INTENT-DRIVEN DEVELOPMENT

Where Human Intent Meets Machine Execution

Richard Stockley

Fractional IT Architect | Executive Coach & Mentor

intentdrivendevelopment.org

Version 0.2 – Free Edition – 13th April 2026

Intent-Driven Development: Where Human Intent Meets Machine Execution

Version 0.2 – 13th April 2026

This is a living document. IDD is an evolving methodology and this ebook will grow alongside it. Expect new chapters, revised frameworks, and deeper explorations as AI capabilities mature and IDD practice develops.

Original articles published at richardstockley.com

ebook hosted at intentdrivendevelopment.org

Foreword

I've spent almost thirty years bridging the gap between human needs and technical solutions. I've built systems for investment banks, charities, government agencies, and global high-street retailers. I've worked across grid computing, microservices, serverless platforms, and heavily multi-threaded architectures in mission-critical environments where speed, security, and resilience were non-negotiable.

Throughout all of it, one pattern has remained constant: the projects that succeed are the ones where the people involved are clear about what they are trying to achieve before they start building. The projects that fail, and I've seen plenty, are the ones where someone starts coding before anyone has truly articulated what the system is for, who it serves, and how they will know it works.

That has always been the case. What has changed is the consequence.

In the old world, unclear intent meant wasted sprints, frustrated stakeholders, and code that missed the mark. The damage was measured in weeks, sometimes months. The feedback loop, although painful, was human-paced. You could course-correct.

In the new world, the world of agentic AI, unclear intent means something different entirely. An AI agent given a vague specification will not ask for clarification. It will not push back. It will build exactly what you asked for, at extraordinary speed, and with total confidence. If your intent was unclear, you have not saved time. You have produced the wrong thing faster than ever before.

Speed without clarity isn't progress. It's just faster failure.

That observation became the starting point for Intent-Driven Development. Not as an abstract framework dreamed up in isolation, but as a practical response to a real shift I was witnessing in my consulting work. Organisations were adopting AI tools enthusiastically, generating code at unprecedented speed, and finding themselves no closer to delivering the right outcomes. The bottleneck had moved. It was no longer about what technology could do. It was about how clearly humans could express what they wanted technology to do.

IDD is my attempt to address that shift. It draws on everything I have learned across nearly three decades of delivery: the rigour of Test-Driven Development (TDD), the shared language of Domain-Driven Design (DDD), the empathy of User-Centred Design (UCD), the stakeholder clarity of Behaviour-Driven Development (BDD). It does not replace any of these. It brings them

together under a single discipline, one that treats human intent as the primary artefact of software development, not an afterthought.

This ebook collects and reshapes the fourteen articles I published between January and March 2026, when IDD began to take form in public. Those articles were written in real time, each one building on conversations, feedback, and challenges from readers, practitioners, and deliveries. This book weaves them into a cohesive journey, but it is not the final word. It is version 0.2 of a living document.

IDD is a living methodology. As AI capabilities grow at breakneck speed, and as more organisations begin to practise intent-first development, the framework will evolve. New chapters will be added. Existing ideas will be refined. This document will grow alongside the discipline it describes.

If you are a developer, an architect, a product leader, or a CTO trying to make sense of how your role changes when agents can build almost anything, this ebook is for you. If you are a non-technical leader trying to understand what your teams should be doing differently in the age of AI, this book is for you too. Intent is a language everyone can and should speak.

The journey starts with a simple question: if AI can build almost anything, are we asking for the right thing?

Richard Stockley
April 2026

How to Read This Book

A Living Document

This is version 0.2 of a book that will continue to grow. IDD is not a finished methodology – it is an evolving discipline, shaped by practice, feedback, and the rapid development of AI capabilities. Think of this as the foundation, not the final structure.

Structure

The book is organised in four parts, designed to be read sequentially if you are new to IDD, or dipped into selectively if you are already familiar with the concepts.

Part I – Foundations: Why Intent, Why Now. Establishes the case for IDD, introduces the core principles, shows how it unifies existing methodologies, and addresses the most common objection: that AI moves too fast for any framework to survive.

Part II – The Framework: Specifying and Measuring Intent. The practical core. What an intent specification contains, how to maintain human control in agentic workflows, how to measure whether implementations honour that intent, and how multi-agent systems interact with intent specifications.

Part III – Scaling Intent: People, Roles, and Hierarchies. How IDD reshapes practitioners, enterprise roles, and organisational structures. Includes the IDD Maturity Model for assessing organisational readiness.

Part IV – The Living System: Evolution, Learning, and the Future. How intent evolves through practice, how organisations learn to manage intent rather than just manage people, and where IDD is heading – including early thinking on Human Intent as a discipline beyond software.

Core Concepts

Intent-Driven Development introduces a number of concepts that together form a coherent way of working in the age of agentic AI. These concepts are introduced progressively through the book; this glossary provides a quick reference.

Intent A clear articulation of why something should exist, what outcome it should produce, and how success will be measured. Intent becomes the primary artefact of delivery teams.

Intent Specification A structured description that allows autonomous systems to execute while remaining aligned with human goals, including intent, domain context, success criteria, validation, constraints, and ethical considerations.

Intent Fidelity Measures how closely an implementation matches the original intent across completeness, correctness, alignment, and consistency.

Risk Dials Governance controls that determine how much autonomy autonomous systems are granted at each step of a workflow: Red (full review), Amber (spot-check), Green (autonomous).

The Intent Hierarchy Organisation Intent, Domain Intent, and Project Intent allow intent to scale through inheritance and promotion.

Intent Shaping The process of transforming human goals into executable, measurable, and governable intent.

Intent Promotion Successful patterns discovered in projects are promoted into domain or organisational intent.

The Human Layer and the AI Layer Humans define intent, governance, and accountability. AI executes implementation and automation.

Govern Intent. Delegate Execution. The core principle of Intent-Driven Development.

Origins

Each chapter is rooted in one of the fourteen articles published on richardstockley.com between January and March 2026. The content has been reorganised for logical flow, de-duplicated, and expanded with bridging material. An article-to-chapter mapping is included in the appendices.

Who This Book Is For

IDD sits at the intersection of engineering, architecture, product, and leadership. Software engineers and architects will likely start with Parts I and II. Product leaders and delivery managers may find Part III most valuable. CTOs and senior leaders will find the Maturity Model and Learning Organisation chapters directly applicable to strategic planning. Non-technical leaders are welcome here too. Intent is expressed in human language, not code.

What Comes Next

Beyond IDD lies a broader question: what does it mean to be good at expressing human intent – not just in software development, but in leadership, strategy, and organisational design? That exploration, which I am calling Human Intent, is the natural next step. You will find early thinking on this at the end of the book. For now, let's start with the foundations.

Prologue: The Boundary Between Intent and Execution

The history of computing is often told as a story of faster processors, better languages, new frameworks, and evolving architectures. But there is a simpler way to understand it. The history of software development is the story of moving the boundary between human intent and machine execution.

Early computing required humans to specify everything in machine instructions. High-level languages moved the boundary by allowing humans to describe logic instead of machine operations. Frameworks moved it further by handling infrastructure and patterns. Cloud computing and platforms automated infrastructure. DevOps automated pipelines. Now AI automates implementation itself.

Each step moved humans further away from execution and closer to intent.

There is a natural limit to how far automation can go. Machines can execute, optimise, and automate, but they cannot decide what should exist, what trade-offs are acceptable, or what outcomes matter. That remains human work. For now.

As execution becomes automated, clarity of intent becomes the most important professional skill. The role of practitioners shifts from implementing systems to designing the intent that governs systems.

Intent-Driven Development is not a break from history. It is the continuation of a long trend. The boundary between human intent and machine execution is still moving. Intent-Driven Development is about learning how to work when that boundary reaches us.

Part I

Foundations: Why Intent, Why Now

Chapter 1: Intent-Driven Development

Based on [the article](#) published 28 January 2026

The Shift

We used to be limited by what we could build. The constraint was always technical capability – the languages we knew, the frameworks available, the hardware we could afford, the time it took to write and test code. For decades, the bottleneck in software development was implementation.

That era is ending.

Agentic AI can now write code, generate tests, scaffold architectures, and deploy systems at a pace that would have seemed absurd even two years ago. But this has exposed a different bottleneck, one that was always there but never mattered as much: the quality of human intent.

When building was slow, unclear requirements were painful but survivable. When building becomes near-instantaneous, unclear intent becomes catastrophic. An AI agent given a vague specification will not ask for clarification. It will build exactly what you described, quickly, confidently, and incorrectly.

Speed without clarity isn't progress. It's just faster failure.

The discipline of software development must shift from mastering implementation to mastering intent.

The Five Questions

IDD forces clarity before code by requiring practitioners to answer five fundamental questions before any implementation begins:

1. **What is the intent?** What problem are we actually solving? What value does this create? Why does this need to exist?
2. **What does success look like?** What are the concrete outcomes we expect? How will we know the system is working correctly? Success must be measurable.
3. **How will we prove it works?** What specific test cases must pass? What scenarios need to be validated? In IDD, the tests are part of the specification, not an afterthought.
4. **What constraints must we respect?** What performance requirements, security implications, integration points, and technical limitations must be considered?
5. **What are the ethical implications?** Who is impacted by this being built? Are we building something we should build, not just something we can build?

Beyond Better Prompting

It is tempting to view IDD as sophisticated prompt engineering. That misses the point. Prompt engineering is a technique for interacting with a specific model at a specific moment. IDD is a methodology for human thinking. Just as Test-Driven Development taught developers to think in tests before writing code, IDD teaches practitioners to think in intent and outcomes before engaging any implementation mechanism, human or artificial.

Intent-Driven Development operates across three distinct but connected levels:

- **Thinking** – the discipline of clarifying intent before action
- **Artefacts** – the creation of structured intent specifications
- **Systems** – the governance, measurement, and organisational models that allow intent to scale

Most methodologies focus on one of these levels. IDD connects all three.

What IDD Creates

When teams adopt intent-first thinking, iteration cycles shorten dramatically. Success becomes measurable because criteria are defined before implementation. Collaboration improves because intent is expressed in human language. Professionalism is restored through a disciplined framework. And skills become future-proof, as AI gets better at coding, defining good intent becomes the differentiating human skill.

A Starting Point, Not a Destination

This first chapter establishes the core proposition: in the age of agentic AI, the quality of human intent determines the quality of outcomes. The chapters that follow explore how IDD connects to established methodologies, why it survives the rapid evolution of AI models, and ultimately how it scales across organisations.

Chapter 2: Bridging UCD, DDD, BDD, and TDD

Based on the article published 29 January 2026

The Question That Followed

Within hours of publishing the first IDD article, a pattern emerged. Developers, architects, and team leads were asking not just where does IDD fit, but whether software development has been structured around the wrong primary artefact, and these methodologies compensate for this limitation? The answer: these methodologies flow together, with IDD as the discipline that connects them into a coherent whole.

What Each Methodology Contributes

- **User-Centred Design (UCD)** teaches you to understand user needs through research, personas, and testing.
- **Domain-Driven Design (DDD)** teaches you to model the business domain with a shared language and bounded contexts. Without DDD, AI builds technically correct but semantically wrong solutions.
- **Intent-Driven Development (IDD)** teaches you to articulate intent – crystallising it into a clear specification with success criteria, validation, constraints, and ethical boundaries.
- **Behaviour-Driven Development (BDD)** teaches you to communicate in stakeholder language – translating specifications into Given-When-Then scenarios.
- **Test-Driven Development (TDD)** teaches you to validate through tests. In an IDD world, test-first thinking is incorporated into the specification stage and amplified by AI.

The Complete Flow

When arranged as a flow: UCD discovers user needs. DDD models the domain. IDD specifies intent, embedding test-first thinking. BDD communicates in stakeholder language. AI implements. Automated validation determines pass or fail. Agentic exploratory testing probes further. UCD validates with real users. Learning feeds back.

Why DDD Becomes More Critical, Not Less

Without a domain model, AI defaults to generic patterns. DDD gives AI the vocabulary, relationships, and rules of your business world. IDD adds the specification layer. Together, they ensure AI builds the right thing in the right way.

A Unified Discipline

IDD does not replace these practices. It reframes them around intent as the primary artefact, with each discipline aligning beneath it. It is the connective tissue that makes them work together when the implementation partner is no longer exclusively human. The question is not whether these methodologies survive the AI era. It is whether we have the discipline to use them properly when agents can build so much, so fast.

Chapter 3: Why IDD Survives Rapid AI Model Evolution

Based on the article published 2 February 2026

The Objection

AI is moving incredibly fast. Models evolve weekly. Tools appear and disappear within months. Why invest in a methodology that might feel obsolete before implementation is complete? It is a reasonable question – but it is based on a misunderstanding. IDD is not built on the parts that keep shifting. It is built on the parts that do not change.

The Coupling Problem

Most AI frameworks fail because they couple human concerns directly to current AI capabilities. Prompt engineering techniques became obsolete within two years. Tool-specific architectures needed reworking with every landscape shift. When frameworks are coupled to technology, technological change forces a restart.

The Separation Principle

IDD enforces a clear separation between two layers. The human layer – user needs, domain models, intent specifications, stakeholder communication, and accountability – is stable. The AI layer – implementation tools, model capabilities, automation levels – changes rapidly. Between them sits the governance interface. When AI improves, you adjust governance controls. When tools change, you swap implementations. The human layer stays stable. The AI layer remains fluid. The interface absorbs the change.

How Adaptation Works

When new AI capabilities emerge, governance controls remain where they are while new implementations are evaluated in parallel. Intent fidelity is measured, not assumed. Improvements are identified first in low-risk areas. The result is that organisations compound their learning rather than restarting with every technology shift.

Teams that succeed are not using better AI. They are using frameworks that survive change.

Part II

The Framework: Specifying and Measuring Intent

Chapter 4: How Do You Specify Your Intent

Based on the article published 17 March 2026

From Philosophy to Practice

Part I established the case for IDD. Now the practical question: what does an intent specification actually look like? This chapter introduces the six elements that make up a complete IDD specification. Teams will adapt these to whatever tools they use. What matters is that the specification captures the elements that allow autonomous systems to execute reliably while remaining aligned with human intent.

The original five questions evolved into six elements as DDD integration made clear that domain context required explicit treatment.

The Six Elements

1. Intent: The Why and the What

Why are we building this? What problem does it solve? What outcome defines success? Without clear articulation, everything downstream becomes interpretation.

2. Domain Context: The World

Which bounded contexts are involved? What entities, aggregates, and value objects exist? What ubiquitous language does the organisation use? When agents lack domain context, they default to generic patterns.

3. Success Criteria: The Done

Functional and non-functional requirements. BDD acceptance criteria. Without clear success criteria, intent fidelity cannot be measured.

4. Validation: The Proof

What must be tested, which edge cases matter, how the system should behave when things go wrong. When well defined, agents can generate test suites exploring far more combinations than humans typically consider.

5. Constraints: The Boundaries

Technical platforms, architectural patterns, integrations, resource limitations, and regulatory obligations. Mature IDD organisations define shared constraints once and let project specifications inherit them.

6. Ethical Considerations: The Should We

Should this system exist at all? Who is impacted? What harms might arise? When autonomous systems can build rapidly, whether to build becomes as important as how to build.

Bringing It Together

A detailed worked example – an abandoned cart recovery system for e-commerce – is available as a downloadable reference on intentdrivendevelopment.org. It illustrates how all six elements work together in practice. The key insight is that the level of specification detail should match the level of risk and complexity involved.

For teams beginning their IDD journey, starting with just Intent and Constraints alone dramatically improves clarity. Additional elements can be added as teams gain experience.

What Specifications Enable

When specifications capture these six elements clearly, autonomous systems can execute reliably, cross-functional teams gain shared understanding, intent fidelity becomes measurable, governance becomes evidence-based, and institutional knowledge accumulates. Delegating execution does not delegate responsibility.

Chapter 5: Human Gates in Agentic Flows

Based on the article published 1 February 2026

The Biggest Barrier to Enterprise AI

The biggest barrier to enterprise AI adoption is not technology. It is the fear of losing control. Enterprise leaders ask: what if the AI makes a critical mistake? How do we maintain compliance? Who is accountable?

They are right to ask. Industry research consistently shows that while the vast majority of organisations experiment with AI, only a small minority scale it beyond pilots. The gap between adoption and impact has never been wider. The bottleneck is organisational transformation capability – most organisations are trapped in pilot mode, lacking a framework for gradual adoption with appropriate governance.

IDD addresses this through risk dials: explicit governance controls at every decision point in an agentic workflow.

The Risk Dial Framework

High Control (Red). Human must review and approve before proceeding. Agent output is a suggestion. This is where every organisation should start.

Medium Control (Amber). Agent proceeds, but human reviews a sample. Human can override or roll back. Delegating, not abdicating.

Low Control (Green). Agent proceeds autonomously. Human receives notification. Only reached after trust is demonstrated through measurement.

These dials are set by the organisation, not by the AI, not by vendors.

Where the Gates Sit

User needs discovery and domain modelling are always human-led. Intent specification starts under full review and can progress to spot-checking. Code review is differentiated by risk: security, payments, and core infrastructure remain under full review permanently. Agentic exploratory testing and deployment follow the progressive pattern from full review to monitoring.

What Never Moves Off Full Control

Strategic intent, ethical boundaries, security and compliance decisions, and final production approval are always human-owned. Agents do not get a vote on strategic direction or ethical judgement. Ethical boundaries are not delegated. They remain permanently under human control, regardless of autonomy level.

From Cautious to Confident

Risk dials resolve the speed-versus-control tension. They provide a mechanism for moving gradually and reversibly from caution to confidence, backed by data. But that requires a way to measure whether AI implementations genuinely align with intent – which is the subject of the next chapter.

Agents amplify your capabilities. You decide which ones, when, and how much.

Chapter 6: Measuring Intent Fidelity

Based on the article published 8 February 2026

The Question Leaders Always Ask

Once governance controls are in place, senior leadership asks: how do we know it is actually working? And specifically: how do we know when it is safe to trust AI more?

Intent fidelity is the primary control metric. It tells you when AI systems are behaving as intended, when they are not, and when it is safe to adjust risk dials. Intent fidelity is a leading indicator. Business outcomes are lagging indicators. Organisations that rely only on outcomes discover misalignment too late. Organisations that measure intent fidelity detect it early, when it is still cheap to correct.

The Four Dimensions

- **Completeness.** Did the implementation address everything specified? Missing features, ignored constraints, skipped ethical considerations.
- **Correctness.** Does it work as specified? High completeness but failing the performance threshold means low correctness.
- **Alignment.** Does it reflect the real intent? The most critical and least automatable dimension – requires human judgement.
- **Consistency.** Does it fit coherently within the existing system? Functional code using wrong architectural patterns is inconsistent.

The Baseline Principle

Intent fidelity applies equally to human and AI implementations. Humans are not exempt from measurement, and AI is not held to a harsher standard. When both scores are low, the issue is intent quality itself.

IDD does not ask leaders to trust AI more than humans. It asks them to trust measurement more than intuition.

When to Adjust Governance Controls

Full review when fidelity is below 85% or during early implementations. Spot-checking when fidelity sustains between 85–95% across twenty or more implementations. Monitoring-only above 95% across fifty or more. Different specialist agents earn trust at different rates.

From Measurement to Business Outcomes

Intent fidelity translates directly into business language. Velocity improvement, risk reduction, and confidence to scale all become quantifiable. Organisations with measurement frameworks move from pilot to production within months.

Chapter 7: IDD via Multi-Agent Systems

Based on the article published 7 February 2026

The Next Architectural Evolution

Multi-agent systems, specialised agents collaborating rather than a single agent handling everything, raise an immediate question: does this change how we specify intent? The short answer is no.

Why Specifications Do Not Change

IDD specifications describe what to build and why, not how the AI organises itself. The specification does not care how the AI layer is structured internally. This resilience is a direct consequence of the separation principle established in Chapter 3. IDD specifications are tool-agnostic, model-agnostic, version-agnostic, and architecture-agnostic.

How Governance Adapts

With multi-agent systems, governance controls can be applied at the specialist-agent level. An architect agent under full review, a test agent under monitoring only, a security agent under permanent full review. The framework adapts to whatever granularity makes sense.

What Comes After

Multi-model teams, human-agent hybrids, hierarchical agent organisations, self-improving agent teams. None of this changes IDD specifications. Every evolution lives in the AI layer. Intent specifications remain in the stable human layer.

But, as the system changes, the people within it must change too.

Build on bedrock. Let the AI layer evolve. Your investment compounds.

Part III

Scaling Intent: People, Roles, and Hierarchies

Chapter 8: The New Practitioner

Based on the article published 14 March 2026

A Structural Shift, Not a Reduction

For decades, the central question has been: how do we build this correctly? As AI takes on implementation, the question becomes: how do we define intent with sufficient clarity that execution can safely be delegated? This is not a reduction in responsibility. It is an elevation. Every role continues to exist. What changes is where expertise is applied.

From Maker to Designer of Constraint

When AI generates implementation at extraordinary speed, the production of artefacts becomes less scarce. What becomes scarce is the clarity of governing intent. Practitioners contribute by defining boundaries within which autonomous systems operate. Deep expertise shifts upstream.

Skills That Compound in Value

- **Ambiguity resolution** – recognising unclear intent before it causes misalignment at machine speed.
- **Cross-disciplinary translation** – bridging product thinking, user insight, architecture, security, and operations into coherent intent.
- **Governance literacy** – risk thresholds, accountability boundaries, and escalation as everyday practice.
- **Intent fidelity judgement** – from "does the code work?" to "does it faithfully represent the governing intent?"

A Profession, Not a Persona

The “AI Engineer” narrative conflates literacy with mastery. Enterprise software requires coordinated expertise. The intent specification is the shared coordination point. The future practitioner is a specialist contributing to a system organised around shared intent.

How Every Role Evolves

UCD becomes the origin point of delivery. UX designs experience constraints. Product governs business intent. Engineers evolve from implementers to architects of constraint. Architecture embeds directly in intent specifications. QA verifies intent fidelity. Security moves to upstream specification. Platform teams become the operational backbone. Engineering Leadership guides the cultural shift.

Delegating execution does not mean delegating responsibility.

Chapter 9: Designing Intent at Scale

Based on the article published 2 March 2026

When Intent Becomes the Governing Artefact

Chapter 8 explored how individual practitioners evolve. This chapter examines how the enterprise adapts – a different challenge requiring a different response. Decisions that once lived inside code reviews move upstream into specification. Implicit assumptions must become explicit constraints.

From Inspection to Articulation

Traditional governance relies on inspecting output after production. When AI accelerates implementation, correcting misalignment after the fact costs far more than preventing it through clear specification.

Implicit intent becomes a liability when implementation happens at machine speed.

Coordinated Specialisation, Not Consolidation

IDD makes collaboration explicit through a shared governing artefact. Product clarifies business outcome. UX safeguards user intent. Architecture defines structural boundaries. Security encodes non-negotiable constraints. Engineering shapes decision space. QA validates alignment. The strength lies in coordinated specialisation, not consolidation.

Designing for Delegated Autonomy

Designing for delegation requires bounded scope, explicit success criteria, transparent constraints, and clear identification of non-movable boundaries. Certain domains may remain permanently human-controlled. Plateau can represent optimal calibration.

Chapter 10: The Intent Hierarchy

Based on the article published 23 March 2026

The Scaling Problem

When teams write their first IDD specifications, elements repeat immediately: regulatory requirements, infrastructure constraints, domain models, architectural patterns. At enterprise scale, this duplication becomes a liability. The Intent Hierarchy solves this: shared intent defined once, inherited by default, refined continuously.

Three Levels of Intent

- **Organisation Intent** – enterprise-wide standards, architectural patterns, security baselines, compliance approaches, ethical principles.
- **Domain Intent** – bounded context definitions, domain models, business rules, ubiquitous language.
- **Project Intent** – feature-specific goals, validation criteria, and unique constraints, referencing inherited intent rather than restating it.

Inheritance and Promotion

Intent flows downward through inheritance. But it must also evolve upward: patterns that prove effective through implementation are promoted into domain and then organisational intent. Intent is not only inherited, it is earned. Projects can extend inherited intent but cannot silently override it. Ethical intent is inherited like any other, but conflicts are escalated, never overridden silently.

Anti-Patterns

God Layer (too much defined centrally), Orphan Projects (no inheritance), Abandoned Layers (defined once, never maintained), Inheritance Spaghetti (overly complex paths), Premature Abstraction (hierarchy before patterns emerge). Start without a hierarchy and introduce it when duplication becomes painful.

Chapter 11: The IDD Maturity Model

Based on the article published 26 February 2026

Measuring Capability, Not Adoption

The IDD Maturity Model measures capacity to expand autonomy while preserving clarity, accountability, and control. Progression is gated by alignment, not adoption.

Level 1: Supervised Learning

Universal conservatism. All risk dials fully engaged. Every AI-generated artefact under human review. The purpose is calibration: learning where AI operates reliably and embedding intent fidelity measurement as structural capability.

Level 2: Selective Delegation

Autonomy begins to differentiate. Low-risk categories transition to spot-checking. Delegation is always evidence-gated. Measurement standardisation is critical infrastructure.

Level 3: Scaling with Sustained Alignment

A structural shift. Engineering effort moves from inspecting output to articulating intent. For many enterprises, this is an optimal steady state – substantial efficiency with preserved human authority over high-risk decisions.

Level 4: Continuous Optimisation

Workflows reconsidered in light of AI capabilities. Specifications evolve for human and machine clarity. Not every organisation will pursue or require this level. Plateau at Level 3 may represent mature equilibrium.

Anti-Patterns and Regression

Perpetual pilots, premature acceleration, symbolic governance. Progression is neither linear nor inevitable. Regression under uncertainty is disciplined risk management.

Organisations that capture enduring value from AI are distinguished by their capacity to scale autonomy while preserving control over what matters.

Part IV

The Living System: Evolution, Learning, and the Future

Chapter 12: Intent Evolution and Intent Shaping

Based on the article published 23 March 2026

The Hierarchy Is Not the Starting Point

The Intent Hierarchy can suggest organisations should start by defining enterprise intent, then domain, then project. In practice, intent begins as work. It first appears locally, in projects where real work happens. Only later does it move upward.

> *The hierarchy is not the starting point. It is the result of learning.*

Intent Shaping

Intent shaping transforms human goals into executable, measurable, and governable intent. Shaping does not happen only at the start; delivery reveals missing constraints, measurement reveals divergence, governance reviews reveal risks. Intent becomes clearer through execution, not just before it.

Bottom-Up Discovery Meets Top-Down Context

From the bottom up, projects discover what works. From the top down, organisations provide strategic direction and regulatory constraints. Neither is sufficient alone. IDD works when they meet in the middle.

Promoting Intent Upward

Project intent becomes domain intent when patterns are shared within a bounded context. Domain intent becomes enterprise intent when patterns span the organisation. This is how

delivery experience becomes organisational knowledge. The hierarchy should emerge from practice, not theory.

Chapter 13: The Learning Organisation

Based on the article published 28 March 2026

A Feedback System, Not a Static Hierarchy

IDD is a model for how organisations learn. Intent flows down through inheritance. Learning flows back up through measurement, feedback, and promotion.

Traditional organisations are hierarchies of authority. Intent-driven organisations are feedback systems.

How the Flows Work

Intent flows down: direction, constraints, standards. Projects inherit what the organisation already knows. Execution produces outcomes. Measurement produces evidence. Learning flows up: successful patterns are promoted. Future projects inherit improved intent automatically. Intent is both imposed (from regulations flowing down) and discovered (from implementation flowing up).

Architecture Is Promoted Intent

In traditional models, architecture can disconnect from delivery. In an intent-driven organisation, architecture evolves from learning. Successful patterns are promoted upward and become standards.

Architecture shifts from prediction to promotion. From control to governance. From ivory tower to learning system.

What Organisations Really Need to Manage

Organisations used to manage people. Then they managed processes. Then they managed software. Next, they will manage agents. But what they really need to manage is intent.

If organisations must learn to govern intent in software, a deeper question emerges: how well do humans understand and express intent in the first place?

Chapter 14: Humans Have Roles, Agents Have Functions

Based on the article published 29 March 2026

Structured Around Execution

Organisations have been structured around human roles: analyst, architect, developer, tester, operations. Roles were containers for execution responsibilities. Once execution becomes automated and orchestrated by agents, the foundation changes.

The Distinction

> *Humans have roles. Agents perform functions.*

A role carries accountability, decision-making authority, domain knowledge, and judgement. A function is narrow: generate code, run tests, deploy a service. Agents do not need job titles, career paths, or departments.

The Trap of Relabelling

The temptation is to insert agents into existing structures. Business Analyst becomes Intent Analyst. Developer becomes Prompt Engineer. But this is the old world with new labels. The interesting question is whether those roles were a product of manual, expensive execution.

Multi-Agent Systems Are Not Organisations

Agents do not need roles or reporting lines. A more natural design is around functions: generate code, generate tests, run tests, deploy services, monitor systems. The human organisation is structured around responsibility. The agent system is structured around functions.

Humans Move Up the Stack

Humans do not disappear. They move up a level – from how to what and why. Setting direction, defining goals, describing outcomes, defining constraints, reviewing results, making trade-offs, deciding priorities, governing behaviour. Roles become containers for responsibility, decision-making, and intent.

Govern intent. Delegate execution.

Beyond Software Development

Agents execute functions. Humans remain accountable not only for outcomes, but for whether those outcomes should exist at all. If humans are responsible for intent and agents for execution, then organisations may stop being structured around delivery pipelines and instead be structured around defining, governing, and evolving intent. That idea takes us beyond software development and into something bigger.

What's Next: Human Intent

This ebook has traced a journey from a simple observation through a complete methodology for specifying, measuring, governing, and scaling intent in software systems. But as the ideas developed, a pattern kept appearing.

The same questions that improve software outcomes, clarity of intent, definition of success, validation of results, and governance of constraints, apply far beyond software.

They apply to leadership. They apply to strategy. They apply to decision-making.

If Intent-Driven Development is about expressing intent for machines to execute, the deeper question is this: How do humans get better at knowing what they want in the first place?

Throughout this ebook, a phrase has appeared again and again: **human intent**. The discipline of articulating intent clearly, measuring alignment, and governing delegation has applications far beyond technology.

The Pattern That Keeps Appearing

IDD's fundamental questions: what is the intent, what does success look like, how will we prove it works, what constraints must we respect, what are the ethical implications, were designed for software. But they apply equally to organisational strategy, coaching, and leadership. A CEO articulating a vision, a coach helping a client, a leader setting direction, all are expressing intent, defining success, and governing delegation.

Human Intent as a Discipline

Human Intent may be a discipline in its own right, one that sits upstream of IDD. Where IDD asks how we express intent for autonomous systems, Human Intent asks: how do humans get better at knowing what they want, expressing it clearly, measuring whether they are getting it, and adjusting when they are not?

In a world where AI can execute almost anything at speed, the quality of human intent becomes the single largest determinant of outcomes – not just in software, but in strategy, leadership, education, and organisational design.

What Future Versions Will Explore

Within IDD: deeper case studies, refined measurement frameworks, tooling that supports intent specification, and patterns from early adopters. Beyond IDD: the exploration of Human Intent as a discipline that touches every domain where humans set direction and others carry it out.

If AI can build almost anything, the question is no longer what we can do. It is whether we know what we want.

This is version 0.2 of a living document. IDD will continue to evolve. This ebook will grow alongside it. Visit intentdrivendevelopment.org to follow the journey. The original articles are published at richardstockley.com.

Govern Intent. Delegate Execution.

– End of v0.2 –

Richard Stockley | April 2026

intentdrivendevelopment.org | richardstockley.com

Appendix A: Article-to-Chapter Mapping

Each chapter in this book is rooted in one of the fourteen articles published on richardstockley.com between January and March 2026. The articles were written in sequence, each responding to conversations and challenges sparked by the last. In reshaping them for this book, the content was reorganised for logical flow rather than publication order.

The table below maps each original article to its corresponding ebook chapter, allowing readers to trace ideas back to their original form.

#	Original Article Title	Published	Chapter	Part	Source
1	Intent-Driven Development (IDD)	28 Jan	Ch 1	Part I	richardstockley.com
2	How IDD Bridges UCD, DDD, BDD, and TDD	29 Jan	Ch 2	Part I	richardstockley.com
3	Human Gates in Agentic Flows	1 Feb	Ch 5	Part II	richardstockley.com
4	Why IDD Survives Rapid AI Model Evolution	2 Feb	Ch 3	Part I	richardstockley.com
5	IDD via Multi-Agent Systems	7 Feb	Ch 7	Part II	richardstockley.com
6	Measuring Intent Fidelity	8 Feb	Ch 6	Part II	richardstockley.com
7	IDD Maturity Model	26 Feb	Ch 11	Part III	richardstockley.com
8	Designing Intent at Scale	2 Mar	Ch 9	Part III	richardstockley.com
9	The New Practitioner	14 Mar	Ch 8	Part III	richardstockley.com
10	How Do You Specify Your Intent	17 Mar	Ch 4	Part II	richardstockley.com
11	The Intent Hierarchy	23 Mar	Ch 10	Part III	richardstockley.com
12	Intent Evolution and Intent Shaping	23 Mar	Ch 12	Part IV	richardstockley.com
13	The Learning Organisation	28 Mar	Ch 13	Part IV	richardstockley.com
14	Humans Have Roles, Agents Have Functions	29 Mar	Ch 14	Part IV	richardstockley.com

All articles remain available in their original form at richardstockley.com.

Appendix B: Further Reading and Resources

The Foundations IDD Builds Upon

Intent-Driven Development does not emerge in isolation. It draws together several established disciplines that, for decades, have each contributed part of the answer to a shared problem: how to build the right thing, reliably, in complex environments.

Domain-Driven Design: Tackling Complexity in the Heart of Software – Eric Evans. Addison-Wesley, 2003. The foundational work on modelling complex domains. Concepts such as bounded contexts, ubiquitous language, and aggregates underpin how IDD expresses domain context within intent specifications. Without this grounding, AI systems default to generic patterns rather than domain-correct solutions.

Test Driven Development: By Example – Kent Beck. Addison-Wesley, 2002. Introduced the discipline of defining success before implementation. IDD extends this principle beyond code-level tests into intent-level specification, where validation becomes part of the governing artefact rather than a downstream activity.

BDD in Action: Behavior-Driven Development for the Whole Software Lifecycle – John Ferguson Smart. Manning, 2014. Demonstrates how intent can be communicated in human-readable form through behaviour and outcomes. IDD builds on this by embedding those behaviours directly within structured specifications that autonomous systems can execute against.

The Design of Everyday Things – Don Norman. Basic Books, 1988 (revised 2013). A reminder that all systems ultimately exist to serve human needs. IDD begins where this work begins: with understanding what should exist, before considering how it will be built.

These disciplines remain fully relevant. IDD does not replace them. It reframes them around intent as the primary artefact.

AI and the Shift in Constraint

The emergence of AI as an implementation partner changes the economics of software development. Execution, once scarce and expensive, becomes abundant. The constraint moves.

Co-Intelligence: Living and Working with AI – Ethan Mollick. Portfolio (Penguin Random House imprint), 2024. Explores how humans and AI collaborate in practice, and the shifting boundary between human judgement and machine execution. This work reinforces a central premise of IDD:

as AI capability increases, the limiting factor is no longer what systems can do, but how clearly humans can direct them.

Much of the current industry focus remains on prompt engineering, tools, and agent frameworks. These optimise interaction with AI systems. IDD addresses a different problem: the clarity and structure of the intent those systems are asked to execute.

Decision-Making and Human Judgement

As execution is delegated, human responsibility does not disappear. It concentrates.

Thinking, Fast and Slow – Daniel Kahneman. Farrar, Straus and Giroux, 2011. A foundational exploration of human judgement and decision-making. IDD implicitly relies on the distinction between intuitive and deliberate thinking: fast interpretation must be supported by slow, structured clarification when defining intent.

In an AI-enabled world, errors in execution can often be corrected. Errors in judgement, what should be built, what trade-offs are acceptable, what risks are tolerable, propagate at speed. This makes clarity of intent a critical professional skill.

Organisational Learning and Systems Thinking

IDD's later chapters – particularly the Learning Organisation and intent evolution concepts – connect to a broader tradition of thinking about how organisations learn and adapt.

The Fifth Discipline: The Art and Practice of the Learning Organization – Peter Senge. Currency, 1990 (revised 2006). The foundational text on learning organisations. Senge's systems thinking, shared mental models, and team learning directly parallel IDD's vision of organisations that learn through the bidirectional flow of intent and feedback.

Thinking in Systems: A Primer – Donella Meadows. Chelsea Green, 2008. An accessible introduction to systems thinking. IDD's treatment of organisations as feedback systems – where intent flows down and learning flows up – draws on this tradition.

Team Topologies: Organizing Business and Technology Teams for Fast Flow – Matthew Skelton and Manuel Pais. IT Revolution Press, 2019. Explores how team structures affect delivery. IDD's Intent Hierarchy and role evolution concepts extend this thinking into a world where team boundaries are defined by intent ownership rather than implementation responsibility.

Related Work on Intent-Driven Approaches

A number of practitioners and organisations are exploring related ideas. These efforts reinforce the direction of travel: toward intent-first thinking in an AI-enabled world.

John Ferguson Smart – "Intent-Driven Development: the hidden key to BDD" (2019). Smart used the phrase "Intent-Driven Development" to describe the practice of announcing your intent before you act – a BDD-adjacent concept focused on individual developer practice. IDD extends this into a complete methodology with specification, measurement, governance, and organisational design. johnfergusonsmart.com

Keyhole Software – "Intent-Driven Development: A Modern SDLC for AI-Accelerated Teams" (2026). Keyhole's approach focuses on aligning product and engineering teams on intent before building, with an emphasis on build-first documentation. Shares IDD's conviction that intent should precede implementation but focuses primarily on SDLC process rather than measurement or governance frameworks. keyholesoftware.com

Binoy Ayyagari – "Adaptive Intent-Driven Development (AIDD)" (2025). Ayyagari's Medium article explores AI-first development with multi-agent collaboration. Focuses on how AI agents collaborate on implementation rather than on how human intent is formally specified and governed.

Exadra37 – AI Intent Driven Development (GitHub, 2025–2026). A practical collection of markdown guidelines for AI coding agents, structured around WHY/WHAT/HOW documents. Valuable at the developer-tooling level; IDD addresses the broader organisational and governance dimensions. github.com/Exadra37/ai-intent-driven-development

intent-driven.dev – Spec-Driven Development with OpenSpec. Nilanjan Raychaudhuri's work on structured context engineering and specification-first workflows for AI agents. Focused on developer tooling and code-generation workflows. Complements IDD's specification framework with practical tooling. intent-driven.dev

These approaches share a common recognition: intent matters. Where they differ is in scope. Intent-Driven Development extends beyond individual practice or tooling. It provides a connected model spanning specification, measurement, governance, and organisational design.

IDD Resources

intentdrivendevelopment.org – The home of IDD. Free ebook downloads, the abandoned cart worked example, and community resources.

richardstockley.com – The original fourteen articles, plus broader writing on technology delivery, leadership, and coaching.

About the Author

Richard Stockley is a technology consultant, executive coach, and mentor with almost thirty years of global experience in software engineering, IT architecture, and delivery leadership.

A Fellow of the British Computer Society (FBCS) and an EMCC-accredited Coach and Mentor, Richard combines deep technical expertise with a commitment to developing people and strengthening the industry. His career spans investment banking, government, healthcare, retail, motorsport, and international NGOs, with particular depth in distributed systems, microservices, and mission-critical architectures.

Intent-Driven Development grew from the intersection of these three threads: consulting on how organisations build technology, coaching leaders on how they make decisions, and a lifelong conviction that clarity of intent determines the quality of outcomes.

richardstockley.com
intentdrivendevelopment.org

Consult | Coach | Create