frends

WHITEPAPER · 2026

Business-native integration: the Frends methodology

Processes, composability and AI. Why the whole is greater than the sum of the integrations.

Asmo Urpilainen
CTO, Frends

FRENDS TECHNOLOGY OY · VERSION 1.0 · OCTOBER 2026

BUILT TO CONNECT WHAT MATTERS.

The process is the unit. Everything else follows.

Most enterprise IT is organized around applications. The ERP owns finance, the CRM owns customers, the ticketing tool owns service. The logic that connects them, how an order becomes an invoice or how a fault becomes a repair, lives in nobody's system. It sits in integration code, in project documents and in the heads of the people who built it.

Frends starts somewhere else. Every integration is modeled and executed as a readable business process. The diagram you review is the thing that runs. The process, with its owner, its steps and its measurements, becomes the unit you design, govern and automate.

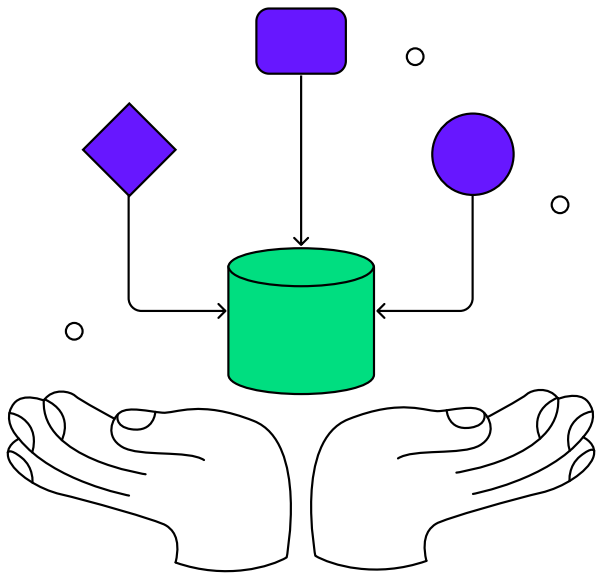
Those processes do two more jobs. Together they form the composable layer of your organization: the place where business capabilities are assembled, and where a system can be swapped without rebuilding everything around it. Increasingly, they are also the context every AI system in your enterprise reasons over. An AI that can read how your work flows can act inside it. An AI that cannot is left guessing.

The value compounds. The first process solves a problem. The tenth gives you an architecture. The fiftieth gives you an organization an AI can work inside. Each addition makes the earlier ones more useful, because they share a language, a catalog and an execution model.

The rest of this paper explains the mechanism, the architecture and what you must commit to for this to work. It is written for the people who have to make the decision together: the business owner, the enterprise architect and the integration lead.

CONTENTS

1	The problem you already have	4
2	The shift: business-native integration	6
3	Where this sits in your enterprise architecture	8
4	Cornerstone I: everything is a process	10
5	Cornerstone II: composable architecture	12
6	Cornerstone III: processes as context for AI	17
7	The compounding value	20
8	The operating model	22
9	Methodology in practice	24
10	Getting started: the first 90 days	25
11	Measuring success	27
A	Glossary	29
B	Comparison with other approaches	30



SECTION 1

The problem you already have

Application-centric IT leaves the logic that runs your business trapped inside systems and projects. You see the symptoms in the boardroom long before anyone names the cause.

The dreaded replacement

Replacing a core system takes years because nobody knows what else depends on it. The integration logic was never written down anywhere a business person could read.

Unknown transaction status

A customer asks where their order is. Four people check four systems. The answer is a reconstruction, never a fact.

Change by project

Every change to how work flows becomes a project with a budget, a vendor and a six-month horizon. Small improvements are never worth the overhead, so they don't happen.

Requirements argued after the fact

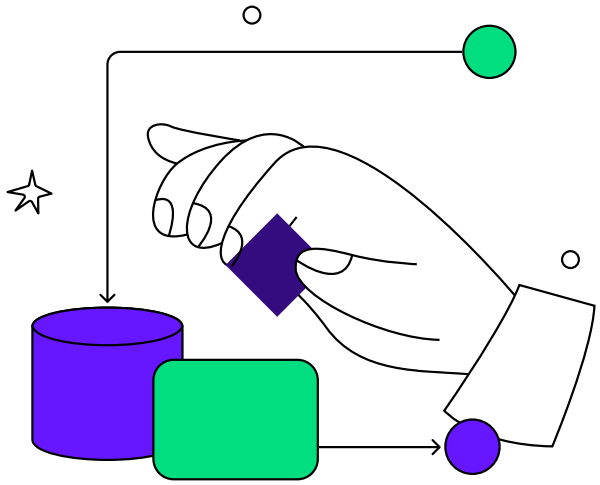
The business describes what it needed once the build is done, because the build was the first time anyone could see how the process behaved.

The AI version of the same problem

Now the same landscape meets AI. You want agents that handle claims, route orders and answer customers. The agent needs to know how a claim moves through the company. In an application-centric landscape, no single artifact holds that answer.

So the AI does one of two things. It invents a plausible process, which is dangerous because plausible is hard to spot in the output. Or it is confined to use cases small enough to fit inside one application, which is disappointing because those were never the expensive problems.

The context problem and the integration problem are the same problem. Nobody can explain how work flows, so neither people nor machines can act on it with confidence. Solving one solves the other.



SECTION 2

The shift: business-native integration

The process is the unit of design, ownership, measurement and automation. Everything else in this paper follows from that one choice.

Business-native integration means every integration is expressed in the language the business already uses to describe its work: a process with steps, decisions and owners. That expression is also what executes. Nothing gets translated into a technical form the business can no longer read, and nothing gets written twice.

The idea is easiest to see against the three approaches it replaces. Each of them optimizes for something real. Each of them breaks in a way that shows up on the executive agenda within a few years.

PROCESS-FIRST

Dated

Processes modeled in a notation tool, then implemented somewhere else. The model drifts from reality within months and nobody trusts it.

TECHNOLOGY-FIRST

Builds the unneeded

Picks the platform, then goes looking for work. Produces integrations nobody asked for and leaves the ones people need unfunded.

AI-FIRST

Automates what isn't understood

Fast at producing output, slow at producing trust. Puts a non-deterministic system in charge of work nobody has written down.

The three cornerstones

The methodology rests on three cornerstones. Sections 4 to 6 unpack each one. Section 7 shows why they are worth more together than apart.

CORNERSTONE I

Everything is a process

Every integration is a BPMN diagram, and the diagram is the implementation. What the business sees is what runs.

CORNERSTONE II

Composable architecture

Capabilities are parts. Processes are the composition layer that assembles, swaps and recombines them.

CORNERSTONE III

Processes as context for AI

A catalog of explicit, validated, executable processes is the most trustworthy context an organization can give an AI.

SECTION 3

Where this sits in your enterprise architecture

You already have an enterprise architecture. The methodology slots into it as the layer that has always been missing: executable processes between the business architecture and the application landscape.

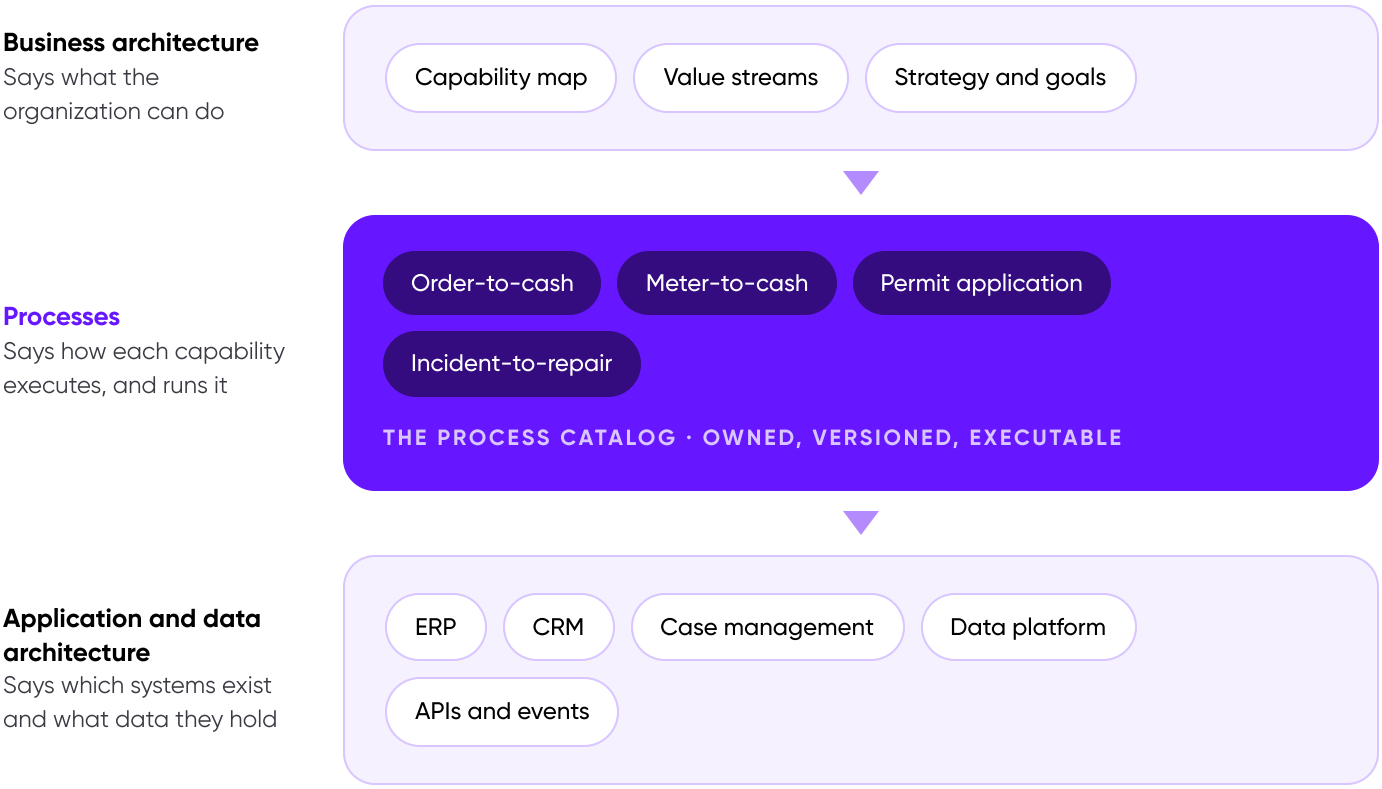


Figure 1. Processes as the executable layer between business and application architecture.

Most EA practices describe the business in capability maps and value streams, and describe the technology in application and data landscapes. Between the two sits a gap. The capability map says the organization can fulfill orders. The application landscape says the ERP, the warehouse system and the carrier portal exist. Nothing in the architecture says how an order travels between them, who owns that journey or what happens when it fails.

Processes fill that gap. Each cross-system process in Frends is a concrete, running answer to the question "how does this capability execute?" The process catalog is the index of those answers.

A living artifact

Architecture documents drift. A process diagram that is also the implementation cannot drift, because changing the diagram changes the behavior. The catalog stays accurate without a documentation effort. Your architects read the real thing.

Fit with what you already use

If you use TOGAF, the catalog lands in the business architecture phase as process models and in the application architecture phase as interface definitions, with one object serving both. If you use capability modeling, each capability points at the processes that realize it. Your architecture review board keeps its role and gains a review surface it can read, plus a change record that matches production.

What it does not do

This is integration and process architecture. It does not replace data governance, application portfolio management or security architecture. It feeds them: every process declares which data it touches, which applications it depends on and which identities it runs as. Those facts flow to the disciplines that own them.

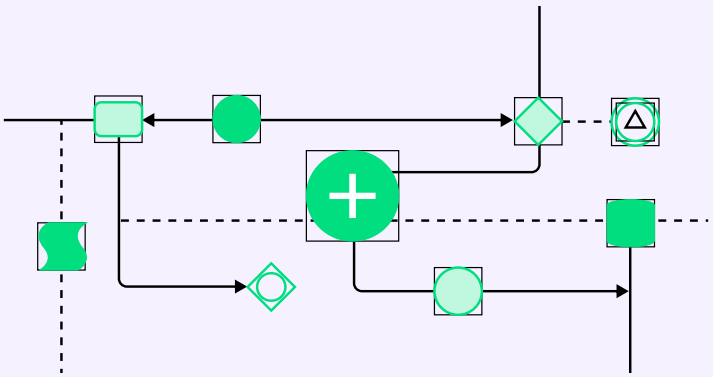
Everything is a process

In Frends, every integration is a BPMN diagram, and the diagram is the implementation. What the business sees is what runs.

Frends models every integration as a process in BPMN 2.0, the standard notation for business processes. A process has a trigger, a sequence of steps, decisions and exception paths. Each step is a Task: call an API, query a database, transform a document, wait for an approval, ask an AI model a question. The process runs on an Agent in your environment, and the Control Plane records every execution.

The diagram runs

There is no second artifact. Nobody translates the picture into code or keeps a document in sync with a deployment. When the process owner approves a change to the diagram, that change is what gets deployed. This removes the most common failure in integration work: the gap between what was agreed and what was built.



A process as the business reads it and as the Agent runs it.

Readable by the people who own the work

The property that matters is that a non-engineer can read the process. A claims manager can look at the claims intake process and say "that decision is wrong, we check the policy before the fraud score, not after." They don't need to understand how the policy lookup is implemented. They need to see that it happens, where it happens and what the outcome routes to. BPMN gives them exactly that.

A shared object

Because the diagram is both readable and executable, it stops being a technical artifact and becomes a shared object. The business owner, the architect and the developer look at the same thing and mean the same thing by it. Requirements discussions happen on the diagram before the build. Incident reviews happen on the diagram after a failure, with the actual execution path highlighted. Audit questions are answered by pointing at a step.

What this gives you

Every integration has a name a business person recognizes. Every execution is traceable to a step in a diagram that person can read. Every change is a visible diff between two versions of that diagram, reviewed by the person accountable for the outcome.

Composable architecture

Your organization is a set of business capabilities that can be assembled, swapped and recombined. Processes are the composition layer that holds them together.

5.1 Capabilities instead of monoliths

The one system that does everything is ending in practice, even where it survives in procurement. You buy packaged business capabilities: a billing engine, a customer identity service, a field-service tool, each best at one job. Composability means those capabilities combine into business processes without each one needing to know about the others.

5.2 Processes as the composition layer

For composition to be real, the orchestration logic must live outside the applications. If the ERP calls the CRM directly, the two are coupled and neither can be replaced alone. If a process calls both, each application exposes a capability and the process owns the sequence. The applications become interchangeable parts. The process is the design.

5.3 Reusable process components

Steps that recur, such as validate a customer, post a journal entry or notify the account owner, become shared sub-processes. Each has one owner, one version and one place to fix. New processes are assembled from them, and a fix in one component reaches every process that uses it.

5.4 Swap without rebuild

Replacing a system means re-implementing the steps it handled. The workflows connected to it keep their shape; only the Tasks that touch the old system change. This is what turns the dreaded ERP replacement into a bounded piece of work with a known scope.

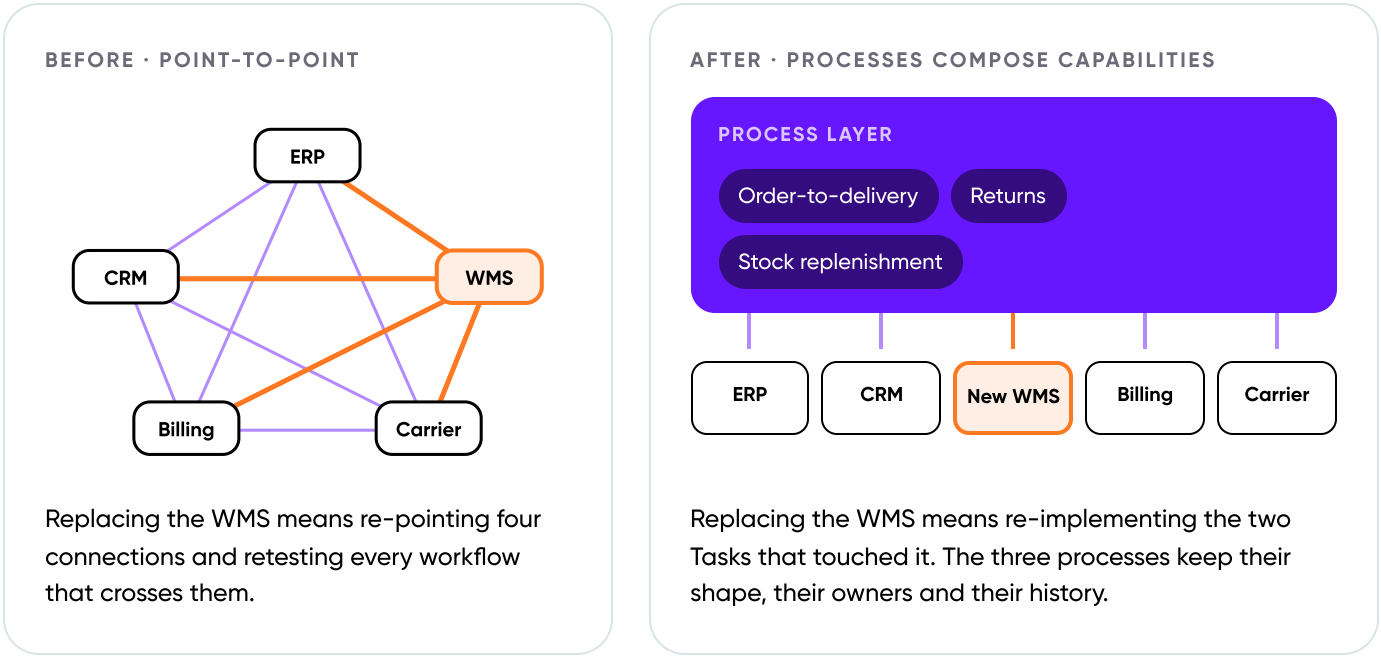


Figure 2. Swap without rebuild: the change is bounded by the steps a system handled, not by everything connected to it.

Composable, loosely coupled, centrally governed

Reuse only pays if it doesn't turn the process layer into a new monolith. Three properties keep it from becoming one.

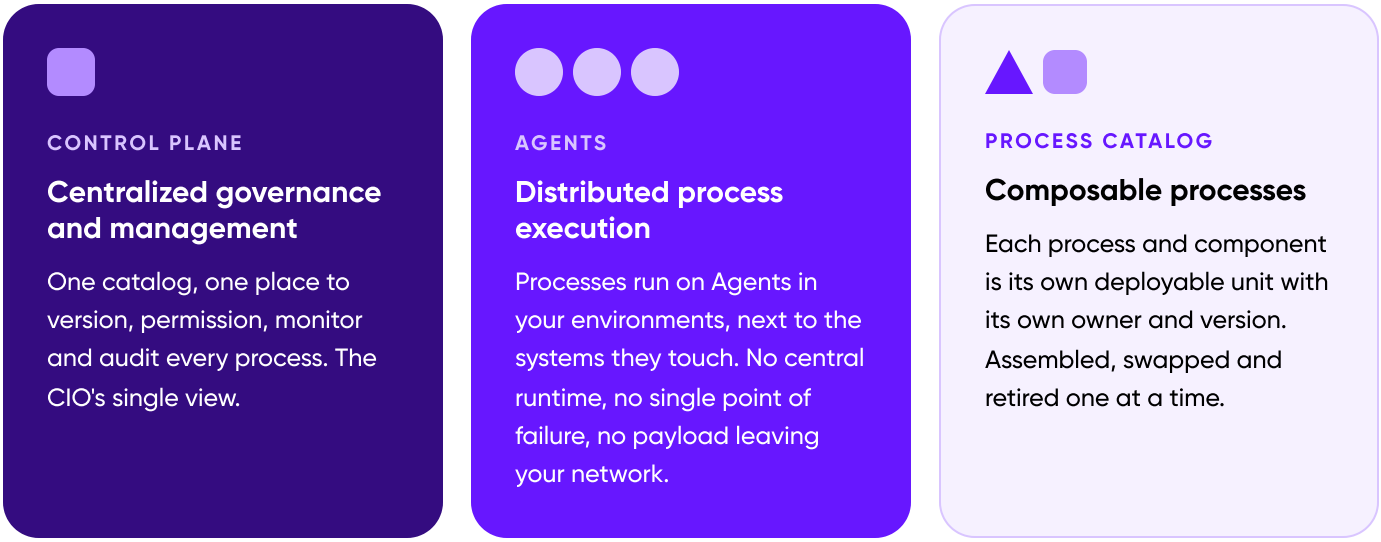


Figure 3. Governance is central. Execution is distributed. Composition happens in between. A monolith has none of these properties.

Loosely coupled by design

A shared component does not hold a rigid contract with the processes that call it. Steps exchange dynamic JSON, the way services do in a microservice architecture. A sub-process can gain new fields or new behavior, behind a feature flag if you want, without any existing caller changing or breaking. Callers read what they need and ignore the rest.

Dependencies still exist. The method minimizes them and makes them visible: every process declares the components it uses, and every component shows the processes that use it. Nothing hides in a codebase.

What limits regression pressure

Three things. Change is additive, so most changes break nothing by construction. Every component is versioned, so running processes finish on the version they started on and a new version is adopted by one process at a time. And the execution record shows exactly which processes touched a component in the last 30 days, so the test scope is a list, never a guess.

WHAT ARCHITECTS WILL ASK

"Nobody in the business reads BPMN."

Nobody reads it for fun. The owner reads the top level of their own process when a decision is wrong or a change is proposed, and that level mirrors the business process. Sub-processes are built for the implementation and the owner never has to open them.

"A component used in 50 processes is a monolith by another name."

A monolith couples everything to one release and one runtime. Here each component is versioned on its own, callers are tolerant of change and execution is distributed across Agents. Fifty callers is a list to check, never a single deployment to fear.

"You say no lock-in. There is always lock-in."

Yes. Every platform creates dependency; the choice is where it sits. Better outside the core systems than inside the ERP. Your processes are BPMN 2.0, so the IPR you build is the process model, and that is portable in a way custom code inside a core system never is.

The patterns that keep the process layer composable

Composition makes technical demands. The patterns below are familiar to any integration architect. What changes is their job: each one exists to keep the process layer composable.

API-led

Capabilities are exposed as managed interfaces so a process can call them without knowing their internals.

Event-driven

Systems announce what happened. Processes react. Nobody polls, and producers stay decoupled from consumers.

Orchestration vs. choreography

Orchestrate when one owner must see the whole sequence. Choreograph when autonomy matters more than a single view. Default to orchestration for anything the business must trace.

Error handling

Exception paths are part of the diagram, with retries, compensation and human escalation modeled where the business can see them.

Versioning

Every process and component carries a version. Running instances finish on the version they started on; new ones take the new one.

Observability

Every execution is recorded step by step, so a transaction can be found, explained and replayed without reading logs from five systems.

Processes as context for AI

AI inside an enterprise is only as good as its understanding of how that enterprise works. A catalog of explicit, executable, human-validated processes is the richest and most trustworthy context you can give it.

6.1 The context problem

An AI model reasoning over an application-centric landscape has to infer the process from fragments: a schema here, a ticket there, a conversation with whoever remembers. It fills the gaps with plausible guesses, and plausible is exactly the failure you cannot detect by reading the output.

A process catalog removes the guessing. The steps, the decisions and the exception paths are written down, validated by their owners and proven by execution logs. The AI reads the same object the business reads, and the answer it gives is grounded in what the organization has agreed and run.

6.2 Deterministic execution for non-deterministic reasoning

The division of labor is simple. AI decides; processes act. A model may classify a document, pick the next best action or draft a response. The action itself, posting the order, releasing the payment, updating the record, runs as a deterministic process with an audit trail, guardrails and predictable behavior.

This is what lets you let an agent do something consequential. The trust rests on a process that has run ten thousand times, with every run recorded, rather than on the model's ability to execute correctly on its own.

Two directions: AI inside a process, processes as tools for AI

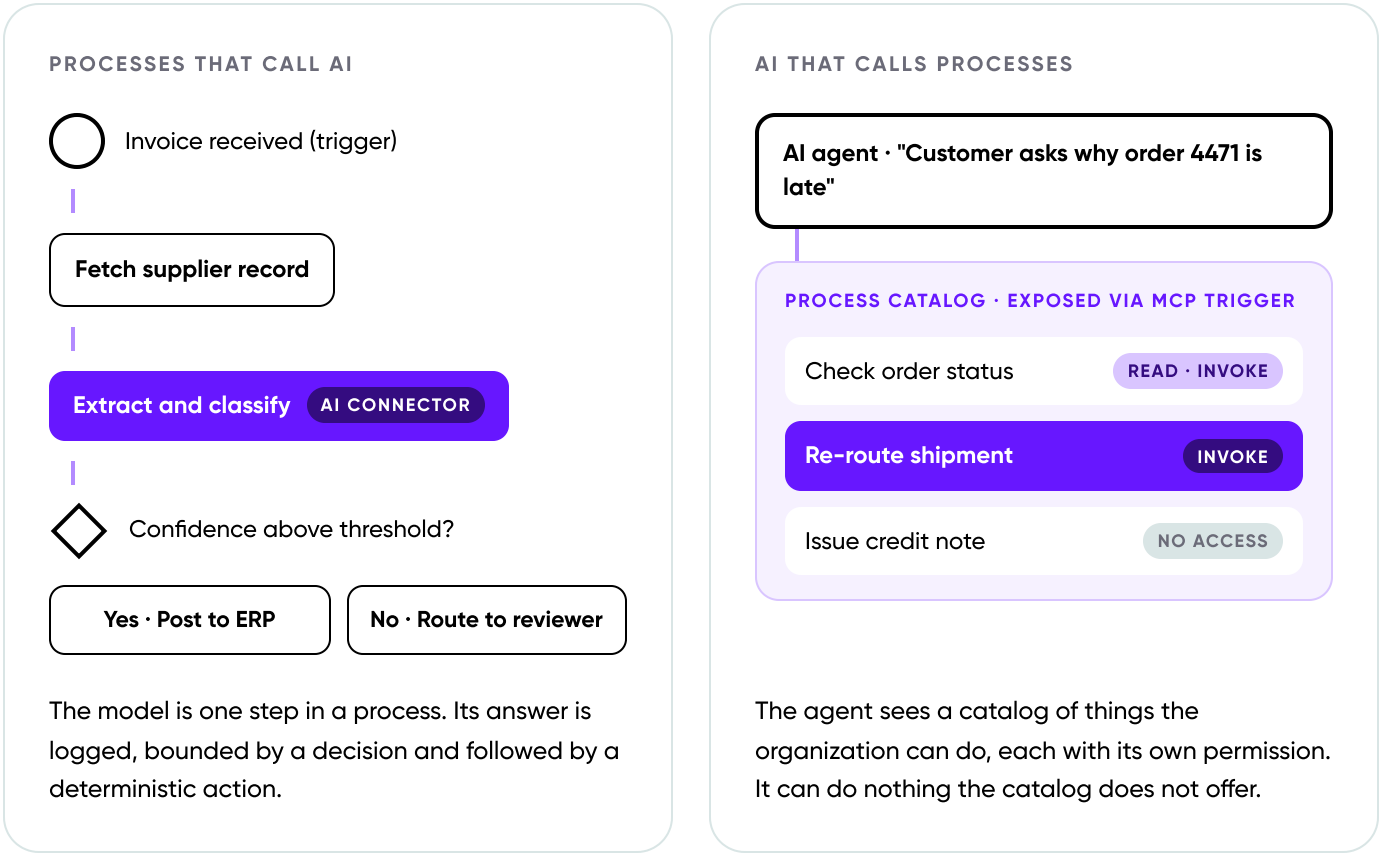


Figure 4. Two directions: AI as a step inside a process, and processes as tools for an AI agent.

6.3 Processes that call AI, AI that calls processes

AI appears inside a process as a step. The AI Connector sends a document, a message or a record to a model and routes on the answer. Classification, extraction and decision support slot into the diagram like any other Task, with the same logging and the same owner.

In the other direction, AI agents invoke processes as tools. Through the MCP Trigger a process becomes a capability an agent can call, with a defined input, a defined output and the permissions of the process rather than those of the agent.

6.4 From intent to integration

AI also accelerates building and changing processes. Describe the intent and Frends AI drafts the process. The human-readable diagram is the review surface: a process owner reads the draft, corrects the decision that is wrong and approves it without reading code. Change gets cheaper again, and the review stays with the person accountable for the outcome.

6.5 Governance

The process catalog is the natural permission boundary. For every process you decide which AI systems may read it, which may invoke it and which may propose changes to it. Read access gives context. Invoke access gives capability. Modify access is reserved for assisted authoring with human approval. Because the boundary is drawn around processes rather than data tables or APIs, the people who own the work can understand it and defend it.

This is what we mean by the deterministic backbone: the layer where non-deterministic reasoning meets predictable execution, and where every action an AI takes is one your organization has already modeled, approved and run.

The compounding value

The whole is greater than the sum of the integrations. Here is why, point by point.

7.1 Anchoring IT in business outcomes

Every integration carries a business name, a business owner and a business metric. IT investment gets discussed as processes improved rather than systems connected.

7.2 A shared language between business, IT and AI

One object, read by three audiences. Meetings get shorter because everyone points at the same diagram, and the AI is pointing at it too.

7.3 Change becomes cheaper

Composability made concrete. A change touches only the steps it concerns, is reviewed on the diagram and ships without a project.

7.4 Visibility and governance at the transaction level

Every execution of every process is traceable: which step, which data, which outcome. "Where is order 4471?" becomes a lookup.

7.5 Measurement and evidence-based optimization

Cycle time, failure rate and volume per process are measured as a by-product of execution. Optimization starts from data, with the process owner in charge.

7.6 Reuse and reduced duplication

Shared process components mean the fifth process is faster to build than the first. Duplicate logic across teams gets retired instead of maintained.

7.7 The AI dividend

Every process added makes every AI in the organization more capable, because the catalog it reasons over got richer and the set of actions it can safely take got larger. This dividend is paid on every process, including the ones built before anyone was thinking about AI.

What the curve looks like

The first process solves a problem. The tenth gives you an architecture. The fiftieth gives you an organization an AI can work inside. Each addition makes the earlier ones more useful, because they share a language, a catalog and an execution model.

THE COMPOUNDING CURVE

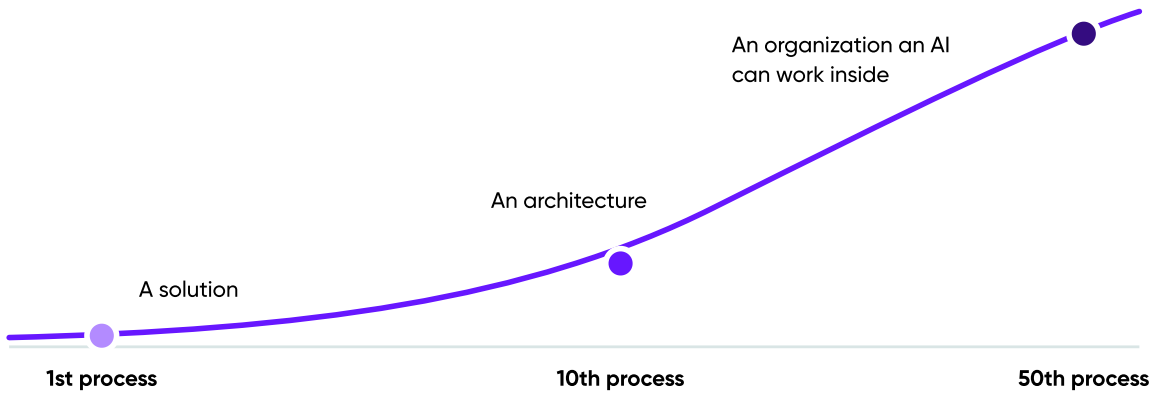


Figure 5. Value per process rises with the size of the catalog. Each process adds a capability, a component and a piece of context.

SECTION 8

The operating model: business and IT at the same table

Four roles, one lifecycle. The method works when the business takes its seat, and this is the section that gives it a reason to.

Process owner

BUSINESS

Owns the outcome, the design and every change. Approves on the diagram. Reads the metrics.

Integration lead

IT

Builds and operates the processes, keeps the component library and the Agents healthy.

Enterprise architect

ARCHITECTURE

Keeps the catalog coherent with the capability map. Decides what becomes a shared component.

AI governance

NEW

Decides which AI may read, invoke or change each process. Reviews what agents did and why.

HOW A PROCESS GETS FROM PAIN TO AI-READY

1
Pain
A business owner names a problem worth fixing.

2
Modeled
The process is drawn together, as it should run.

3
Validated
The owner approves the diagram before any build.

4
Built
The diagram is implemented and deployed as is.

5
Instrumented
Cycle time, volume and failures are visible to the owner.

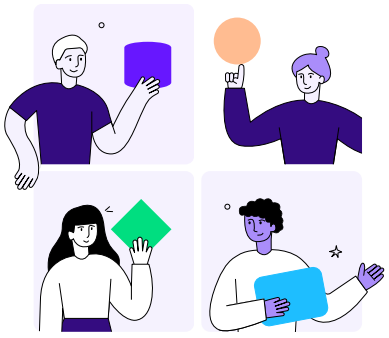
6
AI-ready
Permissions set. The process is context, and a tool.

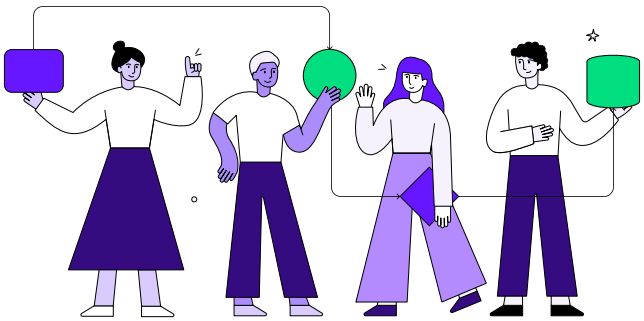
Figure 6. The process lifecycle. Validation happens on the diagram, before the build, with the owner in the room.

Why the business must take the seat

Integration conversations tend to happen between IT and a vendor. That is the pattern this methodology exists to break. A process without a business owner is an integration with a nicer picture. The owner is the person who can say the decision is wrong, who reads the cycle-time number every Monday and who decides whether an AI may act on their behalf.

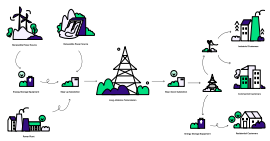
The commitment is modest: one owner per process, an hour to validate the diagram, a look at the metrics each week. The return is a process that keeps matching the business because the business keeps looking at it.





Methodology in practice: three illustrations

Each example is one illustration, not a template. Each shows all three cornerstones: the process, what got composed or swapped and what the AI could then do with it.



Utilities and energy

THE PROCESS

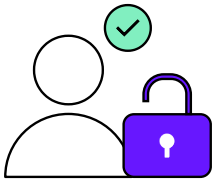
Meter-to-cash for a distribution system operator: reading, validation, settlement, billing, collection. One diagram, one owner in billing.

COMPOSED OR SWAPPED

The legacy metering hub was replaced by the national datahub. The two Tasks that read meter data were re-implemented. Settlement and billing never changed.

WHAT THE AI COULD DO

An agent handles billing disputes by reading the process and the execution trail for that customer, then proposes a correction that a process executes.



Public sector

THE PROCESS

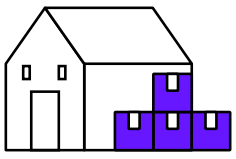
Building permit application: citizen portal, case management, inspection scheduling, decision, notification. Owned by the permits office.

COMPOSED OR SWAPPED

Case management and the GIS stay separate capabilities. Inspection scheduling is a shared component reused from the environmental permit process.

WHAT THE AI COULD DO

A completeness check on each application runs as an AI step. A service agent answers status questions from the execution record, never from memory.



Manufacturing

THE PROCESS

Order-to-delivery across ERP, the plant execution system and the carrier. The sales operations lead owns it and reads lead time weekly.

COMPOSED OR SWAPPED

One plant replaced its execution system. Only the Tasks for that plant changed. Sales and logistics steps, and the other plants, were untouched.

WHAT THE AI COULD DO

An agent triages late orders and invokes the re-route shipment process as a tool, within the limits the process already enforces.

In healthcare, where a centralized EHR holds the clinical record, the same method applies around it: referral, scheduling and supply processes compose the EHR with everything it does not do, and the AI gets its context from those processes rather than from the record itself.

Getting started: the first 90 days

One cross-boundary process. One business owner. Build it, instrument it, then put one AI use case on top of it as proof of the context argument.

DAYS 1 TO 30

Choose and model

Pick one process that crosses at least two systems and one organizational boundary, with a measurable pain. Name the owner. Draw the process together and validate it on the diagram.

DAYS 31 TO 60

Build and instrument

Implement the validated diagram. Deploy it to an Agent in your environment. Put cycle time, volume and failure rate in front of the owner before the process goes live.

DAYS 61 TO 90

One AI use case

Add one AI step to the process, or expose the process to one agent as a tool. Measure whether the AI's answers got better because it could read the process. That is the proof.

WHAT YOU COMMIT

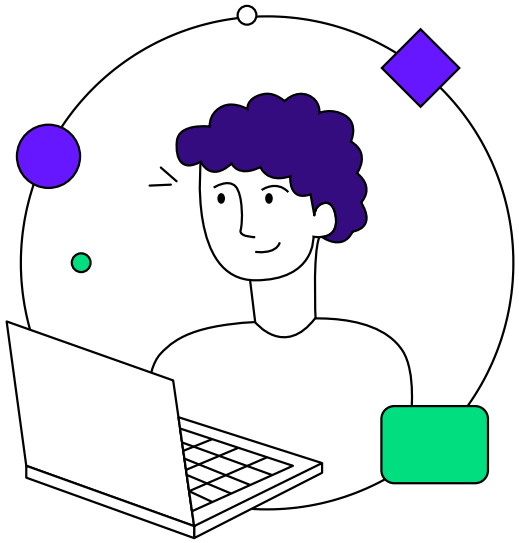
- A named business owner with two hours a week for 90 days
- Access to the systems the process touches, in a test environment
- An architect who can say where the process sits in the capability map
- A decision on which AI may read and invoke the result

WHAT FRENDS AND PARTNERS BRING

- The platform, deployed in your environment or ours
- A lead who has run this cycle before and facilitates the modeling
- Connectors and process components to start from rather than build
- The AI step or MCP exposure for the day-90 use case

Expectations, stated plainly

Visions are visions. At day 90 you have one running process, one owner who reads its metrics, one dashboard and one AI use case that either proved the context argument or didn't. That is a small result and a real one, and it is the only honest basis for deciding whether to build the next ten.



Measuring success

A short scorecard. Six numbers, each readable by the business, each produced by the platform rather than by a survey.

1 Processes under governance

Count of processes in the catalog with a named owner, a validated diagram and a version history. The size of the composable layer.

2 Median time-to-change

Days from a process owner requesting a change to that change running in production. The composability number.

3 Transaction visibility coverage

Share of cross-system transactions whose status can be answered from one execution record instead of several systems.

4 Cycle-time gains

Per process, the change in end-to-end cycle time since it came under governance. Reported by the owner, measured by the platform.

5 Point-to-point connections retired

Direct system-to-system links replaced by a process. Each one retired is a replacement made cheaper.

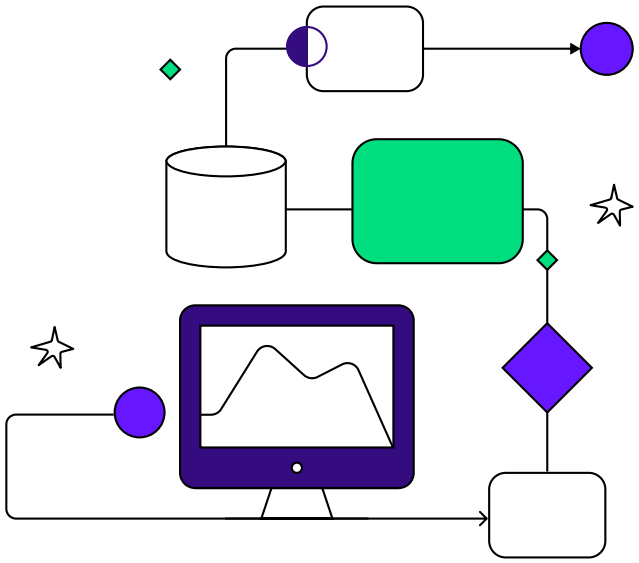
6 Processes exposed as AI-callable capabilities

Count of processes an agent may invoke, with permissions set. The size of what your AI can safely do.

How to read the scorecard

The first number tells you how far the method has spread. The second and fifth tell you whether composability is real. The third and fourth tell you what the business is getting. The sixth tells you what your AI is getting. Review all six quarterly, with the process owners present, and let the ones that are flat set the agenda.

Resist adding more. Six numbers that the business reads beat thirty that nobody does.



Business process. A repeatable sequence of steps, decisions and handoffs that turns a trigger into a business outcome.

Integration. The exchange of data or control between two or more systems so that work can continue across them.

Business-native integration. Integration designed, owned, measured and automated as a business process the business can read and that also executes.

BPMN. Business Process Model and Notation, the standard diagram language for processes, used in Frends as the implementation itself.

Process catalog. The governed inventory of an organization's executable processes, with owners, versions and permissions.

Process owner. The business person accountable for a process's outcome, its design and its changes.

Orchestration vs. choreography. Orchestration has one process directing the steps; choreography has systems reacting to each other's events with no central director.

Composable architecture. An architecture in which business capabilities are independent parts that can be assembled, swapped and recombined.

Packaged business capability. A software component that delivers one well-defined business capability through a clear interface.

Process component. A reusable step or sub-process shared across many processes, with one owner and one version.

Enterprise architecture. The practice of describing how an organization's strategy, processes, applications and data fit together.

Capability map. A structured list of what the organization is able to do, independent of how it does it.

Value stream. The end-to-end sequence of activities that delivers value to a customer or stakeholder.

Event-driven. An integration style in which systems publish events and processes react to them, rather than polling or being called.

API-led. An integration style in which capabilities are exposed as managed, reusable interfaces that processes call.

Deterministic execution. Running a defined process so that the same input and state always produce the same steps and the same outcome.

AI agent. Software that uses an AI model to decide which actions to take toward a goal, using the tools it has been given.

Process-as-tool. Exposing a process to an AI agent as a callable capability with a defined input, output and permission.

Context (in the AI sense). The information a model is given to reason over, including descriptions of how the organization works.

Observability. The ability to see what every execution did, step by step, with enough detail to explain it and fix it.

Approach	Optimizes for	Where it breaks down
Application-centric	Depth inside each system	Cross-system logic has no owner. Change needs a project. No context for AI.
API-led	Reusable interfaces	Interfaces are not processes. The sequence and decisions still live in code the business cannot read.
Low-code and RPA	Speed for small automations	Brittle at scale. Mimics screens rather than modeling work. Weak governance.
Custom code	Full control	Knowledge locked in the codebase and the team. Every change is engineering. Invisible to the business.
AI-first automation platforms	Fast output	Automates what is not understood. Non-deterministic execution of consequential actions. Hard to audit.
Business-native integration	Processes as the unit of design, ownership and AI context	Requires the business to take ownership and keep it. Fewer shortcuts in the first 90 days.

frends

Where data flows, business grows.

Frends is the European iPaaS built for mission-critical automation. Talk to us about the first process, the first owner and the first 90 days.

6,000+
customers

100+
implementation partners

4×
nominated to Gartner Magic Quadrant

G2 Winter 2026, Enterprise
4.6 / 5 G2 · 4.5 / 5 Gartner Peer Insights

DOCUMENT INFORMATION

Author	**Asmo Urpilainen, CTO, Frends**
Version	**1.0**
Published	**October 2026**
Status	**Public**

Version	Date	Change
1.0	October 2026	Initial publication

© 2026 FRENDS TECHNOLOGY OY