asteyo

we get things done

Mainframe Modernization for Automotive

From business-critical legacy applications to modern, modular architecture
A pragmatic transformation approach by Asteyo

What this white paper covers

01. Why now

The modernization pressure facing automotive IT

02. What to avoid

Why big-bang rewrites often increase risk

03. Target architecture

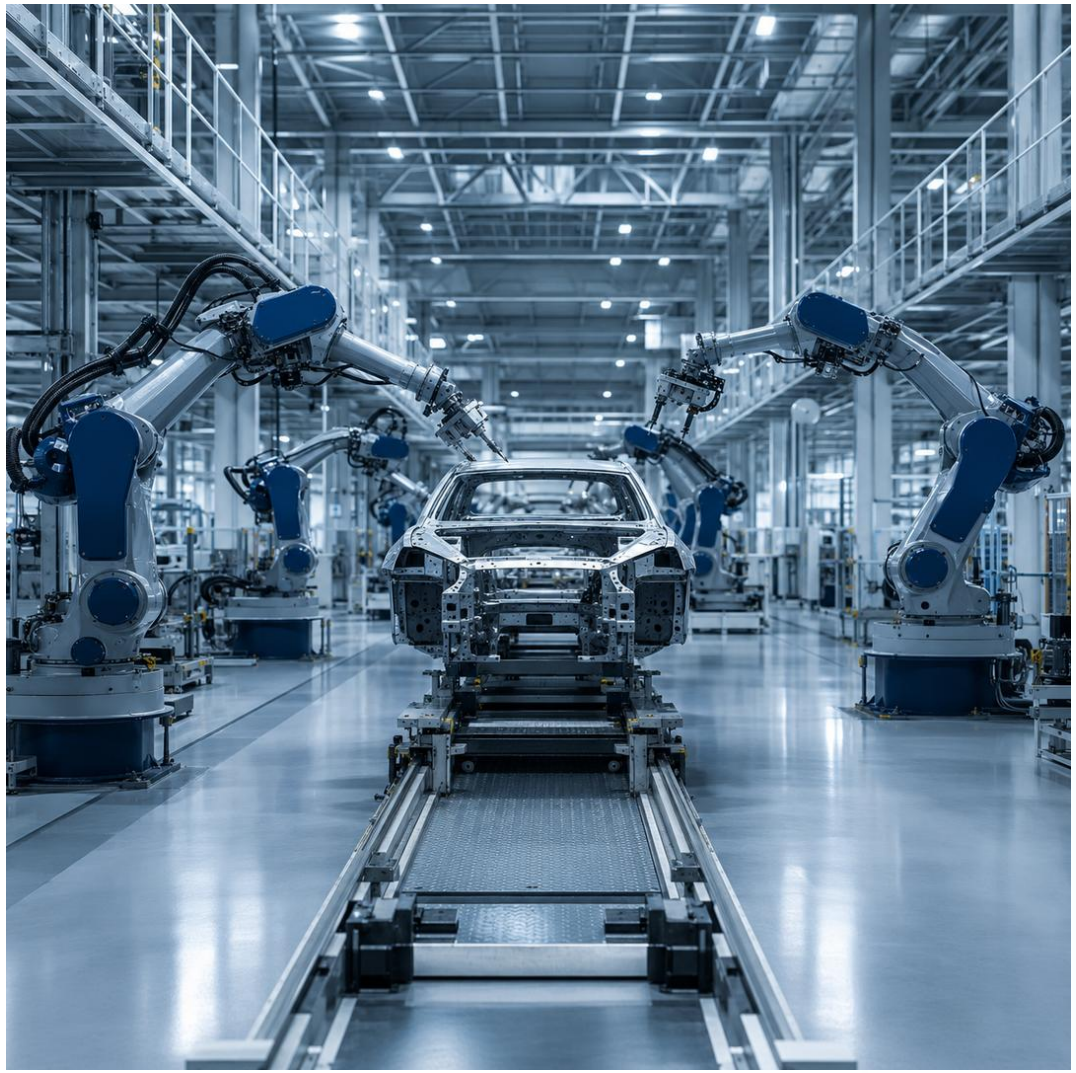
From monolith to modular, API-enabled, cloud-ready systems

04. Transformation path

A controlled journey from legacy stabilization to modernization

05. Why Asteyo

Legacy skills, modern architecture, and scalable nearshore delivery



Automotive IT still depends on systems that cannot simply be switched off

Mainframe applications often sit at the heart of production, logistics, finance, warranty, dealer, or supply-chain processes.

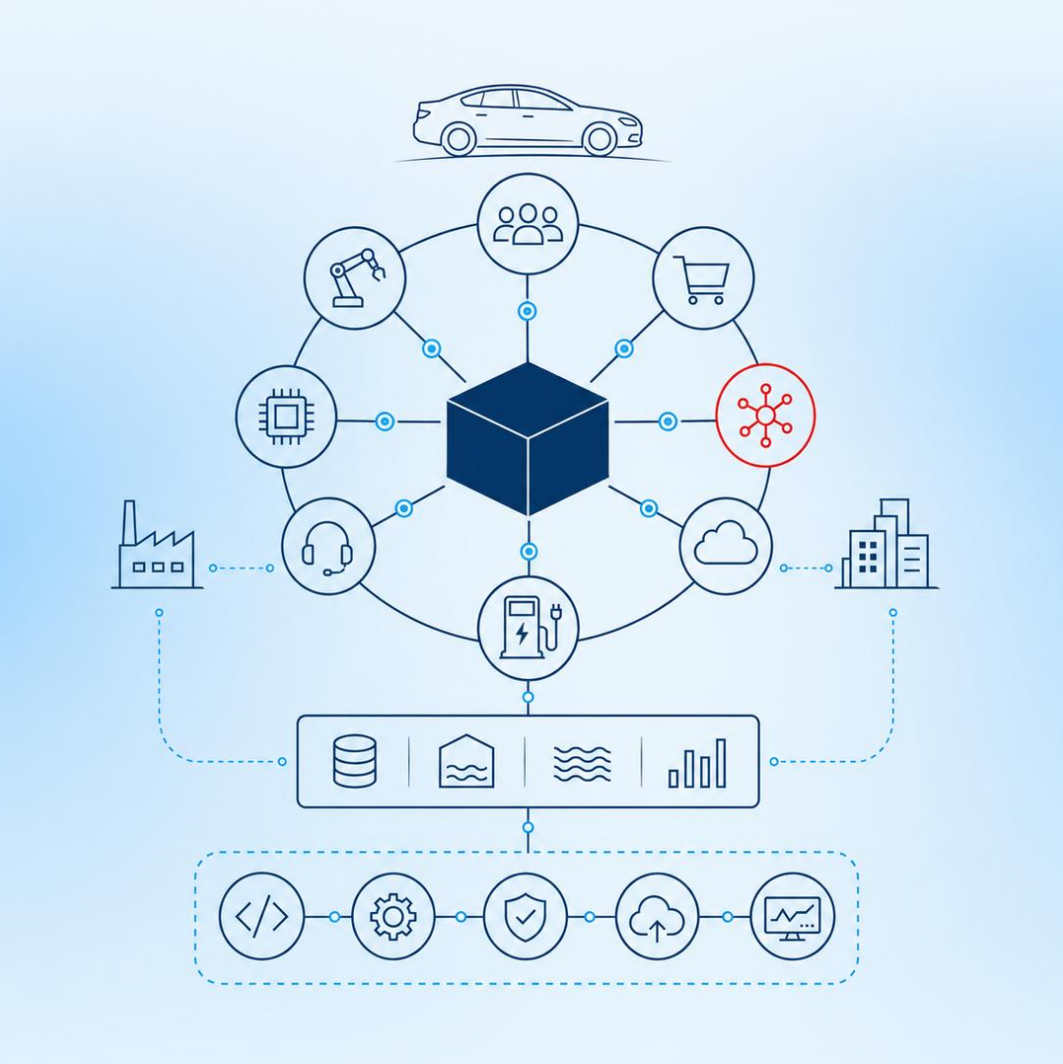
- **Business-critical core** — supports production, parts availability, customer commitments, and financial control.
- **Decades of business logic** — COBOL systems encode rules and exceptions rarely documented in one place.
- **High operational dependency** — surrounding apps, data flows, and batch jobs make small changes complex.
- **The real challenge** — preserving business logic under operational constraints, not just migrating technology.

The system may be old, but the business logic is often mission-critical.

The wrong modernization approach creates more risk than the legacy itself

Automotive organizations need transformation without interrupting operational stability.

- **Full rewrites underestimate complexity** — legacy behavior hides in undocumented rules and years of adaptation.
- **Business users lose confidence** — missing edge cases slow adoption and create shadow processes.
- **Parallel run is expensive** — long dual environments increase cost, complexity, and governance effort.
- **Architecture goals drift** — without clear decomposition, the new stack reproduces the old monolith.
- **Better principle** — reduce risk step by step and keep business continuity at the center.



Modern architecture does not mean replacing everything at once

The target is a modular landscape where legacy value is preserved while technical constraints are reduced.

- **Modular capabilities** — business domains separated into services instead of one tightly coupled core.
- **API-enabled integration** — stable, documented interfaces replace point-to-point dependencies.
- **Data accessibility** — operational and analytical data available without destabilizing transactions.
- **Cloud-ready evolution** — selected workloads move to containers or cloud-native where value is clear.
- **DevOps and automation** — build, test, deployment, and monitoring become repeatable and transparent.

Modernization is a target architecture journey, not a one-time migration event.

A controlled path from legacy stability to modern architecture

Asteyo recommends a staged transformation model that balances business continuity, architecture quality, and delivery speed.

Discover

- Map apps, interfaces, data flows
- Find business logic hotspots
- Assess options by risk and value
- Define target architecture roadmap

Stabilize

- Secure operational knowledge
- Improve docs and monitoring
- Build regression tests
- Joint legacy + modern backlog

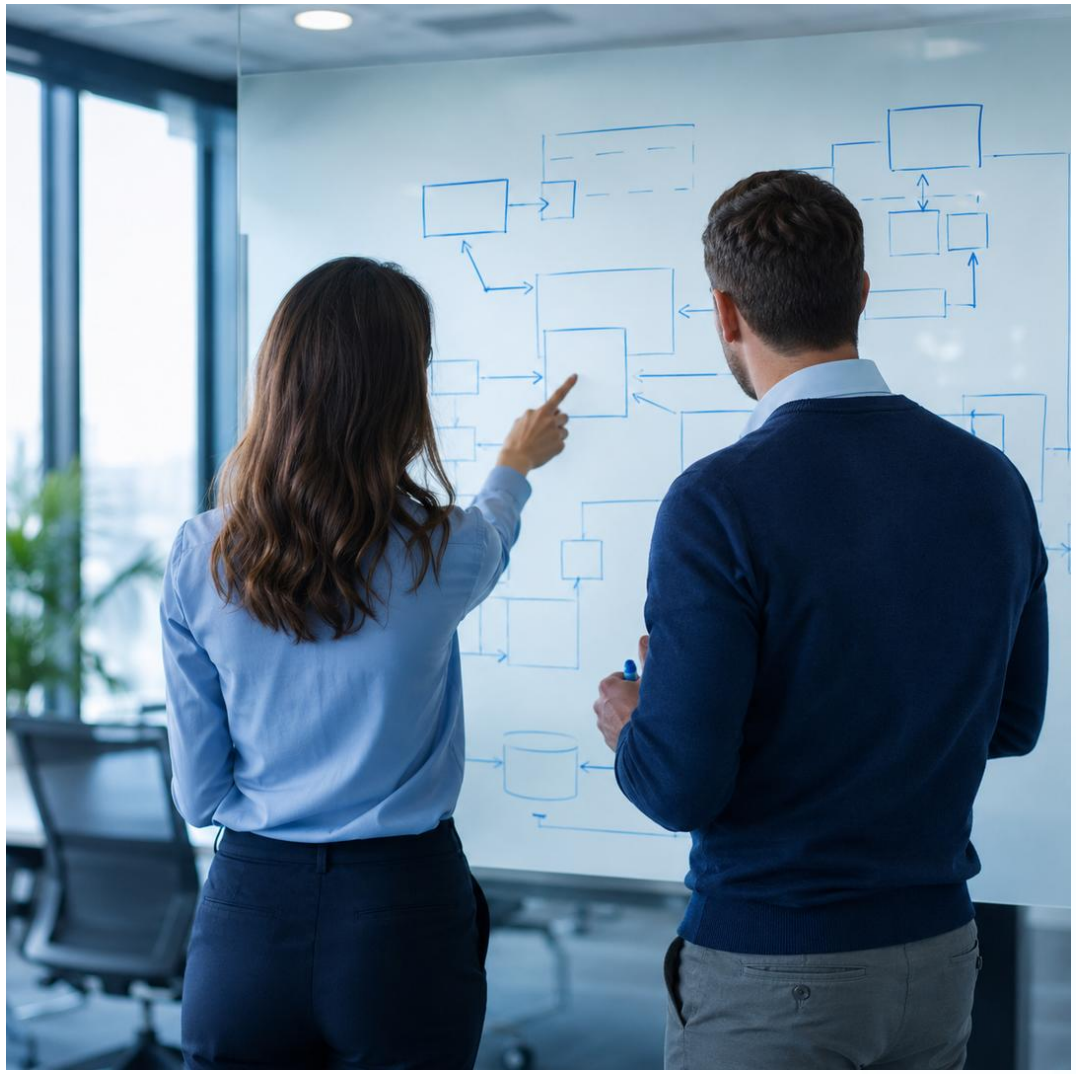
Modernize

- Extract via strangler patterns
- Wrap functions through APIs
- Move data layers progressively
- Mixed legacy + modern teams

Modernization should combine proven patterns instead of relying on one silver bullet

The right pattern depends on business criticality, coupling, data ownership, and change frequency.

- **Encapsulate first** — expose stable interfaces before replacing internal implementation.
- **Strangler pattern** — shift capabilities from legacy core to modern services over time.
- **Domain decomposition** — align modernization slices with business domains, not technical modules.
- **Data transition layer** — separate analytics and integration from transactional constraints.
- **Pragmatic rule** — keep what is stable, modernize what blocks change, replace where the case is clear.



The rare skill is not COBOL alone — it is COBOL plus modern architecture

Many programs fail because legacy and modern teams work in separate worlds.

- **Legacy-only teams protect continuity** — but rarely design the future-state architecture.
- **Modern-only teams drive new platforms** — but underestimate mainframe behavior and batch logic.
- **Business SMEs know the process** — but cannot translate system behavior into modernization decisions.
- **The required profile** — people who change COBOL systems while also designing APIs, DevOps, and cloud-ready architectures.

Automotive modernization needs translators between legacy reality and modern architecture.

Asteyo is built for this exact modernization problem

We combine delivery experience, rare legacy capabilities, and scalable nearshore engineering from Romania.

Legacy

- COBOL & mainframe via our network
- Complex enterprise landscapes
- Stabilize business-critical systems
- Operational continuity & risk focus

Modern

- Java, web, integration, cloud-ready
- API, modularization, DevOps
- Pragmatic delivery focus
- Architecture-aware teams

Delivery

- Nearshore services from Cluj, RO
- 5,000+ Romanian IT professionals
- Track record with EU enterprises
- Reliability · People · Partnerships

asteyo

we get things done

Start with a modernization assessment

Map the legacy landscape, identify quick wins, and define a low-risk roadmap.