tieto Tech Consulting

AI-assisted application modernization: how to modernize once and future-proof forever

Visit tieto.com

Table of content

Executive summary

Software modernization has become one of the most urgent challenges for enterprises today. Legacy systems, built over decades of incremental change, now limit agility, scalability, and innovation. The result is **rising technical debt**, **slow time-to-market**, **and growing business risk**.

AI-assisted application modernization: how to modernize once and future-proof forever

Industry studies indicate that companies spend **60-80% of their technology budget on maintaining operations and legacy systems** ("run-the-business" initiatives), and only 20-40% on "change-the-business" initiatives. From the other side, potential benefits from successful app modernization are enormous:

15–35%	30–50%	74%	10%	14%
Savings on infrastructure year-over-year	Lower application maintenance and running costs	Lower costs on hardware, software, and staff	Improvement in application operational efficiency	Boost in annual revenue

43%
Faster time-to-market

35-40%
Cost reduction

AI-augmented approaches to modernization bring 43% faster time-to-market, while reducing costs by approximately 35-40% through automation, improved visibility, and better decision support.

Artificial intelligence allows enterprises to understand, transform, and optimize software systems at a scale and depth never before possible. This guide focuses on each specific area where AI can assist in software modernization:

- **Understanding existing systems** – AI-driven analysis and developer augmentation.
- **Transforming code and architecture** – automated refactoring and semantic translation.
- **Improving quality assurance and operations** – predictive testing and intelligent monitoring.
- **Migrating and optimizing infrastructure** – AI-assisted cloud and cost optimization.
- **Aligning development with business context** – requirements discovery and validation through AI.

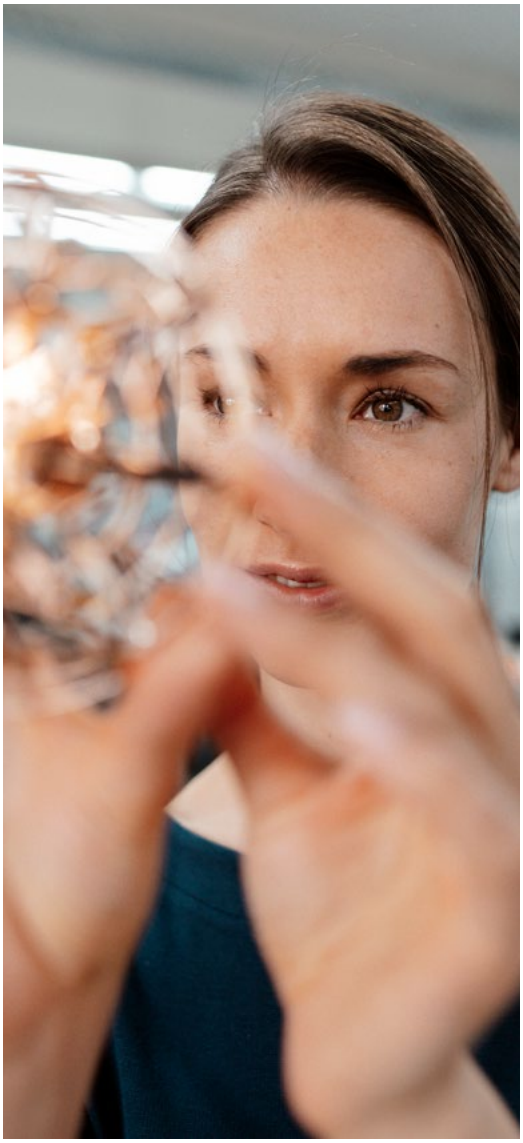
Together, these pillars create a blueprint for intelligent modernization – one that's **scalable**, **explainable**, and **tailored to enterprise complexity**.

Section I.
AI-driven code analysis & developer augmentation

In every mature organization, software systems evolve through years of iterative development. But with growth comes entropy. Layer upon layer, functionality, patches, shortcuts, workarounds, and undocumented decisions accumulate, resulting in a codebase that is hard to navigate, change, or even explain.

Modernization can't begin until this complexity is made visible and manageable.

Traditionally, this visibility required months of manual effort, involving senior developers, domain experts, and trial-and-error testing. But that method no longer scales. Teams are constrained by skill gaps, time pressure, and a lack of institutional knowledge.



What AI brings to the table

Artificial intelligence transforms the analysis phase from a bottleneck to a breakthrough. AI models, trained on vast repositories of open-source and enterprise code, can:

- Map dependencies across thousands of files and services
- Surface hidden architectural patterns and technical debt
- Classify modules by function, volatility, and business domain relevance
- Cluster and tag code based on semantic meaning, not just syntax.

By combining static code analysis with contextual data from documentation, logs, and version history, AI forms a semantic map of the system.

This shift from rule-based scanning to pattern-based understanding marks a new phase in engineering intelligence – one where **AI becomes one of the first lenses through which we see the legacy system**. The following techniques illustrate how this analytical layer is implemented in practice.

TECHNICAL APPROACHES USED

Graph-based code analysis	Static and semantic code models	AI-augmented IDEs
Transforming code into abstract syntax trees (ASTs) and control/ data flow graphs allows ML models to infer structure and intent across interconnected components.	Tools like CodeBERT or PolyCoder help extract meaning from code and comments, enabling classification, summarization, and embedding.	GitHub Copilot, Amazon CodeWhisperer, and TabNine are being piloted to accelerate developer understanding and productivity by suggesting context-aware snippets and refactorings.

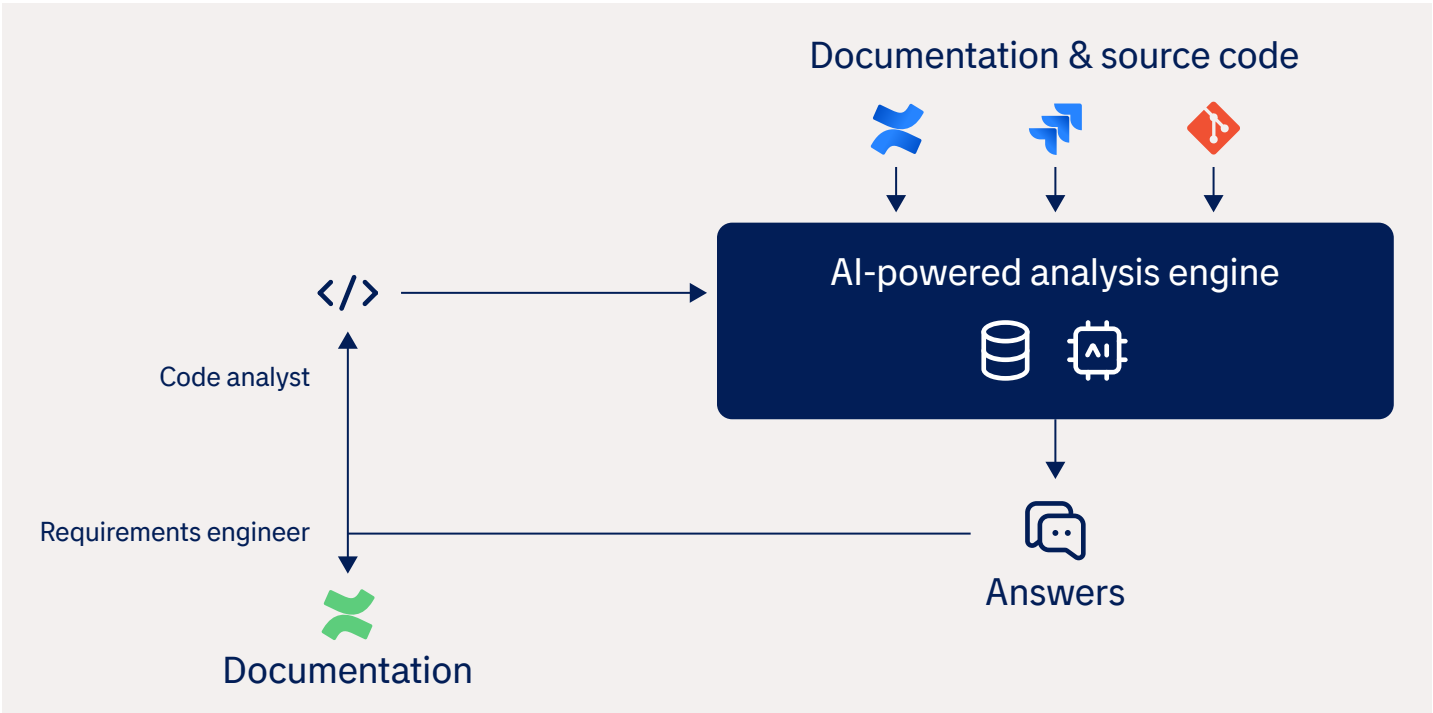
Together, copilots and agents create a continuous feedback loop – one assisting in real time, the other executing broader automation in the background. This way, copilots and agents form an AI development ecosystem – copilots enhance productivity, while agents extend automation and scalability.

Developer in the loop: human-AI pairing

AI doesn't replace engineers – it empowers them. When developers work alongside AI-powered copilots, they gain real-time insight, not just syntax correction. Considering all components involved, the process of AI-assisted code understanding can be visualized as a continuous intelligence pipeline which connects raw source code, documentation, and human feedback into an evolving knowledge loop:

Before AI	With AI
Weeks to map code dependencies	Minutes to generate system graphs
Risky, manual refactoring	Guided, explainable suggestions
Developer burnout, knowledge bottlenecks	Shared understanding through AI-augmented tools
Poor, outdated or no documentation	Real-time, contextual, auto-generated insights

AI-AUGMENTED CODE INTELLIGENCE PIPELINE



AI-generated documentation: closing the knowledge gap

Poor or missing documentation is one of the costliest legacy-system issues. Developers waste hours tracing code behavior, while business analysts struggle to match code to intent.

Modern AI tools **generate documentation on demand**, including inline function and class summaries, architecture overviews, change logs and rationale, API usage examples, and others. These outputs accelerate onboarding, support decision-making, and ensure traceability for audits and compliance.

AI-generated documents are version-controlled with source code, keeping updates transparent and consistent.

Data readiness: the hidden enabler of AI modernization

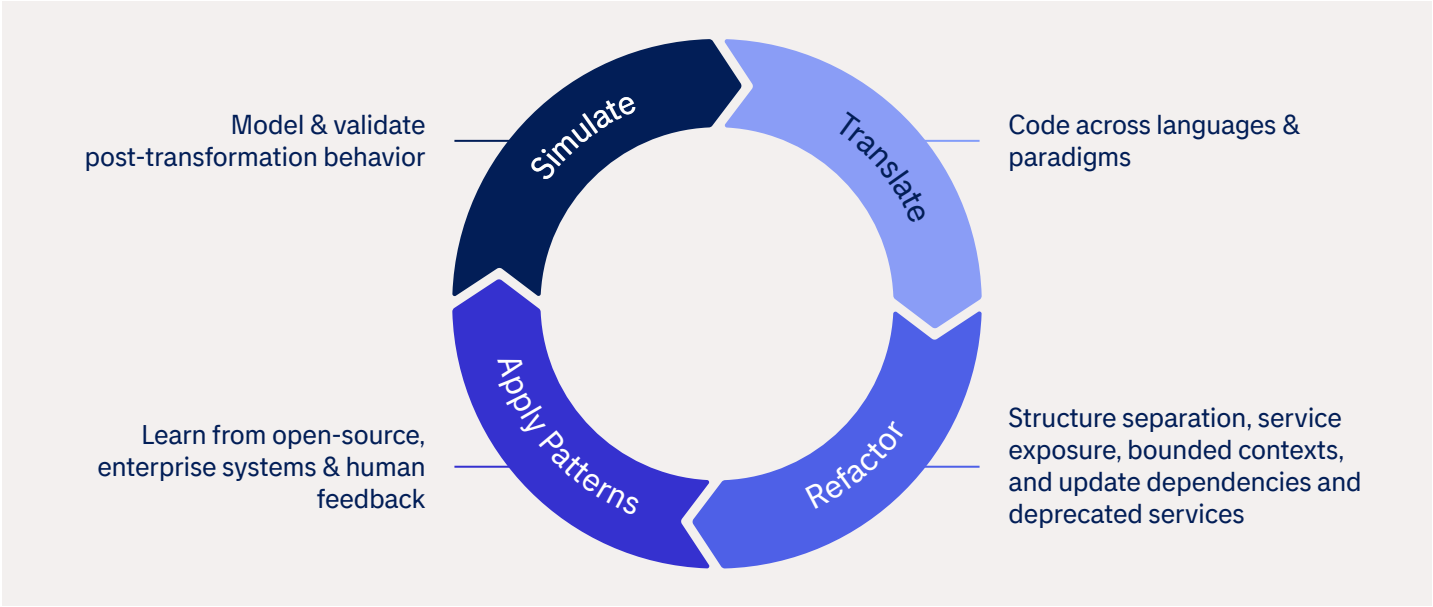
AI modernization relies on clean, contextualized, and secure data across code, documentation, and telemetry. Data preparation is not a one-time step. It is a continuous process ensuring every AI insight stays reliable, compliant, and explainable.

Section II.
From legacy to future-ready: AI-supported code transformation

Moving from analysis to action is where modernization often hits a wall. Teams rewrite code but fail to modernize the system, interpreting only syntax but not the intent. AI-assisted transformation implies restructuring code while preserving the original business logic. It enables **guided**, **incremental modernization**, with the confidence that business logic is maintained.

02

AI-DRIVEN CODE MODERNIZATION LOOP



Key technologies & techniques

Area	Purpose	Examples & capabilities
Large Language Models for Code Translation	Understand syntax & semantics to translate and modernize code	CodeT5+, StarCoder, PolyCoder, Copilot for Migrations, AWS CodeWhisperer • Translate procedural to object-oriented/functional • Modernize platform-specific code (e.g., Windows Forms to React) • Flag deprecated APIs & suggest replacements • Generate, structure, and improve code documentation
Graph-Based Refactoring Engines	Represent code as graphs to identify structure & modularization	Getafix, OpenRewrite, Refact.ai • Isolate business logic • Modularize monoliths • Suggest reusable components AI graph analysis exposes dependencies and service clusters
Architecture Pattern Mining	Discover hidden patterns and boundaries	Embedding & clustering models • Detect microservice candidates • Identify domain boundaries • Spot anti-patterns (god classes, spaghetti modules)
Behavior Simulation & Test Regeneration	Validate functionality post-modernization	Launchable, Diffblue, simulation harnesses • Predict behavior after changes • Auto-generate test cases • Ensure functional equivalence of new and legacy code

Human-in-the-loop: confidence through collaboration

Transformation at scale requires more than automation – it requires trust. That's where the **human-in-the-loop pattern** comes in:

01

AI analyzes and proposes changes:

Suggests code rewrites, modularization, or translations.

02

Developer reviews and validates:

A responsible expert accepts, modifies, or rejects changes using context.

03

Automated tests confirm safety:

Regression and simulation tools ensure parity.

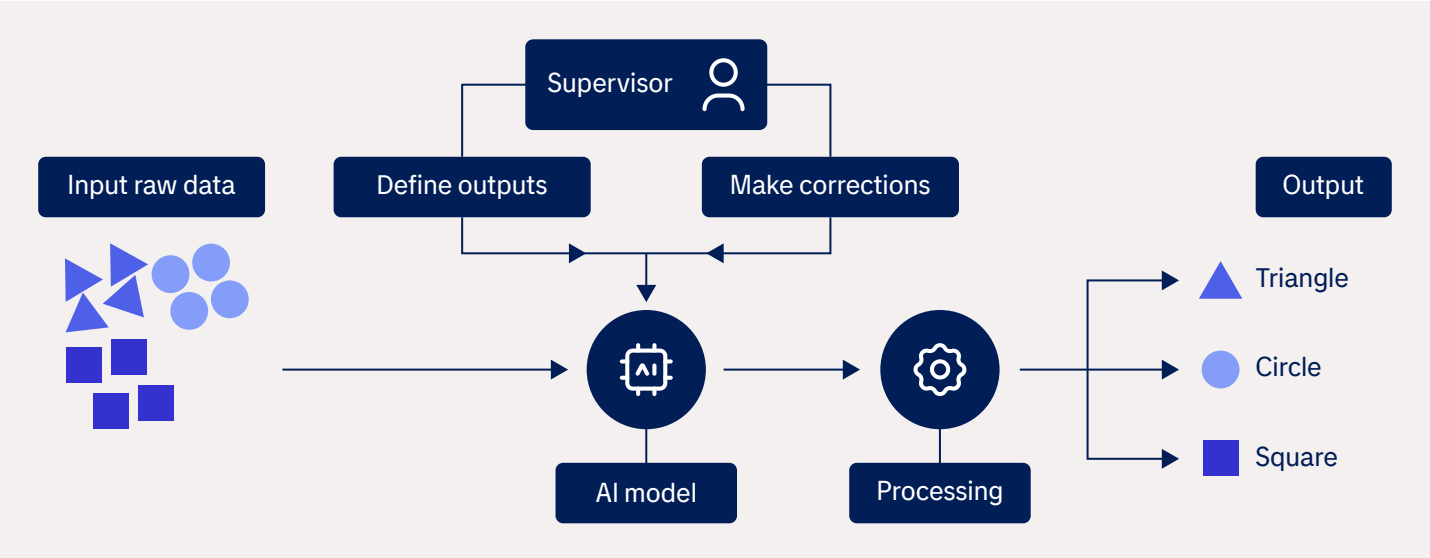
04

Feedback refines the model:

Each interaction improves future suggestions.

The Human-in-the-Loop pattern ensures AI-proposed changes remain transparent and controllable by engineering teams:

HUMAN-IN-THE-LOOP FEEDBACK CYCLE



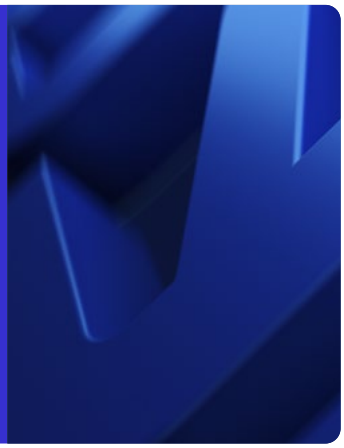
Closing the transformation gap

With AI, teams move faster – not by cutting corners, but by **eliminating uncertainty**:

Transformation is no longer a one-time leap – it becomes a **repeatable process**.

Rewrites are no longer an engineering bottleneck – they become a **collaborative exercise**.

Architecture shifts are no longer guesswork – they are **data-informed, AI-supported strategies**.



Section III. Build once, test forever: AI in quality assurance and predictive IT operations

No matter how advanced your transformation tools are. If you can't ensure quality, **you can't deploy change**. In legacy systems, where there is no clear test ownership, test scripts are often outdated or missing entirely and no longer align with actual code behavior. As a result, even small refactors carry high deployment risks.

AI changes this by enhancing both **quality assurance (QA)** and **IT operations (ITOps)** – transforming them into proactive, intelligent feedback loops.

How AI enhances QA & observability

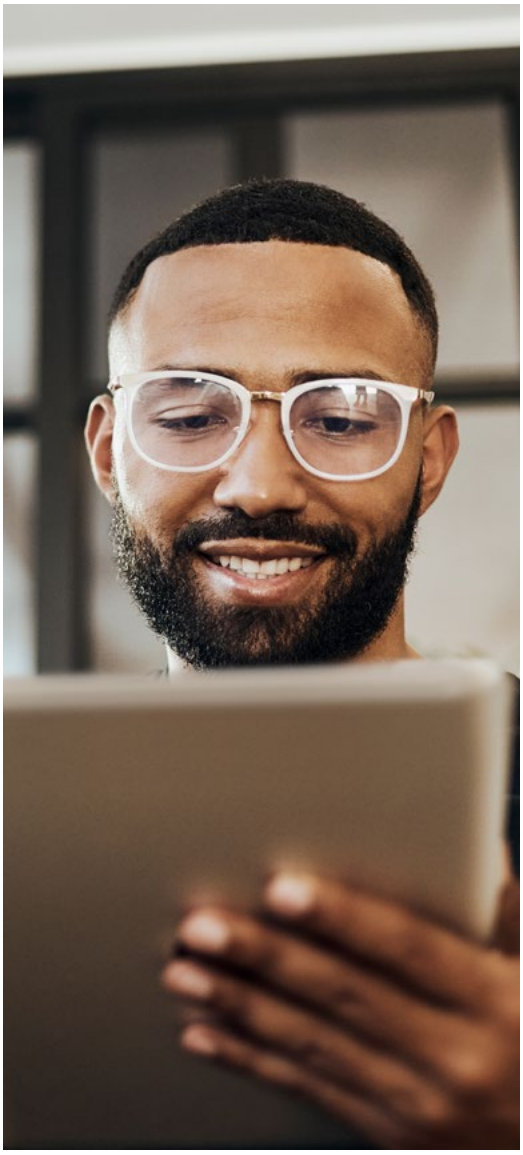
Automated test generation	Intelligent test optimization	Predictive incident detection
AI tools like Diffblue Cover and CodiumAI automatically create unit and integration tests, detect untested paths, and convert business rules into validation logic, closing documentation gaps and increasing confidence in legacy code.	AI prioritizes and selects only the most relevant tests based on code changes, usage patterns, and risk, while identifying redundant or unstable cases, keeping modernization pipelines fast and reliable.	AIOps platforms (e.g., Dynatrace, Moogsoft, Instana) use anomaly detection and pattern analysis to spot issues early, accelerate root cause analysis, and trigger or suggest self-healing actions, shifting operations from reactive to proactive.

From test as a phase to test as a feedback loop

Traditional QA is sequential: write tests › run them › fix bugs. In AI-augmented environments, testing becomes a **feedback system** embedded across the lifecycle:

- Developers write code assisted by AI that *simultaneously* suggests relevant tests.
- Every change triggers coverage analysis and automatic prioritization.
- Failures in staging environments provide learning signals for both AI models and teams.

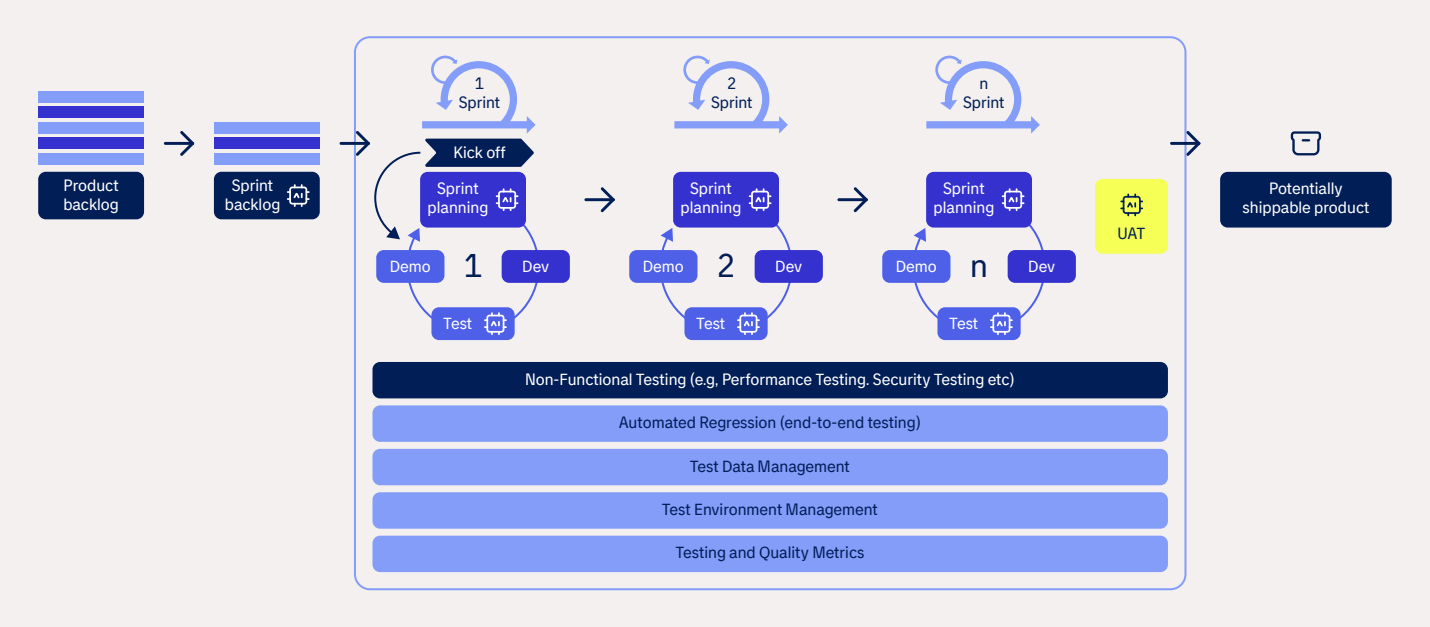
As a result, software testing is no longer a blocker – it's an **accelerator**.



Continuous QA integration in Agile/SAFe pipelines

The diagram below illustrates how AI-driven quality assurance integrates within Agile/SAFe pipelines to maintain continuous feedback across sprints. AI agents assist testing, data, and environment management to ensure reliable, potentially shippable increments every sprint.

CONTINUOUS QA INTEGRATION IN AGILE/SAFE PIPELINES



Benefits for modernization projects

Reduced release risk:
Smarter test coverage + earlier detection = fewer regressions.

Improved transparency:
Code behavior is continuously observed, not just evaluated in staging.

Shorter QA cycles:
Test prioritization means faster, smarter feedback in CI/CD pipelines.

Increased developer confidence:
Safe experimentation encourages more incremental modernization.

Section IV. Smart infrastructure moves: AI for cloud migration and optimization

With quality and reliability established, the next modernization frontier is infrastructure – ensuring that migrated workloads are optimized, scalable, and cost-efficient.

The infrastructure modernization challenges

Poor cost-to-performance ratios

01

Misaligned scalability strategies

02

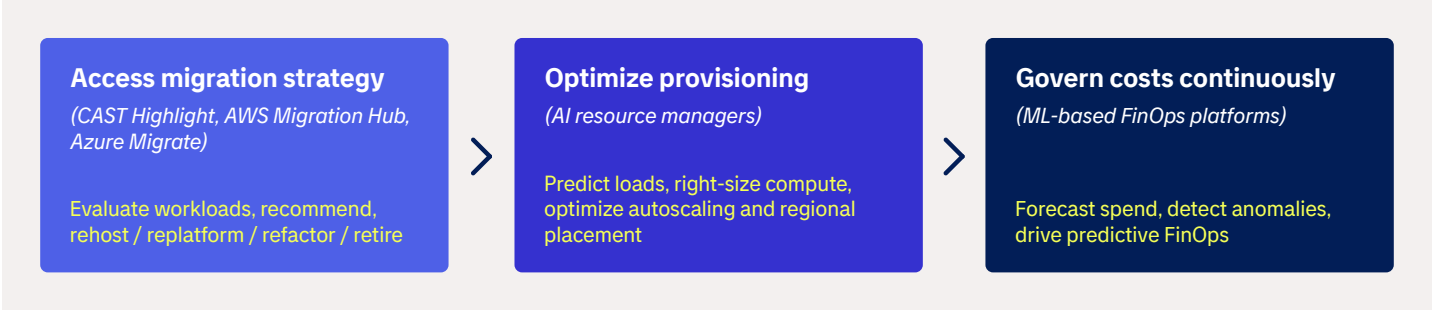
Underutilized services

03

Overprovisioned resources

04

AI-AUGMENTED INFRASTRUCTURE LIFECYCLE



From static design to adaptive infrastructure

In traditional environments, infrastructure is designed once and adjusted manually when something breaks.

In AI-augmented environments:

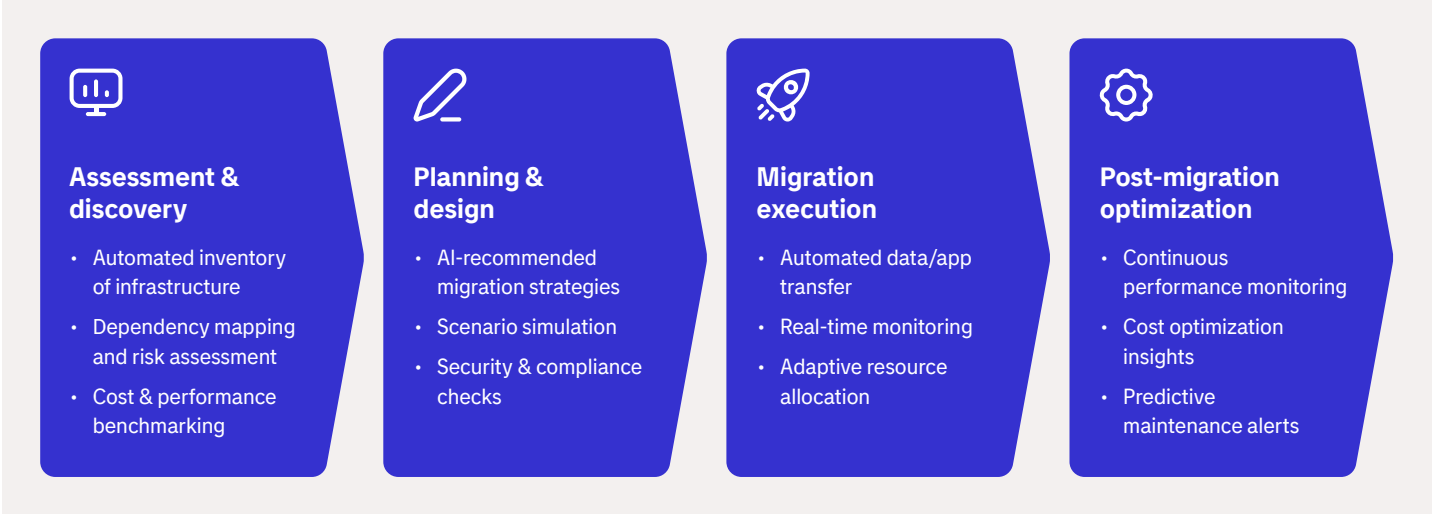
- **Infrastructure adapts in real time** to workload needs.
- **Recommendations evolve continuously**, based on usage telemetry.
- **Architects and DevOps engineers collaborate with AI** to experiment, simulate, and deploy optimal configurations faster.

It **closes the loop between design, usage, and cost** – turning cloud into a strategic asset, not just a hosting model.

AI in the cloud modernization lifecycle

The diagram below illustrates how AI supports every phase of the cloud modernization lifecycle – from assessment to post-migration optimization.

AI IN THE CLOUD MODERNIZATION LIFECYCLE



Strategic benefits

Smarter migration paths:

Move the right workloads, with the right effort, to the right place.

Optimized cost structures:

Avoid post-migration cloud shock with predictive provisioning.

Dynamic infrastructure:

Systems scale, shift, or heal based on real-time conditions.

Architect empowerment:

Cloud decisions are no longer based on guesswork – they're data-driven.

05

Section V.
Making the business visible: AI-assisted requirements engineering

Legacy systems were often built without clear, traceable links between business goals and software behavior. In modernization projects, this creates recurring pain points:

- Outdated or missing documentation,
- Business rules embedded in legacy code,
- Disconnect between what the business expects and what IT delivers.

Modern natural language processing (NLP) and large language models (LLMs) offer powerful tools for reconnecting **business context with technical delivery**.

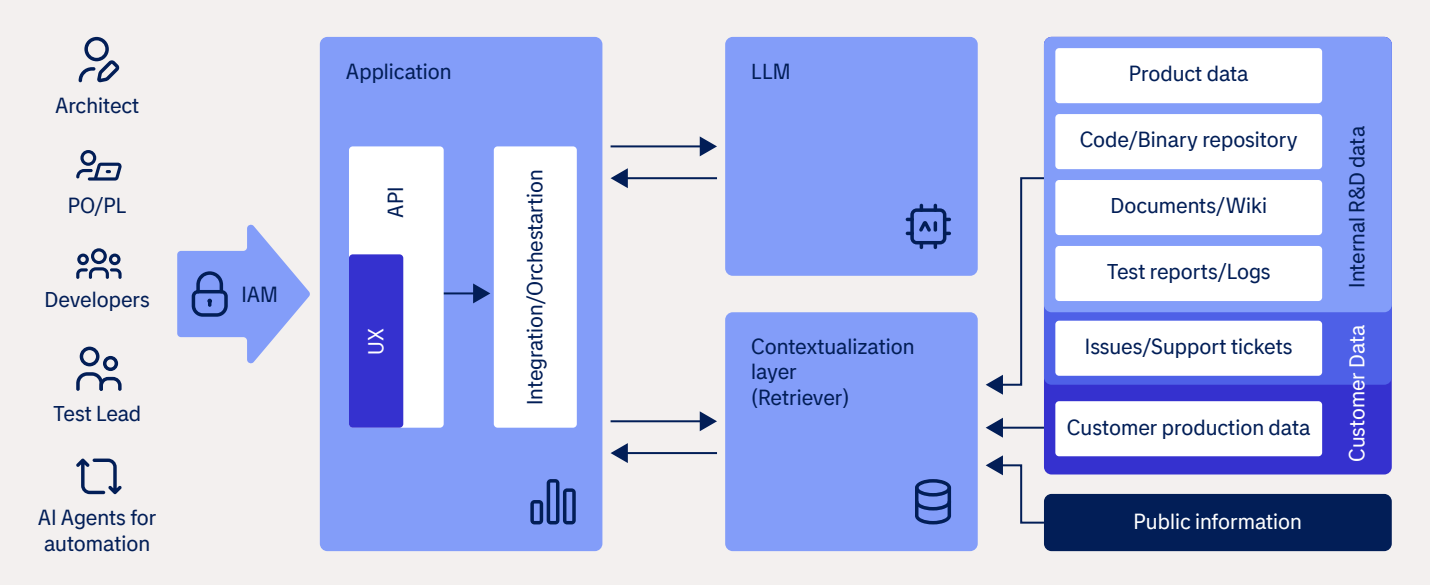
AI-powered requirements lifecycle

Requirements mining	Traceability mapping	Interactive validation
Extract & link requirements from unstructured data (Docs, emails, transcripts): • Extract functional and non-functional requirements • Identify contradictions, overlaps, and gaps • Link requirements to specific components, domains, or workflows.	Map business rules to intent, detect drift: • Map business rules in code to original intent or policy documents • Flag legacy logic that contradicts current objectives • Highlight duplication or drift from expected behaviors.	Ask, simulate, and test scenarios: • Ask natural-language questions like “Where is this SLA enforced?” • Generate user stories or test cases from written policies • Simulate “what-if” changes before implementation.

Context-aware AI system for requirements engineering

AI connects business context with system design through a contextual retriever layer that supplies the LLM with precise, role-specific information. Here is how large language models (LLMs) work in combination with a contextual retriever layer to deliver precise, role-aware responses:

CONTEXT-AWARE AI SYSTEM FOR REQUIREMENTS ENGINEERING



By connecting to internal documentation, support tickets, test logs, and production data, the system enables traceable, explainable alignment between business goals and technical delivery.

Strategic benefits

Clearer alignment:

Ensure what's being built matches what's needed, even across legacy logic.

Faster discovery:

Reduce weeks of analysis to hours with AI-powered mining.

Continuous validation:

Requirements evolve along with the system, not separately from it.

Human-centric interfaces:

Natural language queries democratize understanding across IT and business.

Synthesis. From pilots to platform: scaling AI-assisted modernization

The five areas we've explored – from code understanding to infrastructure, from testing to business alignment – form a **cohesive modernization capability** across teams, disciplines, and systems. In this approach, AI enables **strategic coherence**:

- Code changes are traceable to business goals.
- Infrastructure decisions respond to real-world patterns.
- Testing and monitoring adapt to shifting system behavior.
- Developers, architects, and product owners operate with shared, AI-assisted context.

Building an AI-enabled modernization capability

Organizations typically start with isolated experiments, piloting AI for code review, using a copilot, or trying out automated test generation. However, true value comes when these tools and techniques become **part of the core engineering operating model**. It requires:

- **Strategic alignment:** Leadership commitment to AI-augmented practices.
- **Engineering integration:** Embedding AI into pipelines, processes, and platforms.
- **Governance:** Clear policies around transparency, validation, and model usage.
- **Skills and culture shift:** Upskilling teams to trust and challenge AI outputs.

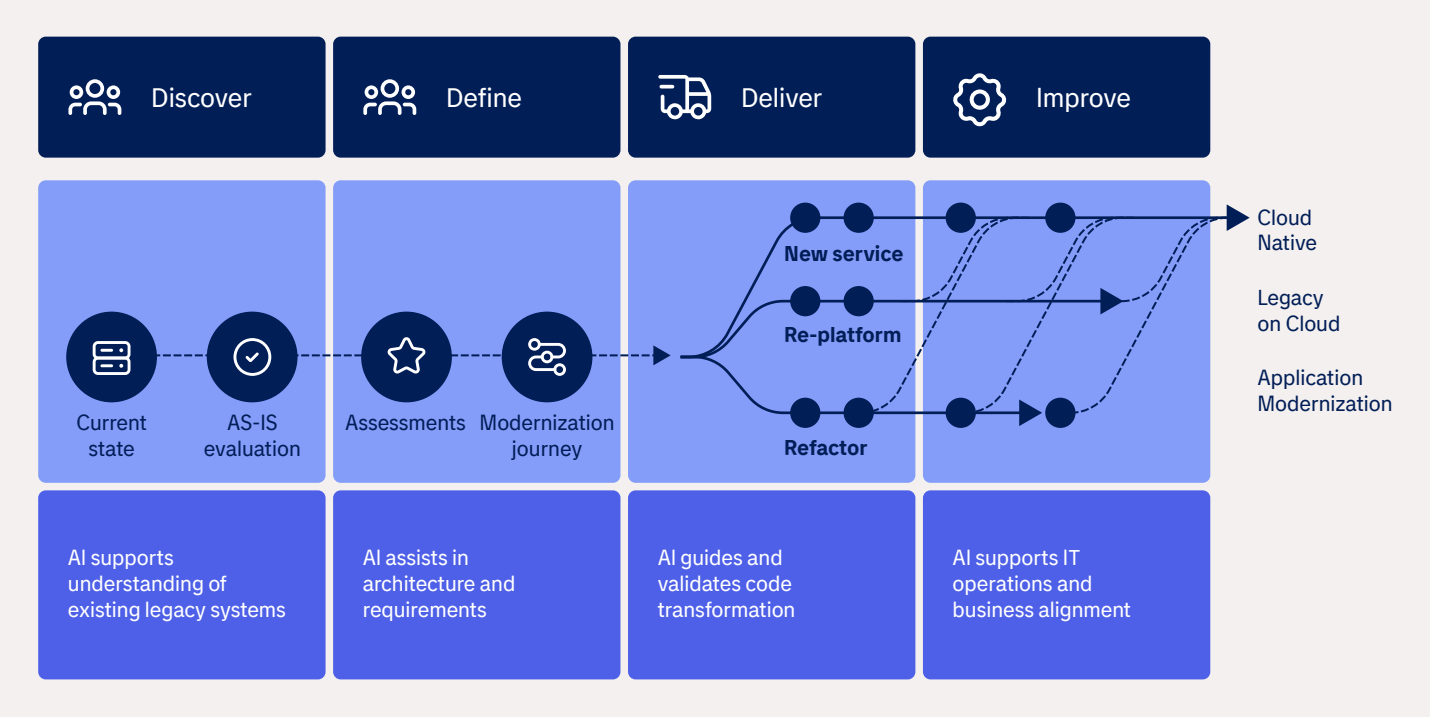
AI-enabled modernization maturity model

We recommend visualizing organizational maturity in four levels:

Level	Characteristics
1. Ad Hoc	Manual modernization, isolated code changes, no AI tools
2. Assisted	AI tools used in silos (e.g., code generation, test creation)
3. Integrated	AI embedded into SDLC; traceability, CI/CD, cloud decisions AI-informed
4. Continuous	Organization-wide modernization as-a-capability, led by data and AI signals

To unify these transformation areas into a single, scalable practice, we map AI contributions across the full modernization lifecycle – from initial discovery to continuous improvement:

AI-ENABLED MODERNIZATION APPROACH ACROSS THE DELIVERY LIFECYCLE



Within such a framework, AI supports each phase of modernization. It aligns technical execution with strategic business goals while enabling continuous evolution toward cloud-native and composable architectures. It is a **transformation discipline**, which becomes a **scalable**, **explainable**, **and business-aligned capability** that differentiates digital leaders from digital survivors.

Next steps: make modernization a capability

To move forward with AI-assisted modernization, consider:

1. **Run an AI modernization readiness assessment**

   Evaluate your systems, tools, and organizational readiness to adopt AI across the SDLC.

2. **Pilot a high-impact use case**

   Start where risk is low and payoff is high: test generation, legacy code understanding, or infrastructure cost optimization.

3. **Create a design-led modernization roadmap**

   Combine AI tools with enterprise architecture, DevOps practices, and business strategy – turning one-time upgrades into a repeatable operating model.

4. **Upskill and enable**

   Invest in teams, governance, and processes to build trust and fluency around AI-augmented engineering.

The organizations that thrive tomorrow are not the ones with the newest tools – but the ones who use them **to continuously evolve**, **align**, **and deliver**.

Let's future-proof your business with a new approach to software modernization!

Contact us:

Doris Raimann

Senior Solution Consultant
doris.raimann@tietoevry.com

Aleksandar Dimitrov

Lead Solution Architect
aleksandar.dimitrov@mentormate.com

About Tieto Tech Consulting

Tieto Tech Consulting provides design-led, data-centric, and AI-powered digital engineering & consulting services to enterprises worldwide. With a strong heritage of Nordic quality, local presence, and global-scale operations, we are backed by 8,500+ IT professionals. Our team elevates people-first experiences and builds tailored digital solutions, cloud, BI, and AI systems. We integrate modern data platforms and unify diverse enterprise software solutions into scalable digital ecosystems, helping you efficiently scale and accelerate your business while making it smarter with purposeful technologies. Our clients trust us for advanced data and analytics skills, top industry-specific expertise, specialized software R&D, and the capabilities of a multinational team. We are part of Tieto, a leading technology company with an annual revenue of approximately EUR 2bn.

tieto Tech Consulting