PROJECT INTENT WORKING PAPER SERIES

# Intent Before Implementation

**Second Edition**

Alan Florschuetz

projectintent.com

---

## The Anatomy of Failure

Every failed project eventually receives an explanation.

The requirements weren't clear. The priorities changed. Engineering misunderstood the business. The architecture was wrong. The timeline was unrealistic.

Sometimes those explanations are true. But they all hide a deeper question.

*Did everyone actually agree on where we were trying to go?*

For decades, organizations treated this ambiguity as a management problem to be solved with more paperwork. We assumed that if we wrote enough pages of specifications, we could force alignment across an enterprise. We built processes to manage human communication because human communication was the messy bridge to implementation.

Implementation was the primary bottleneck of corporate progress. Writing code, structuring data, and managing system environments required months of highly specialized, expensive labor. Because execution capacity was scarce, it earned the right to consume the vast majority of our attention.

Artificial intelligence shatters that economic constraint.

When a capable individual working alongside an intelligence layer can generate multiple valid, production-ready systems before their first meeting of the day, execution ceases to be scarce. It becomes an abundant commodity.

That inversion exposes a reality we are poorly prepared to handle.

If implementation is becoming abundant, what becomes scarce?

The new scarcity is Intent.

---

## The Scarcity Inversion

To understand why traditional requirements fail in an era of abundance, consider a simple thought experiment.

Imagine asking ten engineers to implement the exact same requirements document. You will almost certainly receive ten different solutions. Each engineer will interpret the ambiguities through the lens of their own localized experience, making trade-offs between performance, complexity, and scale based on what they assume matters most.

Now, ask those same ten engineers to explain the core intent behind the work.

If you receive ten different answers, the project was already lost.

Requirements describe the mechanical behaviors of a system. Intent defines its strategic destination. When implementation was scarce, we spent our energy documenting requirements because narrowing the mechanical solution space early was the only way to control the high cost of human engineering variations.

Today, that approach creates immense organizational drag.

Implementation has become abundant. Misalignment hasn't.

When an individual operates with zero technical boundaries but highly fractured strategic alignment, they do not create value faster. They merely generate architectural noise, downstream technical liabilities, and organizational confusion at machine speed.

The quality of the answer can never exceed the clarity of the destination.

---

## The Optimization Trap

Almost every leader experimenting with generative technology experiences a predictable arc of realization.

It begins with the thrill of immediate execution. You type a loose, conversational requirement into an intelligence layer, and within seconds, a comprehensive system design appears on your screen. The code is formatted cleanly, the interfaces are mapped out, and the database migrations are written.

The immediate conclusion feels obvious: we must master prompt engineering. If better instructions yield cleaner results, then the organization that writes the most sophisticated prompts will ultimately win.

This conclusion is dangerous because it mistakes an interface mechanic for an operational strategy.

What these leaders fail to notice is the systemic trap hidden within the technology itself: AI almost always produces an answer. Whether your request is strategically brilliant or conceptually hollow, the machine will gladly return a highly detailed, deeply convincing plan.

That is the dangerous part.

**AI never questions the destination. It optimizes the journey.**

The machine does not tell you your objective is flawed. It optimizes it. Perfectly.

If you provide a confused strategic direction, the intelligence layer will generate a beautifully structured, highly performant implementation of that confusion. It will address edge cases you didn't consider, generate extensive automated documentation, and provision the infrastructure flawlessly.

A perfect implementation of the wrong intent is organizational debt delivered at machine speed.

The faster implementation becomes, the more expensive poor intent becomes. When the technical friction of writing software disappears, the only thing holding an enterprise to its trajectory is the absolute clarity of its human judgment. We can no longer rely on the slow pace of development to act as a natural dampener on poorly conceived ideas.

If you steer a ship toward a reef at five knots, you have time to correct the course. If you increase the speed to one hundred knots, the structural integrity of the hull ceases to matter. Only the navigation vector matters.

---

## The Intent Artifact

In legacy management frameworks, intent is treated as a soft, aspirational concept — a summary of good intentions scribbled at the top of a slide deck.

In an AI-native operating system, intent is treated as a hard technical artifact. It is the explicit operational contract that governs the interaction between human judgment, intelligence layers, and reality.

Consider a typical requirements statement: *"Build a customer portal."*

This statement describes a mechanism, not a destination. It gives both the amplified individual and the machine zero context for making trade-offs. Should the system prioritize high-throughput data processing, extreme security isolation, or frictionless user interactions?

Now, consider an intent statement:

> *Reduce customer onboarding time from ten days to one while maintaining regulatory compliance and preserving our existing identity model.*

This definition provides an unshakeable calibration point. It specifies the strategic boundary, the operational constraints, and the real-world evidence of success, while leaving the implementation entirely open to exploration.

Intent is a rigorous declaration of purpose that deliberately answers five core questions before execution is permitted to begin:

1. What is the precise truth of the customer friction point we are trying to change?
2. Why does this specific change matter to our enterprise position right now?
3. What non-negotiable architectural and security beliefs must guide every down-funnel micro-decision?
4. What financial, performance, and systemic limits absolutely cannot be crossed?
5. What objective, real-world evidence will signal that the outcome has been achieved?

Notice what this framework purposefully leaves blank: it never prescribes the implementation. It does not mandate a programming language, specify a database engine, or sketch out the individual screens of a user experience.

When implementation was scarce, we documented those details because changing them later was prohibitively expensive. Now that implementation is abundant, documenting execution details ahead of time is an exercise in futility.

> *The code can be regenerated, refactored, or replaced in seconds. Your intent must remain an immutable anchor.*

A properly constructed intent statement gives the individual the autonomy to explore dozens of implementation variations with AI, discarding elegant but irrelevant technical paths without ever losing sight of the strategic destination.

---

## The Vector over the Assembly Line

For decades, organizations structured themselves like manufacturing assembly lines. We assumed that human expertise was best coordinated by dividing projects into localized functional steps, passing requirements down a chain from strategy to product, then to architecture, then to engineering, and finally to quality assurance.

This structure was designed to manage the scarcity of knowledge. The people at the end of the line were not expected to understand the strategy; they were expected to master the immediate mechanics of their task.

Abundant intelligence dissolves the assembly line, replacing it with a single vector of alignment.

When an individual contributor can instantly use an intelligence layer to navigate across technical disciplines — evaluating infrastructure, auditing security models, and generating code simultaneously — the physical separation of these roles loses its structural purpose. The individual ceases to be a component on an assembly line. They become the navigator of a vector.

This shift changes where human engineering mastery creates value. Historically, an engineer earned their stature by mastering the microscopic syntax of implementation — memorizing language quirks, configuring boilerplate code, and managing the physical plumbing of environments.

Today, those activities are transitioning into functional commodities.

This does not downgrade the engineer; it promotes them. Their primary value is no longer their ability to act as a human compiler. Their value lives in their capacity for systemic judgment.

Real-world trade-offs are messy, highly contextual, and deeply human. They involve balancing long-term platform health against near-term market pressure, navigating customer ethics, weighing financial risks, and understanding the emotional trust of a user base. An intelligence layer can present the options, predict the performance trade-offs, and generate the structural components for each path.

But the act of choosing between those futures remains an un-delegatable human responsibility.

The elite builders of the next decade will not be those who can generate output the fastest. They will be the individuals who can articulate intent with absolute precision, ruthlessly audit automated implementations against systemic boundaries, and accept total accountability for the evidence their work produces in the real world.

---

## Conclusion

Implementation will continue to change.

Programming languages will change. Frameworks will change. Platforms will change. Even artificial intelligence will change.

Intent won't.

The organizations that endure won't be those that predicted the future most accurately. They'll be the ones that never lost sight of where they were trying to go.

The economics changed.

The organizations haven't.

Reality still charges full price for learning.

---

**From the Book**

**Optimized for Yesterday: What Happens When the Scarcity Changes**

*By Alan Florschuetz*

This working paper is part of the research foundation for *Optimized for Yesterday* — a complete framework for how organizations should operate when intelligence is abundant and implementation is no longer the constraint.

Available in paperback and Kindle. ISBN: 979-8-9971259-0-5

**projectintent.com/book**