seerene
serenity in software production

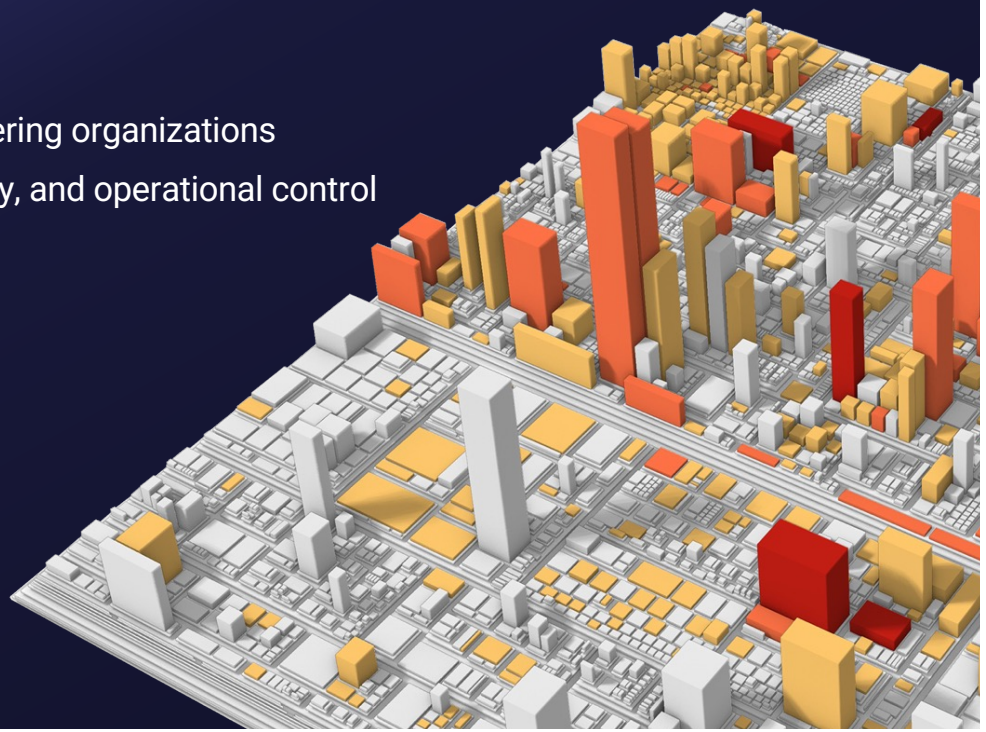
C-LEVEL GOVERNANCE AS KEY FOR SUCCESSFUL TRANSFORMATION INTO THE AGE OF AI CODING

Why software organizations need a new leadership model

- AI is reshaping software delivery at unprecedented speed
- Traditional governance models no longer fit modern engineering organizations
- Executive leadership must regain visibility, steering capability, and operational control

The future challenge is no longer only:

- building software faster
- but governing software organizations effectively at scale



May 7th, 2026 – Executive Exchange at Schloss Fürstlich Drehna

Dr. Johannes Bohnet
CEO & Founder of Seerene

Early Enterprise IT

Software Engineering as Waterfall Projects

The software delivery was sequential:

1) Business defined requirements upfront
2) IT built and operated systems

Role of IT:

→ Primarily an execution function

Clear separation between business and technology



Large-Scale Enterprise IT in Waterfall Times

Governance Through Project Control

- IT organizations delivered many projects in parallel
- Success measured against predefined scope, budget, timeline



Role of CIO / Head of IT

- Track delivery progress of all projects
- Control execution against the plan
- Report project statuses to the business

Governance Principle

- If the project plan was under control, IT was considered under control.

Agile Changed the Organizational Model

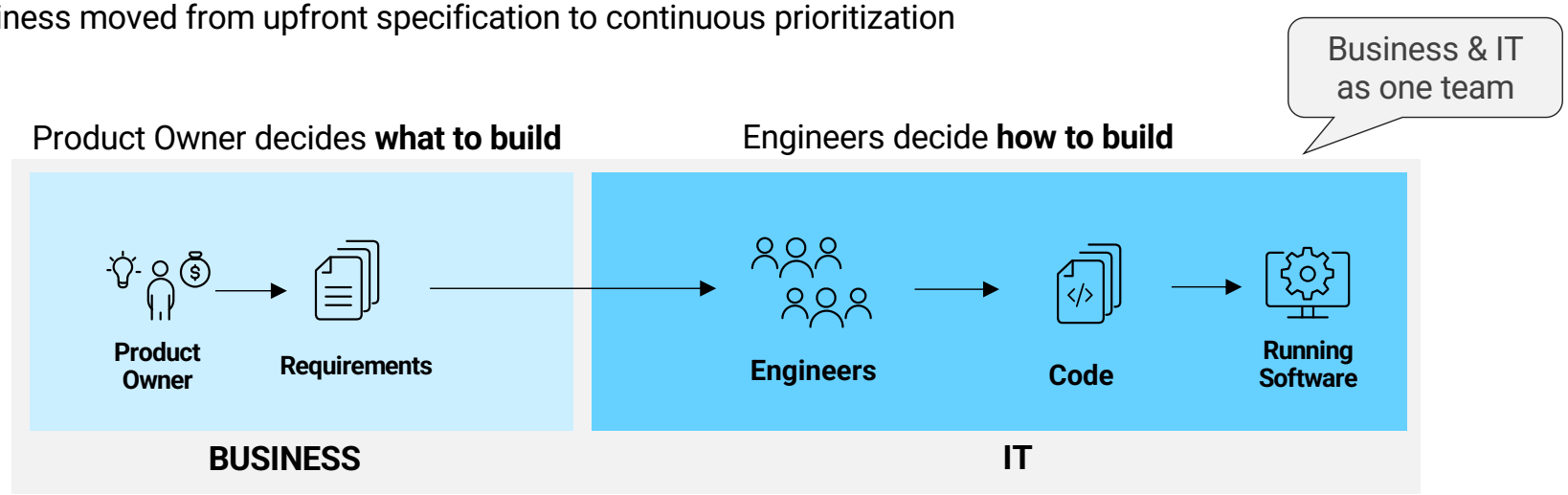
Agile did not just change delivery methods – it changed accountability

The Organizational Shift

- The gap between business and IT was intentionally removed
- Cross-functional DevOps teams became responsible for the entire lifecycle
- Product Owners became embedded into the teams
- Business moved from upfront specification to continuous prioritization

Result

- Teams gained high autonomy, but also
- end-to-end ownership and responsibility.



The Hidden Governance Shift

Agile moved control away from IT leadership

Before Agile

- IT leadership directly governed delivery execution
- Governance happened inside the IT organization

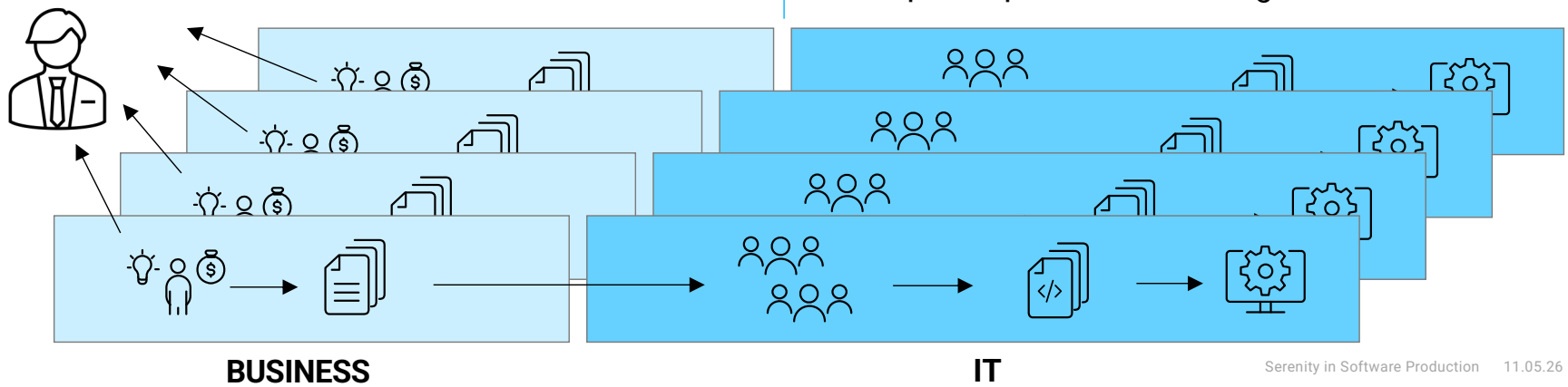
With Agile

Progress reporting shifted to the business side via Product Owners

Emerging Belief System

- “Give teams everything they need.”
- “Do not interfere with autonomous teams.”
- “Swarm intelligence will optimize my organization automatically.”

IT leadership focused increasingly on staffing, enablement, platforms, workforce management



The Governance Gap

Agile created team autonomy — but removed executive observability

Questions Leadership Could No Longer Answer

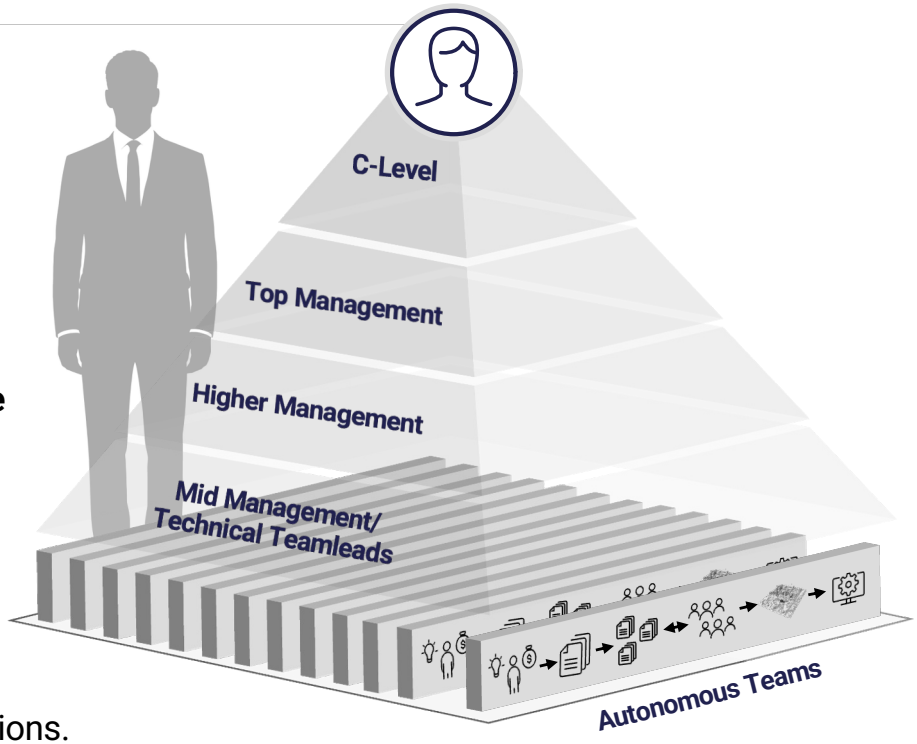
- Which software streams are truly productive?
- Where does engineering complexity slow down delivery?
- Which teams scale effectively?
- Where are structural delivery risks emerging?
- How does engineering performance impact business?

The New Reality – Software Excellence determines Business Fate

- Software excellence determines
  business competitiveness in all business domains
- Weak software execution can threaten the entire company

But There Was a Problem

- No suitable measurement model existed for software organizations.



Why Traditional Metrics Failed

Existing measurement models were conceptually incomplete

Waterfall Metrics

- Worked only because everything was predefined: requirements, budget, milestones, timelines.

Agile Metrics

- Optimized only local team performance.
- Examples: Story Points, Velocity, DORA Metrics, …

Core Limitation

- These metrics were **never designed to govern large-scale engineering organizations**.



Attempts to Solve the Problem

The industry recognized the problem – but lacked operational governance models

Example: Microsoft SPACE Framework

Dimensions

- Satisfaction
- Performance
- Activity
- Communication
- Efficiency



Limitation

- Questionnaire-driven
- Heavily perception-based

Result

- Still no objective, automated governance layer for software organizations.

Seerene: Closing the Governance Gap

First scalable governance model for software organizations

The Breakthrough

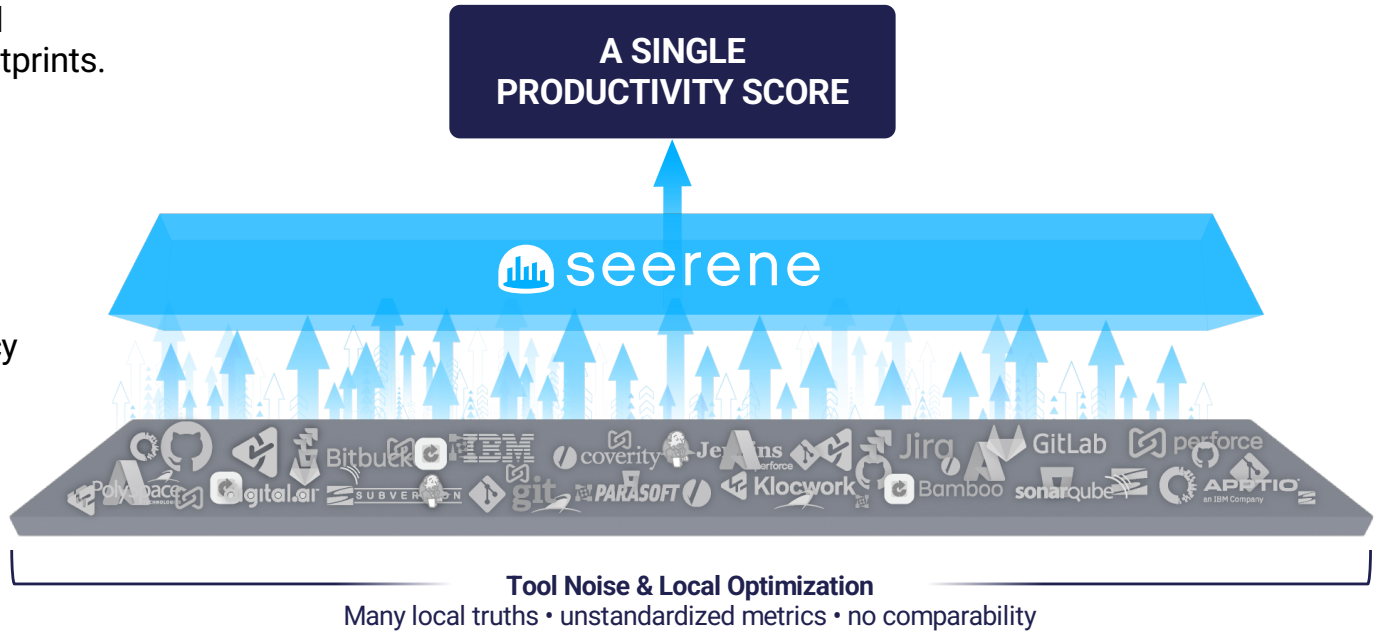
- Automated measurement based on existing engineering data footprints.

Key Characteristics

- No manual reporting
- No methodology dependency
- No technology-stack dependency
- No process changes required

Comparable Across

- programming languages,
- agile, waterfall, V-model,
- Open Source ecosystems, SAP, Microsoft, Oracle,
- frontend, backend, mobile, embedded, mainframe systems.



From Team Metrics to Executive Governance

A new class of executive-grade engineering KPIs

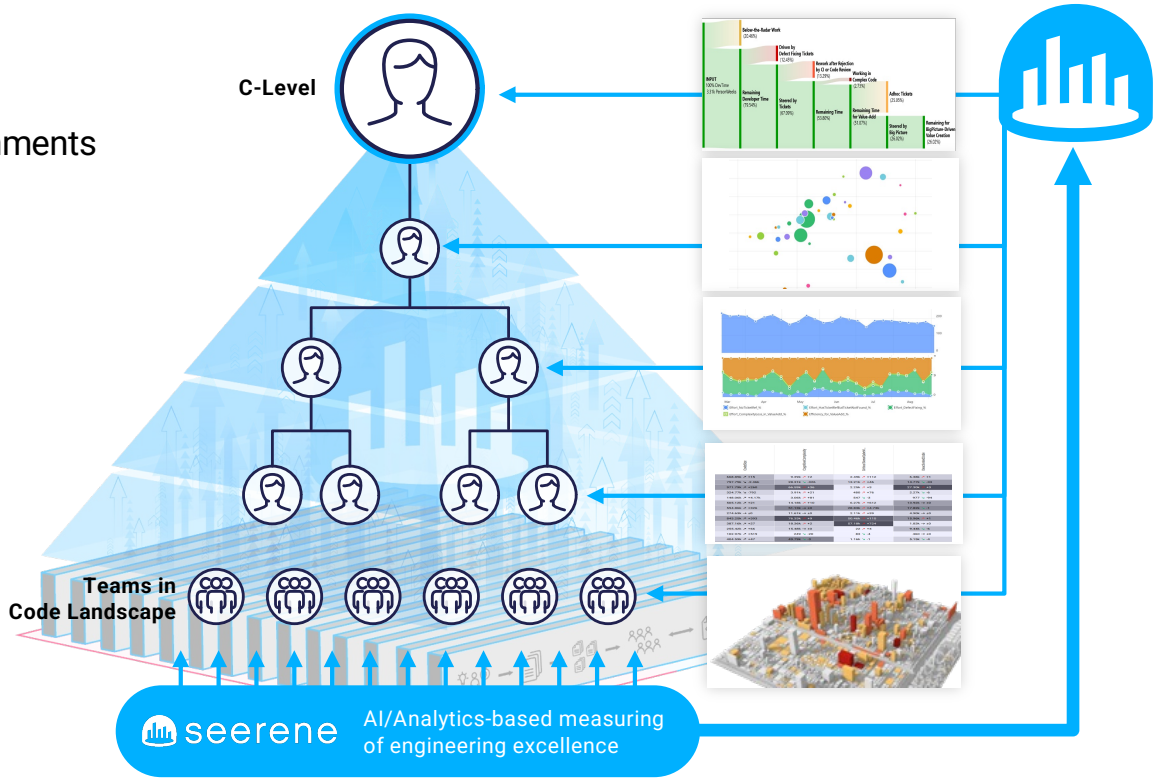
Requirements for Real Governance KPIs

- Automatically measurable
- Comparable across heterogeneous environments
- Aggregatable across management layers
- Understandable for executives
- Actionable at every organizational level

Executive Visibility and Steering Capability

Performance becomes measurable:

- for individual teams,
- delivery streams,
- departments,
- and the entire engineering organization.



The Current State of IT Leadership

A mirror of today’s software leadership landscape



Innovation-Oriented Leaders

- Drive systematic improvement of the organization
- View engineering excellence as a strategic capability
- Embrace transparency and measurable improvement
- Seek scalable governance models

Laggard Leadership Profiles

- Avoid change until it becomes unavoidable
- Prefer established operating models and comfort zones
- Resist transparency because it creates accountability

But Often Still Hesitant

- Fear cross-team measurement could be perceived as distrust
- Struggle to balance autonomy with accountability

Typical Belief Pattern

- “I do not need deeper visibility into engineering operations.”
- “Autonomous teams will optimize my organization automatically.”

The AI Coding Inflection Point

AI massively increases the need for governance

AI changes software development fundamentally

- Higher delivery speed
- Near-zero cost code generation
- Exploding software volume
- Increasing architectural complexity



The New Executive Risk

Without governance:

- complexity grows faster
  than organizations can manage,
- technical debt accelerates invisibly,
- productivity gains become unsustainable.
- **→ In the age of AI coding,**
  passive IT leadership becomes an organizational liability.

Governing the Transition to AI-Ready Software Organizations

AI Coding Changes the Role of the Software Engineer

Before AI: A software engineer had to

1. Understand and challenge the coarse-grained specifications from the business side
2. Translate into fine-grained technical specifications aligned with the underlying code architecture
3. Write and adapt code

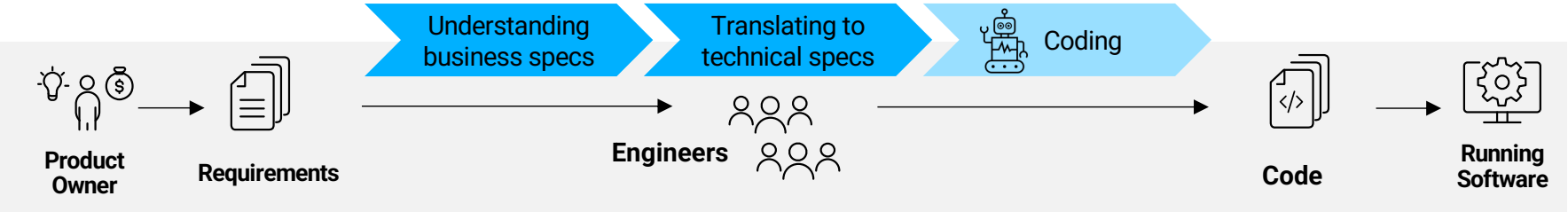
With Agentic AI: Coding itself becomes automated.

The critical skills shift upward:

- precise specification, architectural thinking,
- systems reasoning, validation of AI-generated output,
- maintaining healthy software structures.

Consequence

- Junior coding skills become commoditized.
- Senior engineering capability becomes the key differentiator.



Governing the Transition to AI-Ready Software Organizations

AI Coding Requires Organizational Upskilling

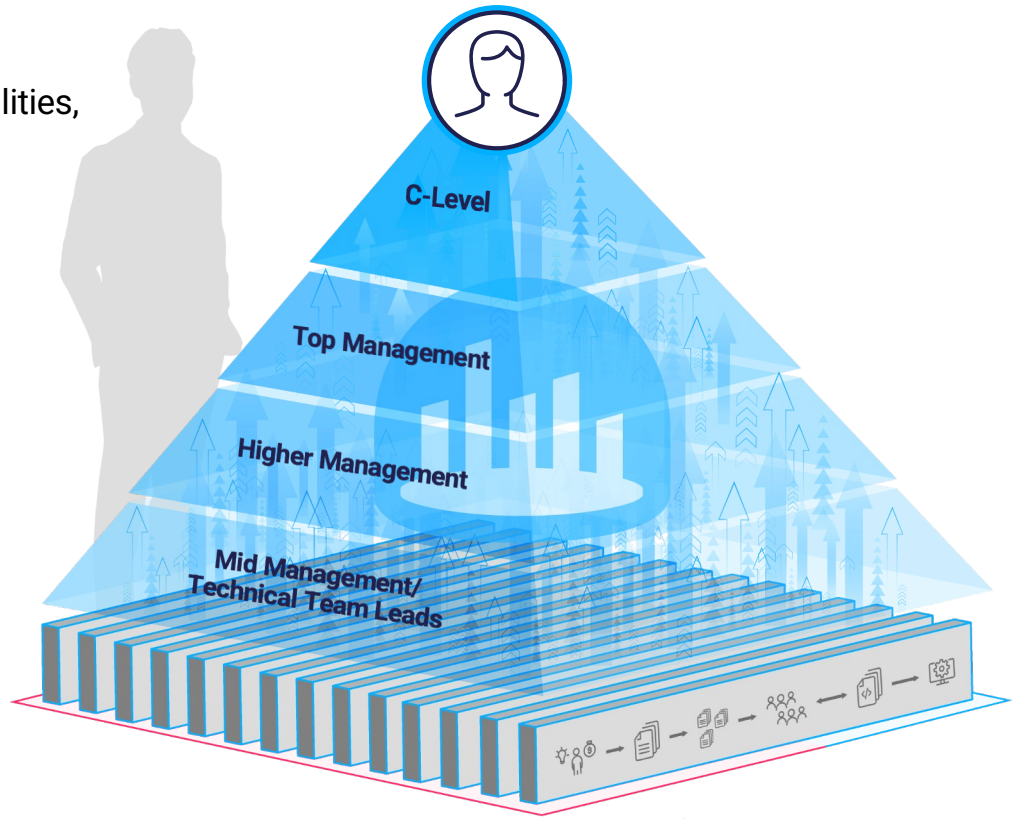
The New Leadership Responsibility

IT leaders need organization-wide visibility to understand:

- which teams already possess senior engineering capabilities,
- where critical skill gaps exist,
- where AI adoption creates structural risks,
- and how to systematically evolve the organization into the age of AI coding.

Core Insight

- AI does not reduce the need for engineering excellence.
  → It increases it.



Key Takeaway

AI Coding requires a new operating model for IT leadership

The future belongs to organizations that combine:

- autonomous engineering teams,
- with transparent executive governance.

Successful transformation requires:

- measurable software engineering,
- organization-wide comparability,
- continuous steering capability,
- executive accountability for engineering excellence.



Thank You!

www.seerene.com



Dr. Johannes Bohnet

CEO & Founder

johannes.bohnet@seerene.com