BONDY

THINKING · PAPER V1.0 · APRIL 2026

How they think, not what they know

Hiring badly today is easier than ever, and it costs more than ever.

The Bondy method for cognitive evaluation of technical talent in the AI era. We're publishing the complete paper, with its limits and bibliography visible. It is not a brochure.

Mara Schmitman · Founder, Bondy Group

Buenos Aires · April 2026

Prologue

I started doing technical recruiting in 2008, when the Argentine market had just discovered that developers were a scarce resource and that they needed to be treated differently. My training wasn't that of a recruiter: I was a psychologist. I came into the world of tech hiring from a side that seemed tangential and ended up being central. What mattered to me wasn't so much what the candidates could do as how they thought when they faced a problem.

For the first few years, this lens felt like an eccentricity of mine. Clients wanted to know how many years of experience someone had with Java, how many projects they had led, how big a team they had managed. I kept pushing them with questions that struck them as odd: how they had made a particular decision, why they had ruled out a particular alternative, what they had learned when something went wrong. The most common response was a kind of patient condescension: "yes, Mara, but we still need the senior with five years of Java."

That worked for a long time. Not because it was right, but because it worked badly enough that nobody questioned it. Global tech hiring was built on an implicit model: evaluate what the person knew how to do, because doing was the scarce thing. Programming languages, frameworks, certifications, years of experience: all of these were reasonable indicators of a harder question, is this person going to solve real problems at our company?

That model no longer works. And it doesn't work for a specific reason that everyone intuits but few name with clarity: artificial intelligence made cheap, fast, and accessible what used to be the main proxy for technical competence. Writing code is no longer the bottleneck. Reading documentation, generating boilerplate, solving syntax problems, even designing complete functions: a tool that any candidate can use does all of that. What used to distinguish a good engineer from a mediocre one (speed, technical fluency, command of the language) became a commodity.

What remains as a differentiator is exactly what I'd been looking at for 18 years without being able to name it completely: how a person thinks when they have to decide what to build and what not to build, what to rule out and why, what to trust to the machine and what to hold with their own judgment. What used to be an eccentricity of a psychologist is today the only question that matters.

I'm writing this document because that shift forces me to make explicit what operated for years as intuition. Bondy grew on the basis of a way of evaluating candidates that I could transmit to the team by osmosis: interviews together, case discussions, years of informal calibration. That is no longer enough. First, because the team grew and I can no longer train each recruiter by direct contact. Second, because clients are starting to ask questions the market doesn't know how to answer with

authority: how do you evaluate a developer in the era of ChatGPT? What do you look at that others don't? Why should we trust your judgment when everyone says they have a method?

Third, and most important: because the problem became urgent. Clients are hiring badly at a speed that wasn't possible before. A bad hire today is detected later, because the candidate's initial output looks good. AI helps them produce. When the problems appear, six months later, the damage is done. Technical debt, bad architectural decisions, misaligned teams, processes that nobody can sustain. And the market keeps evaluating as if it were 2018.

This document is the attempt to articulate, with the greatest possible rigor, the method Bondy has been operating on, which I want to leave written for three different audiences. The first is the Bondy team, because they need a common tool that takes them out of the dependency on personal judgment. The second is our clients, because they deserve to understand what we do when we say we evaluate differently. The third is the recruiting and HR tech community, because I believe what we learned has value beyond Bondy, and I'd rather make it public than keep it as a competitive advantage.

An honest warning before starting: this method isn't a recipe. It's a cognitive evaluation framework built on 18 years of practice, influenced by Will Larson's work on engineering role archetypes, anchored in the organizational psychology tradition on modes of thinking, and tested in hundreds of selection processes at tech companies in LATAM and the rest of the world. It has limits. There are things it doesn't solve. There are situations where another method works better. I try to be explicit about all of that throughout the document, because I believe serious methods are the ones that admit their edges.

If you're reading this as a current or potential Bondy client: what follows is the basis for how we evaluate the candidates we present to you. If you're reading this as a colleague in the industry: I hope it's useful, and I'm interested in your critique. If you're reading this as a member of the Bondy team: this is the source. Use it, question it, adjust it with the real cases you're accumulating.

The tech hiring market is breaking in real time. This is not the moment for lukewarm methods.

Mara Schmitman

Buenos Aires, April 2026

1. The problem

There are two ways to understand what's happening in tech hiring today. The first is to read the numbers. The second is to listen to CTOs.

The numbers are stark. In 2025, according to the Stack Overflow Developer Survey, 84% of professional developers reported using or planning to use artificial intelligence tools in their daily work. Nearly half of the new code written at companies with high tech adoption is generated by AI assistants. GitHub reported 43 million monthly pull requests in 2025, 23% more than the previous year. Code production speed accelerated drastically in less than three years.

But there's another number, more uncomfortable, that's rarely cited in the same conversation: the aggregate productivity of engineering teams did not grow in the same proportion. The 2025 DORA report, published by Google Cloud, says it with a sentence that deserves to become famous: artificial intelligence acts as a mirror and a multiplier. In cohesive organizations, with good practices and clear processes, it amplifies strengths. In fragmented organizations, it amplifies weaknesses. AI doesn't fix broken teams. It accelerates them toward where they were already going.

That explains why CTOs are having conversations they weren't having three years ago. They say their teams produce more code than ever and deliver less value than ever. They say early technical decisions become impossible to reverse faster. They say the juniors they hire seem competent in the interview and fall apart at six months when they have to sustain something in production. They say the seniors they hire are indistinguishable on paper and radically different in real impact.

The broken proxy paradox

For decades, technical hiring operated on an implicit consensus: if a person can demonstrate they know how to do something, they probably know how to think about it. Knowing how to write Python code was a reasonable indicator that the person understood programming. Knowing how to solve an algorithm on a whiteboard was a reasonable signal of problem-solving capacity. Having five years of experience at a large company was a reasonable shortcut for assuming that person had seen enough complex situations to develop judgment.

These indicators were never perfect, but they worked well enough. And what gave them sense was a premise almost nobody questioned: doing is the scarce thing. If the market had few developers, and producing code took a lot of time, then production speed was a good measure of the quality of the human resource.

Artificial intelligence broke that premise. Today, producing code is cheap. What's scarce is deciding what code to produce, what to rule out, when to trust the machine's output and when to doubt it,

how to integrate what AI generates into systems that have to sustain themselves over time. The indicators we built on the scarcity of production stopped measuring what matters.

This creates a situation we could call the broken proxy paradox: the most widespread evaluation tools in the industry (traditional technical interviews, algorithm tests, portfolio reviews, individual productivity metrics) keep measuring things that no longer differentiate the good from the mediocre. And meanwhile, the criteria that do differentiate (decision capacity, judgment under ambiguity, capacity to read and improve someone else's code, translation between technical and business) are evaluated in an implicit, intuitive, untrainable way, dependent on the interviewer's instinct.

The economic consequence

What makes this urgent is not just the technical change. It's the economic consequence. A bad tech hire was always expensive, but today it's more expensive for three converging reasons.

The first is that bad hires are detected later. Before, a person who couldn't solve problems became obvious in the first few weeks: they couldn't write code, they couldn't keep up with the team's rhythm, they couldn't deliver their tasks. Today, AI acts as a prosthetic: someone who can't think can still produce decent output for months, until a situation appears that requires real judgment and everything falls apart.

The second is that early technical decisions are harder to reverse. The speed at which software gets built today means teams make significant architectural decisions in weeks, not months. If those decisions are bad, the technical debt they generate is enormous and spreads through the entire system before anyone can stop it.

The third is that the opportunity cost went up. Good engineers are more productive than ever when they work with AI. Their capabilities multiply. That means the gap between a good hire and a mediocre one widened. The first one isn't 1.5× better. They're 5× or 10× better. Paying the salary of a mediocre engineer when you could be paying that of a good one is losing that multiplied difference.

The commercial translation of all this is brutal: **hiring badly today is easier than ever, and it costs more than ever**. Traditional hiring methods are optimized for a world that no longer exists, and the cost of continuing to use them is no longer abstract. It's a measurable cost in technical debt, lost productivity, and time from technical leaders that goes into fixing what was hired badly.

Why the market hasn't reacted yet

If all of this is true, why are most companies and recruiting firms still evaluating like it's 2018? Three reasons are worth naming.

The first is institutional inertia. Large companies have hiring processes that took years to build, with legal compliance, ATS systems, review committees, and established metrics. Changing those processes requires political energy and internal consensus that few organizations are willing to mobilize as long as current processes still produce acceptable results, even if those results are increasingly worse in relative terms.

The second is the absence of structured alternatives. There's a lot of discourse about the need to evaluate judgment, criterion, decision-making. But there are no widely adopted frameworks that operationalize those concepts in concrete questions, evaluable rubrics, and repeatable protocols. Hiring teams know they have to change something, but they don't know exactly what, and nobody is giving them a ready-to-use tool.

The third is the measurement problem. The things that do matter today (how a person thinks, how they decide under ambiguity, how they read and improve someone else's code) are harder to measure than the things that mattered before. It's easier to evaluate whether someone knows how to write a binary search algorithm than to evaluate whether they have the judgment to rule out three architectural proposals. And markets, when they face a change where what's important is hard to measure, tend to keep measuring what's easy even after it stopped being relevant.

Bondy is writing this document from the conviction that the third reason is the only one that can be attacked with serious intellectual work. Institutional inertia will be broken by clients who get tired of hiring badly. The absence of alternatives gets broken by writing alternatives. That's what follows.

2. Why existing frameworks fall short

Before proposing a new method, we have to be fair to the methods that already exist. We don't dismiss them because they're bad. We describe them to show what they solve well and where they fall short for the specific problem tech hiring faces today.

The skills and certifications model

The most widespread way of evaluating technical talent is still the skills list: programming languages, frameworks, tools, certifications. A company posts a job with a list of technical requirements, recruiters filter by match, interviewers verify that the person actually knows what they say they know. This model has the advantage of being easily operationalizable (any ATS can filter by keywords) and producing metrics that compare across candidates.

Its limit is structural: it measures accumulation of technical knowledge, not capacity to apply it under real pressure. A person can have every AWS certification and not know when to use a serverless architecture versus a container-based one. Another might have no certifications and have made that decision correctly dozens of times in production. The skills model doesn't distinguish between these two cases.

Traditional technical interviews (FAANG style)

The generation of technical interviews popularized by Google, Facebook, Amazon, Netflix and other big tech companies is built on two pillars: whiteboard algorithms and systems design. Algorithms evaluate the capacity to solve structured problems under pressure. Systems design evaluates the capacity to think architecturally.

This model has real merits. It's repeatable, calibrable between evaluators, and puts all candidates in the same situation. But it has three problems that became critical in the AI era. First, whiteboard algorithm problems are exactly the kind of task AI tools solve best. Training candidates to solve them without help evaluates a skill that's increasingly less relevant. Second, traditional systems design assumes hypothetical scales (millions of users, suboptimal latencies) that don't reflect the real problems of most companies. Third, the pressure of a whiteboard interview rewards those who are prepared to interview, not necessarily those who are good engineers.

Will Larson's archetypes

The most serious work that exists today on how to describe the work of senior engineers is Will Larson's, in his book *Staff Engineer*. Larson identified four archetypes to describe how Staff+

engineers operate: the Tech Lead, who guides a team's execution; the Architect, responsible for the technical design of a critical area; the Solver, who dives into complex problems without a clear path; and the Right Hand, who extends an executive's attention in large organizations.

Larson's taxonomy is valuable for two reasons. First, it's built from empirical observation. Larson interviewed dozens of Staff+ engineers at different companies and grouped real patterns, not theoretical categories. Second, it gives the recruiting world a common vocabulary to talk about something that used to be ineffable: what the different forms of being senior in engineering actually look like.

But Larson's taxonomy has a specific limit for our purpose: it's designed to describe how someone operates in a role, not to evaluate them in a selection process. Larson tells us what an Architect does when they're already an Architect. He doesn't tell us how to detect in a 45-minute interview whether a person has the cognitive capacity to be one. His framework is descriptive, not evaluative. And that distinction matters: a recruiter can't use Larson directly in an interview. What they need is a framework that takes Larson's observation as input and turns it into something measurable.

The "new AI archetypes"

In the last two years, several attempts have appeared to propose new archetypes specifically for the AI era. Some talk about the Vibe Coding Validator, the Prompt Expert, the Domain Expert. Others propose taxonomies of Connector, Innovator, Systems Thinker, Translator. These attempts are legitimate reactions to the change the industry is going through, and some of them identify real dimensions of the problem.

But most of these frameworks share two weaknesses. The first is that they're superficial: they describe visible styles of working with AI, not deep cognitive modes. The second is that they're not built to be used in evaluation: they're taxonomies to classify candidates after the fact, not tools to interview them with. They work for LinkedIn conversation, not for serious hiring processes.

The gap that justifies this method

What's missing in the market is a framework that combines four things that are not integrated in any available tool today:

- That it be evaluative, not just descriptive: usable in an interview to diagnose a candidate, not only to label them afterwards.
- That it be anchored in how the person thinks, not in what they know how to do: because the latter stopped being a good indicator.
- That it integrate the AI dimension transversally, not as a separate mode: because the use of AI

runs through every cognitive mode.

- That it produce diagnoses that are comparable between evaluators: with rubrics and protocols, not just intuition.

That's what the Bondy method tries to build. It doesn't replace Larson: it uses him as conceptual input on what senior engineering roles look like. It doesn't replace the skills model: it complements it, because you still need to know whether a person knows the relevant tools. What it does is add a layer that doesn't exist anywhere today: a layer of structured cognitive evaluation, usable by real recruiters in real interviews, calibrable between evaluators, and designed for the specific historical moment we're in.

3. The conceptual turn: from what they do to how they think

The Bondy method rests on a conceptual turn that's simple to state and hard to operationalize: instead of evaluating what a person knows how to do, we evaluate how they think when they face technical problems. This chapter explains why that turn matters, what it means exactly, and what its practical implications are.

What "how they think" means

When we say how they think, we don't mean personality, learning styles, or Myers-Briggs profiles. We mean something more specific: the dominant mode through which a person relates to open technical problems. There are people who, when they face a problem, first decompose it into alternatives and evaluate trade-offs. There are people who first look for what existing pieces they could combine. There are people who first read the context before acting. There are people who first identify what could go wrong.

These are not personalities. These are cognitive modes: structured ways of approaching technical reality. The same person can activate different modes depending on the situation, but everyone has a dominant mode: the first one they reach for by default, the one they apply with the least effort, the one they use when they're tired. And that dominant mode has enormous consequences in the kind of work they're going to do well and the kind of team they're going to be a good complement to, or a bad fit.

Why cognitive modes are evaluable

A reasonable objection to this framing is that cognitive modes sound like something hard to measure, almost like intuition disguised as method. That objection deserves a concrete answer.

Cognitive modes are not directly visible, but they're inferable from how a person talks about their work. When you ask someone to tell you about an important technical decision they made, you're not just listening to what they decided: you're listening to the structure of the reasoning they used to decide. If the person tells you about three alternatives they ruled out and why, they're showing that their dominant mode includes explicit structuring of decisions. If the person tells you only the chosen solution and justifies it with "it seemed better to me," they're showing that their dominant mode does not include that structuring.

This is exactly what organizational psychology has done for decades with concepts like locus of control, achievement orientation, or decision-making styles. You don't measure it by observation; you measure it through speech. And speech, when listened to with a clear framework of what to look for, is surprisingly revealing.

The difference from evaluating personality or soft skills

It's important to mark what differentiates this approach from the personality tests and soft skills evaluations that already exist in the hiring market. Personality tests measure general traits (extroversion, openness to experience, conscientiousness) that have little specific relation to how a person approaches technical problems. Soft skills evaluations measure communicational and relational capacities (teamwork, communication, leadership) that are important but generic.

The cognitive modes of the Bondy method are specifically technical: they describe how a person thinks when they face a software engineering problem, not how they behave in general. A person can be extroverted and have a cognitive mode of Technical Editor (preferring to improve what exists over building what's new). Another can be introverted and have a cognitive mode of Translator (excellent at converting business problems into technical specs). Personality and cognitive modes are distinct and relatively independent dimensions.

Practical implications of the turn

The turn from "what they know" to "how they think" has four practical implications for tech hiring.

First: interview questions change. Instead of "tell me what technologies you know," you ask "tell me about an important technical decision you made last year." The first measures knowledge accumulation. The second measures dominant cognitive mode. Both are valid, but the second differentiates the good from the mediocre much better in the current era.

Second: calibration between evaluators becomes possible. If what you're evaluating is cognitive mode and you have a clear framework of what signals correspond to what mode, then two evaluators can reach the same diagnosis independently. That's what differentiates a method from a personal intuition.

Third: hiring criteria become situational. Instead of looking for "the best candidate" in the abstract, you look for the cognitive mode the team needs at its current moment. A team that already has three Decision Architects doesn't need a fourth. It needs a Technical Editor to clean up the debt those architects generated. That decision is invisible if you only evaluate skills.

Fourth: the candidate's professional evolution becomes a relevant data point. A person can be starting to develop a new cognitive mode, or consolidating one they already have. That matters for hires where the client isn't looking only to cover a position, but to add someone who will grow with the team.

These four implications turn tech hiring from a filtering problem into a diagnosis problem. Filtering is deciding whether someone meets requirements. Diagnosing is understanding how that person operates and where they fit best. The Bondy method is, in essence, a cognitive diagnostic protocol applied to hiring.

4. The Bondy modes

The Bondy method organizes cognitive evaluation around six modes. Each mode describes a dominant way of relating to technical problems. They're not personality types or job roles: they're structured ways of thinking that manifest when a person faces a decision, a problem without an obvious solution, or a situation of technical ambiguity.

Four of the six modes are core: they're evaluated in every candidate, in every process. The other two are optional: they're activated when the specific role requires it. This distinction is not hierarchical. The optional ones aren't "less important." It's operational: it reflects the reality that in a 45-minute interview you can't evaluate everything with the same depth, and you have to prioritize what differentiates most.

Before introducing each mode, an important clarification. Nobody is a single mode. Every candidate is going to show signals of several. What the method seeks to identify is the dominant mode (the strongest one) and the range (how many other modes the person can activate when the situation calls for it). We develop that distinction in chapter 5.

Mode 1: Decision Architect (core)

The Decision Architect doesn't design systems in the abstract. They design the decision process that precedes the system. When they face a problem, their first move is to identify possible alternatives, evaluate explicit trade-offs, and argue what they're ruling out and why. Their value isn't in choosing the right solution. It's in structuring the reasoning that makes the choice defensible.

The conceptual roots of this mode are in the literature on decision-making under uncertainty, particularly Herbert Simon's work on bounded rationality and the distinction between programmed and non-programmed decisions. In the current tech hiring context, this mode is critical because most relevant technical decisions are non-programmed: they don't have a single correct answer, and what differentiates a good engineer is the quality of the decision process, not the final choice.

A person with a dominant Decision Architect mode usually talks about their projects in terms of "we chose X accepting that we'd lose Y." They distinguish intuitively between reversible and irreversible decisions, and treat the latter more carefully. They consider who is going to maintain what they build. And, crucially, they can articulate what they ruled out even when not asked directly.

Mode 2: Connector (core)

The Connector solves problems by combining pieces that already exist. Their value isn't in building from scratch. It's in knowing what to build and what to borrow. They know the broader technical

ecosystem (APIs, services, libraries, platforms) and their criterion for choosing tools is based on fit with the problem, not familiarity or popularity.

This mode is probably the one most transformed by the arrival of AI tools. Before, connecting existing pieces was seen as "less creative" than building from scratch. Today, in a world where almost everything you need already exists in some form, the Connector became one of the most valuable cognitive modes in the market. Companies that grow fast do so because they have Connectors who put solutions together in weeks with existing pieces, not because they have heroic engineers building everything by hand.

A person with a dominant Connector mode usually describes their solutions in terms of "I connected X with Y using Z." They speak with familiarity about API contracts, failure modes, external dependencies. They have informed opinions about the ecosystem, not just the tools they used. And, crucially, they can explain why they didn't build something from scratch.

Mode 3: Technical Editor (core)

The Technical Editor improves what already exists. Their value is in reading other people's code, identifying relevant technical debt, and executing improvements that reduce complexity without changing behavior. In the AI era, where enormous volumes of code are generated without the discipline a trained engineer would have writing it by hand, this mode became critical.

The common intuition is that a good engineer is someone who writes quality code. But in a world where AI produces a large part of the code, the bottleneck moves. The scarce thing is no longer writing code: it's reading it critically, identifying which parts are problematic, and improving them without breaking what works. A Technical Editor is a person who has the discipline and the judgment to do that work, which is probably the least glamorous and most critical work of the moment.

This mode has conceptual roots in the literature on refactoring (particularly Martin Fowler's work) and in the practice of code review as a quality mechanism. But the Bondy method treats it as a cognitive mode, not as a technical skill: what it evaluates is not whether someone knows how to refactor, but whether their natural way of relating to existing code is one of deliberate improvement or avoidance.

A person with a dominant Technical Editor mode usually talks with enthusiasm about refactors they did. They distinguish between types of technical debt (urgent, tolerable, structural). They have opinions on readability, naming, file organization. And, crucially, they don't propose rewriting from scratch as the default answer to inherited code.

Mode 4: Translator (core)

The Translator converts between worlds. They take a business problem and bring it down to an actionable technical specification. They take a technical limitation and explain it in terms of business

impact without losing rigor. Their value isn't in knowing more on one side or the other. It's in operating as a bridge between two languages that rarely meet.

The conceptual roots of this mode are in the literature on boundary spanners in organizational psychology, particularly Michael Tushman's work in the seventies on individuals who connect groups with different communicational codes. Tushman showed that in complex organizations, boundary spanners are disproportionately important for the overall performance, although they're hard to identify with traditional evaluation methods.

In current tech hiring, the Translator became critical for a specific reason: engineering teams are increasingly exposed to conversations with product, business, and end clients. Tech companies can no longer afford engineers who only speak with other engineers. And at the same time, AI made it easier than ever to generate code without understanding why. That makes the capacity to articulate the business reason behind a technical decision one of the strongest differentiators between a good engineer and a replaceable one.

A person with a dominant Translator mode usually explains technical things to non-technical people without being condescending. They identify when a "technical" problem is actually a communication problem. They adapt their level of detail to the interlocutor. And, crucially, they don't look down on non-technical stakeholders.

Optional mode 1: Operator

The Operator executes well what they're asked to do. They have discipline, attention to detail, capacity to follow complex instructions without losing context. It's the most common dominant mode in competent juniors and, at senior levels, it's assumed as the base on which the other modes rest.

That's why it's optional: we only evaluate it explicitly when the role is junior, when the candidate is going to work with close supervision, or when the client specifically needs execution capacity. In senior+ roles, if someone can't execute, that appears naturally in the other modes. A Decision Architect who can't translate their decisions into execution isn't really an Architect. Operator as an independent mode only adds value when it's the central dimension of the role.

Optional mode 2: Guardian

The Guardian obsesses over what can go wrong. They think about security, edge cases, observability, resilience. They don't slow the team down: they make sure that what the team builds doesn't explode in production. Their cognitive mode is defensive in the best sense of the term: they don't reject change, they prepare it to survive contact with reality.

This mode is optional because its criticality depends on context. In a startup in exploration phase, where moving fast matters more than preventing, a pure Guardian can be a brake. In a fintech, a

healthcare company, or any system with critical production impact, a Guardian is indispensable. We activate this mode when the role requires it and leave it out when it doesn't.

The conceptual roots of the Guardian are in the literature on safety engineering and high reliability organizations. In current tech hiring, this mode became critical for a specific reason: the speed at which software is built today makes errors in production more frequent and more costly. And AI, by producing code without deep human context, introduces new categories of failures that only a trained Guardian can anticipate.

5. Dominant mode and range

The Bondy method is built on a claim that deserves to be defended explicitly: nobody is a single mode. Every person shows traits of several cognitive modes, and what differentiates a strong candidate from a weak one is not only what their dominant mode is, but how many other modes they can activate when the situation requires it.

This distinction between dominant mode and range is central because it solves two problems a simpler method would have.

Why identifying the dominant mode is not enough

A method that only identified the dominant mode of each candidate would produce diagnoses that are too rigid. It would say things like "this candidate is a Decision Architect" and leave the client with the impression that the person can do nothing else. But the reality is that a good Decision Architect also needs to be a Connector and a Translator, not as dominant modes but as capacities they can activate when the context calls for it.

The range metric captures exactly that: how many additional modes the person can activate when the dominant one isn't enough. A candidate with a dominant Decision Architect mode and strong range toward Translator and Technical Editor is radically different from a candidate with the same dominant mode but poor range. The first is a potential technical leader. The second is a specialist who needs a team that compensates for what they don't do.

How range is measured

Range is measured through the numerical scoring (1–5) that the method applies to each core mode. A candidate with three modes at score 4 or higher has wide range. A candidate with only one mode at 4+ is a specialized profile. A candidate without any mode at 4+ is in trouble, regardless of which mode is their dominant one.

This measurement has two practical virtues. First, it makes explicit something experienced recruiters intuit but rarely articulate: a "well-rounded" candidate is not vague: they're a candidate with wide range across specific and observable modes. Second, it lets the client understand exactly what they're buying: not "a senior developer," but "a Decision Architect with range toward Translator and Editor, who's going to be strong in X kind of problem and weak in Y kind of problem."

Why some candidates change their dominant mode

An observation the method incorporates from the organizational psychology literature is that a person's dominant mode can evolve through their career, but not quickly or voluntarily. Cognitive modes form through years of repeated experience in similar situations, and they consolidate when that experience is validated by positive results.

This means you can't "train" someone to change their dominant mode in six months. What you can do is expand the range: help a person develop secondary cognitive modes that complement the dominant one. That's valuable information for clients who are thinking about medium-term hires: a candidate with a consolidated dominant mode but emerging range can be an intelligent bet if the client is willing to invest in their development.

The pathologies of each dominant mode

There's a point worth making explicit that distinguishes the Bondy method from more optimistic frameworks: each dominant mode has an associated pathology when it operates without range or counterweights. The method does not celebrate all modes equally. It recognizes that each one, taken to the extreme, becomes dysfunctional.

The pure Decision Architect becomes the engineer who paralyzes the team with endless analysis and never gets to execution. The pure Connector becomes the engineer who puts together fragile solutions because they don't deeply understand any of the pieces they're connecting. The pure Technical Editor becomes the obsessive engineer who blocks features because there's always something more to improve first. The pure Translator becomes the technical politician without substance, someone who communicates well but doesn't produce.

These pathologies are not reasons to dismiss a candidate: they're signals to understand their range. A candidate with a dominant Decision Architect mode and wide range isn't going to fall into the pathology, because they can activate other modes when the situation requires it. A candidate with the same dominant mode and narrow range will fall into the pathology sooner or later. Identifying that difference is exactly what the method seeks to do.

6. Application in hiring

Up to this point, this document is a conceptual framework. This chapter describes how that framework translates into a concrete hiring process, from the first contact with the client to the presentation of the selected candidate. The detailed operationalization (interview questions, rubrics, templates) lives in the Implementation Manual, which is a separate document. Here we present the logic of the process, not its mechanics.

The client kick-off

Every serious hiring process starts by understanding what the client is buying, and most processes fail at this first step. The client typically describes what they need in terms of seniority and skills ("I'm looking for a senior backend with Python and AWS experience"), but rarely articulates what kind of cognitive mode the team needs at its current moment. That translation is the recruiter's responsibility, and the Bondy method turns it into a structured conversation.

The ideal kick-off covers three questions. First: what kind of problems is this person going to have to solve in their first six months? The answer to that question reveals which modes are critical. If the problems are architectural, the client needs a Decision Architect. If the problems are about integration, they need a Connector. If the problems are about technical debt, they need a Technical Editor. Second: what team are they going to work with, and which cognitive modes are already covered? That avoids hiring redundancy and allows you to look for complement. Third: what's the production criticality level of this person's work? That decides whether to activate the Guardian mode.

Pre-screening

Traditional pre-screening filters by CV keywords. Bondy method pre-screening filters by initial evidence of cognitive mode in the candidate's LinkedIn, GitHub, or portfolio. This doesn't produce a diagnosis (it's too early for that), but it produces early hypotheses that orient the deep interview.

For example, a candidate whose LinkedIn is full of project descriptions where they mention "I led the decision to migrate from X to Y considering trade-offs A, B, C" probably has Decision Architect traits, and the interview can start by verifying that hypothesis. A candidate whose GitHub shows many forks and contributions to open source projects probably has Connector traits. These hypotheses are not conclusions: they're hypotheses to be tested in the interview.

The deep interview

The Bondy method interview lasts 45 to 60 minutes and is structured to evaluate the four core modes (plus the optional ones if they apply) through a reduced set of multipurpose questions. Each question illuminates two or three modes simultaneously, and the recruiter doesn't focus on asking "one question per mode," but on listening for which modes activate in each response.

The conversation matters, but the discipline is what to listen for. A recruiter trained in the method knows that when a candidate tells a technical decision, they have to listen for three things: whether they mention alternatives they ruled out (Architect), whether they mention existing tools they evaluated (Connector), and whether they connect the decision to business impact (Translator). That simultaneous listening of multiple dimensions is what makes the 45-minute interview efficient.

Post-interview scoring

Immediately after the interview, the recruiter completes a numerical scoring (1–5) for each core mode and a qualitative evaluation (green/yellow/red) for each optional mode. This scoring is not done during the interview. It's done cold, with the raw notes at hand, in the following 30 minutes while the conversation is still fresh.

The scoring rules are explicit to avoid grade inflation: a 5 is never assigned without evidence that the candidate is a reference for others in that mode; a 1 is never assigned without active evidence of red signals. A candidate who didn't show evidence of a mode receives a 3 (neutral) noted as "not evaluable with available data." This discipline is crucial for scorings to be comparable between candidates and between evaluators.

Presentation to the client

The presentation to the client is where the method becomes visible. Instead of presenting the candidate as "senior with 8 years of experience," we present them as "Decision Architect with range toward Translator and Technical Editor." That difference in language is not aesthetic: it's what lets the client make an informed decision about whether the candidate fits in their specific context.

The presentation includes at least one direct quote from the candidate. Direct quotes are the empirical anchor of the diagnosis: they show the client that the cognitive mode we're describing isn't our opinion, but an inference anchored in what the candidate actually said. Without a direct quote, the diagnosis is debatable. With it, it's defensible.

Calibration between evaluators

No cognitive evaluation method survives without a protocol for calibration between evaluators. If two recruiters interview the same candidate and reach different diagnoses, the method loses credibility and dissolves into individual intuitions. That's why the Bondy method includes a monthly

calibration protocol: a recorded interview that all recruiters score independently, and the differences are discussed until reaching a common interpretation or explicitly documenting the points of ambiguity.

This practice is what keeps the methodology alive. Without periodic calibration, each recruiter develops their own interpretation of the modes and, in six months, the method stops being a shared method and goes back to being a set of individual criteria. With calibration, the method sharpens over time and is taught to new recruiters without depending exclusively on one-on-one mentoring.

7. Honest limitations

This chapter is what distinguishes a serious method from a consultancy white paper. Every method has limits. There are situations where it doesn't work well, types of candidates it doesn't capture, problems it doesn't solve. What follows is an explicit inventory of those limitations.

It doesn't replace the technical evaluation

The Bondy method evaluates how a candidate thinks when they face technical problems. It doesn't evaluate whether they know the specific technologies the role requires. That evaluation is still necessary and must be done separately, ideally by someone on the client team who has the technical knowledge to verify whether the candidate can use the tools they're going to need.

The reason is simple: the Bondy method is a cognitive diagnostic protocol, not a technical test. Confusing it with the latter would be a serious mistake. A candidate can be a brilliant Decision Architect and not know how to use Kubernetes. That makes them inadequate for a role that requires Kubernetes, regardless of their cognitive mode. The two evaluations are complementary, not substitutable.

It works better at mid and senior levels than at junior levels

The method is based on listening to how a candidate talks about their own professional experience. At junior levels, where experience is limited, the available material to evaluate is scarcer. This doesn't make the method useless for juniors (the optional Operator mode was designed precisely for those cases), but it does mean the cognitive diagnosis is less precise when the candidate has less professional history to discuss.

In practice, this means that for juniors the method produces early hypotheses rather than firm diagnoses. And the hire decision for juniors should weigh other factors more (learning capacity, motivation, cultural fit) than the cognitive diagnosis alone.

It depends on the quality of the recruiter

Although the method seeks to produce diagnoses comparable between evaluators, it doesn't eliminate the importance of the recruiter's individual judgment. A recruiter without experience in tech hiring, or without training in active listening, is going to have difficulties applying the method with precision even with all the rubrics at hand.

This means the method has a competence floor: it requires recruiters with enough technical experience to understand what a candidate is saying about their projects, and with trained active listening skills. It's not a plug-and-play tool for hiring teams without experience. It's a professionalization tool for teams that already have experience and want to systematize it.

It doesn't predict cultural fit

A person's cognitive mode tells you how they think, not how they're going to behave in a specific culture. A Decision Architect can thrive in a culture that values structured technical debate and fail in a culture that rewards speed and intuitive action. That dimension, the fit between the person and the team's culture, is orthogonal to the cognitive diagnosis and must be evaluated separately.

Bondy handles that dimension through its Dynamic Compatibility work, which is a complementary component of the hiring process but isn't part of the Bondy method in the strict sense. The cognitive method and the cultural evaluation are two distinct lenses on the same candidate, and both are necessary for an informed hire decision.

It doesn't predict long-term evolution

The method produces a diagnosis of how the candidate thinks today. It doesn't predict how they'll evolve in the next 18 months or 5 years. There are candidates who have cognitive modes still in development and can grow significantly with the right context, and there are candidates who have consolidated cognitive modes that aren't going to change much. Distinguishing between these cases requires longitudinal data the current method doesn't have.

This limitation matters for clients who are thinking about long-term hires or building a talent pipeline. The method can tell them what they have today, but it can't promise what they'll have in three years. It's a real limitation that I prefer to make explicit rather than leave implicit.

It doesn't work the same across all cultures

The method was developed and tested primarily at LATAM tech companies with global clients. There are some signals that certain dimensions work differently in cultures with communicational codes very different from those of the region. For example, the Translator mode can express itself in more subtle ways in Asian cultures where direct communication is less valued, and the method may underestimate candidates with those modes.

This limitation is something the method will have to address as it gets applied in more cultural contexts. For now, we mark it as a known edge and an invitation to colleagues from other regions to test it and report results.

8. Closing and projection

This document articulates version 1.0 of the Bondy method. That numbering is not decorative: it implies an explicit expectation that this method is going to evolve. The next versions will incorporate what we learn over the next 12 to 18 months of operational use, and will adjust what reality shows to be poorly calibrated.

There are three concrete lines of work the method will develop in its next phase. The first is the incorporation of resolved cases: real examples (anonymized) of candidates evaluated with the method, with scoring, diagnosis, and outcome at the client six months later. These cases are the most important component for teaching the method to new recruiters, and can only be built with prolonged operational use.

The second is the calibration of the longitudinal dimension: identifying which early signals predict cognitive evolution over the following 18 months. This requires tracking candidates hired through the method, and a feedback protocol with clients who report how the hire performed after the process. It's slow work but indispensable if we want to move from "current diagnosis" to "reasonable prediction."

The third is opening the method to the community's critique. This document is going to be published in some form: probably as a white paper on the Bondy site, possibly as a short book. The intention isn't commercial: it's to invite other tech recruiting professionals, CTOs, and HR researchers to use the method and report what works and what doesn't. Bondy benefits more from having a method externally validated than from having a kept commercial secret.

The underlying conviction, the one that sustains this entire document, is that tech hiring is going through a historical moment that requires new tools. The methods that worked between 2010 and 2020 stopped working, and the ones that will replace them are not yet consolidated. This is exactly the moment when it's worth writing, publishing, debating, and refining. Things become obvious after someone articulates them with enough clarity for the rest to see them. This document attempts that articulation.

I don't expect the Bondy method to be adopted as-is by the entire industry. I expect it to open a conversation about how to evaluate technical talent when what's scarce is no longer knowing how to do, but knowing how to think. If someone reads this document, critiques it, improves it, and builds something different that works better, that's already a success. The whole industry needs this kind of work, and I'd rather be part of the problem of having too many competing methods than part of the problem of having none.

Bibliography

Primary sources

Larson, W. (2021). *Staff Engineer: Leadership Beyond the Management Track*. Stripe Press. Archetypes chapter available at: https://staffeng.com/guides/staff-archetypes/

Stack Overflow. (2025). *Developer Survey 2025*. https://survey.stackoverflow.co/2025/

Google Cloud. (2025). *DORA State of DevOps Report 2025*. Concept of AI as "mirror and multiplier" in cohesive versus fragmented organizations.

Context of the change in technical hiring

Bruneaux, T. (2026). *Hiring for AI-native developers in 2026*. DX. https://getdx.com/blog/hiring-developers/

Built In. (2025). *How to Reshape the Developer Hiring Process for the AI Era*. https://builtin.com/articles/developer-hiring-process-ai-era

Ewerlöf, A. (2025). *Staff archetypes can be anti-patterns*. https://blog.alexewerlof.com/p/staff-archetypes-are-anti-patterns

Theoretical foundation (organizational psychology and decision-making)

Simon, H. A. (1947). *Administrative Behavior: A Study of Decision-Making Processes in Administrative Organization*. Macmillan. Concept of bounded rationality and the distinction between programmed and non-programmed decisions, conceptual base of the Decision Architect mode.

Tushman, M. L. (1977). Special boundary roles in the innovation process. *Administrative Science Quarterly*, 22(4), 587–605. Concept of boundary spanners, conceptual base of the Translator mode.

Morin, E. (1990). *Introduction à la pensée complexe*. ESF éditeur. Complex thought and the logic of dominant mode with activatable range.

Fowler, M. (1999). *Refactoring: Improving the Design of Existing Code*. Addison-Wesley. The practice of refactoring as a technical discipline, conceptual base of the Technical Editor mode.

Note on original development

The six Bondy modes (Decision Architect, Connector, Technical Editor, Translator, Operator, Guardian) are original development of Bondy Group, built on 18 years of practice in technical recruiting and prior training in organizational psychology. The cited sources are conceptual influences and theoretical antecedents, not direct sources of the framework. The integration of the evaluative framework, the scoring rubrics, the multipurpose questions, and the calibration protocol are original work.