determinIQ

Governed data intelligence

Research and development case study

Trusted analytics, built on evidence.

determinIQ proves it.

AI explains it.

Independent research and development
October 2026
Conceived, designed and developed by John Stutely

Governed AI, Verifiable Analytics and Human-Led Product Development

A practical account of building an analytics product around deterministic execution, evidence, verification and controlled use of AI.

Contents

01 Introduction
02 The original problem
03 Separating analysis from explanation
04 Authority boundaries
05 The governed query contract
06 Verification, not reassurance
07 From result to finding
08 Ask / Explore and the blank question problem
09 Summary and proactive orientation
10 Discovery without noise
11 Attention as a policy
12 Modular analytical capability
13 Context without changing truth
14 Modular domain packs
15 Privacy, PII and the AI boundary
16 Reports and governed communication
17 Evidence for decisions, not automated decisions
18 Personal Support
19 Documentation that comes from the source
20 From analytics engine to product
21 The end-to-end customer journey
22 Operational resilience
23 Scale and engineering validation
24 Accessibility and cognitive load
25 UI, UX and reducing cognitive load
26 Building a coherent product identity
27 AI as a force multiplier
28 Human-led, not AI-led
29 Recursive development
30 Convergence rather than accumulation
31 Checks and balances
32 One philosophy, two applications
33 What is unusual about the project
34 What I set out to achieve
35 Current position

This paper describes the product principles, R&D approach and development method at a deliberately high level.

Proprietary implementation detail is not included.

determinIQ in 30 seconds

determinIQ helps business users understand data without making AI the authority over the answer. Natural-language intent is converted into a governed analytical request, calculations run deterministically, evidence is retained and findings are verified before AI is allowed to explain them.

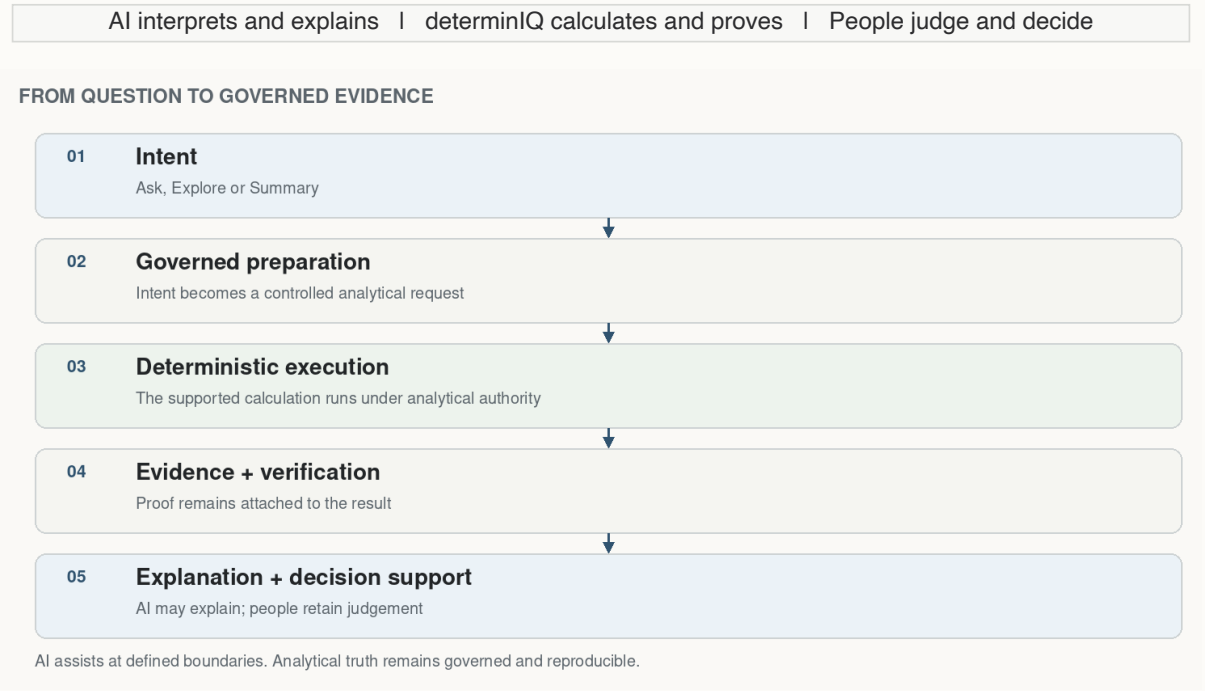


Figure 1. The governed analytical path. AI assists with interpretation and explanation, while analytical truth remains within the governed core.

In practical terms, determinIQ helps organisations understand what is happening in their data, identify meaningful patterns and changes, and explore questions without requiring specialist analytical skills.

It combines automated discovery with straightforward, natural-language exploration, presenting findings in a form that business users can understand and act upon.

Crucially, every analytical result is produced through deterministic, governed processes, supported by verifiable evidence. AI helps people understand the findings, but does not determine what is true.

Introduction

determinIQ started with a fairly simple concern: the more I worked with generative AI, the more useful I found it. It could interpret language, summarise complex material, help reason through problems and dramatically speed up development work.

But there was an obvious weakness. I could not always fully trust the outcome. More importantly, there was often no practical way to verify it.

An AI-generated answer could be correct, mostly correct, subtly wrong, based on a misunderstanding of the question, or based on an assumption that was never made explicit. The problem was not that AI was useless. Quite the opposite. The problem was that the fluency of the answer could make it look more reliable than it really was.

> *If AI gives me an analytical answer, what exactly am I being asked to trust?*

The model? The prompt? Its interpretation of the question? The calculations it claims to have performed? The data it decided to use? Its explanation of the result? And if the answer matters, how do I independently prove that it is correct?

That became the core research problem. I wanted to see whether I could use AI to make analytics much easier to work with, while removing the need to trust AI as the source of analytical truth.

The result is determinIQ Analytics, a governed analytics platform built around deterministic execution, evidence, verification and controlled use of AI. The project has also become an experiment in how AI itself can be used to build a complex product, not by replacing judgement, but by amplifying it.

The original problem

Traditional analytics has its own weaknesses. Business intelligence tools can be very powerful, but they often assume the user already knows what they are looking for. They also tend to expose a lot of complexity.

The user may need to understand the data structure, know which dimensions and measures matter, construct the right view and then interpret the result correctly. That is fine for analysts. It is less helpful for a business user who simply wants to understand what is happening.

Generative AI seems to solve that problem because the user can simply ask a question. That sounds ideal, but it introduces another problem.

If the model is both interpreting the question and producing the analytical answer, it has become the analytical authority. That means the user is relying on a probabilistic system to decide what is true. For some applications that is acceptable. For business analytics, particularly where decisions, money, performance or risk are involved, I did not think that was good enough.

> *Can natural language make analytics easier without giving AI authority over the answer?*

Separating analysis from explanation

One of the first decisions I made was to separate analytical truth from narrative.

In determinIQ, the analytical engine is the authority. The AI is not. The browser is not. The narrative layer is not.

A question has to move through a governed process before it becomes an answer.

The exact internal process is more detailed than that, but the principle is straightforward. The system first establishes what the user is asking. It then determines whether that question can be answered using a supported analytical operation. The calculation is executed deterministically. Evidence is retained. The result can then be verified, qualified, promoted, explained and presented.

AI can help interpret the user's intent and explain the governed result, but it is not allowed to independently decide what the data says.

> *AI can assist with understanding. It cannot decide the analytical truth.*

Authority boundaries

As the project developed, this idea grew beyond AI. I found it useful to be explicit about where authority sits throughout the system.

Different parts of determinIQ have different responsibilities. The narrative layer can explain, but it cannot change evidence. The browser can present and interact, but it does not decide analytical truth.

Business context can influence relevance and wording, but it cannot rewrite a calculation. Customer configuration can control permitted behaviour, but it does not become platform authority. Reports can communicate findings, but they do not create new analytical facts.

> *Every important boundary should have a clear authority, and competing authorities should be avoided.*

That reduces ambiguity and makes the system easier to reason about. It also makes errors easier to locate because the system has a defined answer to the question: which part is actually responsible for deciding this?

The governed query contract

At the centre of determinIQ is what I call the UQC, the governed contract that sits between a user's intent and the analytical engine.

The user may arrive with a natural-language question. That question is useful as an expression of intent, but it is not safe or precise enough to execute directly.

Before anything analytical happens, determinIQ converts that intent into a controlled contract. Its purpose is to define, in a governed and machine-checkable form, what the system believes the user is asking and what analytical operation is actually permitted.

At a high level, the contract establishes

- the analytical intent
- the relevant fields
- the permitted operation
- the expected direction or comparison
- the scope of the analysis
- the constraints that must be satisfied
- the evidence that must be produced

The important point is not the internal format of the contract. The important point is that it exists.

```
what the user says → governed contract → what the analytical engine is allowed to do
```

Without that boundary, a natural-language system can move too easily from interpretation to execution. The UQC forces an intermediate step. The system has to make its interpretation explicit enough to validate before it calculates anything.

That makes it possible to reject unsupported requests, ask for clarification where the intent is ambiguous, and prevent apparently plausible questions from reaching the execution layer in an uncontrolled form.

> *Natural language can be open-ended. Analytical execution should not be.*

Why the contract matters

The contract acts as a control point, separates interpretation from execution, gives the system something that can be validated before analysis begins, and provides a stable reference for evidence and verification after the result has been produced.

A technically correct calculation based on the wrong intent is still a wrong answer. The governed contract helps prevent that.

It also gives determinIQ permission to stop. If the intent is unclear, the system can clarify. If the requested analysis is unsupported, it can refuse. If the available data cannot support the requested answer, it does not need to manufacture one.

Verification, not reassurance

I did not want determinIQ to tell users that an answer was trustworthy. I wanted it to be able to show them why.

That led to a stronger emphasis on evidence and provenance.

Answer → explanation → evidence → calculation → provenance

Most users do not want to inspect calculations every time. They should not have to. But when an answer is important, disputed or simply interesting, the evidence should be there.

That has influenced calculation transparency, verification information, the evidence model and the Ledger.

The aim is not to make every interaction feel technical. It is to make the technical detail available when trust needs to be established.

From result to finding

A technically correct calculation is not automatically something a user needs to see. There is a difference between a valid result and a finding worth somebody's attention.

determinIQ therefore continues working after execution. A result can be evaluated against a wider set of governed criteria before it is promoted into something more visible.

- materiality
- salience
- novelty
- relationship to other findings
- direction and scale of movement
- contribution to a wider change
- relevance to the current context
- evidence quality

The objective is not to manufacture significance. It is to decide whether a technically valid result deserves attention.

Result → evidence → verified finding → qualification → promotion → attention

That distinction has become an important part of the product.

Ask / Explore and the blank question problem

Once I had started looking at conversational analytics more critically, another weakness became obvious. A blank text box only helps if the user already knows what they want to ask.

Many business users do not. They know they want to understand the data, but they may not know which analytical question is valid or useful.

That led to one of the core interaction models in determinIQ: Ask / Explore.

Ask

Ask is for users who already know broadly what they want to know. A natural-language question is interpreted, validated and converted into a governed analytical request rather than being executed directly.

Explore

Explore is for users who do not know what they should ask. The system looks at the structure of the dataset and presents analytical routes that are actually relevant.

Here is a toolbox. Work out what to do with it. → Here are the things this data can sensibly tell you.

> *Ask what you know. Explore what you do not.*

The same model has since been applied to Personal Support. If the user knows the problem, they can ask. If they do not know how to frame it, they can explore guided support routes.

Summary and proactive orientation

Ask and Explore both begin with some form of user intent. I also wanted determinIQ to be useful before the user asks anything.

That led to the Summary and governed Insights model. The purpose of Summary is not to generate a generic narrative about the dataset. It is to provide an orientation layer based on what the governed analytical process has actually established.

> *Here is what currently appears to matter.*

That creates five complementary ways into the data: Summary for orientation, Ask when the user has a question, Explore when the user needs help finding one, Insights when the user wants to see what matters, and Change Intelligence when the user wants to understand what changed.

Together they reduce dependence on a single conversational interface. The user can enter through orientation, a known question, guided exploration, qualified findings or change, rather than having to start with a blank prompt.

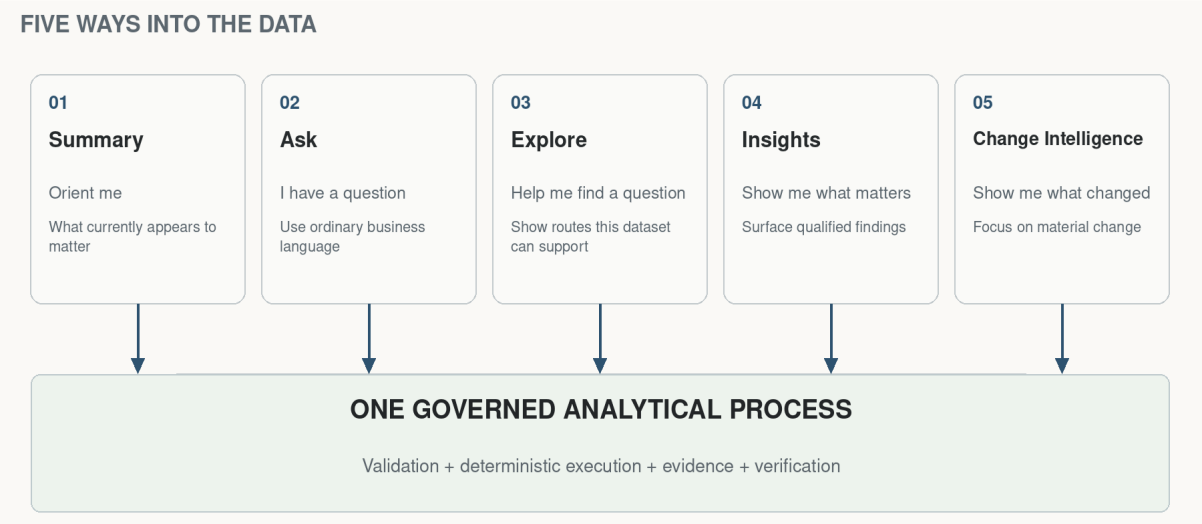


Figure 2. Summary, Ask, Explore, Insights and Change Intelligence provide different ways into the data but converge on the same governed analytical process.

Discovery without noise

Another problem I wanted to solve was the way analytics products often produce too much. It is relatively easy to generate observations. It is much harder to decide which observations are actually worth a person's attention.

I developed a number of analytical families to support this process:

- ranking disparity
- concentration
- movement
- acceleration
- volatility
- comparative displacement
- dispersion
- contribution to change

The aim is not to maximise the number of insights. One of the design decisions I am most comfortable with is that zero insights can be a valid result.

If nothing is sufficiently important, the system should say so. That is preferable to producing weak observations simply because the interface expects something to appear.

> *There is a difference between a system designed to always say something interesting and a system designed to say something only when the evidence warrants it.*

Attention as a policy

As this work developed, it became clear that relevance is not absolute. Some users want only the most important findings. Others want a broader exploratory view.

Focused ← Balanced → Exploratory

Balanced is the default. This gives the system a way to vary how much it surfaces without changing the underlying evidence.

It treats attention as a governed product decision rather than simply presenting everything the analytical engine can calculate.

Modular analytical capability

The analytical families themselves are modular. The current set of analytical capabilities is not intended to be the final boundary of the product.

New analytical families can be introduced without creating a separate execution path or bypassing the existing governance model.

validation → deterministic execution → evidence → verification → qualification

That gives the product room to expand analytically without creating multiple competing analytical architectures.

Context without changing truth

The same finding can mean very different things to different organisations or roles. A finance director may care about one aspect of a result. A sales lead may care about another. An operations team may interpret the same movement differently again.

determinIQ supports organisation context, user role and narrative behaviour so that the explanation can be more relevant. But the analytical result does not change.

> *Context can influence interpretation and presentation. It cannot rewrite the evidence.*

That distinction allows the product to feel more intelligent and relevant without creating different versions of analytical truth.

Modular domain packs

Another important part of the architecture is the use of modular domain packs. The core analytical and governance model is deliberately kept separate from domain-specific interpretation.

That means determinIQ does not need to be rebuilt for every industry, dataset or business problem. A domain pack can add the vocabulary, field meaning, analytical priorities, contextual rules and interpretation required for a particular use case while still relying on the same governed analytical engine underneath.

Core: governance → validation → execution → evidence → verification → provenance

A domain pack can define

- recognised field types and business meaning
- domain-specific terminology
- preferred analytical routes
- relevant measures and dimensions
- interpretation rules
- contextual weighting
- expected relationships
- business-specific thresholds
- narrative language
- guidance for Ask and Explore

Domain: business meaning → context → terminology → relevance → interpretation

The same analytical capability can therefore be used across very different business environments. Movement, concentration, volatility, ranking or contribution to change can have very different meanings depending on the domain.

The calculations remain reproducible. The domain-specific knowledge changes.

That makes it possible to support bespoke business cases without creating a separate analytical product for each one. It also avoids pushing domain interpretation into an unrestricted AI layer.

> *A common governed platform underneath, with highly specific behaviour at the business level.*

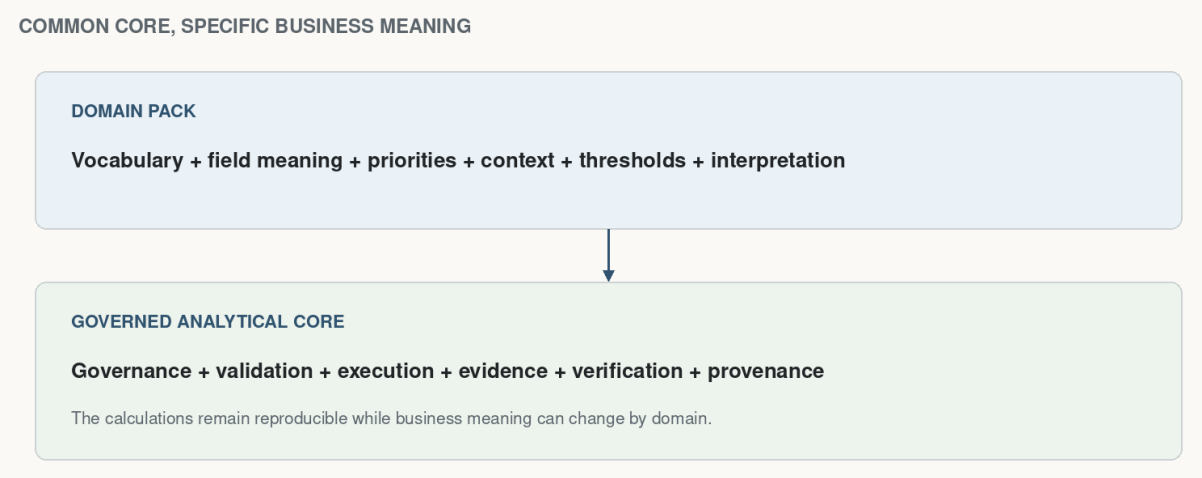


Figure 3. Domain packs change business meaning and relevance without creating a separate analytical authority.

Privacy, PII and the AI boundary

Privacy has been treated as part of the architecture rather than a policy statement added afterwards.

> *What is the minimum information AI actually needs in order to perform its permitted role?*

In many cases, the answer is far less than the complete raw dataset. That led to a design where personally identifiable information can be separated from the analytical and narrative process before AI is involved.

Sensitive values can be replaced with controlled tokens or anonymised representations. The analytical system can still work with the governed structure and meaning of the data, while the AI layer receives only the information it needs to explain or clarify the result.

Where the original value needs to be shown again, it can be rehydrated later inside the controlled application environment.

Raw data → detect sensitive values → anonymise or tokenise → analyse and narrate → rehydrate safely for presentation

The aim is not simply to hide a few columns. It is to create a clear privacy boundary between the original customer data, the governed analytical process, the information exposed to AI and the final presentation shown to the user.

Some geographic or customer-related fields can still carry analytical value even when their raw form is sensitive. Instead of discarding them completely, they can be transformed into a safer analytical representation.

> *Preserve analytical meaning where possible. Minimise identifiable data wherever possible.*

AI is not given more information simply because it is convenient. Narrative generation receives a governed and privacy-aware representation rather than unrestricted raw data.

For me, the important question is not “Can AI see this data?” It is “Does AI actually need to see this data to do its job?” If the answer is no, it should not receive it.

Reports and governed communication

I did not want the analytical journey to stop at an answer on screen. In real organisations, the useful outcome is often the ability to communicate that analysis clearly to somebody else.

That might be a manager, department head, board member, client or another stakeholder who was not involved in the original analysis.

> *How do you move from governed analysis to communication without losing the evidence and context that made the result trustworthy in the first place?*

Reports provide that bridge. A report can bring together governed findings, narrative, supporting evidence and relevant context into a form that can be reviewed and shared.

data → analysis → finding → evidence → communication → decision

Preserving provenance

The reporting model is designed to preserve the relationship between the presented finding and the governed analytical process that produced it. The narrative can be simplified for the reader, but the analytical result does not become detached from its source.

Different audiences, same evidence

The same underlying finding may need to be explained differently for different audiences. The presentation can vary while the analytical result remains stable.

> *Presentation can change. Evidence should not.*

Controlled sharing

The reporting system supports controlled sharing, allowing governed analytical outputs to reach people who may not be regular users of the application while retaining control over what has been published and how it is accessed.

Persistent analytical artefacts

Reports create persistence. Important findings can be preserved as governed artefacts and revisited later. Combined with the Ledger and evidence model, this creates a longer-term analytical history rather than a collection of disconnected queries.

Discover → verify → understand → communicate → decide

Evidence for decisions, not automated decisions

determinIQ is not intended to make the business decision. Its role is to reduce the work required to establish good evidence for that decision.

A product can help identify what changed, show why it matters, preserve the evidence and communicate the result. The final judgement still belongs to the person or organisation responsible for the decision.

> *determinIQ does not automate judgement. It automates much of the work required to arrive at evidence that a person can judge.*

For me, that is a better use of automation than pretending that every decision should be delegated to a machine.

Personal Support

The Personal Support feature developed quite late in Phase 1, but it turned out to fit the wider product unusually well. I wanted support to be more useful than a contact form.

The Ask / Explore model gave me a way to do that. A logged-in user can raise a specific issue or move through guided support routes.

- what the user was trying to do
- which support route they took
- what was asked
- what response was given
- whether the issue was resolved
- the status and history of the interaction

This turns support into another source of product information. Over time, it should be possible to see where users repeatedly struggle, where documentation is weak and where the product itself is creating avoidable friction.

User problem → guided route → support interaction → resolution → product intelligence

Documentation that comes from the source

Documentation raised a surprisingly similar problem. If the product is governed carefully but the documentation is maintained independently, the two can drift apart.

The software says one thing. The user guide says another. An old screenshot survives. A behaviour changes but a paragraph does not. Eventually the documentation becomes a description of what the product used to do rather than what it actually does.

I wanted to reduce that risk. The documentation model is therefore based around governed source documentation that can feed the customer-facing documentation environment.

> *The documentation should derive from the maintained source of truth, rather than becoming a competing source of truth.*

That gives documentation a role within the product lifecycle rather than treating it as an unrelated publishing exercise.

The source material can be maintained alongside the product and then used to generate or populate the documentation that customers actually see. This improves consistency across product behaviour, terminology, user guidance, support, release information, onboarding and operational documentation.

It also creates a useful feedback loop. If support shows that users repeatedly misunderstand something, the source documentation can be improved. That improvement then flows back into the published documentation rather than being patched separately in several places.

> *Maintain the authority once. Derive the presentation from it.*

Operational resilience

A real product also needs to survive change. That has meant thinking about updates, diagnostics, release integrity and recovery.

One principle I considered particularly important was the separation between application release state and customer state.

Updating or rolling back application code should not mean rolling back customer data, reports, configuration, organisation context, users or other tenant information. Those are different authorities.

This allows recovery to focus on the software without treating customer state as disposable.

> *A production system should have a controlled way to change, verify and recover itself*

From analytics engine to product

I did not want determinIQ to remain a research prototype. A major part of the work has therefore been turning the analytical architecture into something that could operate as a real product.

- multi-tenant customer instances
- reseller administration
- platform operator controls
- user management
- evaluation and live states
- customer provisioning
- branding
- organisation context
- entitlement controls
- diagnostics
- support
- update management
- rollback
- integrity checking
- documentation
- marketing
- onboarding

The product is built around a modular application architecture with clear separation between the analytical engine, customer experience, tenancy, configuration and operational control.

A system may analyse data correctly and still not be a usable product if it cannot be provisioned, supported, updated, governed and operated safely.

Can the analytical idea work? → Can this operate coherently as a real product?

The end-to-end customer journey

As Phase 1 progressed, individual features became less important than the complete journey. The product has to work from one end to the other.

Provisioning → configuration → first login → data ingestion → orientation → Ask / Explore / Insights / Change Intelligence → governed analysis → findings → verification → reporting → sharing → support

A failure anywhere along that path can make the wider product feel incomplete.

This has changed the nature of the later development work. Earlier work was largely about proving that the architecture could do what I wanted. Later work has increasingly been about convergence.

Duplicated controls are removed. Status is clarified. Transitions are improved. Terminology is made consistent. Users can return to where they came from. Support, documentation, reporting and administration are made to feel like parts of the same product.

That is a different type of engineering from adding more features. It is about reducing competing paths and making the product behave as one system.

Scale and engineering validation

I have tried to avoid designing determinIQ only around demonstration-sized datasets. The architecture has been tested against substantially larger volumes to see whether the assumptions still hold.

- large dataset ingestion
- memory-efficient processing
- deterministic execution at scale
- analytical opportunity generation
- evidence generation
- modular analytical families
- state management
- constrained memory environments

One useful test involved running the wider lifecycle across a dataset in the region of a million rows with multiple fields while remaining within a constrained memory budget.

The exact benchmark is less important than the reason for doing it. I wanted to know whether the architecture still behaved like the same system when the amount of data became uncomfortable.

That kind of testing has repeatedly exposed assumptions that would not have appeared in a small demonstration dataset.

Accessibility and cognitive load

Accessibility has been designed in from the outset. The application and documentation support light, dark and high-contrast presentation.

The same principle applies to cognitive load. determinIQ is intended for business users who need the information, not necessarily the mechanics.

That has driven repeated simplification of navigation, controls, language, status and hierarchy.

> *The aim is not to hide complexity. It is to avoid making the user carry complexity that the system can deal with itself.*

UI, UX and reducing cognitive load

A large part of the determinIQ project has been concerned with usability. The analytical architecture can be correct and still fail as a product if the user cannot understand where they are, what they can do next, or why the system is showing them something.

The target user is business competent, but not necessarily an analyst. That distinction has driven much of the interface work. I did not want determinIQ to behave like a traditional BI tool with a large number of exposed controls, nor like a chatbot where everything begins with a blank text field. The aim has been to create a guided interface where the system carries as much of the complexity as possible.

A consistent interaction model

The main user journey has gradually converged around a small number of distinct purposes: Summary for orientation, Ask when the user has a question, Explore when the user needs help finding one, Insights when the user wants to see what matters, Change Intelligence when the user wants to understand what changed, Verification when the evidence needs to be inspected, and Reports when the result needs to be communicated and preserved.

> *Summary / Ask / Explore / Insights / Change Intelligence → Verify → Report*

The objective is not to expose every capability at once. Each area should answer a clear user need and lead naturally to the next useful action.

Progressive disclosure

Most users do not need to see the full analytical machinery all the time. They need the answer first. If they want to go further, they can inspect the explanation, evidence, calculation and provenance.

This allows the interface to remain approachable without weakening the underlying rigour. Simplicity and verification do not have to be opposing choices.

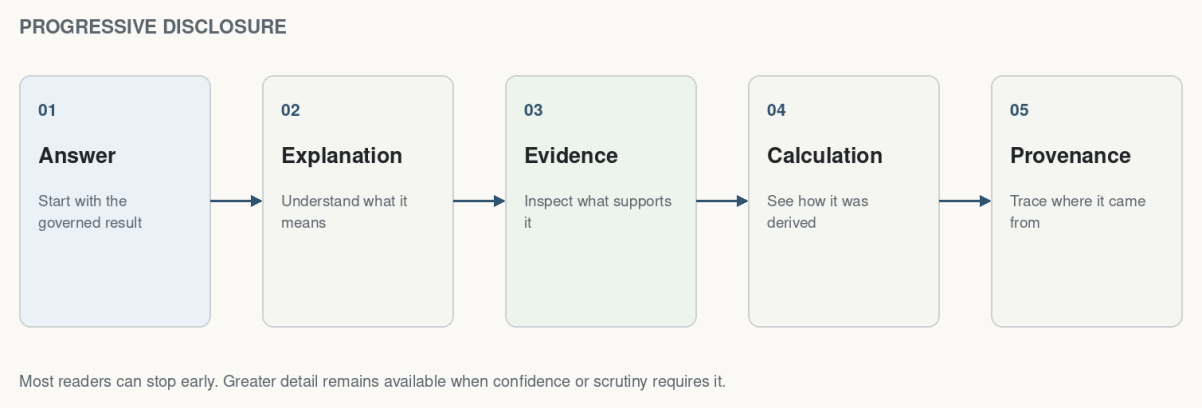


Figure 4. Progressive disclosure keeps the answer accessible while preserving a route to evidence, calculation and provenance.

State, continuity and navigation

A considerable amount of work has gone into making the application feel continuous rather than like a collection of screens. If a user asks a question, follows a clarification, inspects the answer and then returns, the relevant state should still make sense. The same principle applies when moving between Summary, Ask, Explore, Insights, Change Intelligence, Reports, Ledger, Reference and support.

This has led to repeated work on navigation, active-state indicators, breadcrumbs, drawers, return paths, status messages, preserved query state and clearer transitions. These details are small individually, but together they have a large effect on whether a product feels predictable and trustworthy.

Convergence through removal

Much of the later UX work has involved simplification rather than adding functionality. Duplicate controls have been removed, competing paths collapsed, labels clarified, status made more visible and visual hierarchy reduced where it was doing too much.

> *What can be removed or simplified without reducing capability?*

That question has become an important part of product convergence. A mature interface should expose the smallest coherent set of paths needed to let the user succeed.

Visual hierarchy and calmness

The visual design is deliberately restrained. Muted colour, controlled spacing, clear typography and limited use of borders and containers are intended to keep attention on the information rather than the interface itself.

The aim is not to make a technically sophisticated product look technical. It is to make the sophistication manageable.

UX as part of the trust model

I came to see UX as part of the trust architecture. Strong governance underneath the product is not enough if the interface leaves the user uncertain about what is active, what changed, whether an action succeeded, how to return, or where an answer came from.

> *The product needs to be clear enough to use, structured enough to trust, and simple enough that the user is not carrying unnecessary complexity.*

The UX itself has been developed recursively. Running screens are inspected, screenshots are reviewed, friction is identified, changes are made and the result is checked again. This is the same evidence-led development loop used elsewhere in the project.

Building a coherent product identity

As determinIQ developed, branding became more than a logo or colour palette. The product was growing into a family of environments: the analytical application, customer administration, partner controls, documentation, support, reports, shared content and the public-facing product presence.

If each of those evolved independently, the result would feel like a collection of systems rather than one product. A considerable amount of work therefore went into creating a consistent visual and behavioural identity across the estate.

One product, different contexts

Different parts of determinIQ have different jobs. An analytical workspace should feel focused. Documentation needs to support sustained reading. Partner controls need to make status and account actions immediately understandable. Support needs to feel approachable. A report needs to work as a persistent business artefact rather than looking like a screenshot from an application.

The challenge was not to make every screen identical. It was to make them unmistakably part of the same product.

That has produced a common design language across typography, colour, spacing, navigation, page structure, controls, status treatments, cards, headings, iconography, forms, notifications, mastheads, footers and light, dark and high-contrast presentation.

Branding as a system

I increasingly treated the brand as a system rather than a collection of graphical assets. The visual identity needed to communicate many of the same characteristics as the architecture itself: calm, controlled, precise, credible and understated.

That influenced decisions such as muting colour, reducing unnecessary decoration, rationalising borders and containers, using whitespace more deliberately and making important state changes clear without making the interface visually aggressive.

Consistency builds trust

A customer may move from marketing to login, into the application, then into a report, documentation or Personal Support. Each transition contributes to their perception of the product. Consistent terminology, hierarchy and interaction reduce uncertainty and help the user build familiarity.

> *Marketing → Login → Application → Reports → Documentation → Personal Support → Customer and Partner environments*

The brand therefore has to survive the whole customer journey, not just the main application.

The brand emerged from the product

The identity was not designed once and imposed on determinIQ. It emerged iteratively alongside the product. Screens were reviewed, spacing changed, navigation simplified, colours muted, controls rationalised, shared mastheads and footers aligned, and new environments tested whether the visual language could extend without losing coherence.

> *The brand emerged from the product, rather than being applied on top of it.*

Taken together, the project now has several related forms of architecture: analytical architecture governs how intent becomes evidence; experience architecture governs how people move through the product; product identity architecture keeps those experiences coherent; and operational architecture allows the whole system to be provisioned, supported, updated and recovered.

AI as a force multiplier

There is another side to this project which I think is just as interesting as the product itself. I have used AI throughout the development of determinIQ.

Not as a replacement for product judgement. Not as a substitute for architecture. And not as what is often called “vibe coding”. I have used it as a force multiplier.

I have been working in software and digital product development for more than 30 years. That experience matters because AI becomes far more useful when there is already a strong understanding of architecture, product design, user behaviour, delivery and the consequences of technical decisions.

Over the course of the project I have progressively built up a detailed AI working context around determinIQ. That context includes the product architecture, terminology, design rules, analytical lifecycle, governance model, UX principles, current implementation and previous decisions.

The AI has not been trained on the product in the machine-learning sense. Instead, it has become a context-aware development partner.

- product analyst
- software engineer
- architecture reviewer
- UX reviewer
- QA partner
- test analyst
- research assistant
- technical writer
- documentation support
- release assistant

In effect, I have been able to turn AI into something closer to a multidisciplinary product team working under my direction.

The distinction is important. I define the product. I set the architecture. I decide what belongs and what does not. I decide whether something is acceptable. AI expands the amount of work I can do around those decisions.

Human-led, not AI-led

The development process has always remained human-led.

Human direction → AI-assisted execution → human review → test → correction → repeat

rather than:

Prompt → code → accept

AI can produce plausible code, plausible explanations and plausible solutions that are wrong. I never treated plausibility as proof. The actual product had to demonstrate that the work was correct.

> *I did not use AI to avoid doing the thinking. I used AI to increase how much thinking, building, testing and checking I could do.*

That is the real force multiplier.

Recursive development

The way determinIQ has been developed has been highly recursive. Changes are made, inspected, tested, challenged and then revisited.

A piece of code can work technically and still be wrong for the product. A screen can look correct and still produce the wrong interaction. A fix can solve one issue and introduce another. A feature can function in isolation and still conflict with the wider architecture.

Define → implement → inspect → test → challenge → correct → retest → document → repeat

The codebase is checked. The running interface is checked. Logs are checked. State is checked. Screenshots are reviewed. User journeys are repeated. Regression issues are traced. Documentation is updated.

Nothing is accepted simply because AI generated it. Nothing is accepted simply because I expected it to work. The product has to prove itself.

Convergence rather than accumulation

In the later stages, the emphasis moved from adding capability to making the product cohere.

> *Does this make the product more coherent?*

That often means removing something rather than adding something. Duplicated controls are removed. Competing paths are collapsed. Terminology is standardised. State is made clearer. Features that belong in a later phase are deliberately kept out.

The aim is convergence. A mature product should not expose every internal capability simply because it exists. It should present the smallest coherent set of paths needed to let the user succeed.

Checks and balances

One of the main weaknesses of generative AI is that it can sound convincing while being wrong. Rather than treating that as a reason not to use AI, I treated it as a constraint on the development process.

The answer was not to trust AI more. The answer was to build more checks around it.

That mirrors the philosophy of determinIQ itself. The product exists because I did not want users to trust an analytical answer simply because it had been expressed confidently. The development process follows the same rule.

> *Do not trust plausibility. Verify the result.*

One philosophy, two applications

There is a symmetry in the project that I did not set out to create, but which has become increasingly obvious.

Inside determinIQ: AI provides leverage, but not analytical authority.

During development: AI provides leverage, but not product authority.

In both cases, the same principles apply. Use AI where it is useful. Keep authority explicit. Preserve evidence. Verify the outcome. Do not confuse fluency with correctness.

What is unusual about the project

Looking back, there are several aspects of determinIQ which are unusual in combination.

- AI authority is treated as an architectural problem rather than just a prompting problem.
- Analytical truth is separated from narrative.
- Verification is part of the core product rather than an audit feature added later.
- A governed contract sits between natural-language intent and execution.
- A qualification process sits between technically valid results and findings that deserve attention.
- Ask / Explore addresses the problem of users not knowing what to ask.
- Summary can orient users without requiring a question at all.
- Insights provide a direct route to governed findings that have already qualified for attention.
- Change Intelligence provides a further route focused on understanding material change.
- Zero insights is a valid outcome rather than a failure to generate content.
- Attention is configurable without changing the underlying evidence.
- Business context is kept separate from analytical truth.
- Both analytical capability and domain interpretation are modular.
- Domain packs allow the same governed core to support very different bespoke business cases.
- Privacy boundaries are placed around AI rather than assuming raw data should be sent to it.
- PII-sensitive data can retain analytical value without unnecessarily exposing identity.
- Reports remain connected to governed evidence rather than becoming disconnected exports.
- Controlled sharing allows outputs to leave the application without immediately losing provenance.
- The final business decision remains with the human.
- The same interaction model is reused across analytics and support.
- Support becomes a source of product intelligence.
- Documentation derives from maintained source documentation rather than becoming an independent version of product truth.
- The commercial and operational control plane is developed alongside the analytical engine.
- Customer state is separated from software release state.
- The architecture is tested beyond demonstration-sized datasets.
- Accessibility and cognitive load are treated as architectural concerns.
- Convergence and removal of competing paths are part of the development method.
- AI is used as a context-aware product team while human judgement remains in control.
- UI and UX are treated as part of the trust model, with progressive disclosure, preserved state and deliberate reduction of competing paths.
- A coherent product identity is carried across the application, reports, documentation, support and customer and partner environments.

None of those ideas is unique in isolation. What I think is different is how they fit together.

What I set out to achieve

The project was never really about building another dashboard. I was trying to see whether several things that are often treated as trade-offs could exist together:

- Natural language and deterministic execution
- AI assistance and limited AI authority
- Simple answers and inspectable evidence
- Automated discovery and controlled attention
- Proactive orientation and user-led exploration
- Business context and stable analytical truth
- Privacy and useful AI
- Advanced analytics and non-specialist users
- A common analytical core and bespoke domain behaviour
- Analysis and governed communication
- Software change and preservation of customer state
- Documentation and a maintained source of truth
- Fast development and strong architectural control
- AI-assisted delivery and human-led product development
- Advanced capability and low cognitive load
- Different product environments and one coherent identity

The result is my working answer to those questions.

Current position

determinIQ is now approaching the end of Phase 1. The work has moved from proving whether the architecture can work to making the whole product converge.

Ask, Explore, Summary, Insights, Change Intelligence, Verification, Ledger, Reports, Personal Support, customer administration, partner controls, documentation, the visual system and the operational layer are increasingly behaving as one product rather than separate features.

The full customer journey is becoming the test.

- Can the system be provisioned?
- Can the user enter it and understand where they are?
- Can they bring in data?
- Can determinIQ orient them?
- Can they enter safely through Summary, Ask, Explore, Insights or Change Intelligence?
- Can it surface meaningful findings?
- Can those findings be verified?
- Can they be turned into reports and shared?
- Can the user get help?
- Can the system be updated and recovered without compromising customer state?

If that whole journey works coherently, then the research architecture has become a product.

The original problem was simple: how do you use AI in analytics without asking the user to trust AI as the source of truth?

> *Use AI aggressively where it gives leverage. Keep truth, evidence, judgement and authority under control.*

The way I currently describe determinIQ is:
Trusted analytics, built on evidence. determinIQ proves it. AI explains it.

That is still the clearest description of what I set out to build.

determinIQ ANALYTICS

The commercial proposition was an outcome of the research, rather than its starting assumption.

determinIQ began as an investigation into whether AI could make sophisticated analytics easier to use without becoming the authority over analytical truth. Pursuing that question into a working system required research across analytical governance, evidence, privacy, contextual interpretation, human-computer interaction, accessibility and AI-assisted software development.

As those ideas were implemented, tested and refined, they converged into a coherent product architecture with practical commercial application. The resulting product therefore represents more than a potential route to commercialisation. It is working evidence of the research itself: an implementation through which the architectural principles, development methods and product decisions described in this paper can be examined and tested.

A governed approach to useful AI

The central idea behind determinIQ is not to remove AI from analytics. It is to put AI in the role where it is most useful, while keeping analytical truth, evidence and authority somewhere that can be checked.

> *AI provides leverage. Evidence provides confidence. Authority remains governed*

John Stutely, October 2026.