A Function Paper from GRAIL

What does an AI-augmented engineering organisation look like?

Engineering organisation

Where software development and R&D engineering are heading in the next two to three years, and what GRAIL believes it takes to get there first.

"Execution runs in parallel. Judgment stays with the engineer."

01 Is your engineering bottleneck code, or the decisions behind it?

02 What breaks when agents reveal weaknesses already in your engineering?

03 Who is teaching your future engineering leaders to exercise judgment?

WHAT DOES AN AI-AUGMENTED ENGINEERING ORGANISATION LOOK LIKE?

Where software development and R&D engineering are heading in the next two to three years, and what GRAIL believes it takes to get there first.

Implementation is becoming abundant. Clear intent, sound architecture and acceptance evidence are not. **The change that matters is that engineers stop carrying every task from first draft to finished change and start supervising several bounded execution threads.** The agent produces and checks. The engineer defines, judges and remains accountable.

THE ENGINEER'S DAY, 2028

01
BRIEF

Combining overnight incidents, customer signals, dependency alerts, failed pipelines.

02
SPECIFY

Inspect the relevant code, propose a plan.

03
RUN

Create branches, write tests and prepare small pull requests.

04
REVIEW

Challenge correctness, maintainability, security and missing documentation.

05
RELEASE

Check tests, dependencies, secrets and licences.

Execution runs in parallel. Judgment stays with the engineer.

GRAIL's view of the engineering function ahead. A position, not a survey.

THE FUNCTION TODAY

Engineering capacity disappears into the delivery *system*.

A European mid-market engineering function may sit inside an industrial company, a software product company or a combined hardware, firmware and software organisation. It normally includes a CTO or Head of Engineering, engineering managers, technical leads, software or embedded engineers, test and QA specialists, and some platform, DevOps or security capacity. Architects often remain senior individual contributors. External developers and nearshore teams may supply a substantial share of delivery capacity.

The function affects revenue through time to market, product capability and reliability. It also determines infrastructure spend, warranty work, defects and technical debt. Its performance cannot be read from speed alone. The scorecard combines roadmap predictability, change lead time, deployment frequency, failed deployment recovery time, change failure rate, escaped defects, availability, security findings, test coverage and backlog age. Industrial teams add verification, certification, prototype and engineering-change measures.

Eight core processes

01	Discovery, specification, backlog and planning	05	Release and deployment
02	Architecture and technical decisions	06	Incidents, maintenance and technical debt
03	Coding, code review and documentation	07	Hardware and embedded R&D
04	Testing, QA, security and dependencies	08	Vendors and nearshore coordination

Where the information lives is the problem. Repositories, issue trackers, pipelines, documentation, chat, observability, customer support and spreadsheets each hold part of the engineering record. Industrial teams add requirements systems, CAD, PLM, simulation, lab results and compliance files. The rationale for critical choices still sits disproportionately in senior engineers' memories.

The visible task is writing code. The real constraint is turning intent into a small, tested, secure and supportable change.

A faster first draft does not create a faster engineering function when specification, review, testing or release cannot absorb it. Throughput and instability have to be managed together.

WHAT CHANGES STRUCTURALLY

Implementation stops being the scarce *unit*.

Two structural shifts redefine the function.

Engineers move from producing every first draft to supervising execution. They define intent, constraints and acceptance evidence, then direct several bounded threads of work. Agents investigate code, propose plans, prepare changes and run checks. The engineer remains accountable for architecture, material review and the decision to merge. More output follows only where specifications, tests and release controls are already sound.

The business can prove desirability before engineering commits to production. Product, sales, finance and operations can turn a customer problem into a disposable prototype inside an approved sandbox. Engineering receives observed behaviour, feedback and a runnable artefact rather than a long requirements document. The boundary stays clear. The business can test the idea. Engineering retains authority over architecture, production data, security, support and release.

The prize is not a longer feature list. It is a larger portfolio of tested options, with engineering capacity moving toward the products, reliability work and customer problems that deserve it.

> WHERE IT IS WEAKEST the combination is weakest in poorly specified products, legacy estates and safety-critical systems. More generated code can enlarge the review queue and technical debt faster than validated delivery improves.

THE 2028 OPERATING PICTURE

A day, a week, a *month*.

A function that invests now still employs engineers in 2028. Its shape changes. Fewer roles centre only on ticket implementation. More capacity sits in platform engineering, test architecture, security, systems design and product-minded generalists. Manual QA gives way to quality engineering. Junior intake remains deliberate because the function still needs experienced engineers in 2031.

The day

MORNING

Each team receives a generated brief combining overnight incidents, customer signals, dependency alerts, failed pipelines, open pull requests and work at risk. Engineers choose the exceptions that need judgment. Routine status collection no longer sets the agenda.

SPECIFY AND RUN

An engineer gives a bounded problem its intent, constraints and acceptance evidence. Agents reproduce the bug, inspect the relevant code, propose a plan, create branches, write tests and prepare small pull requests. Several contained runs can proceed in parallel. The engineer resolves ambiguity and keeps generated volume within the team's review capacity.

REVIEW AND RELEASE

Separate review agents challenge correctness, maintainability, security and missing documentation. Deterministic tools check tests, dependencies, secrets and licences. A person reviews material changes, owns the merge and approves customer-impacting releases.

THE WEEK Product and business teams demonstrate disposable prototypes built with synthetic or approved demo data. Engineering rejects some, accepts some as discovery evidence and admits a small number into a productionisation lane. Acceptance begins a fresh engineering process. Architecture reviews arrive with prepared alternatives, dependencies, security consequences and migration implications. The chosen rationale returns to the repository.

THE MONTH The function reviews customer outcomes beside delivery flow, escaped defects, security findings, debt movement and experiment throughput. Gross agent activity stays separate from validated delivery. The leader's week centres on technical and investment decisions, quality-system design, talent development, cross-functional prototype choices and supplier performance.

Continuous modernisation, wider test matrices, current documentation and several prototypes per customer problem used to be unaffordable. They become normal work for the same team.

PROCESS BY PROCESS

What runs, and what stays with the *person*.

PROCESS	WHAT THE AGENT DOES	WHAT STAYS WITH THE PERSON
Discovery, specification, backlog and planning	Turns curated research, strategy and examples into user flows, acceptance criteria, edge cases and open questions; prepares disposable prototypes in an approved sandbox	Priority, scope, desirability and production intake
Architecture and technical decisions	Searches repositories, standards and prior decisions; drafts alternatives, dependency effects, security implications and a decision record	The architecture choice, its rationale and the migration judgment
Coding, code review and documentation	Inspects the codebase, plans bounded changes, writes code and tests, runs checks, opens small pull requests and flags review issues	The specification, review of material changes and ownership of the merge
Testing, QA, security and dependencies	Derives test cases, generates fixtures, expands regression coverage and traces findings to deployed components	Test architecture, security policy, risk acceptance and serious findings
Release and deployment	Assembles test evidence and approvals; drafts release notes, rollout steps and rollback steps; monitors production health	Approval for customer-impacting or high-risk releases
Incidents, maintenance and technical debt	Builds incident timelines, correlates telemetry with recent changes, proposes tested fixes and prepares bounded refactors	Mitigation, incident command and debt priority
Hardware and embedded R&D	Checks requirements traceability; drafts tests, embedded or PLC code, simulation setups and compliance evidence	Physical design, tolerances, safety judgments, testing and certification
Vendors and nearshore coordination	Reconciles contracts, issues, pull requests, review findings and accepted outcomes; exposes rework and ownership gaps	Maintainability, integration ownership, supplier judgment and knowledge transfer

The accountabilities on the right deserve a second reading. Every line is a decision or an accountability. The left-hand column exists to give engineers more attention for it.

DATA AND CONNECTIONS

Four stages, and the *engineering-system position* today.

A realistic path maps the engineering connection sequence onto the rungs in The Access Ladder. The order matters.

01 **Documents only**

Curated repository snapshots, standards, prior decisions and approved examples support specifications, architecture analysis, test plans and code review. There is no live-system access. Engineers supply current context and inspect every output.

02 **Read access**

Repository and issue-tracker access grounds planning, investigation and traceability in current work. Agents can see issues, dependencies, pipeline results and architecture context without changing the authoritative record.

03 **Read and act with approval**

Repository write access connects with CI, test and security systems. Agents create issues, branches and pull requests, never direct production commits. Automated evidence closes the loop between generation and validation. People approve material merges and releases. For industrial R&D, PLM, requirements, simulation and lab systems enter here but remain read-heavy while traceability and product-safety controls are tested.

04 **Bounded autonomous action**

Observability, customer feedback and deployment controls connect engineering decisions to production behaviour. Reversible runbooks and low-risk actions can operate within preset limits after measured performance. Material production changes retain named human approval.

The engineering-system position, as of September 2026

GitHub	Hosted MCP server generally available since September 2025. OAuth exposes repositories, issues, pull requests, Actions and licensed security tools. Repository permissions and branch rules still govern action.
Atlassian	Rovo MCP server generally available in February 2026 for Jira, Confluence and Compass Cloud. Governed reads and writes carry admin controls and audit logs. Bitbucket Cloud coverage includes repositories, commits, pull requests and pipeline results.
Microsoft	Azure DevOps remote MCP server generally available in August 2026. Hosted coverage includes work items, repositories and pipelines through Microsoft Entra authentication.
GitLab	MCP server remains in beta. OAuth can expose project, issue and merge-request operations across hosted and self-managed environments. Production write authority should stay narrow.
Linear	Hosted OAuth endpoint can find, create and update issues, projects and comments. Read-only access is available; the default endpoint is read-write.
Datadog	Official remote MCP server covers logs, metrics, APM, monitors and security signals. Its Audit Trail records the user, client, arguments and tool name.
Grafana	Hosted Cloud and open-source MCP servers inherit RBAC. The open-source server supports read-only mode.
Sentry	A generally stable public API and German regional endpoint are documented. Treat MCP access as a custom connector unless the current contract confirms supported first-party capability.

Point connections work for one team and a few repositories. End-to-end analysis across planning, code, tests, releases, incidents, customer behaviour and supplier delivery needs a shared data layer of normalised events and relationships, not an indiscriminate copy of source code.

> In the headless model, GitHub, Jira or Azure DevOps keeps the authoritative record, permissions and history. Engineers work through briefs, specifications, reviews and exceptions. Every action log captures the initiating specification, model and version, context sources, tool calls, code diff, test results, security findings, approving person and deployment result.

GRAIL'S THESIS

Six things we believe, from building *this*.

The tools are moving quickly. The operating truths are more durable. Six beliefs should shape how a leadership team redesigns engineering work.

01 The floor is faster code. The ceiling is better engineering decisions.

Drafting speed is visible, so every rollout notices it first. The return sits higher up, in clearer specifications, better architecture, stronger tests and sharper choices about which work deserves production investment. A team can generate more and still deliver less if those decisions do not improve.

02 Implementation stops being scarce before judgment does.

Agents can inspect, draft and test several bounded changes at once. That makes intent, interfaces, acceptance evidence and review attention the new constraints. Engineering leadership must design for those constraints rather than count the volume produced.

03 A prototype proves the question, not the product.

Business teams should test customer problems before joining the production queue. A working sandbox artefact is useful evidence, but it does not inherit the architecture, security, support or release qualities of a product. Productionisation starts a new engineering process.

04 The engineering system amplifies whatever is already there.

Strong specifications, modular architecture, automated tests and protected pipelines turn agent output into validated change. Weak controls turn the same output into larger reviews, false confidence and debt. The tool does not repair the delivery system by arriving.

05 Augment execution, never automate accountability.

Agents can create branches, tests, evidence and proposed remediations. People still own architecture, safety, material merges, incident command and customer-impacting release decisions. That boundary lets the system move quickly without pretending that passing checks is the same as sound judgment.

06 The apprenticeship must be redesigned on purpose.

Senior engineers gain time for systems design and evidence review, but future senior judgment cannot appear by itself. Junior engineers need end-to-end slices, incident rotations, generated work to explain and criticise, and periods of debugging without assistance. The new operating model has to build capability as well as output.

These six beliefs are not positions on tools. They are positions about how a company turns intent into dependable products, and every agent inherits them.

WHAT IT ASKS OF PEOPLE

Roles, rhythm, and where it *fails*.

Senior engineers become specification authors, systems designers and reviewers of evidence. Engineering managers manage flow across people and agents rather than count activity. QA moves from manual execution toward test architecture, risk analysis and quality gates. Platform and security roles gain influence because their controls determine whether generated work can reach production.

Junior engineers need protected apprenticeship. They should own complete slices, explain generated code, challenge acceptance tests, rotate through incidents and periodically debug without agent assistance. Across the function, problem decomposition, specification, test design, code and security review, permissions and usage control matter more than prompt technique. The important skill is recognising an inadequate result.

The rhythm

START	THREE MONTHS	SIX TO TWELVE MONTHS	TWELVE TO EIGHTEEN MONTHS
Select approved tools, owners and standards for bounded workflows	Establish approved tools and bounded workflows	Change team routines for specification, review, testing and prototypes	Connect planning, delivery, security and observability

The sequence follows the delivery chain. Access expands only when specifications, tests, permissions, review and release controls can carry the additional work.

> WHERE IT FAILS Measuring generated code. Accepting large pull requests. Leaving tests weak. Connecting write access before the controls are ready. Letting every team invent its own stack. Removing roles before the delivery system is stable. Adding review work without removing the old routine. Mandating tools while engineers cannot contest findings or influence standards.

Bring the people whose work changes into the design of the standards, measures and approval boundaries. Adoption becomes durable when engineers can see what the system is for, challenge what it produces and remain accountable for what ships.

HAVE YOU THOUGHT ABOUT THIS?

Twelve questions for the engineering *leader*.

1 Product managers, sales and operations could turn customer problems into working prototypes inside an approved sandbox, giving engineering observed feedback before production work begins. Which product decision should they test this way next month?

2 Each well-specified issue could go to an agent that prepares code, tests and a pull request, while a separate agent challenges the result. Which bounded part of your backlog most needs that second opinion?

3 Every architecture decision could be recorded with its reasoning and made available to future engineers and agents. How much rework comes from decisions whose reasons have been forgotten?

4 An overnight engineering brief could combine failed builds, incidents, dependencies and delivery risks before the team starts work. Which morning meeting would that replace or shorten?

5 Your incident team could receive a timeline, likely cause and tested remediation proposal within minutes. Which recurring incident should become the first controlled case?

6 Your technical-debt backlog could be linked to incidents, delivery delay and customer impact rather than engineer opinion alone. Which debt item would move up or down under that evidence?

7 Your release process could assemble test results, approvals, rollback steps and customer notes. What currently prevents a completed change from reaching customers?

8 Your nearshore supplier could be measured on accepted outcomes, escaped defects and knowledge transfer instead of hours and ticket counts. Which contract would look different under those measures?

9 Your embedded team could connect requirements, generated tests, simulation results and compliance evidence into one traceable record. Which verification step consumes the most engineering time today?

10 Your junior engineers could train by reviewing, explaining and debugging generated work rather than only producing routine code. What apprenticeship will create your senior engineers of 2031?

11 Your engineering leader could compare delivery gains with stability and customer outcomes every month. Which current productivity number could rise while the product quietly gets worse?

12 Low-risk fixes could eventually deploy within preset limits while material changes still require approval. Which reversible change is safe enough to prove that operating model?

These are the questions the programme is built to answer with the people who run the function, on their own products, repositories and engineering system, rather than from the outside.