AI SW Factory – zkušenosti s jejím vytvářením a nasazením

Jan Antoš; Trask Solutions

15. října 2026

ISACA
Czech Republic Chapter

TRASK VELA

Executive summary

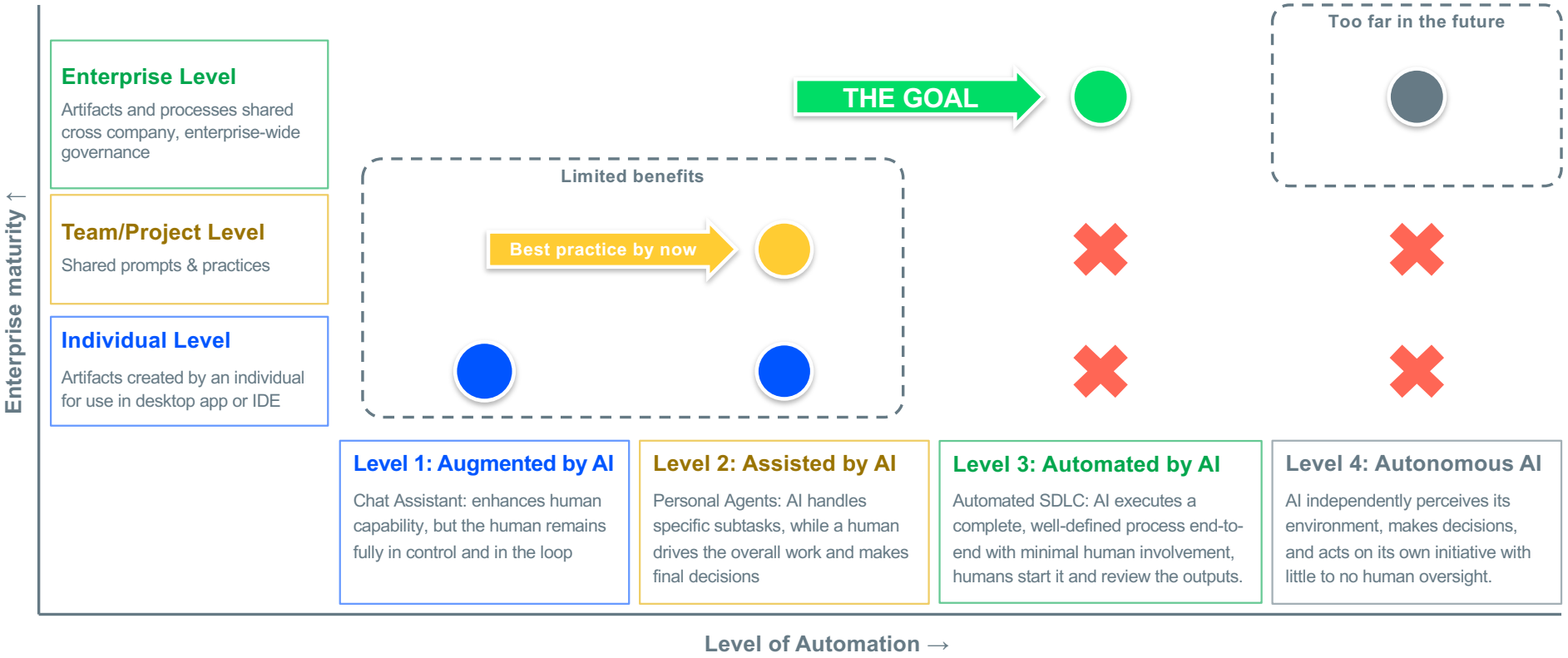
01	**Why**	We need to optimize beyond code optimization because code is only 25–35% of delivery effort. Commercial products/factories focus on code generation and not on project delivery into enterprise environments. Competition offers factories but only as akcelerátor for their deliveries – they keep the factory for themselves and dont share the factory with the client.
02	**What**	Vela is Trask's delivery methodology turned into a runnable AI software factory: 22 AI roles, 19 modules, four phases with human gates. Covers end-to-end delivery or it can be used to automate partial aspects: test automation from analysis, planning, automation to execution. Or automated bug triage and auto-bugfix. Or Knowledge base modul to work with large contexts for analytical and design work automation.
03	**Proof**	Our measured results show overal effort reduction on projects by 28–42% and for individual specific deployments they range from 66% to 85% of effort reduction. Proven on projects for clients as ČEZ, Moneta Money Bank, Škoda Auto and others.
04	**Ask**	A 90-day proof of concept on one application, in your environment and on your tools, with a 3-month free license.

© 2026 Trask Solutions. All rights reserved.

Why we built Vela
(The AI software factory)

trask

AI Maturity Levels



© 2026 Trask Solutions. All rights reserved.

WHY WE BUILT VELA

The real AI gain lies beyond code generation: Vela measured 28–42%

WHAT THE MARKET SHOWS

25–35%

of delivery effort is writing code

the rest is analysis, design, testing, documentation and release

~10%

gain from code assistants alone

because they only touch the code share; typical 2025 baseline

25–30%

gain expected from AI across the whole SDLC

Gartner outlook for 2028

WHAT TRASK VELA HAS MEASURED

28–42%

less delivery effort with Trask Vela

measured in man-days on six real assignments at Škoda Auto, Moneta Money Bank and ČEZ, against plan or a conventional build, 2025–2026

Vela already exceeds the 2028 outlook, today. Code assistants stop at ~10% because code is only a quarter to a third of the work. Vela applies AI to analysis, design, testing, documentation and release as well, which is where the measured 28–42% comes from.

Sources: Gartner, AI in software engineering outlook (2025); Trask delivery records, effort in man-days versus conventional plan or actual.

© 2026 Trask Solutions. All rights reserved.

trask

WHY WE BUILT VELA

Existing options leave a gap: code-first tools, full autonomy, or a factory you never own

	Commercial coding assistants and code factories	Consultancy-internal factories	Trask Vela
Scope	Focus on creating code. Documentation, analysis, testing and project-delivery governance stay manual.	Cover more of the process, but the method is the consultancy's, not yours.	Covers the whole delivery, analysis to release; documentation and gate records are a product of every phase.
Autonomy	Aim for the highest autonomy the model allows.	Varies; typically optimized for the vendor's own throughput.	Deliberately bounded: AI works inside a phase, people decide at every gate. Speed you can defend to audit.
Ownership and capability after the project	A license per developer; no delivery capability is transferred.	Kept internal to the vendor. When the project ends, the capability leaves with them.	Installed in your estate and handed over. The delivery capability stays with you after we leave.

Vela is the only option that closes all three gaps: whole-process scope, governed autonomy, and a factory BMW keeps.

© 2026 Trask Solutions. All rights reserved.

WHY WE BUILT VELA

Regulated enterprises need speed they can prove



Speed with control

Business wants features in weeks; audit wants every decision traceable.

VELA ANSWER

Deterministic process flow, no check skipped, human gates between phases, full audit



Documentation focus

Requirements, architecture, decisions and test evidence must exist as documents, not in a chat history.

VELA ANSWER

Documentation is more important then the code – code can be generated any time again.



Brownfield reality

Greenfield projects are very rare. And every project is in heavily integrated environment with many teams working on it.

VELA ANSWER

Knowledge base to ingest very large context, reverse engineering, focus on integration contracts



One way of working

Single experts with AI tools are hard to be governed. Enterprise needs standard set of tools and enforced methodology

VELA ANSWER

Delivery is **standardized**. Process as code, versioned. New tools added as **governed** decision.

© 2026 Trask Solutions. All rights reserved.

What Vela is and how it works

trask

WHAT VELA IS

Vela is Trask's delivery methodology turned into a runnable AI software factory

Brain

Our delivery know-how, written down and runnable

- **22 specialist AI roles**: analyst, architect, developer, tester, security engineer, reviewer, …
- **Methodology as code**: standards, templates, quality criteria
- **Regulation packs** for regulated industries

+

Engine

Orchestration that runs the process end to end

- Delivery **state machine** on your Jira, GitLab or GitHub
- Runs as **background agents**, one story at a time
- AI **critic and judge loops** inside every phase

=

Modules

19 capabilities, from bug triage to a complete delivery line

- Knowledge base
- Analysis of legacy systems and documents
- Specification, clickable mockups, architecture
- Build, test automation, code review, operations
- Bug triage and auto-fix

Deployed as background-running agents triggered from a task management system.
Interactive modes (console, web) where needed.

© 2026 Trask Solutions. All rights reserved.

HOW VELA WORKS

Four phases run by AI roles, four gates decided by people

Loop for every user story

1 Discover & understand

Analyst · Architect · UX engineer

Documents in; cited facts, requirements, HL architecture; UX mockups; work split into stories; change requests.

2 Design

Analyst · Architect · UX · Tester

Per-story specification, design and plan, GUI design, clickable mockups, test scenarios.

3 Build

Developer · Tester · Security

Test-based development; implementation; code review; security tests; it runs as an audit-and-fix loop.

4 Test & release

Security · DevOps engineer

Thorough security; packaging, release notes and documentation; deployment and operations support; triage and auto-bugfix

HUMAN GATES BETWEEN PHASES

Business approves analysis, architecture and mockups

G0

A reviewer approves each story's design before build

G1

A human merges every merge request

G2

A human approves every release

G3

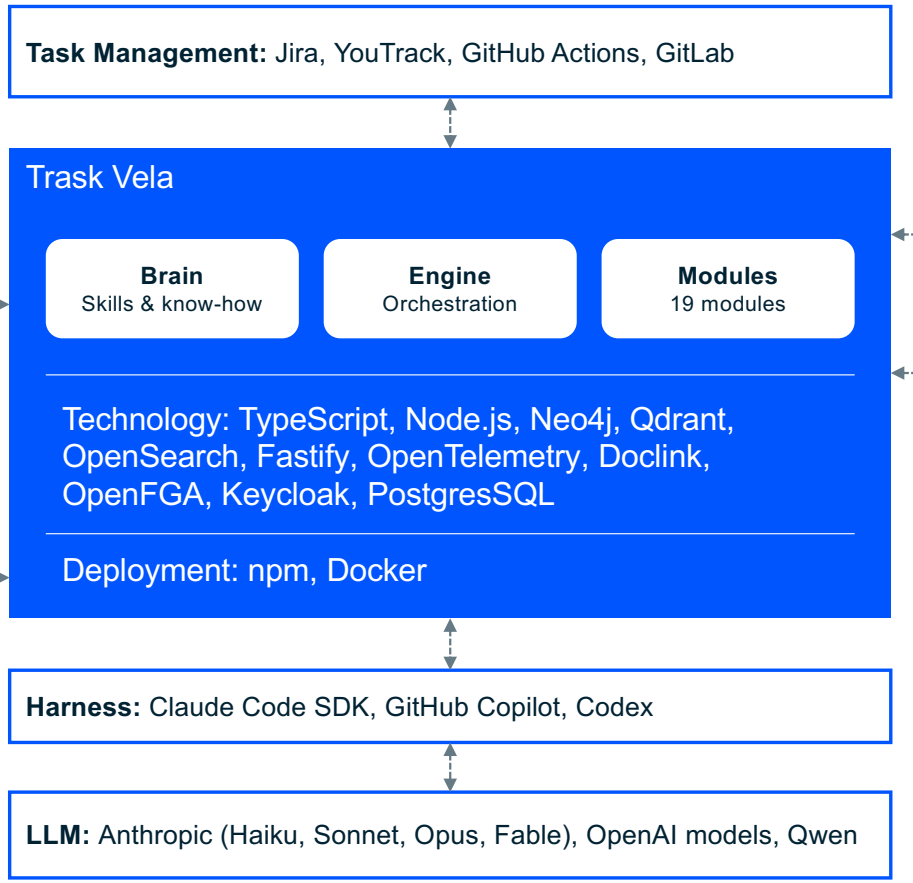
> Inside a phase, AI critics check the work; between phases, people decide. Every story's state is visible on your own board, never 'somewhere in the AI'.

© 2026 Trask Solutions. All rights reserved.

trask

Technology Landscape

Trask vela is built to fit into various enterprise landscapes. We need to integrate Vela with various tools/systems to make it work.
If your tools are not on the list, we are happy to create integrations to your company's standard tools as well.



An integration to task management is prerequisit for Vela to run as background agents. These agents autonomously interact with tasks assigned to them.

Data Intake
Confluence, Figma
MS Office formats, PDF, MS Teams, images, JetBrains YouTrack, source code

We continuously add support for new type of documents and information sources to be able to automate information intake.

Repositories
GitHub, GitLab

Codebase repository, where the project/system files and code lives is the core integration requirement for Vela to work.

Security Tools
SonarQube, AISLE

Testing & QA Tools
Vitest, Playwright

As we deploy to more and more clients, we broaden our list of supported tools that we use in Vela to complete tasks. Vela is designed to use the standard tools that the client's organization has defined as internal standards.

Vela is as technology neutral as possible. Yet currently we support Claude Code SDK with other harnesses on the roadmap.

In Vela it can be configured for which tasks/activities what models to use. The selection can be fully customized.

© 2026 Trask Solutions. All rights reserved.

trask

Agentic development — with Target Security Model in mind

Immature models secure only the produced software – we secure the AI factory and the models behind it as well.

What we protect & secure – three layers, robust target model on our roadmap

1 PRODUCT software the AI factory delivers

- Every build stays secure and trustworthy
- Code scanning by default – SAST, SCA, secret scanning
- Artifact integrity and provenance
- SBOM and CVE monitoring
- 3rd party dependencies – smart selection, continuous testing
- Approval gates / human in the loop

2 AI FACTORY agents, tools & infrastructure

- Each agent has an identity, an owner and managed access – least privilege by default
- Full audit trail as regulatory evidence – who generated what, who approved
- Project workspace isolation – experiments never touch production
- Hardened MCP and infrastructure
- Verified agent skills only
- AIBOM for ICT asset inventory
- Policy-based approved tools / models / MCP servers, enforced at environment level

3 LLM LAYER models behind the factory

- Direct & indirect prompt injection – all input is untrusted
- Malicious / hallucinated dependencies
- AI supply chain risk assessment
- LLM gateway with central policy enforcement

Standards & regulations

Frameworks we build on

- NIST AI Risk Management Framework
- CSA AI Controls Matrix
- MITRE ATLAS – real threat framework
- OWASP Top 10 for LLM, MCP, agents

Compliance targets

- AI factory: EU AI Act, ISO 42001, GDPR, CRA
- Product: TISAX, DORA, PCI-DSS, Network Code

Target: 100% enterprise-ready security – the full model is on our roadmap.

© 2026 Trask Solutions. All rights reserved.

PART 2 · WHAT VELA IS

Trask Vela‘s Modules

Coverage is complete; maturity is transparent and continuously advancing.

Matured = mature component that is ready to be shipped also as a standalone tool; Core = v1.x operational in every end-to-end run; Preview = early version

Discover & Understand

- **Intake connectors & converters** (Preview)
  Integrate, collects and converts various information formats and sources (MSOffice, Figma, Confluence, Wiki, Jira, …) into MD files to enhance performance
- **Intake & Analysis** (Core)
  Documents in → cited evidence registry, business requirements analysis, epic & story breakdown.
- **UX & Mockups** (Core)
  Clickable HTML mockups with in-mockup business feedback.
- **Brownfield Introspection** (Preview)
  Reads legacy codebases into the same analysis flow.
- **Industry Knowledge Packs** (Core)
  banking, utility, energy, automotive, healthcare regulations and NFR defaults, applied automatically.
- **Knowledge base** (Preview)
  Enables humans and agents to work on large documentation sets through vector search and knowledge graph

Design

- **Architecture & ADRs** (Core)
  System and per-story architecture — C4 views, decision records.
- **Requirements Grill** (Core)
  Adversarial questioning that hardens specs and tracks open questions.
- **Adversarial Design Gates** (Core)
  Critic-vs-judge review every story must pass before coding.
- **Traceability & Audit Evidence** (Core)
  Requirement → task → commit chain with preserved gate logs.

Build & Verify

- **Implementation Line** (Core)
  Test-driven coding, one auditable commit per task.
- **Vela Test Line** (Matured)
  Designs, generates, runs and repairs tests; full reporting.
- **AI Code Review** (Core)
  Audit-and-fix loop to approval; runs standalone in CI too.
- **Security Assessment** (Core)
  Per-story OWASP and domain checks, static-analysis gate.

Run & Improve

- **Bug Triage** (Matured)
  Triage issues based on code and available specs and documentation. Does root-cause analysis. Routes issues to fast-track fix, full design loop or to humans.
- **Auto-Bugfix** (Matured)
  Fast-track fix pipeline: AI fixes issues, humans do review and merge.
- **Release & Docs** (Core / Preview)
  Merge requests, release notes, runbooks; deploy daemon.
- **Cost Telemetry & Governance** (Core)
  Per-phase and per-ticket spend, scorecards, budget caps.
- **Self-Improvement Loop** (Core)
  The factory analyzes its own runs and files improvements.

© 2026 Trask Solutions. All rights reserved.

PART 2 · WHAT VELA IS

Documentation produced by Vela — key documents

PROJECT · epics/_project/ · factory analyze

FACTS.md — evidence registry (the engagement's "ground truth"): every factual/quantitative claim from the intake materials as one evidence row; everything downstream (BRD, architecture, story specs) must trace back to it.

BRD.md — engagement-wide business case: customer vision & executive summary, stakeholder analysis incl. RACI responsibility matrix, scope & phasing (in/out), As-Is/To-Be process analysis, functional requirements US-NNN (user story) and BR-NNN (business rule) prioritised with MoSCoW, non-functional requirements, Requirements Traceability Matrix (RTM), Definition of Done.

SYSTEM_ARCHITECTURE.md — solution overview: system boundaries + context diagram, key use cases, technology stack with per-layer justification, component & deployment views, architectural decision records (ADRs), core non-functional requirements; optional evidence-based extensions (security, data architecture, DRP, monitoring, cost estimation, AI/ML).

ADR-NNN.md — architectural decision record (1 file = 1 decision): context, ≥2 options considered, decision, consequences & trade-offs, status and review date.

DEPLOYMENT.md — single source of truth for deploy & operations, maintained over time: Dev/CI/Staging/Prod environment strategy, CI/CD pipeline (build → test → SAST → deploy → verify), containerisation, Infrastructure as Code (IaC), rolling/blue-green/canary deployment, DB migrations, feature flags, monitoring & alerting, release + rollback process, operational runbook.

PMO/ (optional) — PRINCE2-style PM package: Project Initiation Document (PID), risk register, quality register (acceptance criterion → test → result), project plan with milestones, highlight reports, acceptance letter, kick-off minutes, lessons learned.

EPIC · epics/<epic>/ · multi-story epics only

brief.md — one-page epic summary: links to related BR/US/ADR IDs + sizing rationale (why one releasable increment, why this story count).

BRD-SLICE.md (optional) — epic-scoped slice of the project BRD when an epic needs its own requirement refinement.

architecture/ · technical-analysis/ · security/ · deployment/ — tech spec, API contracts, data model, integration specs; same templates, epic scope; only when stories share cross-cutting concerns.

STORY · epics/<epic>/stories/<KEY>-<slug>/ · factory design / implement

spec.md — compact, gate-verified business spec for one story: problem statement, user stories US-NNN with Given/When/Then acceptance criteria, functional requirements FR-NNN traced to a US, only genuinely relevant NFR-NNN, story-level Definition of Done mapped to FR/NFR, references — every intake material cited or marked irrelevant.

architecture.md — story-scoped architecture (boundaries only, no detailed design): solution overview, in/out scope & where the code lives, technology choices with one-line rationale, an ADR per non-obvious decision, integration & runtime landscape; omitted on a fast-track bugfix run.

plan.md — story technical specification — the contract every later phase consumes (UX, tasks, implementation, e2e, MR): concrete file/directory structure, component/module breakdown, data model (entities, fields, relationships, storage), API surface (endpoints/commands/events with request/response shapes), error-handling contract, test strategy (unit/integration/e2e split).

UX/ — UX specification (UI stories only): README (summary, personas, screen index), screens.md (purpose, wireframe, UI elements, loading/empty/error/success state matrix, accessibility, traceability to FR/US), navigation.md (navigation flow + happy path), components.md (shared vs. screen-specific), mockups/ (clickable HTML preview).

tasks.md — engine-parsed, dependency-ordered work breakdown; the implementation phase executes it task by task.

analyses/open-questions.md — requirements-grill output: every undecided point as question + context + AI suggestion + why it matters + owner role; feeds HUMAN GATE 1 (human approval).

analyses/cycle-log.md — transcript of the adversarial design gate (critic ↔ judge); implementation cannot start before the log ends "GATE PASSED".

review.md + reviews/ — per-PR code review output (the actual audit result, not a standards document).

integration-tests/ — backend API tests, three parts: test-plan.md (user story → test case, ACTIVE/SKIP status, linked to AC and endpoint), crosscheck-report.md (gaps between plan and implemented endpoints + remediation), report.md (summary table, requirements conformance per acceptance criterion, INTEGRATION PASSED/FAILED verdict, findings).

e2e-tests/ — GUI tests in Playwright, same plan/crosscheck/report triple for browser flows; E2E PASSED/FAILED verdict.

security/SECURITY_ASSESSMENT.md — security review scoped to the story diff per OWASP Top 10: findings table SEC-NNN (severity, location, description, remediation, blocking?); rule: no unresolved HIGH/CRITICAL finding without explicit risk acceptance.

deployment/RELEASE_NOTES.md — the only deployment doc a story emits: user-facing change summary, new/changed config & env vars, DB migrations (or "none"), concrete rollback steps, impact on the project-level DEPLOYMENT.md runbook.

© 2026 Trask Solutions. All rights reserved.

Benefits and proof

trask

BENEFITS AND PROOF

Great results evidenced by case studies

Selected achievements of AI deployed to our projects

	**Analysis & specification**	**−85%** analysis duration	– System analysis of a 7-year-old system cut from 2.5 weeks to 2 days – Facts registry, numbered requirements and decision records produced at the gate
	**Testing**	**−66%** testing effort	– Automation of 1000 test cases – 160 MD saved on the assignment originally planned for 240 MDs; testing finished 2 months earlier
	**Build**	**+28%** Productivity increase	– Agile development of car configurator recommendation engine: productivity increased by 10% while reducing team from 10FTE to 8FTE (eliminating offshore resources)
	**Operations**	**−75%** incident diagnosis and resolution effort	– AI-driven root cause analysis and incident resolution completed at 75% of original efforts.

Source: Trask delivery records, Car Sharing, Vehicle Intake, Spare Parts, Recommendation Engine and test automation cases, 2025–2026

© 2026 Trask Solutions. All rights reserved.

Lessons Learned from „Closing the Loop“

Enterprise-wide Collaboration

PROBLEM

Outputs of the fully-automated system were rejected by human integration teams.

ROOT CAUSE

Reviewers recognized the AI origin and the volume overwhelmed them.

SOLUTION

Added a human gate wherever team-to-team communication is required.

Cross-Team Collaboration

PROBLEM

Testing agents had insufficient coverage.

ROOT CAUSE

Analytical documentation was missing key information needed to design tests.

SOLUTION

Cross-role play: testers shipped a skill to analysts that enriches story descriptions and enables auto-generated test plans.

Roles Merge

PROBLEM

Role boundaries blurred as automation expanded.

ROOT CAUSE

Automating development merged FE and BE responsibilities into one workflow.

SOLUTION

Defined a new role „Feature Engineer“ handling both sides through an AI-automated system.

© 2026 Trask Solutions. All rights reserved.

ISACA
Czech Republic Chapter

Contact

Jan Antoš
Trask Solutions
jantos@thetrask.com

ISACA Czech Republic Chapter