Simplifying Business Complexity and Maximising GenAI Investment value

How a Moonshot Sparked a Shift to 8x Faster Delivery

at RTL's Content Supply Chain - and what it means for the enterprise

Author: Giovanni Piccirilli

Reviewers: Ygor Geurts, Ingeborg Voetberg

RTL Technology - June 2026

Executive summary

Most enterprise GenAI investments are failing to deliver business outcomes. Organisations automate their existing complexity rather than simplifying it, and pilot projects never graduate to production. The result: significant technology spend with marginal returns.

RTL took a different approach. In Q2 2025, the Senior Leadership Team (SLT) sponsored a **"moonshot"** experiment in the Content Supply Chain domain: a 5-week, fully protected sprint with one objective: achieve a 10× acceleration in feature delivery. The team was freed from all other obligations. Failure was explicitly accepted as a possible outcome.

The initial result: a **3.5× improvement in delivery speed**, achieved without adding headcount. More importantly, the approach is now the **production operating model** for the domain, with the team simultaneously building a new product and a new platform capability. The transformation rested on two pillars: first, the way of working was redesigned from a sequential, specialist-driven handover model to a collaborative micro-team model; second, GenAI was introduced to perform the bulk of the mechanical coding work, freeing humans to focus on intent, quality, and business decisions. The experiment demonstrated that a small team can deliver what previously required a full multi-disciplinary team.

The most important lesson, however, is what happened next. The moonshot was not a one-off event but the catalyst for a continuous transformation. Its biggest legacy was cultural: the team became willing to change its own way of working again and again. Over the following months they kept going, eventually rebuilding their technology and infrastructure to put AI agents, rather than humans, at the centre of the work. **Today, measured against the pre-AI baseline, the team delivers 8x faster, with the average business idea reaching production within 24 hours**. Human work has shifted almost entirely from writing code to directing intent, orchestrating AI agents, and reviewing outcomes. This paper tells the full story: the moonshot that started it, the turning point that deepened it, and the operating model that now runs in production.

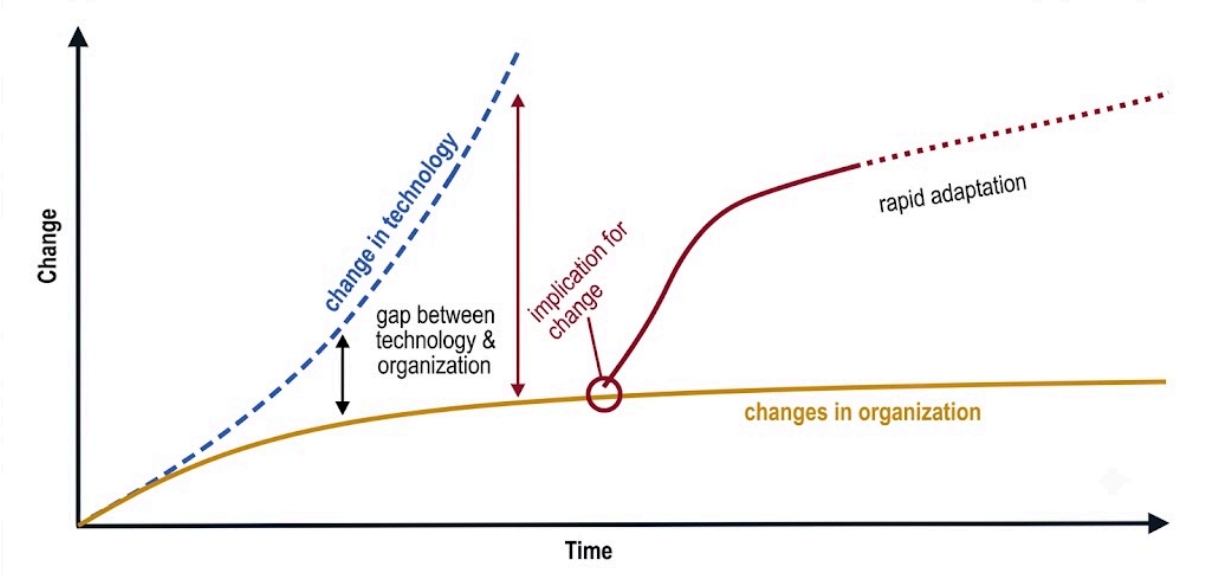
> *GenAI alone does not transform delivery. Combining AI tools with a fundamentally different operating model (fewer priorities, smaller work units, faster decisions, and changed roles) is what converts AI investment into measurable business value.*

Why this matters

The strategic problem

The media industry has undergone a fundamental shift from linear distribution to digital, on-demand delivery. Customer expectations for speed, i.e. faster content availability, shorter time-to-market, instant multi-platform distribution, are accelerating. Meanwhile, most media organisations still operate with supply chain processes and technology stacks designed for a slower era.

Scott Brinker's "Martec's Law" captures the challenge: technology changes exponentially, while organisations change logarithmically. This creates a widening gap that incremental improvement cannot close. The fastest organisations that close this gap gain a durable competitive advantage.



Martec's Law: the gap between technology capability and organisational change

The AI investment risk

Large organisations face a specific trap with GenAI: when AI is applied to automate existing processes, it replicates current complexity rather than simplifying it. The "as-is" state gets faster, but not fundamentally better. This leads to a pattern seen across industries: significant AI spend, pilot projects that never scale, and growing boardroom scepticism about returns on AI investment.

RTL's moonshot was designed to break this pattern: rather than automating what existed, the team was tasked with **rethinking how work is done**, using AI as an accelerator within a completely redesigned operating model.

Competitive implications for media

For a media company, the Content Supply Chain (ingest, metadata, subtitles, rights management, and distribution) is a core competitive capability. A 3.5 or even 8x improvement in feature or capability delivery speed is not an engineering metric; it is a **time-to-market advantage** that translates directly into viewer engagement, advertising revenue, and platform competitiveness.

Results at a Glance

Metric	Result	Relevance
Delivery speed	**8x faster**	Current state versus pre-AI baseline. The moonshot first delivered ~3.5×; continuous change took it further
Team size	**2–3 people**	Down from 10; fundamentally changes the cost model. Critical to achieve the higher speed
AI code contribution	**~100%**	Humans no longer write code; they direct intent, orchestrate agents, and review
Lead time, idea to production	**~24 hours**	Average ticket now reaches production in just under a day. The moonshot itself ran 5 weeks, plus 2 weeks of preparation
Target achieved	**8x**	Moonshot framing drove radical change; 10× set the ambition. The initial moonshot delivered ~3.5×; the scaled model built on it since reached 8x
Approach status	**In production**	Not a pilot anymore. This is now the standard way of working

A note on the numbers: the figures above describe the current production state, roughly a year after the moonshot. The moonshot itself, run in mid 2025, produced an immediate step change of about 3.5× (nearer 2 to 3× under the strictest measurement). What turned that into today's 8x was a sustained period of continuous change, described in the sections that follow.

What the team built

The team's mission was to build a hands-off content supply chain: one where content flows through ingest, enrichment, subtitling, rights, and distribution without manual intervention at each step. The intent was to free operations staff from routine processing so they could concentrate on judgment and the handling of exceptions, which is where human attention adds the most value. Everything the team built served that mission.

During the experiment and the production phase that followed, the team delivered a unified portal (internally named "Sherlock") for RTL's Content Supply Chain. This portal consolidates the end-to-end content journey, from ingest and metadata enrichment through subtitling, rights management, and distribution, into a single interface replacing workflows previously scattered across multiple legacy systems. Content operators have a single view of every asset's status and can act without switching between tools or waiting for updates from other teams. The platform reached production recently and its user base and feature set are still growing.

The team also built a new data delivery capability replacing a legacy integration layer, and delivered video marker insights for editorial and production teams. These are now production systems in daily use, not prototypes.

Critically, the approach was not to replace legacy applications one by one. The team assessed the overall application landscape end-to-end and designed a coherent new solution from scratch with AI as the development engine. Piecemeal replacement preserves the complexity of the old landscape; end-to-end redesign simplifies it. The 8x result was only possible because the team solved for the whole, not the parts.

What Senior Leadership did

This experiment succeeded because of Senior Leadership-level sponsorship, not despite it. The SLT's role was not to manage the work but to create the conditions in which a radical change could happen. Without this sponsorship, the experiment would have been diluted into incremental improvement. Here are some of the principles used:

Set an audacious target: the 10× moonshot was deliberately unreachable. Its purpose was to force the team to abandon incremental thinking and redesign the operating model from scratch.

Protect 100% of capacity: for 5 full weeks, the team was freed from all other deliverables, competing priorities, and ad-hoc requests. This is the single most important enabler: AI transformation cannot happen "on the side."

Accept the risk of failure: the SLT explicitly sanctioned that the experiment might produce no usable outcome. This gave the team psychological safety to try radical approaches.

Hold non-negotiables firm: quality, security, privacy, and compliance standards were never relaxed. Speed was achieved through a better model, not by cutting corners.

It is worth being precise about where the freedom applied: The licence to fail covered how the team worked, the roles, the process, and the tools it was free to reinvent. It never extended to production standards: quality, security, privacy, and compliance held throughout. The experiment was bold in method and conservative in what it allowed to reach customers, and that distinction is what made the risk acceptable.

Keep stakeholders informed: short demos and clear communication on what was prototype versus production maintained organisational trust.

Grant full decision-making authority: one of the main problems in large organisations is that decision-making is distributed across layers and committees. During the experiment, the team had full authority to make every decision needed to deliver (technical, process, and stakeholder-facing) without escalation. This removal of decision latency was a critical accelerator.

> *The SLT's role in AI transformation is to set the ambition, protect the space, accept the risk, and hold the guardrails. The operational model change is for the team to design and execute.*

The delivery problem in most enterprises vs the Moonshot approach

Before describing what changed, it is important to name the problem clearly. The delivery challenges RTL faced are not unique: they are endemic across large enterprises. Most organisations suffer from the same structural inefficiencies.

High lead times

Implementing and releasing even modest changes takes weeks or months. Work passes through sequential stages, analysis, architecture review, development, testing, deployment, with waiting time between each handover often exceeding the actual working time. For the SLT, this means strategic initiatives take quarters to materialise, and market windows close before products reach customers.

Excessive non-productive work

A significant share of team capacity is consumed by refinement sessions, status meetings, cross-team alignment, architecture reviews, and handover documentation, activities that coordinate work rather than deliver it. In many organisations, less than half of engineering capacity translates into delivered features. The rest is overhead. For the SLT, this represents a hidden cost: teams appear fully utilised, but actual output is a fraction of what the investment should deliver.

Complex integrations and dependencies

Cross-system changes consume disproportionate effort and create dependencies on external teams and vendors. Each integration point adds coordination cost, risk, and delay. As systems grow more interconnected, this problem compounds, slowing even well-funded teams to a crawl.

What the moonshot fixed

The moonshot operating model was designed to systematically attack each of these inefficiencies. High lead times were eliminated by replacing sequential handovers with whole-team, simultaneous delivery. Non-productive overhead was eliminated by replacing meetings and documentation with real-time collaboration where decisions happen as part of the work itself. Integration complexity remains a constraint, but internal delivery speed has been radically improved. The 8x result is, in essence, the measurable outcome of removing enterprise delivery dysfunction from one product domain.

The shortfall against the original 10x ambition had concrete causes. Legacy systems that were not retired continued to constrain what the team could change quickly, and some infrastructure steps were still manual despite established automation practices. The largest remaining constraint, integration with external systems, is addressed in its own section below.

What changed — The Operating Model shift

The improvement came from fundamentally changing how work is organised, decided, and delivered having AI as the accelerator; the operating model was the transformation.

From complex teams to micro-teams

Teams moved from a 10-person unit with specialist handovers to micro-teams of 2–3 people each owning end-to-end slices. A **subject matter expert and a software developer, working together with AI tools, could now accomplish what previously required a full multi-disciplinary team.** This fundamentally changes the economics and structure of technology delivery across the enterprise.

This shift also redefines the profile of people needed. The traditional enterprise model relies on **deep vertical specialists**: a business analyst writes requirements, a developer writes code, an architect reviews design, a tester validates etc…, each operating in their own lane and handing work to the next. AI-enabled micro-teams require a fundamentally different profile: **T-shaped people** who combine deep expertise in one (or more) area with the ability to contribute meaningfully across the full delivery chain. When a subject matter expert can prompt AI to generate code and validate the output alongside a developer, the boundary between “business” and “technology” dissolves. For the SLT, this has direct implications for hiring strategy, talent development, and role design: the organisation should invest in building T-shaped capability at scale, because the teams that move fastest are the ones where every member can contribute across the end-to-end slice, not just within a functional silo.

How the team was composed

The shift to micro-teams was not planned from the outset. It emerged from what the teams observed during the experiment itself. The teams started as single units of 10 people each with varied specialisations: product owner, lead developer, business analyst, etc... As GenAI accelerated the speed of product design and development, an imbalance quickly became visible. Some team members were working at full intensity while others found themselves waiting with nothing to contribute, because the AI tooling was absorbing much of the detailed work that had previously justified their individual roles. At the same time, the size of the group became a coordination burden. Ten people meant dozens of communication lines, and the overhead of keeping everyone aligned was actively slowing the team down rather than helping it. What emerged alongside the problem, however, was also the solution. Because AI handled the mechanical and specialist work, the remaining human contribution was increasingly about intent, judgement, and validation. Anyone in the team could participate meaningfully in designing and building the product, regardless of their original job title. This combination of roles being absorbed by AI, coordination costs rising with team size, and the realisation that every team member could contribute across the full delivery chain, is what pushed the teams to break into multiple micro-teams of 2–3 people, each owning an end-to-end slice. A product/tech lead and a software developer, both deeply Agentic AI-savvy, could now accomplish what previously required the full multi-disciplinary unit. What began as a response to friction within one large team became the blueprint for how the domain now organises itself.

Mindset

Mindset and cognitive diversity mattered as much as technical skill. Most team members were shaped by years of traditional delivery: ticket-driven, individually accountable, and instinctively cautious about output they had not personally crafted. Not all were ready for the experiment. Several felt, at the outset, that their professional purpose, the craft they had spent years developing and genuinely loved, was being challenged, perhaps even made obsolete. That concern was real and had to be acknowledged openly, not dismissed. What made the difference was that they were willing to try and see what happened. That willingness, combined with cognitive diversity (people with genuinely different mental models and professional backgrounds) proved essential. A team that thinks alike will prompt alike, accept the same AI outputs, and share the same blind spots; the most valuable contributor was often the one who found the AI’s answer plausible-looking but instinctively wrong. The micro-teams were led by an individual with both technical and business understanding, well respected by stakeholders, who could make rapid trade-off decisions and maintain coherence across the work.

From handovers to whole-team delivery

The sequential handover model (analysis → design → build → review) was replaced by whole-team sessions where decisions and delivery happened simultaneously. AI drafted code and prototypes in real time while humans directed intent, set constraints, and reviewed quality.

From sprint batching to continuous stakeholder feedback

Instead of waiting for sprint reviews, stakeholders saw working prototypes continuously. This compressed the feedback cycle from weeks to hours, ensuring the team built the right thing faster.

Dimension	Before	After
Team structure	10-person unit	Micro-teams of 2–3 people
Delivery cycle	2-week sprints	Continuous flow, small slices
Code authorship	100% human-written	~95% AI-generated in the model following the initial moonshot; ~100% in the scaled model
Stakeholder feedback	End of sprint	Continuous, via prototypes
Handover model	Analysis → Build → Review	Whole-team, simultaneous

How the model evolved after the moonshot

The moonshot was the beginning of the story, not the end. Its most durable outcome was not the immediate speed gain but a change in mindset: the team became comfortable changing its own way of working, and kept doing so. What followed was a year of continuous change that took delivery far beyond the original result. It is worth describing that journey honestly, because the path matters as much as the destination.

From moonshot to continuous change

After the moonshot, the team did not stand still. Roles kept blending, the team kept shrinking, and as new and more capable AI models arrived, the team folded them into the way it worked. They experimented with approaches such as specification-driven development to put even more weight on clear intent. This was a period of steady, incremental improvement, made possible by the cultural shift the moonshot had created: process changes that once took weeks of debate could now be tried in a day.

The turning point: we are the problem, not the technology

Around the turn of the year, a single meeting changed the trajectory. The team had gathered to discuss a handful of bugs that had lingered on the backlog for a long time, to the growing frustration of the business. While the group debated how to prioritise and resolve them, one engineer quietly assigned the bugs to AI agents and let them work in the background. By the end of the meeting, while the discussion was still going, the bugs were fixed.

The realisation was uncomfortable and liberating in equal measure: **the team itself, not the technology, had become the bottleneck.** The process was still built around people, around human discussion, human scheduling, and human hand-offs, when the technology had already moved on. This insight, combined with a step change in the capability of background agents, prompted a far more radical move.

Rebuilding the operating model, and the technology, around AI agents

The team decided to redesign not only the process but the technology itself around AI. Until then, the technology and infrastructure had been organised for human convenience. The team rebuilt both to be agent-first: suited primarily to the AI agent rather than to the individual contributor. In practice this meant consolidating many separate code repositories into a single one, introducing clear vertical product slices with well-defined boundaries, and building an internal platform where one or more agents could operate on their own.

The shift in human posture was just as important. The team moved from "**babysitting"** the AI, watching over each step, to **orchestrating** it: setting up the work, launching agents, and reviewing what came back. This was the change that unlocked the largest gains.

A day in the new operating model

The work now runs on a simple division of labour. People decide what to build and describe it in plain business terms; AI agents build it, test it, and prepare it for release, working in the background and through the evening. Humans set the intent and review the outcome; they no longer write the code. Routine checks are largely automated, so human attention is reserved for the decisions that carry real risk, such as changes that affect the wider architecture.

The effect on pace is striking: an idea now reaches production within 24 hours, against weeks under the old model. Each team also keeps a shared memory of its decisions and context that is fed automatically to the agents, so they build on the team's accumulated knowledge rather than starting from scratch each time. The full technical detail is in the appendix for readers who want it.

What this produced

A careful internal analysis compared how much output the team delivers now against the moonshot period and the pre-AI baseline. To keep the comparison fair, the team used AI to size each piece of work as small, medium, or large from its description, and confirmed that the mix of sizes was similar across periods. On that basis, the team now delivers 8x more than before AI. External validation has followed: a major hyperscaler now recommends a similar approach to other organisations, though a less far-reaching version than the one RTL runs.

Because the teams are small, the freed-up capacity was redeployed rather than removed. The same group now builds several products in parallel and has extended the model beyond its original domain into marketing, a meaningful proof point: it shows the approach is not specific to the Content Supply Chain but transfers to a different problem space. Further domains are next. The opportunity ahead is to internalise work currently handled by external vendors, reducing cost over time.

The role of GenAI in the new Operating Model

GenAI was not a peripheral add-on to the moonshot. It was the engine that made the new operating model viable. Without AI handling the mechanical bulk of coding and prototyping, the initial 3.5× acceleration would not have been achievable regardless of how the team was reorganised and the same clearly applies to the 8x further acceleration. At the same time, AI

without the operating model change would have produced only marginal gains, as is the case in most enterprise pilots. The two are inseparable: the operating model creates the conditions for AI to be effective, and AI makes the operating model sustainable at speed.

During the initial moonshot effort, AI performed roughly 95% of the code generation. In the current model it performs essentially all of it. Humans direct intent through natural-language prompts, orchestrate the agents that build and deploy the work, review the outcomes using as reference the enterprise engineering guidelines, set architectural constraints, and make business trade-off decisions. The AI's role is to draft rapidly and explore multiple options in minutes rather than days, eliminating the mechanical bottleneck that has historically dominated software delivery. This shifted the human role from "writing code" to "directing, orchestrating, validating, and deciding", a fundamentally different and higher-leverage use of human expertise.

Regarding tooling, each team was given access to standard enterprise AI tools but also the freedom to identify and select the best tools for their specific goals. Once chosen, these tools were quickly assessed for fitness before any enterprise-specific data was introduced. This balance of freedom to innovate within a governance framework is a model the SLT should consider replicating as AI adoption scales across the organisation.

People and collaboration outcomes

Beyond the delivery speed improvement, the moonshot produced measurable changes in the way people work together, changes that matter to the SLT because they determine whether the model is sustainable and scalable.

Faster alignment and fewer decision bottlenecks

Questions that previously bounced across multiple meetings and email chains were resolved in-session, in minutes rather than days. With the whole team present, decisions were made once, with full context, and acted on immediately. For an organisation accustomed to multi-week decision cycles, this represents a step-change in organisational responsiveness.

Shared understanding and reduced key-person risk

Because work happened collectively, more people understood both the "why" (business intent) and the "how" (technical implementation). This directly reduces single points of failure (or single points of knowledge), the "hero" or "only person who knows how it works" problem that creates operational risk in most organisations. For the SLT, this means greater resilience and lower continuity risk across critical product domains.

Stronger retention

An outcome that matters to leadership is retention. Once the team had worked this way, people did not want to go back, and they did not want to leave for organisations that still work in the traditional manner. Engineers who watched a conventional, code-by-hand demonstration described it as travelling back in time. The work became more engaging rather than less, because people spend their time on intent, design, and judgment rather than routine production. For a critical product domain, this lowers the risk and cost of losing key people.

Amplified creativity and broader skill development

AI tools amplified the team's creative output across product development, process design, and solution engineering. Because AI handled the mechanical work of coding, team members contributed to dimensions previously outside their role: product managers explored technical options, engineers shaped business requirements, and analysts validated prototypes directly. This skill broadening reinforces the shift toward T-shaped profiles (professionals who possess deep expertise in one specialized field combined with broad knowledge and collaboration skills across multiple disciplines) and increases the versatility of the workforce.

Lighter portfolio management

At this delivery speed, **backlog management and prioritisation become significantly less burdensome** because the teams can address almost all demand. Resource scarcity, the traditional constraint driving painful prioritisation debates, is no longer the binding limitation. Instead, the constraint shifts to external system integration speed, which is where the next wave of improvement should focus.

Sustainability and intensity

The main risk for people is intensity. Whole-team sessions are cognitively demanding, particularly in remote or hybrid settings. The team mitigated this through structured breaks, role rotation, and deliberate attention to psychological safety. The SLT should treat sustainable pace as a non-negotiable as this model scales, as speed gained through burnout is not durable.

Following initial scepticism from some individuals at both management and team level, once the experiment was running and results became visible, the whole team embraced the approach. This shift in belief from doubt to conviction required active, sustained sponsorship from both the SLT and senior management. It is a reminder that transformation is as much a psychological journey as an operational one.

Strategic implications for the enterprise

Workforce and cost model

If micro-teams of 2–3 people can replace teams of 10, the implications for workforce planning, cost structure, and organisational design are significant. This does not mean eliminating roles immediately. It means **redeploying capacity to higher-value work** and **doing more with the same investment**. At scale, this changes the economics of build-vs-buy decisions, reduces dependency on external vendor roadmaps, and enables faster internal response to market needs.

An 8x improvement in delivery speed may appear to imply direct cost reduction, but the reality is more nuanced. In this initial phase, the acceleration and higher efficiency in resource usage were reinvested: the freed-up capacity was redirected to feed the teams with a more efficient delivery mechanism and additional resources that support ongoing operations, rather than cutting headcount. At this stage, the most responsible approach was strengthening the model and building organisational confidence. However, if this approach is scaled across a broader section of the enterprise, genuine efficiencies and cost reductions

can be claimed in the future as the organisation matures its ability to operate with smaller, faster teams at scale.

It is worth being clear about where the financial return ultimately comes from. The engineering efficiency gain was reinvested rather than banked, but the automated supply chain reduces the routine operational work that currently requires people, and over time this will lower operational headcount and/or increase scale (so be able to do more with the same people). That, together with the phasing out of external vendors as capabilities are brought in-house, is where the measurable savings will be realised. This is an honest point for leadership to hold: part of the return falls on operational roles whose work is being automated, where scale is also able to dramatically increase, which is - in both cases - a transition to manage with care.

Build vs. buy recalibration

With dramatically faster internal delivery, the cost curve shifts. Capabilities that were previously outsourced or purchased as SaaS may be more efficiently built in-house, with the added benefit of full customisation and IP ownership. The SLT should expect build-vs-buy assessments to be revisited as this model scales.

Measurement and value realisation

Faster delivery is only valuable if the organisation can measure whether it creates the intended business impact. As teams ship more, the need for robust product KPIs, outcome-based investment criteria, and mechanisms to stop low-value work becomes critical. The SLT should prioritise investment in measurement capability alongside delivery capability.

Scaling beyond one domain

This approach has been proven initially in one domain and since extended to others, including marketing. Scaling it across the organisation requires:

Senior Leadership Team mandate: teams need protected focus time, clear decision rights, and a consistent risk appetite. Without executive air cover, operational pressure will pull teams back to old patterns.

Talent and culture investment: not everyone will adapt at the same pace. The organisation needs to invest in upskilling, build communities of practice, and make difficult talent decisions where individuals actively block the change.

Governance and responsible AI: clear policies on data handling, AI-generated output, quality standards, and accountability must be in place before scaling. These should be enabling, and avoid being bureaucratic.

Bottom-up and top-down alignment: grass-roots enablement (tools, training, examples) must be matched by strategic direction (vision, bold goals, visible leadership role-modelling).

Experience since the moonshot has sharpened where this scales easily and where it does not. For other product teams, the recipe transfers directly: the same operating model can be applied with confidence. The binding condition is leadership sponsorship. Where a senior leader protects the space and backs the change, it works; where that sponsorship is absent, it stalls, regardless of the size or talent of the team. This is the single most reliable predictor of success or failure when extending the approach.

Functions without their own product and engineering capability, such as finance, are a different case, and it would be a mistake to assume their operating model can be made “agent-first” in the same way. The realistic route there is twofold: revisit the vendor systems and tools those functions depend on, and stand up an engineering capability that builds the automation around them. In other words, the product engineering model described here is the means by which automation reaches the wider operation rather than a template that every department adopts unchanged. The honest position is that the team has proven this for product engineering and believes it extends to the operations those teams support, but does not yet claim proof for every corner of the enterprise.

The recurring obstacle is not technical. It is the tension between the urgency of running the operation and the willingness to pause in order to go faster later. Resolving that tension is a decision only senior leadership can make, which is why this remains, above all, a change-management challenge rather than a technology one.

External systems as the next bottleneck

One clear finding was that internal delivery now outpaces the external systems it integrates with. This was one of the main reasons the moonshot reached 3.5x rather than 10x, and it remains the primary constraint today. Vendor integrations hold things back where partner

systems lack usable APIs or do not expose the necessary data, and legacy partner systems and cross-domain dependencies compound the problem. Assessment to renegotiate integration approaches and SLAs with external parties, or to invest in decoupling strategies are crucial.

Experience pointed to three things an organisation must control to automate a chain end-to-end. It must own the data, so that information is available wherever an agent needs it. It must own the integration surface, or at least be able to integrate fully with the systems in the chain. And it must own the business-process interface, meaning the ability to drive the process itself rather than only read its data. This last point is the one most often overlooked, because in many vendor products the process is locked behind a user interface and only the data is exposed through APIs.

Risks and Guardrails

The following risks should be considered as this model scales:

Quality risk: AI-generated code can introduce subtle errors if accepted without mandatory human review, automated testing, and clear ownership for production code. In the current model, quality is managed through a combination of automated AI validation and human oversight: agents generate and check the work, including building and running tests, and humans still review the outcomes that carry the most risk. This dual layer is what allows the team to move at speed without relying on traditional test coverage alone. Review is now the main constraint, because as output rises people can no longer read every change at this volume, and the team is progressively shifting review to agents that check the work of other agents so that human attention concentrates on the changes with real architectural or business impact. The intent is to let AI close the loop as far as possible, with agents that continuously analyse production logs, trace each error back to the change that caused it, and fix it, while user-reported issues are fed back for the agents to resolve.

IP ownership risk: ownership of AI-generated software is legally unsettled and enterprises cannot ignore it. AI-generated code is only protectable where a human has made sufficient creative decisions: a developer who prompts, accepts, and ships without meaningful review may leave the enterprise with no IP claim over what was built. Speed and ownership must be managed as a conscious trade-off, with review processes designed to establish the human creative contribution required for IP protection.

Responsible AI risk: sensitive data exposure, compliance gaps, or loss of accountability require clear data-handling policies, synthetic data for AI prompts, and human accountability for all decisions.

People risk: cognitive intensity of whole-team sessions, potential exclusion of quieter contributors, and over-reliance on AI weakening foundational skills. Sustainable pace demands structured breaks, role rotation, mentoring, and explicit learning moments.

Speed-over-value risk: delivering faster without measuring outcomes. The organisation needs robust product KPIs and mechanisms to stop low-value initiatives quickly.

Change resistance: managers and individuals who are not ready for radical change can slow or block adoption. Scaling requires active leadership sponsorship, clear communication, and willingness to make tough talent decisions.

> *Quality, security, privacy, and compliance are never traded for speed. The operating model change delivers speed through better ways of working, not reduced standards.*

Where to start: Choosing the right processes

The highest-leverage entry point for this approach is not greenfield projects or low-risk experiments, but broken back-office workflows: the processes already on fire, with clear pain, defined failure, and an audience desperate for relief. These are the ideal conditions precisely because the current state is full of workarounds and complaints that requirements write themselves. Speed is safe here: the floor is already low, the blast radius is internal, and any working replacement is an immediate win.

Organisations that move first on these burning platforms will not only fix what is broken faster than traditional development ever could, but will build the internal champions, the muscle memory, and the organisational confidence to scale this way of working across the enterprise. The RTL Content Supply Chain was exactly this kind of target: a domain with visible pain, urgent demand, and stakeholders ready to embrace a different approach. Selecting the next domains should follow the same logic: look for the processes where the cost of the status quo is highest and the appetite for change is already present.

Recommendations for the Senior Leadership team (SLT)

Endorse scaling: extend the moonshot operating model to additional product domains, starting with the areas of highest strategic value and business impact.

Protect capacity for transformation: mandate that teams pursuing AI-driven operating model change are given dedicated, protected time rather than asked to transform "on the side." This was the single biggest success factor.

Invest in measurement: ensure product-level KPIs and business outcome tracking are in place to validate that faster delivery translates to competitive advantage.

Set AI governance standards: establish enterprise-wide policies for responsible AI use that enable speed while protecting quality, data, and compliance.

Address talent and culture: invest in upskilling at scale, build communities of practice, and support leaders in making the difficult talent decisions needed to sustain the transformation.

Communicate the vision: use this result as a proof point to build organisational confidence in AI-driven operating model change, and to set expectations for what "good" looks like.

> *AI adoption can be a durable competitive differentiator for RTL only if it is treated as an operating model transformation, not a technology rollout.*

Conclusion

The Content Supply Chain moonshot demonstrates that GenAI investment delivers measurable business value when combined with a fundamentally different operating model. The moonshot's immediate 3.5× improvement was a production result, and the continuous change it set in motion has since taken delivery to 8x the pre-AI baseline, which is now the standard way of working for this domain. Two changes made this possible: the way of working was transformed from a sequential, specialist-driven model to a collaborative, small-team approach where decisions and delivery happen together; and GenAI was embedded as the primary engine for the mechanical work of building software, allowing humans to operate at a higher level of intent, orchestration, quality assurance, and business judgement.

The experiment also proved a more profound point: **the traditional multi-disciplinary team structure is no longer the only viable model.** A subject matter expert and a developer, working with AI tools, can deliver outcomes that previously required a much larger team. For an SLT overseeing workforce strategy, cost optimisation, and competitive positioning, this finding has implications far beyond one product domain.

Moonshots are enabled at the top. The SLT set the ambition, protected the space, accepted the risk, and held the guardrails. That sponsorship was the essential condition for success. As RTL considers scaling this approach, the same leadership commitment will be required, along with investment in measurement, governance, and talent, to turn a single experiment into an enterprise-wide competitive advantage.

Implications for organisations

For any organisation reading this, regardless of industry, the question is not whether GenAI will change how work is delivered. It is whether you will lead that change or follow.

Identify one domain where the pain is visible and stakeholders are ready for something different. Protect a small team fully for a fixed period. Give them decision authority, AI tooling, and permission to fail. Set a target ambitious enough to force a redesign of the operating model, not an optimisation of the existing one. When scoping the work, resist replacing applications one by one; assess the end-to-end landscape and build a simplified architecture with AI rather than replicating old complexity faster.

If the results confirm what RTL experienced, scale immediately: select the next domains, invest in measurement that links speed to business outcomes, and start the talent and culture work that determines whether the change lasts.

“Computers are no longer just tools: they are becoming members of the organization.”
Thomas W. Malone, MIT Sloan School of Management

Summary Infographic

Metric	Legacy Operating Model	Agent-First Operating Model
Delivery Speed	Baseline (1x)	**8.5x Faster**
Team Size	10-person specialist unit	**2-3 person micro-teams**
Code Authorship	100% Human-written	**~100% AI-generated**
Lead Time	Weeks/Months	**< 24 Hours**
Feedback Loop	End of 2-week sprint	**Continuous via prototypes**

Appendix: Operating Model detail

This appendix provides additional operational detail for readers who wish to understand the execution methods used.

A.1 Mob programming and vibe coding (agentic engineering)

Mob Programming is a whole-team way of working, pioneered by Woody Zuill, where everyone focuses on the same item together. One person drives the keyboard while the rest guide, review, and decide in real time. "Vibe coding," now more commonly referred to as Agentic Engineering, is a prompt-driven development style popularised by Andrej Karpathy in 2025: people describe intent in natural language and an AI coding assistant drafts the code. Combined with mob programming, these methods let the team move fast while maintaining shared understanding and quality.

A.2 Agentic mob programming loop

Align on outcome: clarify user value and what “good” looks like.

Choose a small slice: the smallest end-to-end piece that can be delivered.

Let the AI draft: generate an initial solution quickly with the coding agent.

Review together: validate quality and risks; iterate fast.

Deliver and learn: release safely, gather feedback, repeat.

A.3 Flow principles

The approach was informed by DevOps Research and Assessment (DORA) findings on delivery performance, Brinker’s Martec’s Law on the technology-organisation gap, and Alistair Cockburn’s “Elephant Carpaccio” exercise on slicing work into thin, shippable increments. The core principle: reduce parallel work, deliver in the smallest possible end-to-end slices, and tighten feedback loops to hours rather than weeks.

A.4 Role changes in detail

Product owner: shifted from managing tickets to shaping outcome slices, constraints, and acceptance criteria in-session.

Software engineers: shifted from writing code to reviewing AI output, integrating changes, and ensuring production readiness.

Business analyst + system analyst: merged into a “product engineering” function: intent, prompt, and validation, instead of writing handover documents.

Domain architect: shifted from a separate approval step to an in-session constraint setter and coach.

Application manager: shifted from after-the-fact support to production readiness, monitoring, and controlled cut-over.

Chapter lead: focused on protecting capacity, reducing parallel work, removing blockers, and maintaining a sustainable pace.

In the months after the moonshot these changes went further. The product manager and engineer roles blended to the point where both now specify work and create tickets in the same way, and purely coordinating roles such as scrum master and business analyst were

no longer needed. The engineer's identity shifted most of all: no longer someone who builds technology, but someone who directs it in product language, deciding what is needed and orchestrating the agents that build it. Product managers and engineers work very closely, leaning on each other as the engineers grow into the product side of the work.

A.5 Change readiness

Not all employees and managers were ready to embrace a radical change. It took weeks of discussion to align on the core hypothesis. Inside the team, readiness varied due to different concerns (risk, credibility, loss of control, fear of failure) and it had to be handled through repeated explanation, involvement in decisions, practical coaching, and creating safety to try. Once results became visible, the whole team embraced the approach. This "psychological work" of building belief, safety, and commitment was a fundamental part of making the change succeed.

A.6 Quality guardrails

Review by at least two people, basic testing where feasible, and monitoring for new behaviour.

Security and compliance checks: redacted examples and synthetic data for AI prompts; no sensitive data in external tools.

Explicit cut-over from prototype to product: production code requires readability, tests, and clear ownership.

Retrospectives focused on flow and well-being: sustainable pace treated as a requirement, not an afterthought.

A.7 Media industry context and references

The strategic urgency described in this paper is supported by well-documented trends in the media industry. The shift from linear broadcast to digital, on-demand distribution has fundamentally altered the competitive landscape. Global OTT (over-the-top) revenue has grown at double-digit rates annually, with platforms such as Netflix, Disney+, Amazon Prime Video, and regional players driving a fragmented, hyper-competitive market. Traditional broadcasters face simultaneous pressure from declining linear advertising revenue and the need to invest in streaming infrastructure, content personalisation, and multi-platform distribution.

Industry analyses from PwC's Global Entertainment & Media Outlook, Deloitte's Digital Media Trends, and the European Audiovisual Observatory consistently highlight that the speed of content delivery, from production to multi-platform availability, is a critical differentiator. Audiences expect instant availability across devices, and the window between content creation and monetisation continues to shrink. The content supply chain is no longer a back-office function; it is a front-line competitive capability.

The DPP (https://www.thedpp.com/) has published extensively on how AI is reshaping media workflows, from automated metadata generation to intelligent content routing. Their reports emphasise that media companies which fail to modernise their supply chains risk being outpaced not only by digital-native competitors but also by traditional peers who invest earlier in AI-driven operations. For RTL, operating in one of Europe's most competitive media markets, these trends make the case for the moonshot approach not just compelling but urgent.

A.8 The agent-first technical setup

As the model matured, the team rebuilt its technology specifically so that AI agents, rather than people, could operate it. Several choices made this possible. Many separate code repositories were consolidated into a single one, which agents handle more reliably. The architecture was reorganised into clear vertical slices with well-defined boundaries, so that a change normally stays within its own slice and any spillover is an immediate signal that something needs human attention. The team built an internal platform on which an agent can provision a complete environment on its own, including database, routing, and supporting services, and created a fresh, temporary environment for every change so work can be previewed in isolation.

Work is tracked in a modern tool chosen partly because it integrates directly with the AI coding assistant and supports hosted background agents that keep working unattended. In practice, the team used Cursor as the AI coding assistant and Lovable for UI prototyping. A team member describes a feature in conversation with the assistant, which creates a ticket; a background agent then picks it up and builds it. Routine code review is largely automated, with most changes passing cleanly, and a dedicated agent scans for security vulnerabilities and fixes them, keeping time-to-fix very short. Human review concentrates on the changes that matter most, and the team is building an agent that flags tickets with potentially high architectural impact so a person looks at those specifically.

Finally, each small team maintains a shared knowledge base, a collective memory into which meeting notes and context are continuously added. This memory is automatically supplied as context to every new ticket, so agents work with the team's accumulated understanding rather than starting from scratch each time. The guiding principle throughout was to move humans from babysitting the AI, supervising each step, to orchestrating it, setting up and directing the work and reviewing the result.