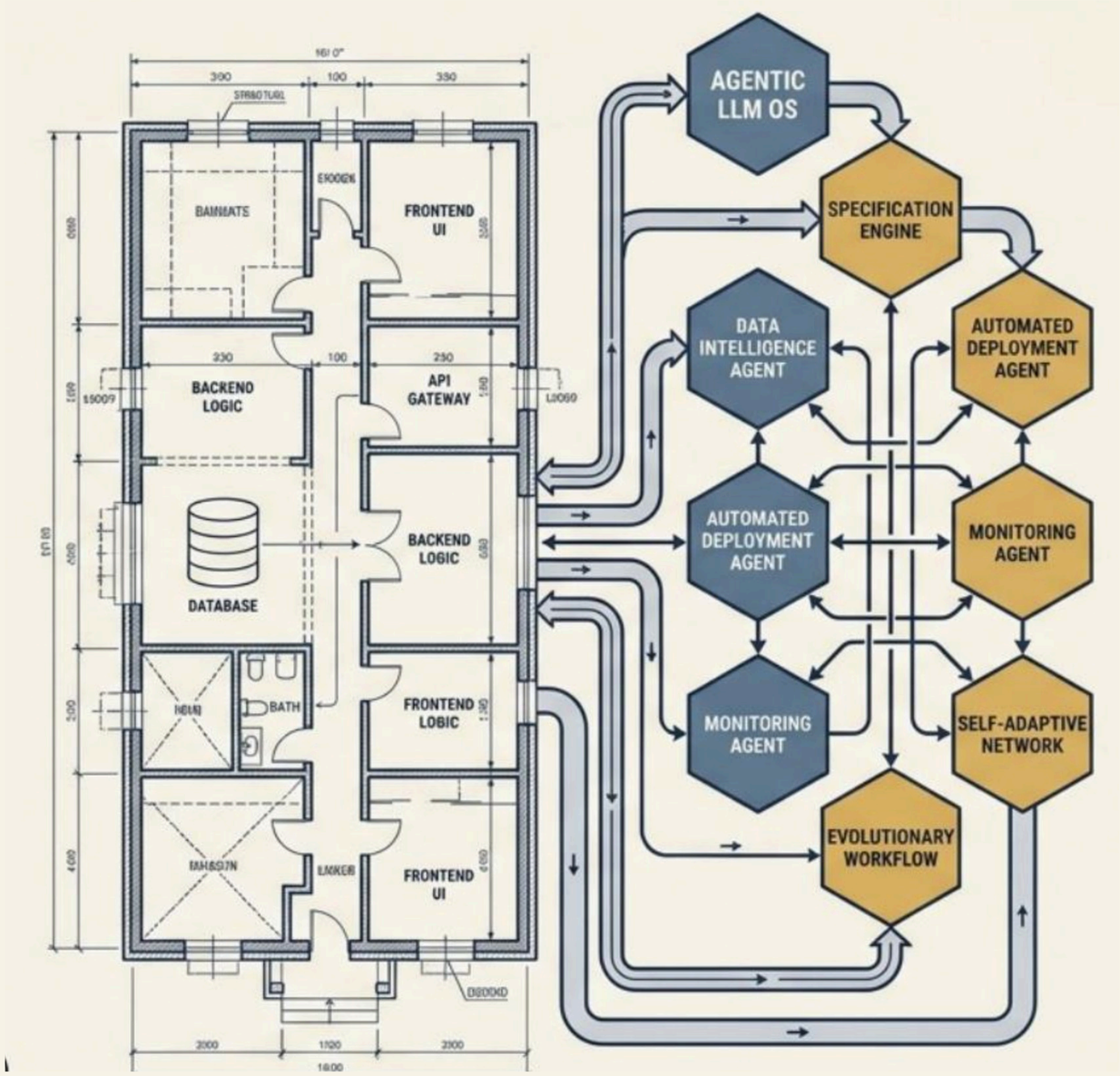
Open Source Survival Strategy for Solo Founders in the AI Era

Beyond simply publishing code: the evolution toward spec-driven development and an agentic LLM OS

(feat. uEngine's 20-year answer)



Jinyoung Jang | **CEO, uEngine Solutions**

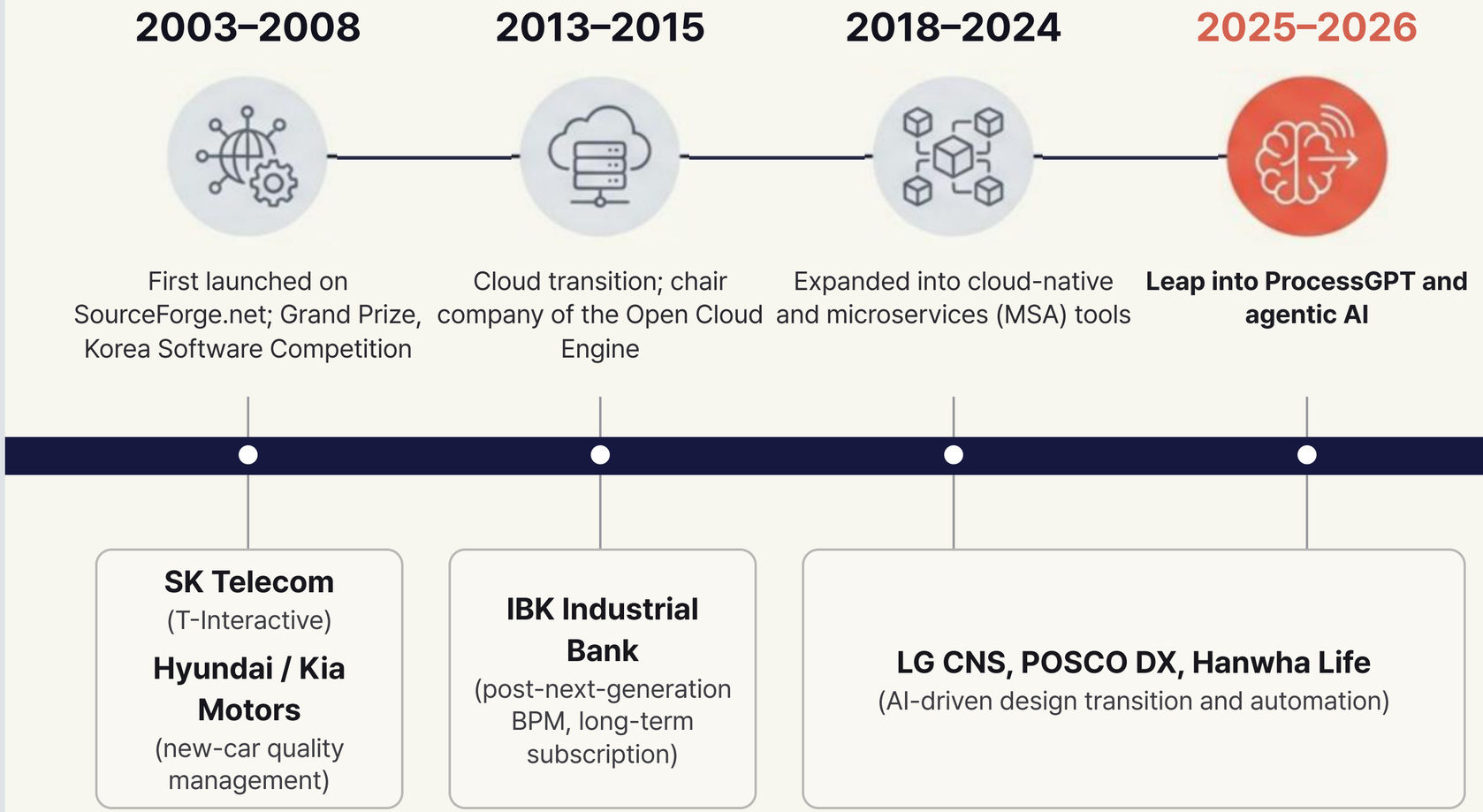
(Founded Korea's first open-source BPM community on SourceForge.net in 2003)

uEngine

A High-Trust Architecture That Has Run Through Korea's Core Businesses for 20 Years

Open source is not just a hobby.

It is the foundation of the core “brain” of large enterprises that must process thousands of transactions per second without a single error.



Premium Enterprise Blueprint

Three Survival Rules for Solo Founders in the AI Era



1. Marketing Leverage

“Don't mistake code for the product.” Use open source as a powerful distribution channel for earning trust and for “charging for the customer's operational risk.”



2. Specs as Assets

“Break free of code labor.” Become the architect who owns customers' recurring problems with structured domain specs and evaluation sets, not coding skill.



3. Own the Process

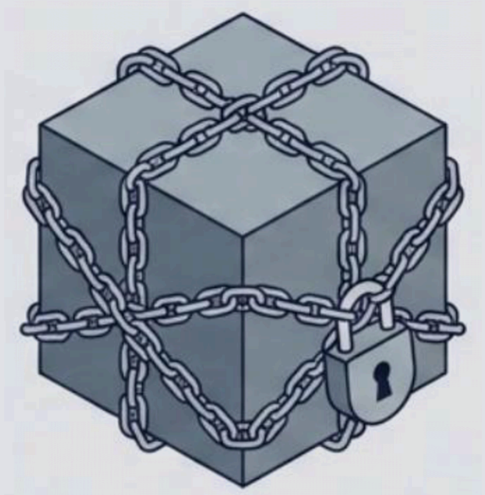
“Drop the lightweight chatbot.” Instead of fragmented AI demos or utilities, control an end-to-end (E2E) value chain that is complete from start to finish, however narrow.

UEngine

The Strategic Essence of Open Source: A Marketing Asset and a Beachhead of Trust

The only way an unknown startup can win over a giant enterprise customer is not to keep its source code to itself, but to give the customer unlimited control and freedom, removing their primal fear of vendor lock-in.

Proprietary Vendor Dependency (Lock-in)



– Closed software
– Permanent technical dependency (lock-in)
– Low mutual trust

Open Source: Freedom to Customize

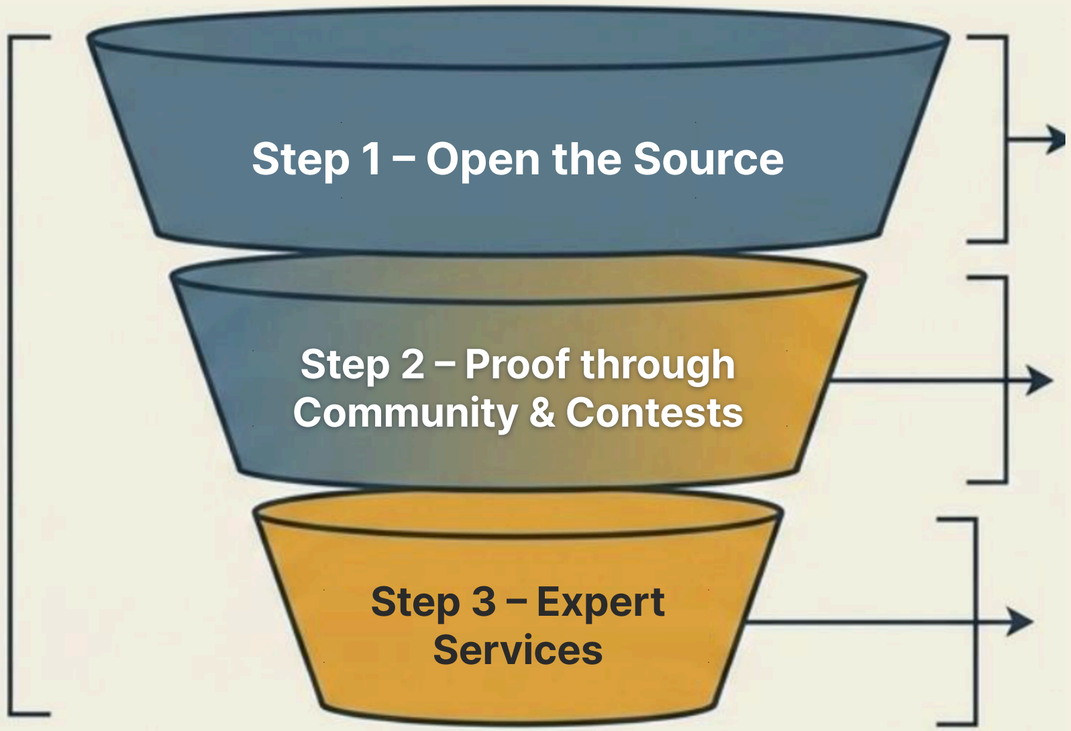


– Core source code fully open
– Guaranteed right to customize freely
– Faster shift to technical consulting and enterprise subscription models

UEngine

Open Source as Marketing Leverage: Charge for the “Risk”

Trust Funnel



Removes the fear of lock-in and provides an environment to verify it yourself
(lower adoption barrier)

Third-party validation and a public record of use
(builds trust)

Integration, operations, training, support
(turns into revenue)

> **The line between free and paid is not the “source code.”**
> **The “operational risk” customers find hard to handle themselves is where the business model begins.**

Small Teams, Huge Impact: 2025 AI-Native Company Leaderboard

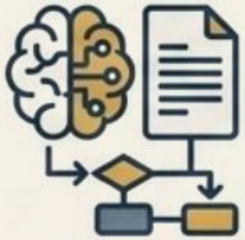
Telegram	30 employees	≈ ₩48.3 billion / person
Midjourney	40 employees	≈ ₩18.1 billion / person
Anysphere (Cursor)	20 employees	≈ ₩7.2 billion / person

Companies that have actively adopted AI tools are **creating unicorn-level value** with just a few dozen people.

UEngine

The Open Source 2.0 Era, Triggered by AI





Open Source 1.0 (Past)

CODE-centric.
An era of reusing and validating engines and tools.
(The preserve of developers with STEM backgrounds)

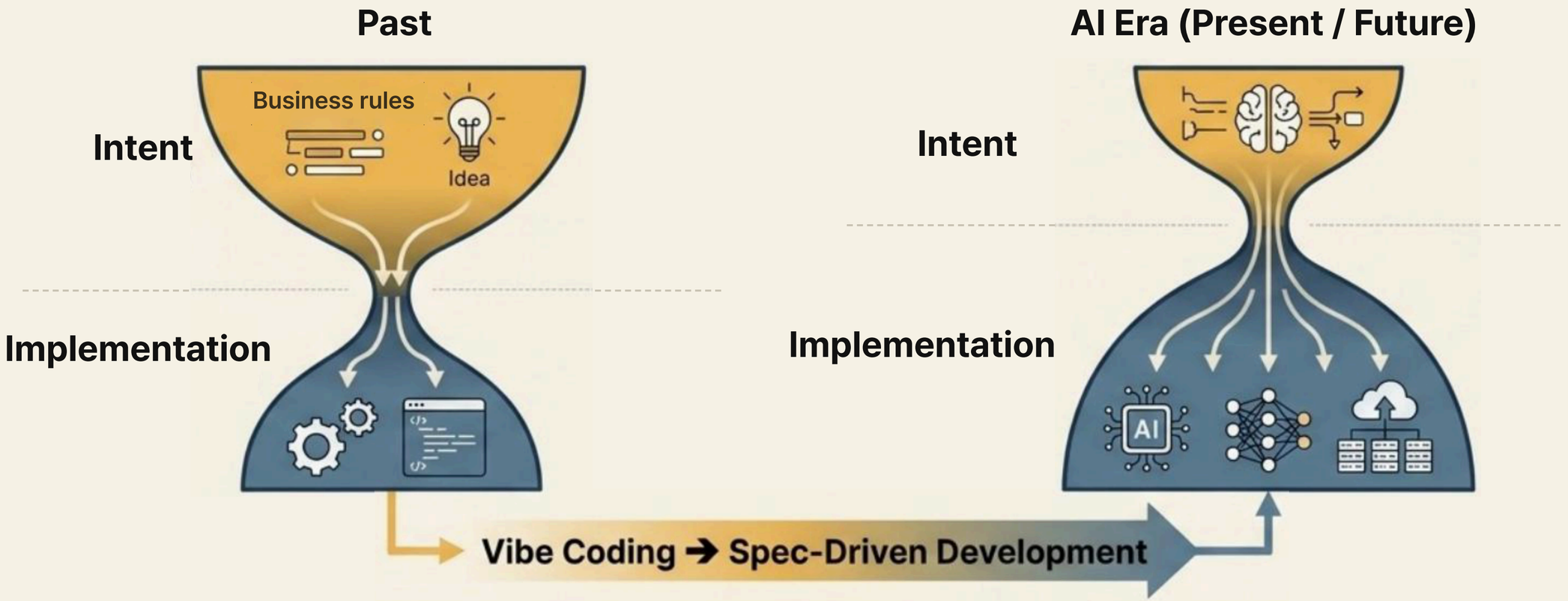
Open Source 2.0 (Present/Future)

SPEC & KNOWLEDGE-centric.
An era of co-developing business rules, domain specifications and organizational knowledge, and accumulating agents' execution records (Operating Learning).

UEngine

Trend 01. The End of Coding, the Reign of the “Spec”

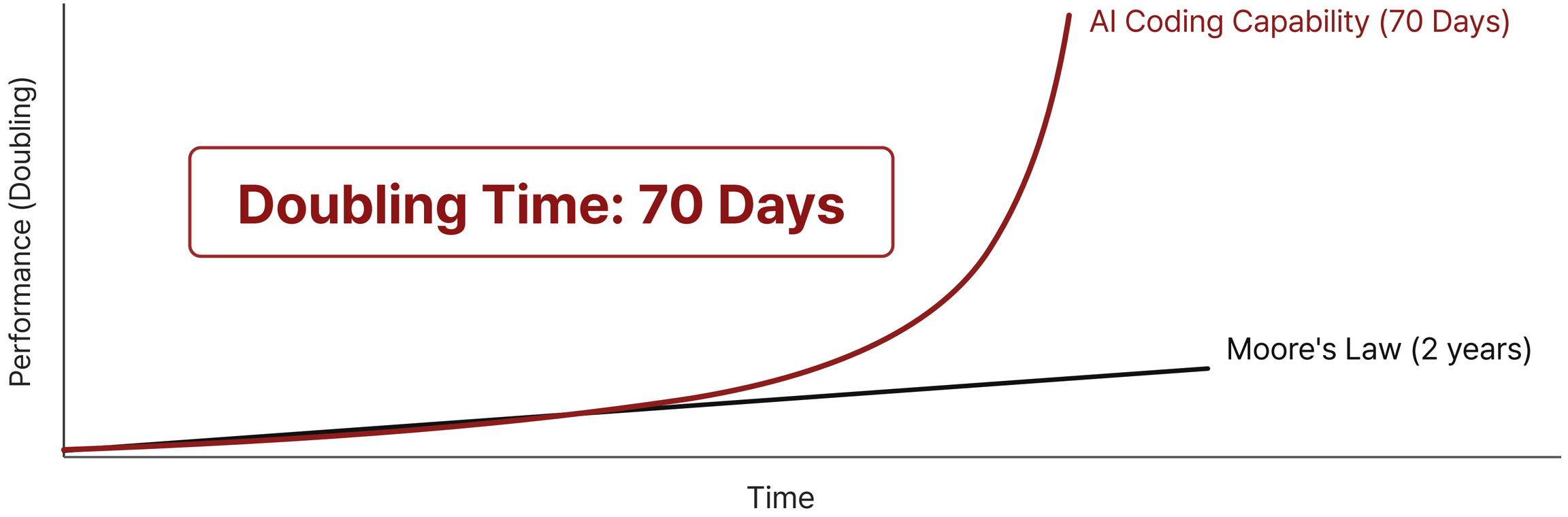
AI coding removes the implementation bottleneck and exposes the “intent bottleneck.”



Problem definition, domain concepts, normal / exception flows, roles / permissions.
Domain knowledge structured in natural language becomes the top layer of the source code.

UEngine

Real Metric 2: An “Explosion of Coding Intelligence” 5× Faster than Moore's Law

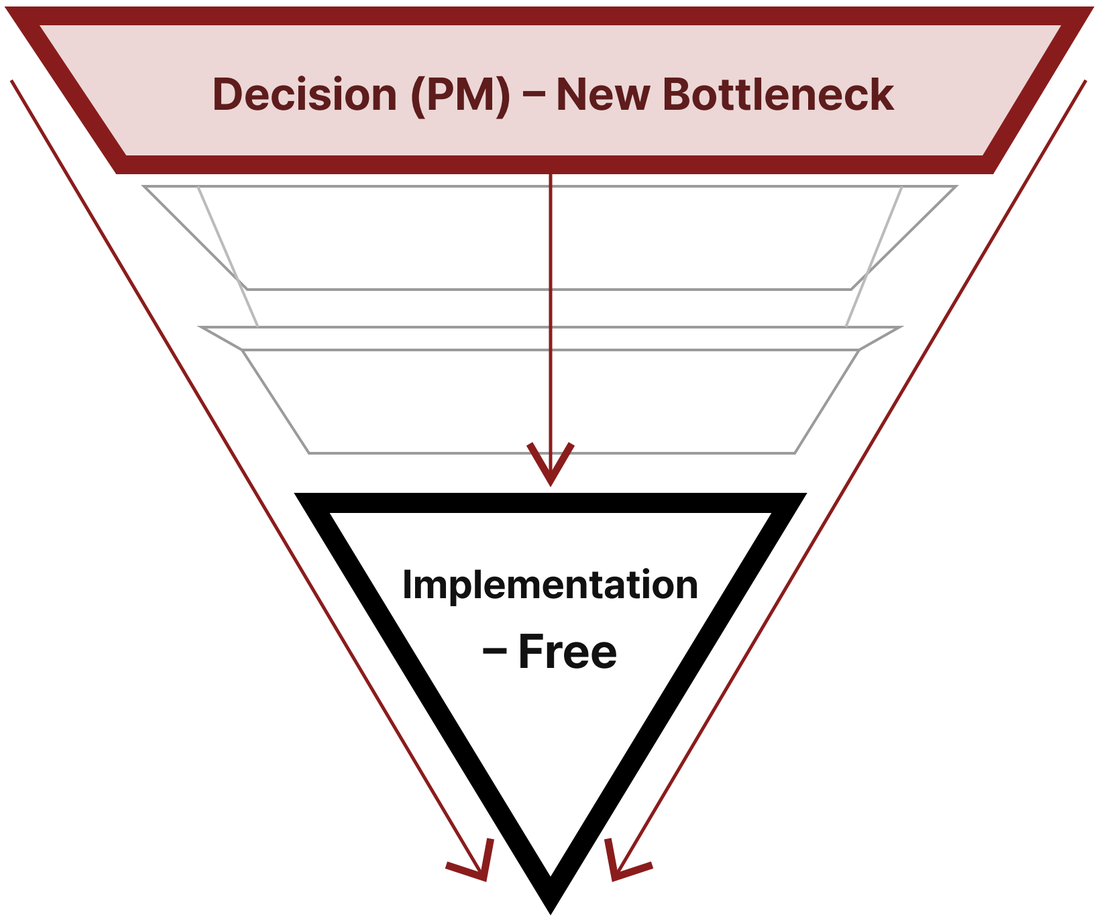


Chips get faster every two years, but AI coding capability doubles every 70 days.

The AI of 2026 will handle the code that took you an hour today without batting an eye.

UEngine

The Great Bottleneck Shift: Implementation Has Become Free

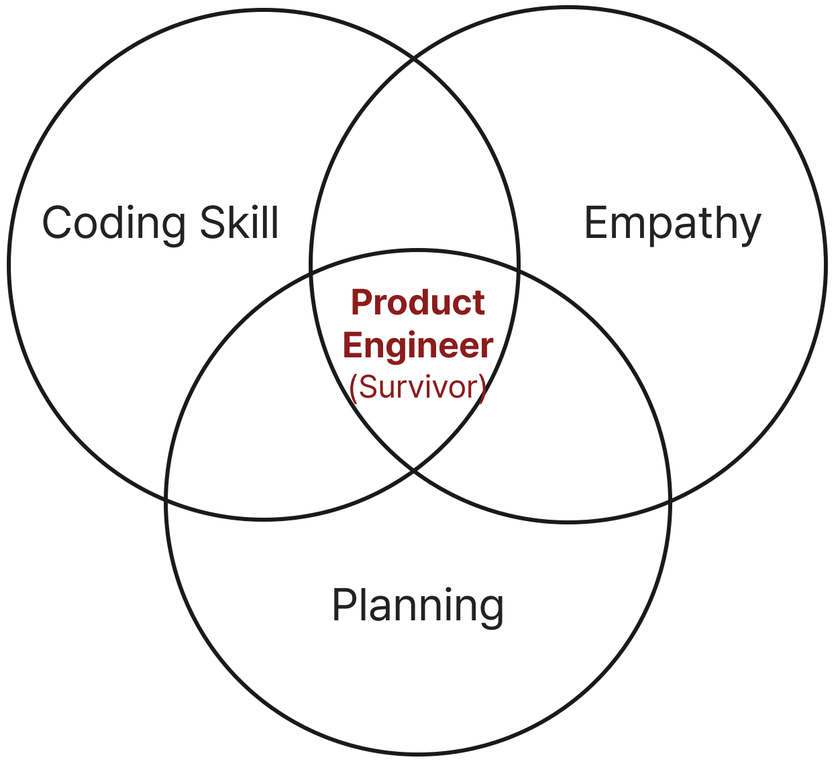


As the cost of implementation converges to zero, the battleground has moved completely from **“how we code it (How)”** to **“what we code (What).”**

Old Bottleneck: technical implementation
(dozens of engineers, days of work)

New Bottleneck: what should we build?

The New Survivor: A Product Engineer, Not a Coder



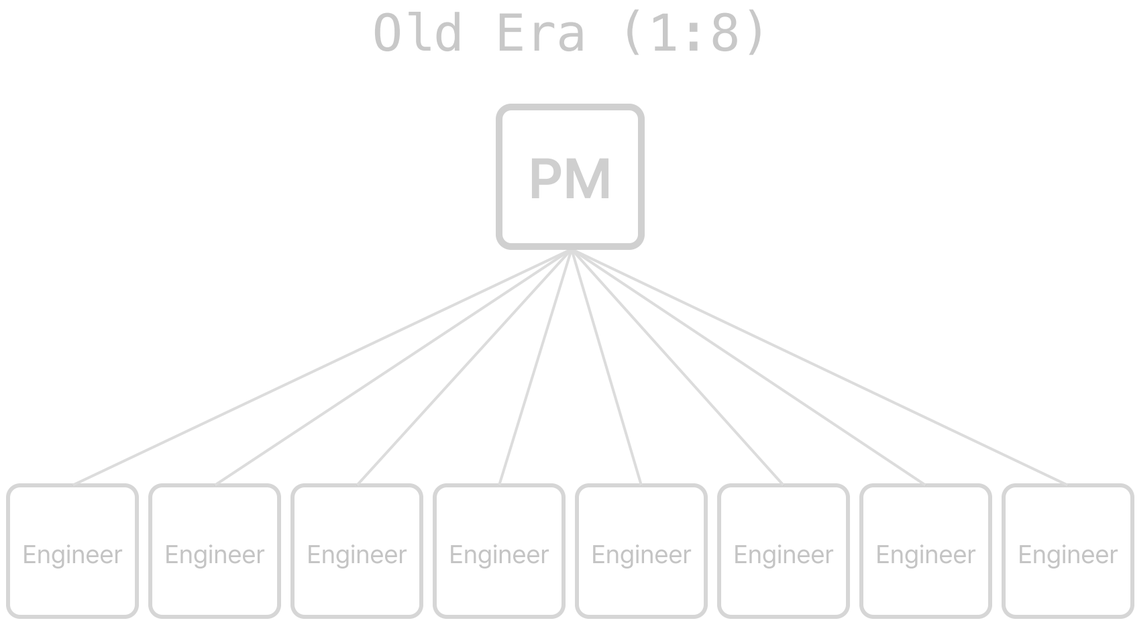
Past: a bricklayer who waits for the spec.

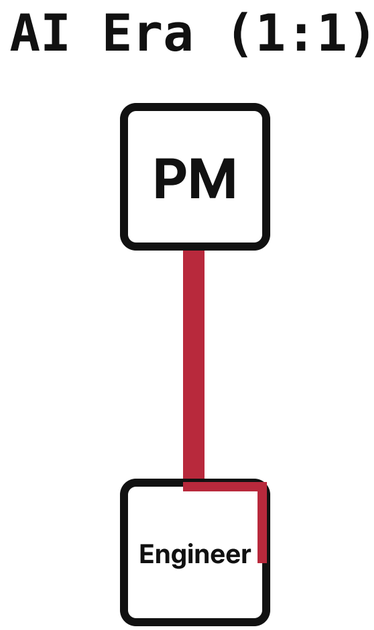
Future: a site manager who revises the blueprint and rebuilds.

“Don't wait for someone to hand you a plan. You have to make the decision yourself.”

UEngine

The Broken Golden Ratio: The 1-to-8 Era Is Over





AI coding has made engineers as fast as a spaceship, but human judgment **(PM)** still moves at bicycle speed.

In the end, the engineer and the planner will merge into **a single person**.

UEngine

Our Reality: From Chaotic Requirements to Valuable Software



The biggest challenge is turning customers' ambiguous, unbounded pain points into clear, structured implementation specs.

Implementable Spec

The traditional approach makes this conversion too slow, relying on heavy manual work and guesswork. It cannot keep up with the market.

What Is the Real Capability of a Senior Engineer Worth Their Salary?

~~Architecture design skills~~

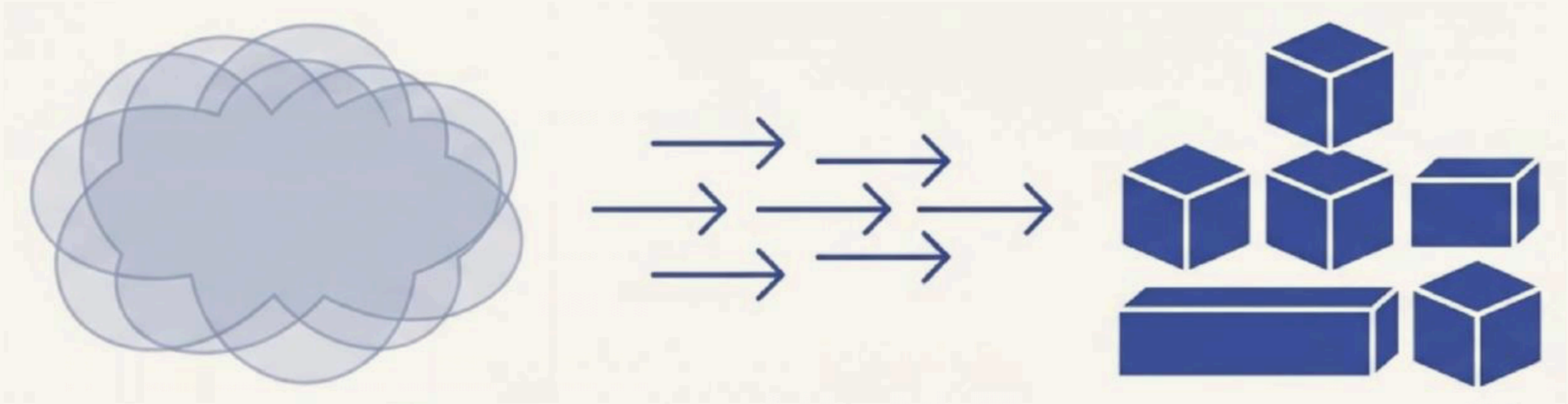
~~Outstanding leadership~~

~~Strong coding skills~~

The decisive difference is not on this list.

UEngine

The Ability to Make Uncertain, Ambiguous Problems Concrete



Abstract requirements

Executable plan

“A senior's value comes not from coding skill itself,
but from removing project risk and turning it into an executable plan.”

UEngine

The Two Core Axes of Vibe Coding



Requirements Engineering

Clearly defining “what to build.”
Deriving value-aligned, bounded requirements from customers' pain points.



Model-First Design

Designing and validating the essence of the problem, the domain model, first, without a UI.
The UI, which brings complexity, is implemented later.

UEngine

Your New Role: Analyst and Team Lead

The developer's role is shifting from writing every line of code yourself to directing AI, verifying its results and making strategic decisions. You are the leader of a teammate called AI.

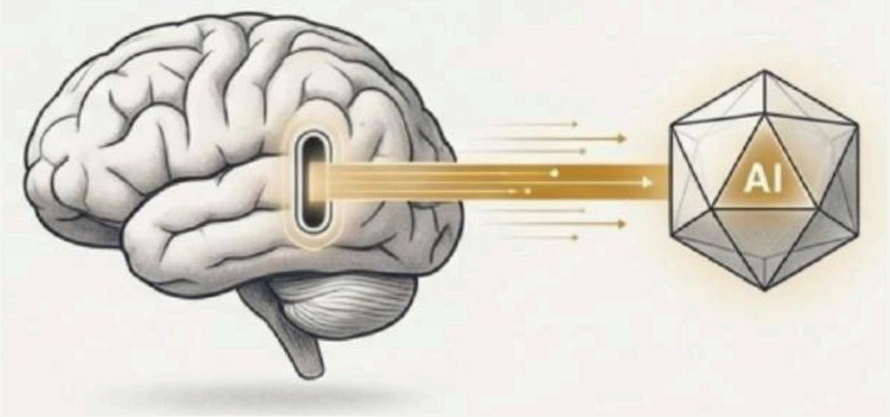
“I only set the direction AI should take.
AI leads the detailed thinking.”

A Shift in Mindset: Entrust Your Brain to AI

“Detailed thinking is led by the AI; my part is to set the direction.
Use AI the way you would use a member of your team.”



OLD MINDSET

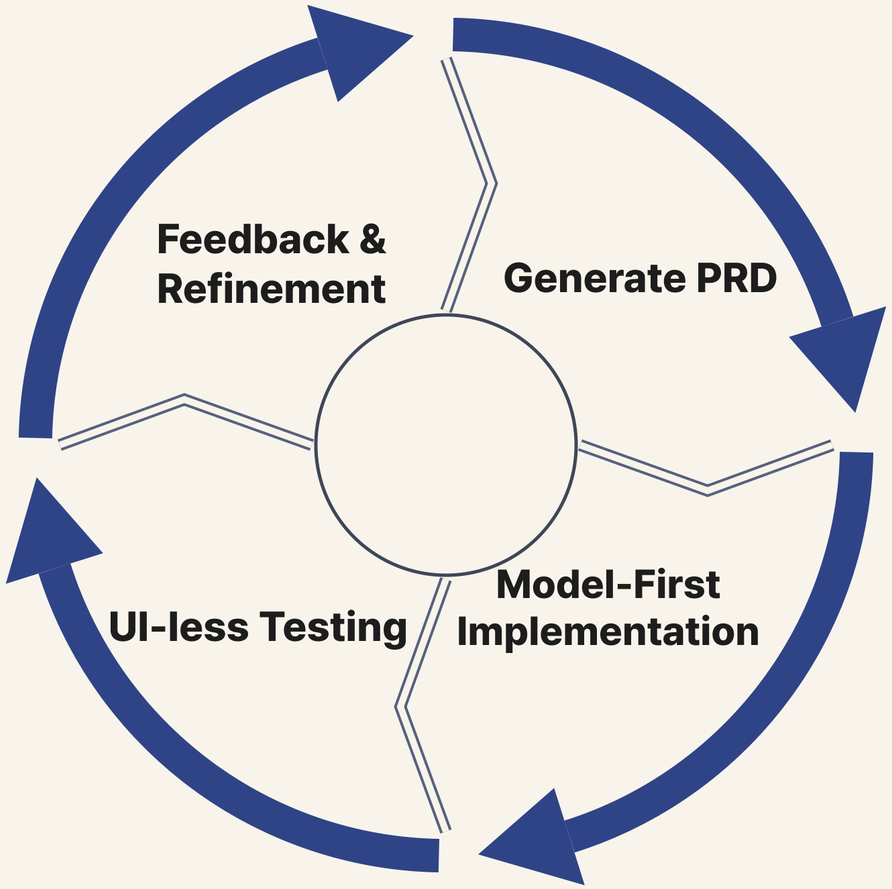


NEW MINDSET

OLD MINDSET (old-school thinking)	NEW MINDSET (new thinking)
I have to know everything	I know the direction and the end goal
Make a perfect plan from the very start	Complete it step by step through fast experiments
Code can be trusted only if I write it myself	I can understand and maintain AI-generated code

UEngine

A High-Speed Iteration Cycle Done in 30 Minutes



> **Key insight:** assume that whatever the first cycle produces will be thrown away. The goal is learning and refinement through fast execution.

UEngine

Step 1: Drafting the Requirements Spec (PRD) with AI

Human Input: Background & Pain Points

Users struggle to manage their newsletter subscriptions. They want to cancel multiple subscription services easily in one place and to filter out emails they are not interested in. Please write a PRD to solve this problem.

AI Output: Structured PRD

User Story 1:
As a user, I want to see all my newsletter subscriptions in a single dashboard so that I can manage them easily.

Acceptance Criteria (GWT format):

- Given I am logged into my account,
- When I navigate to the 'Subscriptions' page,
- Then I should see a list of all active newsletter subscriptions.



Value-aligned

Does it solve the customer's real problem?



Bounded Scope

Is it a small unit that can be implemented?

Steps 2 & 3: Validating Core Logic Without a UI

- **Model First:** focus on designing the domain model, the essence of solving the problem.
- **Test First:** verify through test code that every acceptance criterion in the PRD is met.

Engineer's Role:

Reading and understanding the generated code is what matters. If needed, ask the AI something like "Explain the structure of this code as a UML diagram" to deepen your understanding of the code.

"Bring in a UI and things get complex. The UI can be built later."

Architecture Must Evolve Together with the Business

Market validation comes first, not technical perfection. Introduce complex architecture (event bus, microservices) once the need has been proven.

Phase 1 (Initial Stage)



Goal: market validation

Architecture: single DB, monolithic module integration

Note: *Build an MVP quickly and focus on checking how customers react.*

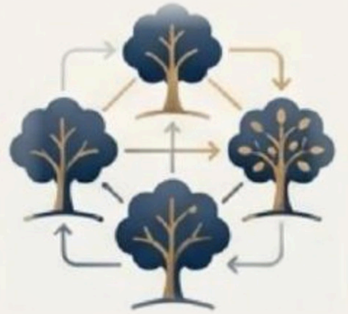
Phase 2 (Growth Stage)



Trigger: when the product starts selling, the number of modules grows and shared-storage problems begin to appear

Action: introduce an event bus to loosen the coupling between modules.

Phase 3 (Scaling Stage)



Trigger: large-scale traffic and growing complexity

Action: consider moving to a fully event-driven microservice architecture.

UEngine

Developer Manifesto for the AI Era



1. We are conductors, not coders.
We conduct AI to create value.



2. Speed over perfection.
We execute fast and learn in 30-minute cycles.



3. The model always comes first.
We validate core logic before the UI.



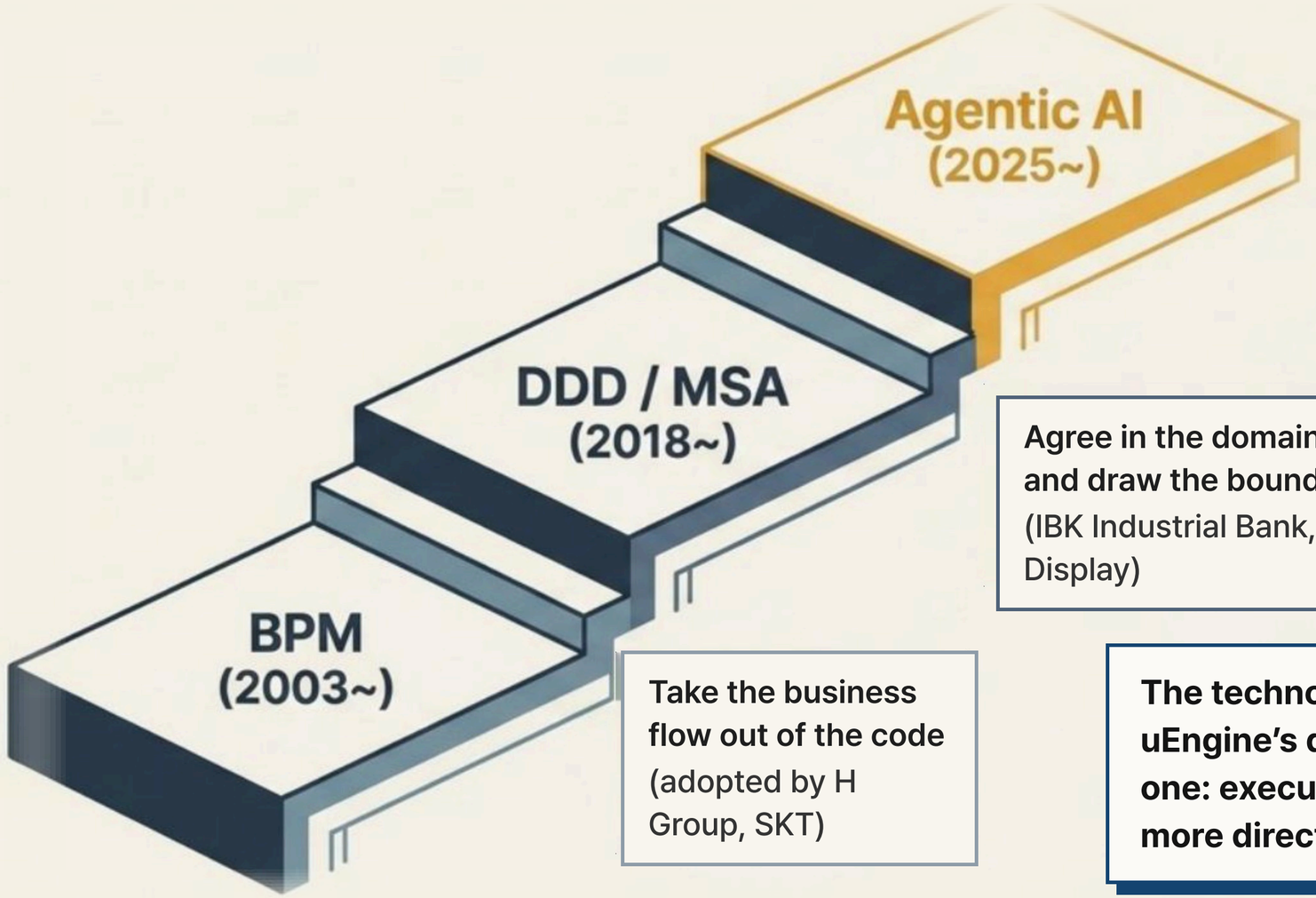
4. We automate integration.
We leave repetitive integration work to AI.



5. Architecture evolves.
We grow the system as the business grows.

UEngine

20 Years of Evolution: “How do we turn business intent into an executable structure?”



Agents execute natural language directly, grounded in knowledge and specifications
(Hanwha Life, POSCO DX)

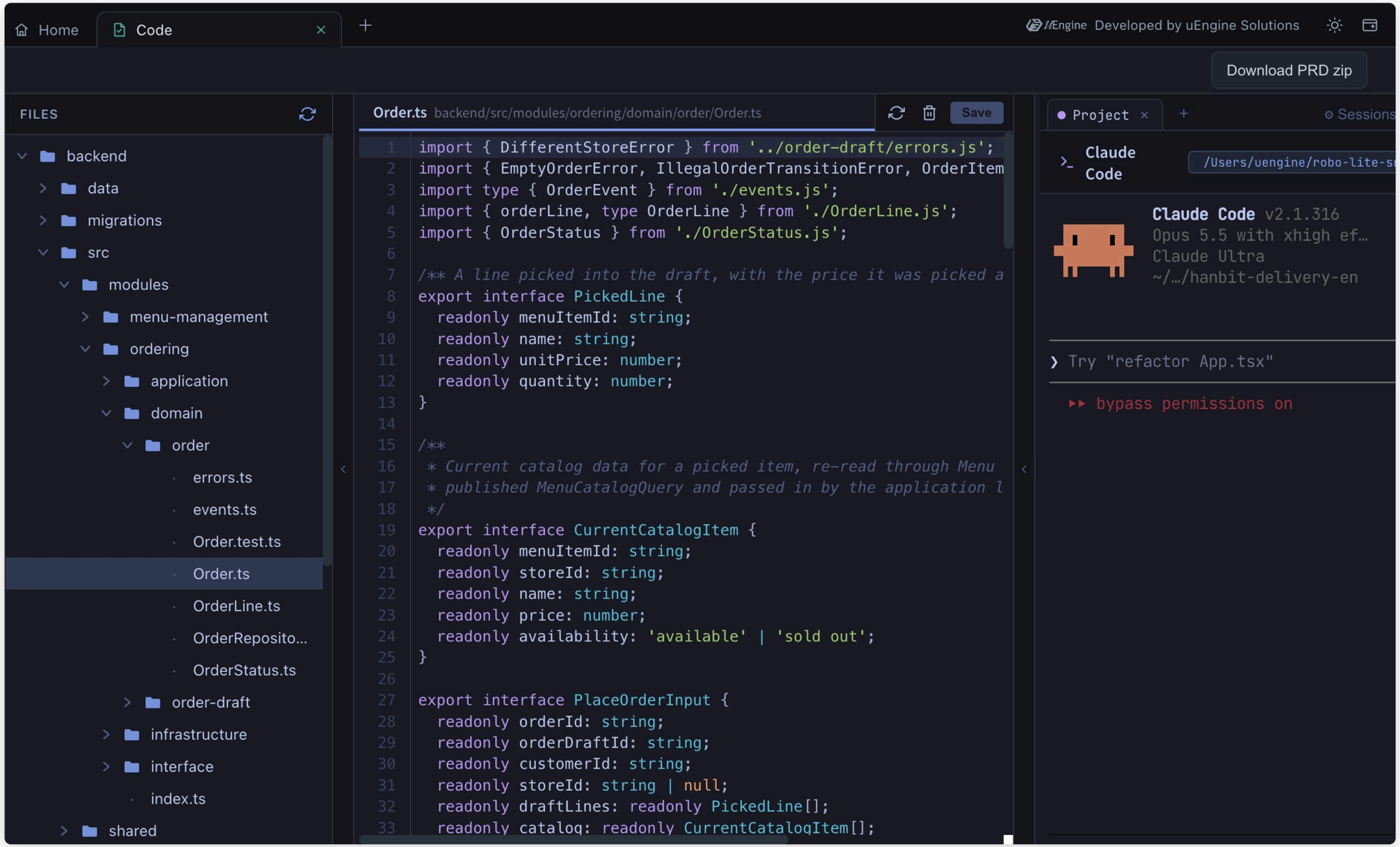
Agree in the domain language and draw the boundaries
(IBK Industrial Bank, LG Display)

Take the business flow out of the code
(adopted by H Group, SKT)

The technology names have changed, but uEngine’s direction over 23 years has been just one: executing what the business says it intends, more directly.

Spec-Driven Development, Completed: Robo Architect

	Off-the-shelf LCNC	uEngine (Robo Architect)
Code generation	Proprietary runtime scripts	Pure, standard general-purpose code such as Java and Python
Vendor dependency (Lock-in)	Very high	Zero: modify freely
Architecture scope	Small back-office apps	Large-scale cloud native / MSA
Domain mapping	Simple DB mapping / CRUD	Aligned with DDD and Aggregates



https://www.uengine.org/en/contents/roboarchitect.html

uEngine

Trend 02. From Personal Assistants (Co-pilot) to E2E Process AI

Assist (Personal Tasks)

Summaries, email, search.

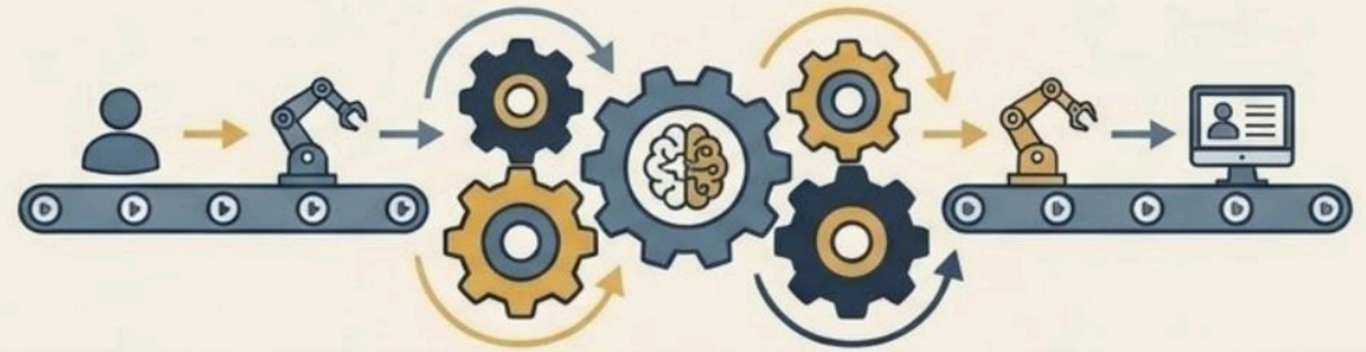
(Isolated chat windows, context lost between departments, and failed enterprise ROI because there are no process KPIs)



Orchestrate & Operate

Handoffs and collaboration among people, agents and systems.

Execution, observation and learning, completed end to end.

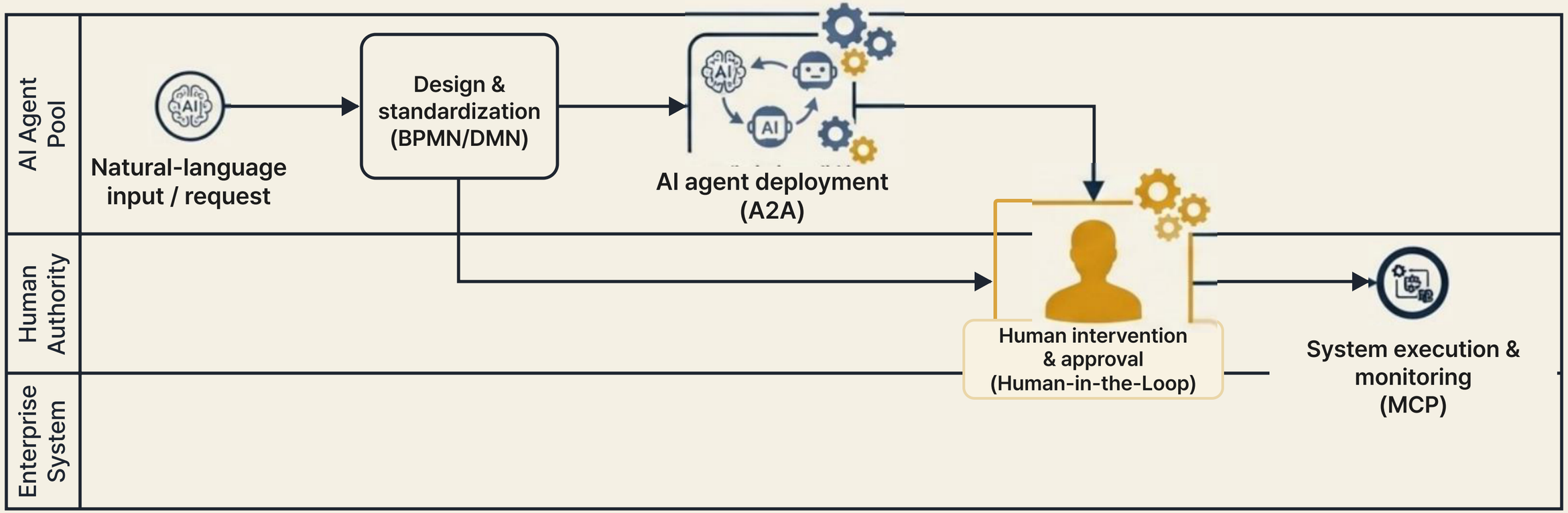


No matter how many personal assistants you deploy, end-to-end enterprise results do not appear on their own.
Enterprise results come from completing the value chain.

UEngine

Premium Enterprise Slide

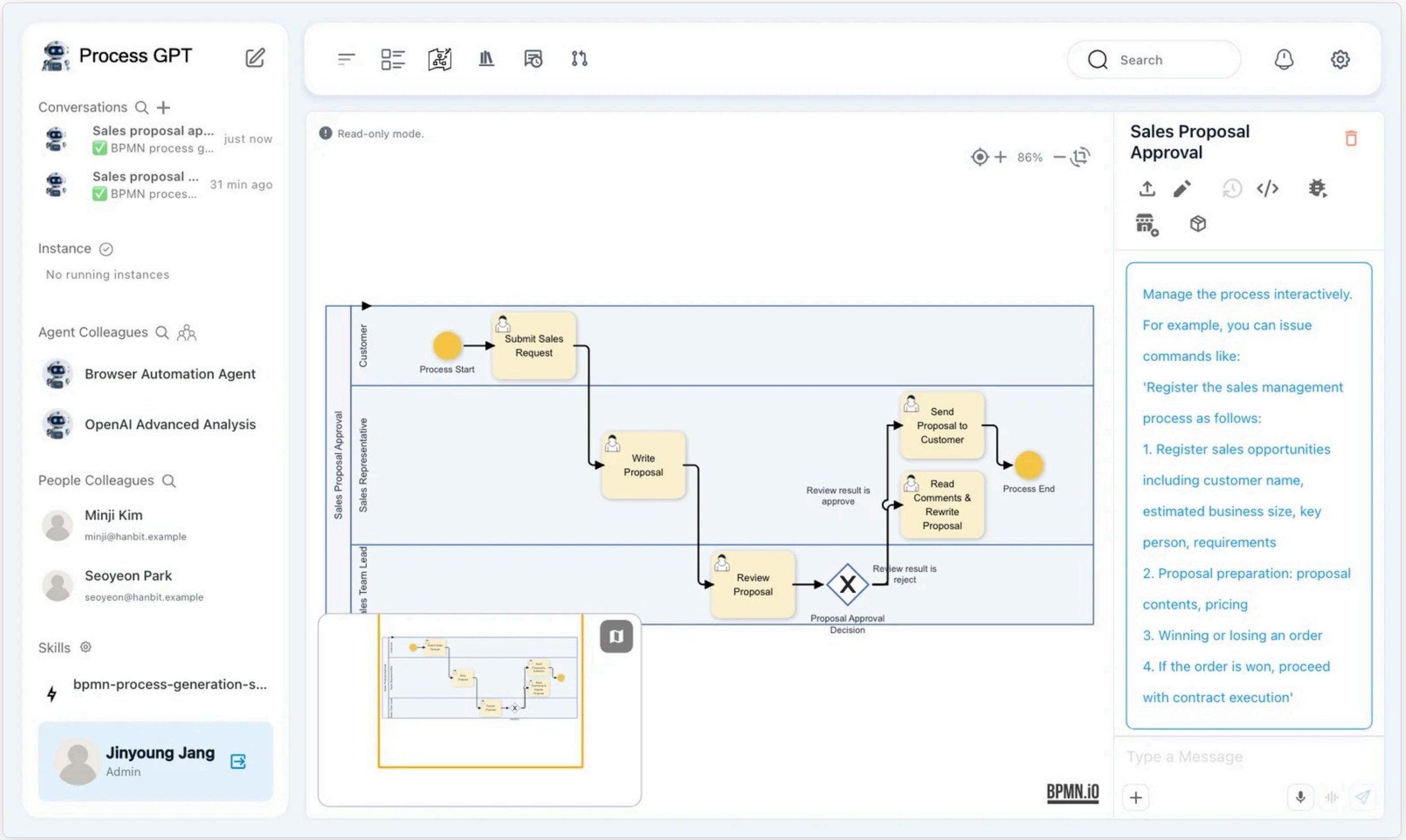
From Fragmented AI to a Business Execution Engine: ProcessGPT



Natural-language design → BPMN/DMN standard model conversion → Automatic agent deployment (A2A protocol) → Human intervention & approval → System execution (MCP protocol)

In ProcessGPT, BPMN is the complete ‘Operating Contract’ that balances agent autonomy with enterprise control.

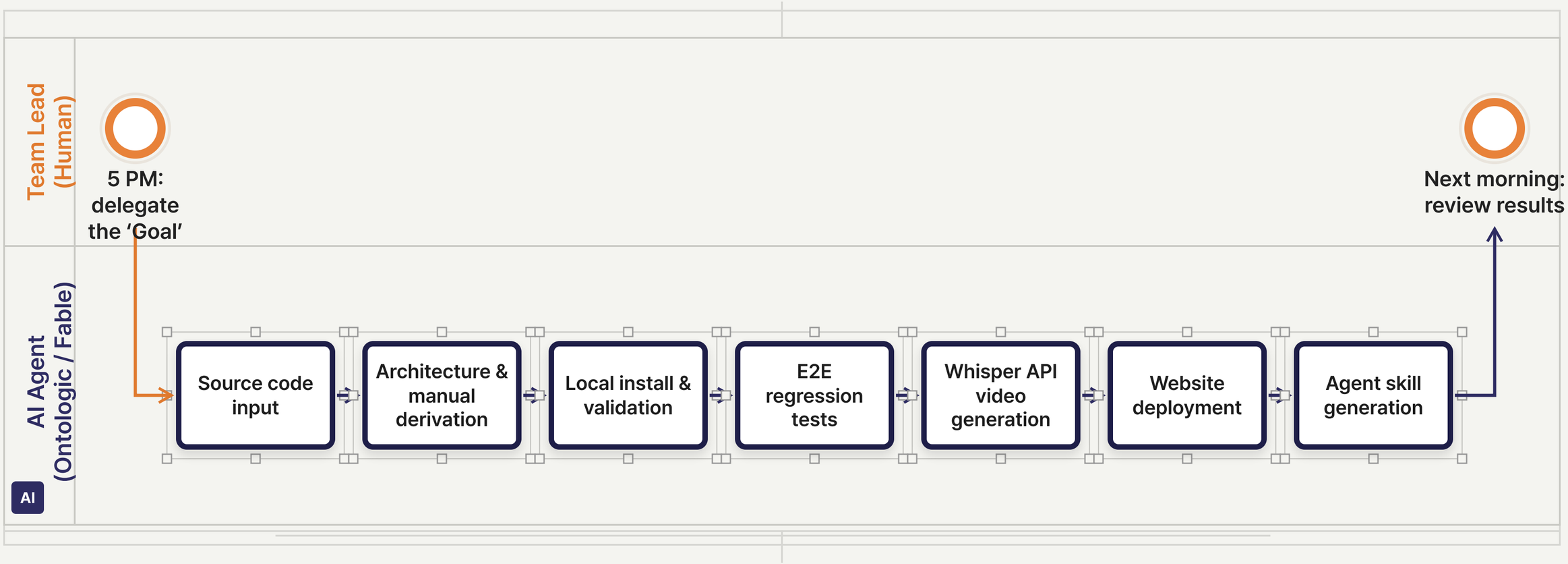
UEngine



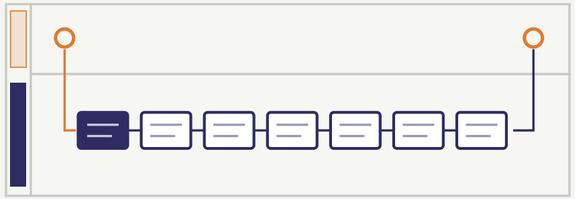
https://www.uengine.org/en/contents/processgpt.html

Autonomy Blueprint: The End-to-End Automation Master Process

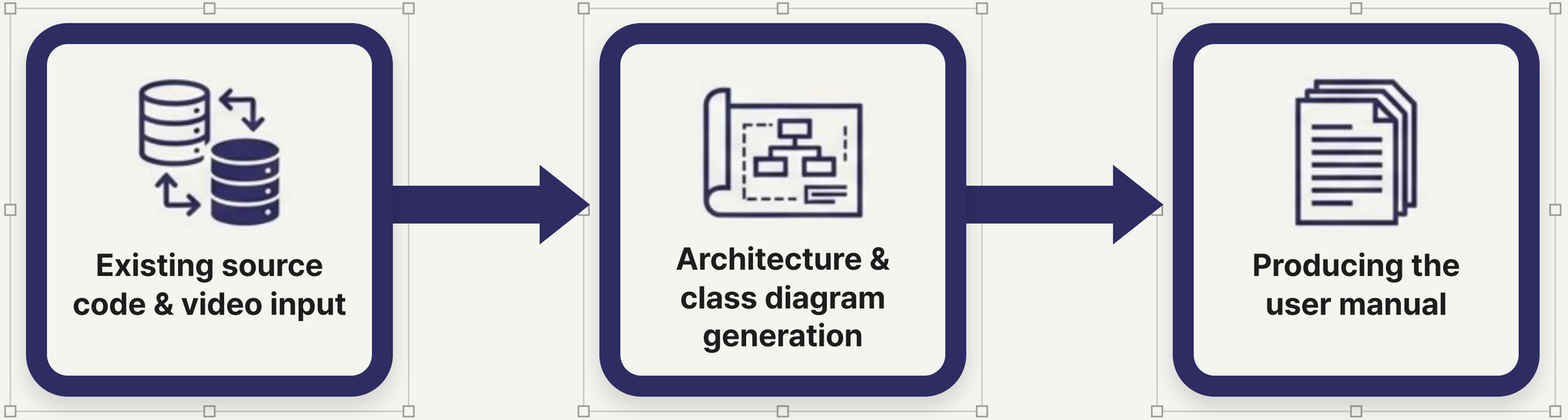
Development, QA, marketing asset creation and deployment, which used to take 3–4 days, are merged into one cohesive workflow that runs on its own overnight.



Step 1: Structuring Knowledge and Reverse Engineering

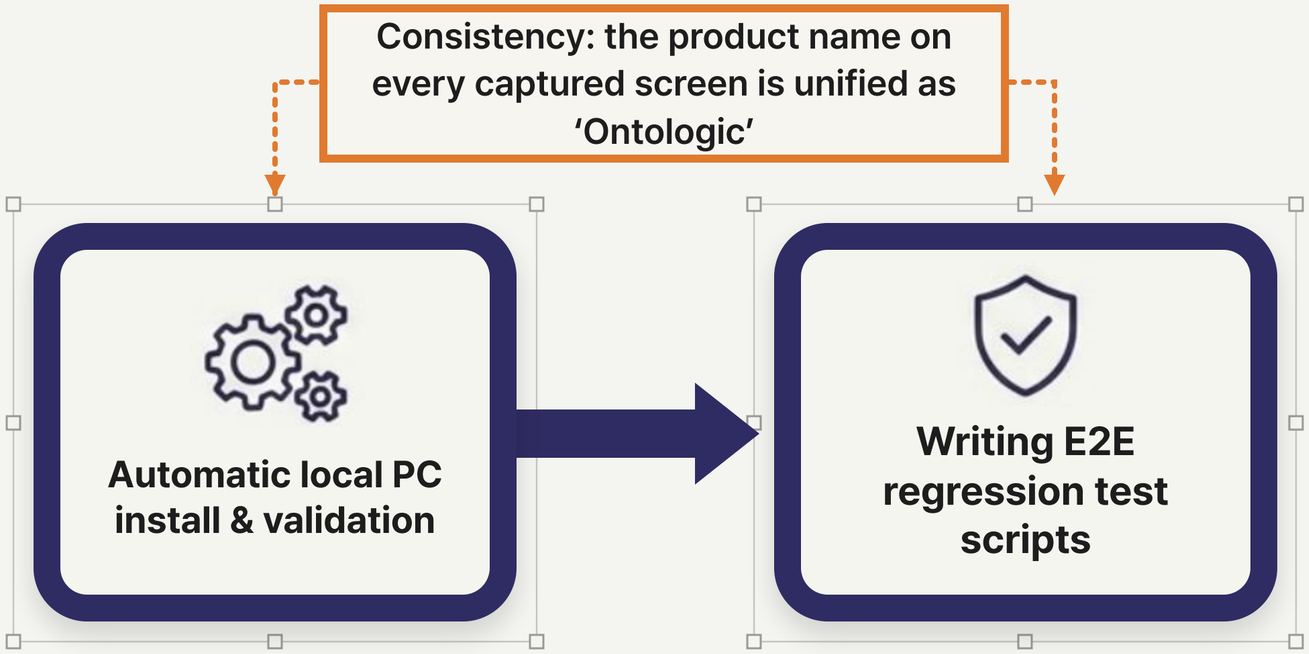


AI analyzes the existing code and past videos on its own to understand the architecture, traces back where each feature is implemented, and writes a complete user manual. The whole process runs without human intervention.



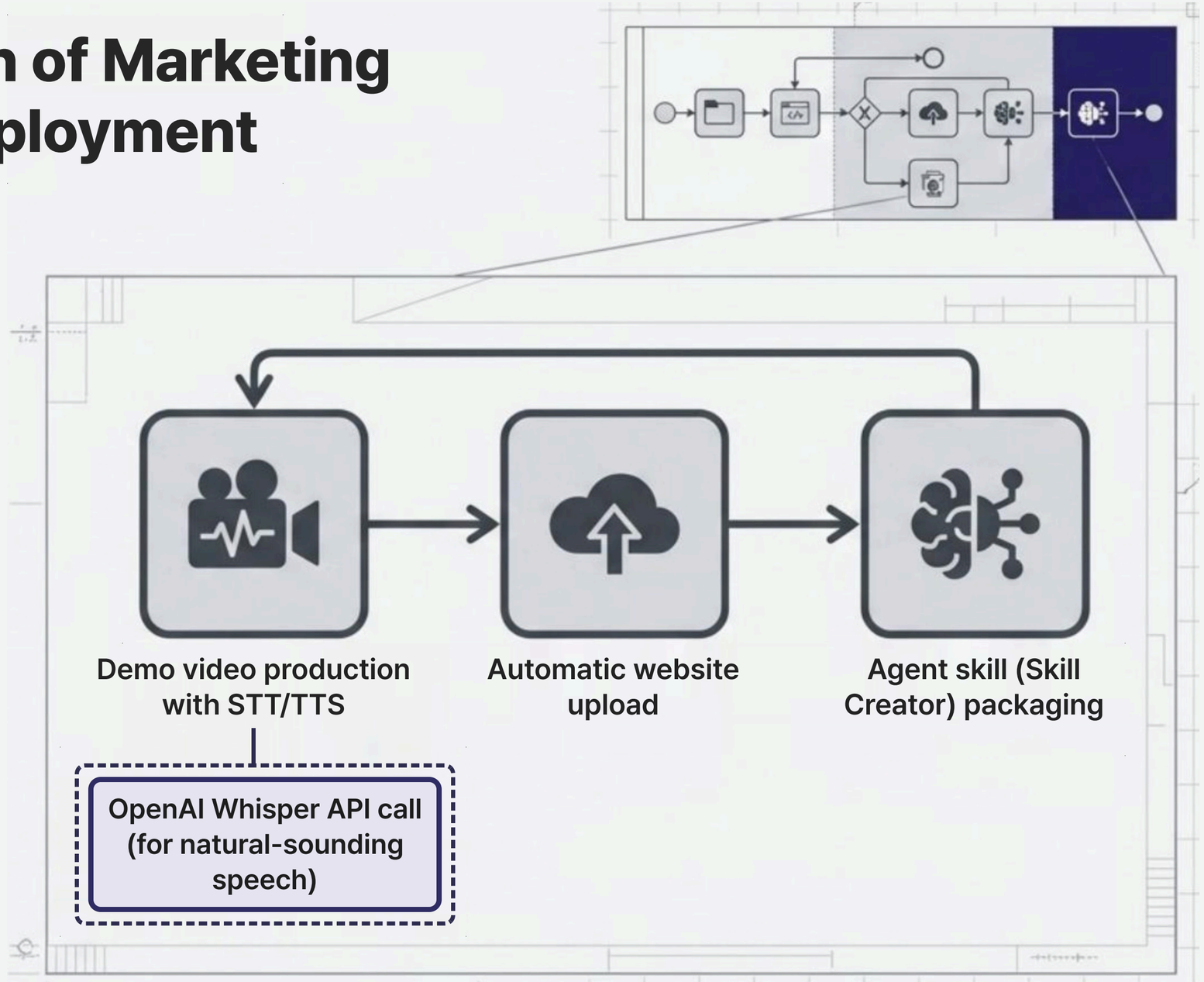
Step 2: Unattended Install Validation and E2E Test Automation

Based on the manual it wrote itself, the AI installs the product in a real environment and validates it. To prevent errors from ripple effects when features are added later, it also writes, ahead of time, end-to-end (E2E) regression test scripts that verify the whole environment.



Step 3: Full Automation of Marketing Asset Creation and Deployment

Using the screens captured during testing, it writes an explanation script. To avoid an awkward, robotic voice, it explicitly calls the Whisper API to produce a smooth video, then uploads it to the website and, in the same pass, generates an installation agent skill that others can use.



UEngine

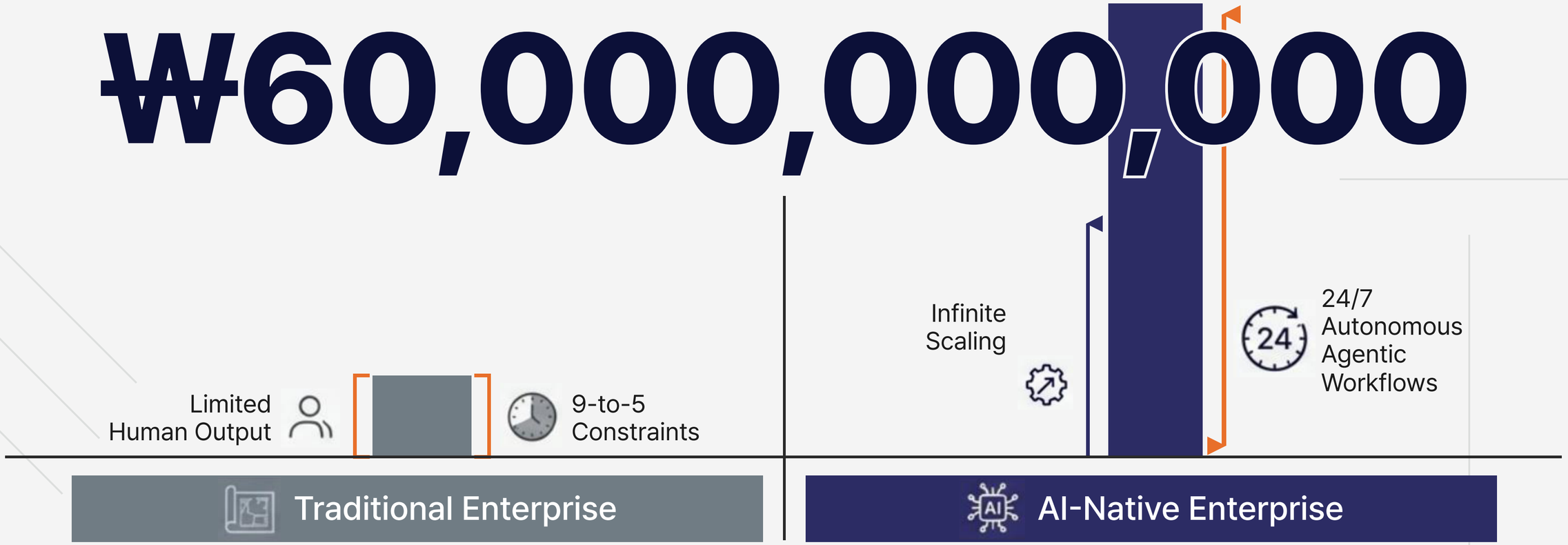
Micro-Prompting vs Goal-Oriented Delegation

When instructions are too long or too detailed, the AI loses its autonomy, and its performance drops or it spits the input back (Reject). The key is to give the AI as much autonomy as possible so that it schedules its own work.

Criteria	Micro-prompting (beginner)	**Agentic delegation (AI-native)**
How you instruct	Detailed step-by-step descriptions (nagging: “do this, do that”)	**High-level instructions centered on business purpose and intent**
Token efficiency	Inefficient, with frequent context breaks	**AI allocates resources itself to optimize**
Role of the AI	Simple executor	**Autonomous scheduler (e.g. Fable)**
Unit of work	Small chunks of 1–2 hours	**Huge chunks of 1–4 days of work**

₩60 Billion Revenue per Person: The Economics of AI-Native Companies

None of this is a mere technical curiosity. AI-native companies that have built agent workflows working autonomously day and night have already passed ₩60 billion in revenue per employee.



SYNTHESIS INSIGHT

Owning an infinitely scalable intelligent workforce is the only way to maximize results without overtime.

UEngine