Experimentation Design for Two-Sided Labor Market Matching:

Measurement Windows, Information Signals, and Hire Rate Optimization in Online Labor Markets

Anjan Goswami

Abstract

Online labor markets face a fundamental matching problem: connecting job owners with contractors through recommendation and search while managing information asymmetries, delayed conversions, and cross-side spillovers. As Chief Scientist at Elance (later Elance-oDesk, now Upwork), I led the design and implementation of experimentation infrastructure for measuring and optimizing hire rate in a two-sided marketplace. This case study describes three contributions: (1) an empirical analysis of measurement windows in multi-step hiring funnels, establishing 3-day metrics as the minimum viable window for causal inference in delayed-conversion marketplaces; (2) the design of a real-time feature pipeline built in Scala and Scalding with NLP and pricing signals that improved matching efficiency, producing a 4% hire rate improvement; and (3) a Bayesian contractor reputation model that provided quality signals independent of the matching algorithm. Elance during this period was also a site of active academic research: economist John Horton studied the platform's labor market dynamics during his doctoral work, and Steve Tadelis (UC Berkeley) consulted on marketplace design questions. Their published findings on online labor market intermediation and reputation systems provide useful theoretical context for the applied science problems described here.

1 Context: The Online Labor Market Matching Problem

Online labor markets are two-sided platforms where the intermediary's core function is reducing information asymmetries between buyers and sellers of labor. Horton, who studied Elance's labor market dynamics during his doctoral research (using platform data), described the platform creator's role as extending beyond simple matchmaking—they design the institutional infrastructure that determines "the space of permissible actions within the market," including matching algorithms, reputation systems, and search technology. Separately, Tadelis, who consulted for Elance on marketplace design questions, had built an extensive body of work on reputation systems and information disclosure in online marketplaces through his research at eBay.

These academic perspectives provide useful framing for the applied science problems we were solving in production. While the economists studied the platform's market structure from a theoretical and empirical angle, the science team—which I led as Chief Scientist—was building the real-time systems that operationalized matching, experimentation, and reputation scoring at scale.

At Elance, the matching problem had a specific structure. A job owner posts a project describing work they need done. The platform then operates through two parallel pathways:

- **Recommendation pathway**: The platform recommends contractors to the job owner based on skill match, availability, pricing, and reputation signals. The job owner reviews recommendations and sends connection requests.
- **Search pathway**: Job owners search for contractors directly and send requests. Separately, contractors search for jobs and are recommended jobs matching their profiles.

This creates a **four-surface matching system**: job owners receive contractor recommendations, job owners search for contractors, contractors receive job recommendations, and contractors search for jobs. Treatment on any surface can create spillovers on the others—a contractor hired through recommendation is no longer available for search-based matching, and vice versa.

The hiring funnel was a multi-step process with significant delays between stages:

1. Job posted
2. Contractors recommended (or found via search)
3. Job owner connects with contractor
4. Contractor responds
5. Contract agreed
6. First payment made

The business objective was to optimize contractor recommendation for faster, higher-quality hiring. The key metric was **hire rate**: the probability that a posted job results in a completed contract. But measuring the causal effect of recommendation changes on hire rate required solving a non-trivial experimentation design problem: *how long do you need to wait after treatment before you can measure the outcome?*

2 Contribution 1: Measurement Window Analysis

2.1 The Problem

In standard e-commerce A/B testing, the outcome (purchase) typically occurs within minutes or hours of treatment (product impression). The measurement window is short and the feedback loop is tight. In labor market matching, the outcome (hire) occurs days or weeks after treatment (recommendation), with multiple intermediate steps that each introduce delay and noise.

This creates a direct tension between **statistical validity** and **experiment velocity**:

- **Too short a window** (e.g., 1-day): Many hires still in progress are counted as non-hires. The measurement systematically underestimates conversion, attenuating treatment effects and reducing statistical power. You conclude experiments are ineffective when they are simply unmeasured.
- **Too long a window** (e.g., 15-day): You capture more true outcomes, but the signal is diluted by noise accumulated over a longer period. Confounding events multiply. And critically, you slow your experiment velocity—at 15-day measurement windows, you can run at most 2 experiments per month, creating an organizational bottleneck.

This is not merely an operational question. It is a **statistical design decision** that determines whether the experimentation platform can detect real effects at all.

2.2 The Analysis

We analyzed hiring funnel data across the full spectrum of measurement windows: 1-day, 3-day, 7-day, and 15-day post-treatment windows. For each window, we measured:

- **Conversion capture rate**: What fraction of eventual hires have occurred by day k?
- **Signal-to-noise ratio**: How does the treatment effect estimate change as the window extends?
- **Statistical power**: At what window length do we reliably detect a given effect size?

The analysis decomposed the hiring funnel by stage, measuring the time distribution of each transition: post-to-recommendation, recommendation-to-connect, connect-to-response, response-to-contract, contract-to-payment. This revealed where the delay was concentrated and which transitions were time-sensitive versus time-insensitive.

2.3 The Finding

Three-day metrics were the minimum viable measurement window. At 1-day, too many in-progress hires were misclassified as failures—the noise dominated the signal. At 3-days, a sufficient fraction of hires had completed (or progressed far enough through the funnel) to produce stable treatment effect estimates. Extending to 7 or 15 days improved point estimates marginally but did not change directional conclusions, while cutting experiment throughput by 2–5×.

This finding had immediate operational impact: it set the cadence for the entire experimentation program. Every A/B test on the recommendation and search systems used the 3-day window as the primary decision metric, with 7-day as a confirmation check.

2.4 Why This Generalizes

Any marketplace with delayed conversions faces this same tradeoff—real estate platforms, B2B procurement, enterprise SaaS, recruiting. The specific optimal window varies, but the analytical framework transfers directly: decompose the funnel by stage, measure time distributions per transition, find the minimum window where the conversion capture rate stabilizes, and validate that directional conclusions are invariant to longer windows.

3 Contribution 2: Real-Time Information Signals for Matching

3.1 The Theoretical Frame

Research on information disclosure in marketplaces offers a useful lens for understanding why real-time features matter. Tadelis's work on information disclosure as a matching mechanism (AER 2015) establishes a counterintuitive result: in markets with horizontally differentiated buyers and vertically ranked sellers, disclosing *more* information—even negative information—improves matching efficiency by helping buyers sort themselves to appropriate sellers. The implication for marketplace ML systems is direct: **richer, faster feature computation is a form of information disclosure that improves match quality.**

In Horton's intermediation framework, the recommendation algorithm *is* the intermediary's primary tool for conveying information about workers to buyers. The quality of that information—its accuracy, recency, and granularity—directly determines matching efficiency.

3.2 The Implementation

The pre-existing recommendation system used batch-computed features updated on a daily or weekly cycle. This meant that the matching algorithm was operating on stale information: a contractor's availability, recent project completions, pricing changes, and skill updates were reflected with significant lag.

We built a **real-time feature pipeline** that computed matching signals at query time, incorporating:

- **NLP features**: Real-time extraction of skill requirements, project complexity signals, and domain specificity from job descriptions. These were matched against contractor profiles to produce fine-grained relevance scores that went beyond keyword matching to capture semantic fit.

- **Pricing features**: Historical pricing data from previous contracts, adjusted for project type, contractor experience level, and market rates. This addressed a critical information asymmetry: job owners often have poor calibration on what projects should cost, leading to mismatches where highly qualified contractors ignore underpriced jobs and underqualified contractors apply to overpriced ones.
- **Availability and responsiveness signals**: Real-time indicators of contractor engagement, response latency, and current workload. A contractor who is actively seeking work and responds quickly is a fundamentally different match candidate than one who is fully committed to existing projects.

The matching model itself was deliberately simple: **logistic regression**. The insight—consistent with the diagnostic-first philosophy described in our companion case study on applied science team prioritization—was that the model architecture was not the bottleneck. The bottleneck was feature quality and freshness. A logistic regression with real-time NLP and pricing features outperformed more complex models with batch features because the information content of the inputs, not the expressiveness of the model, was the binding constraint.

3.3 Results

The real-time recommendation system produced a **4% improvement in hire rate** measured at the 3-day window and confirmed at 7-day. In a marketplace where small percentage improvements in hire rate translate to significant revenue impact (through platform fees on completed contracts), this represented substantial business value.

The improvement was not uniformly distributed. It was concentrated in:

- Jobs with specific, well-articulated requirements (where NLP features could extract precise matching criteria)
- Mid-range pricing tiers (where pricing calibration had the highest variance and thus the highest information value)
- First-time job owners (who had the weakest priors about contractor quality and benefited most from algorithmic matching)

4 Contribution 3: Bayesian Contractor Reputation Model

4.1 Reputation as Information Infrastructure

Research on reputation systems in online marketplaces—particularly Tadelis's extensive body of work—establishes that feedback mechanisms are not merely quality indicators. They are **structural components of market design** that affect entry, exit, effort, and pricing decisions. His research on eBay demonstrated that reputation signals affect both adverse selection (which sellers enter the market) and moral hazard (how much effort sellers exert). In online labor markets, where the "product" is a human being's effort over time, reputation plays an even more central role.

The challenge with reputation in labor markets is the **cold-start problem** and **score inflation**. New contractors have no reputation data. Established contractors accumulate inflated scores because dissatisfied buyers often leave no feedback rather than negative feedback (a well-documented phenomenon in marketplace reputation research). A reputation system that simply averages historical ratings provides a noisy, biased signal.

4.2 The Bayesian Approach

We built a separate **Bayesian reputation model** that estimated contractor quality as a posterior distribution rather than a point estimate. The model incorporated:

- **Prior**: Category-level quality distribution for contractors with similar skills, experience levels, and pricing. This addressed cold-start by providing an informed prior for new contractors based on their observable characteristics.
- **Likelihood**: Observed outcomes (hire success, project completion, buyer satisfaction, repeat hiring) weighted by recency and relevance. More recent projects and projects in the same skill domain received higher weight.
- **Posterior**: A distribution over contractor quality that naturally expressed **uncertainty**. A new contractor with two perfect reviews and a veteran contractor with 200 reviews averaging 4.5/5 might have similar point estimates but very different posterior variances—and the matching algorithm should treat them differently.

The reputation model fed into the recommendation system as a feature but was **architecturally separate** from the matching model. This separation was deliberate: it allowed the reputation model to be updated, validated, and improved independently of the matching algorithm, and it prevented the matching model from learning to "game" the reputation signal through correlated training.

5 The Two-Sided Experimentation Challenge

5.1 Cross-Side Spillovers

A/B testing in two-sided markets is fundamentally harder than in one-sided systems because treatment on one side creates spillovers on the other. If we improve contractor recommendations for job owners (demand side), this changes which contractors get hired, which changes contractor availability, which changes the supply side for other job owners. The standard Stable Unit Treatment Value Assumption (SUTVA) is violated.

We managed this through careful experiment design:

- **Job-level randomization**: Treatment was assigned at the job level, not the user level, to minimize within-user contamination across multiple job postings.
- **Supply-side monitoring**: We tracked contractor utilization rates in treatment versus control to detect supply-side depletion effects. If the treatment arm was disproportionately hiring the best contractors, control group outcomes would degrade for reasons unrelated to the treatment.
- **Short measurement windows**: The 3-day window, in addition to its statistical benefits, limited the time horizon over which spillover effects could accumulate.

5.2 Four Surfaces, One Metric

The four matching surfaces (job-owner recommendations, job-owner search, contractor recommendations, contractor search) were not independent. An improvement in recommendation quality could reduce search activity (substitution) or increase it (complementarity, if better recommendations teach job owners what to search for). We instrumented the recommendation A/B tests with funnel metrics across all four surfaces to detect cross-surface effects.

6 The Real-Time Infrastructure: Scala, Scalding, and Engineering Choices

The 4% hire rate improvement was ultimately an infrastructure achievement as much as a modeling achievement. The pre-existing batch pipeline computed features on a daily cycle, meaning the matching algorithm operated on information that was 12–24 hours stale. In a marketplace where contractor availability and job urgency change by the hour, this staleness was a first-order problem.

We rebuilt the feature pipeline as a **real-time streaming system in Scala using Twitter's Scalding framework** (a Scala API over Cascading/Hadoop that enabled type-safe MapReduce pipelines). I hired Vojin Jovanovic, a PhD student from Martin Odersky's programming research group at EPFL—the group that created Scala. Vojin was a core contributor to the Scala language itself, including co-authoring the Scala Futures and Promises specification (SIP-14). His expertise in functional programming, type systems, and distributed computation was instrumental in building a pipeline that was both performant and maintainable at scale. Vojin later noted that the production challenges he encountered at Elance influenced his subsequent academic work on domain-specific language embedding, published as the Yin-Yang framework (GPCE 2015, with Odersky as co-author).

The key engineering decisions:

- **Scalding for feature computation**: NLP feature extraction (skill parsing, complexity estimation, semantic similarity) and pricing feature computation ran as streaming jobs that updated contractor and job representations in near real-time. The functional programming model enforced immutability and composability, reducing the class of bugs that plague mutable-state pipelines.
- **Feature store with low-latency serving**: Computed features were written to a serving layer that the recommendation API could query at request time. This replaced the batch feature table that the previous system joined against.
- **Logistic regression as deliberate simplicity**: The matching model was intentionally kept simple. With real-time features, a logistic regression could capture the signal that mattered—current availability, current pricing fit, current skill match—without the debugging complexity of deep models. The bottleneck had been feature freshness, not model expressiveness.

This architectural choice—investing engineering effort in the data pipeline rather than the model—reflected a principle that has proven durable across every system I have subsequently built: **the quality of information flowing into a model almost always matters more than the model's capacity to extract signal from stale information.**

7 Academic Context

Elance during this period (2012–2014) was a site of active academic interest in online labor market economics. John Horton, then completing his doctoral research, studied the platform's labor market dynamics using data from the platform, and subsequently published work on online labor markets that formalized many of the intermediation concepts relevant to our matching systems. Steve Tadelis consulted for the company on marketplace design questions; his PhD students interned with the marketing team and published research on reputation and demand generation. Their work proceeded independently from the science team's production system development, but the broader intellectual environment—where economists were formalizing the theory of the very markets we were building—provided useful conceptual vocabulary for reasoning about our design decisions.

Horton's framing of the platform as an intermediary that conveys "high bandwidth information" about workers to buyers maps directly to what the real-time pipeline was doing: reducing the information latency between a contractor's current state and the matching algorithm's representation of that state. Tadelis's work on reputation systems and information disclosure informed our thinking about the Bayesian reputation model, particularly the insight that disclosing more information—including uncertainty estimates—improves matching efficiency rather than degrading it.

8 Generalizable Principles

1. Measure the measurement system first. Before optimizing the matching algorithm, we had to determine whether our experimentation infrastructure could detect improvements at all. The measurement window analysis was a prerequisite for everything else. In any delayed-conversion marketplace, the first investment should be in understanding the funnel's temporal dynamics.

2. Information freshness often matters more than model complexity. The 4% hire rate improvement came from logistic regression with real-time features, not from a complex model with stale features. In matching systems where the state of the world changes rapidly (contractor availability, pricing, workload), feature latency is a first-order concern. This is consistent with the broader finding in marketplace research that information disclosure improves matching—the ML system's job is to disclose more, better, faster information to the matching decision.

3. Separate the reputation model from the matching model. Architectural separation preserves the integrity of both systems and enables independent improvement. The reputation model captures long-run quality; the matching model captures query-specific fit. Coupling them tightly makes both harder to debug, validate, and improve.

4. Two-sided experimentation requires supply-side monitoring. A treatment that improves demand-side outcomes may do so by consuming disproportionate supply-side resources. Without monitoring for supply-side depletion, you overestimate treatment effects and create problems that only manifest at scale.

5. The prior determines the value of exploration. This principle echoes our findings in e-commerce search (see companion case study): in the Upwork context, recommending contractors with strong observable signals (skills match, pricing fit, availability) but limited platform history is informed exploration. Recommending random available contractors is naive exploration. The difference in hire rate outcomes is substantial.

6. Invest in the pipeline before the model. The Scala/Scalding real-time pipeline was the highest-ROI engineering investment in the entire effort. It unblocked every downstream improvement—better NLP features, fresher availability signals, real-time pricing—by reducing the latency between the world changing and the model seeing the change. In any matching system, the first question should be: *how stale is the information my model is operating on?* If the answer is "hours" or "days," the pipeline is the bottleneck, not the model.