Fixed-scope robotics engineering,
done by engineers who've solved it before.

Ekumen Blueprints

EKUMEN
A GRID DYNAMICS COMPANY

TABLE OF CONTENTS

EKUMEN
A GRID DYNAMICS COMPANY

Engineering Foundations

The estimated project timelines throughout this document are approximate and may vary depending on the specific project.

EKUMEN
A GRID DYNAMICS COMPANY

Pipeline Ready

Pain point:
manual, slow QA and mounting technical debt.

THE SITUATION

All industries. Especially relevant for startups, research labs, and mid-size companies without in-house infrastructure.
Any application type.

01 WHAT WE TYPICALLY SEE

- CI has no clear owner, it's always someone's side project
- Tests exist, but they run manually, so they get skipped under deadline pressure
- Code changes break existing functionality with no automated way to catch it before merge
- There's no test strategy that accounts for overall quality

02 BEFORE WE START

- An existing test suite (even if it's currently run manually)
- A defined environment and scope
- Existing infrastructure to run CI on

03 WHAT EKUMEN DELIVERS

- We assess your existing CI setup and testing practices
- We provide a strategy report on how to improve quality assurance
- We build up to 3 automated pipelines that run your tests on every change, on the platform you already use (GitHub Actions, GitLab CI, etc.)

SNAPSHOT

DURATION:
~3 months
(access setup, infrastructure assessment, testing)

CORE STACK:
GitHub Actions
YAML
GitLab CI
Linux
Git
Shell Scripting

EKUMEN
A GRID DYNAMICS COMPANY

Build Foundations

Pain point:
QA challenges, key-person risk.

THE SITUATION

Teams with an existing codebase and no standardized way of building, testing, or onboarding new engineers to it. All robot types and industries.

01 WHAT WE TYPICALLY SEE

- Build times are slow and inconsistent, so nobody fully trusts a clean build reflects what's shipping
- Testing coverage depends on which engineer wrote the code, some parts well tested, most aren't
- Onboarding a new engineer takes weeks, the workflow isn't documented, it lives in senior engineers' heads
- Different local environments cause “works on my machine” bugs that eat into engineering time

02 BEFORE WE START

- Access to the current codebase and build system
- A defined subset of the codebase to prioritize (can't cover everything in one engagement)
- At least one engineer available to pair during the engagement, for knowledge transfer

03 WHAT EKUMEN DELIVERS

- We standardize the build system (colcon, Bazel, CMake, or whatever fits your stack) for reproducible builds
- We define a testing strategy (unit, integration, system-level) prioritizing coverage where it reduces the most risk
- We document the workflow and add guardrails (linting, pre-commit hooks, CI checks) so it's enforced automatically

SNAPSHOT

DURATION:

1 week
for a project built from scratch;

up to 3 months
for an existing project

CORE STACK:

Colcon · Bazel · CMake · CCache · Docker

EKUMEN
A GRID DYNAMICS COMPANY

ROS 2 Launchpad

Pain point:
migration, technical debt.

THE SITUATION

Teams migrating an existing robot, from ROS 1, a custom stack, or a prototype OS setup onto ROS 2 in production. All industries.

01 WHAT WE TYPICALLY SEE

- The OS wasn't set up with ROS 2's real-time/networking needs in mind (ad hoc distro, missing RT kernel, driver gaps)
- Nobody can fully reproduce the current OS/robot environment, so new deployments or hardware units are risky
- ROS 2 networking (DDS) and real-time behavior are unreliable because the OS isn't configured to support them

02 BEFORE WE START

- A defined target hardware/compute platform
- The current OS and driver setup, documented or accessible
- Access to hardware for testing and validation

03 WHAT EKUMEN DELIVERS

- We assess your current OS/hardware setup against ROS 2's real-time and networking requirements
- We prepare a reproducible OS image for your target hardware (RT kernel where needed, drivers, DDS-ready networking)
- Documentation and handover so your team can reproduce and maintain the environment going forward

EKUMEN
A GRID DYNAMICS COMPANY

Behaviors Design

Pain point:
fragile architecture, hard to extend, hard to test.

THE SITUATION

All robot types (AMRs, AGVs, drones, and more). All industries, especially logistics, inspection, and manufacturing.

01 WHAT WE TYPICALLY SEE

- The robot has no mission management layer
- New and current behaviors can't be validated early, and regressions can't be caught without expensive manual testing

02 BEFORE WE START

- A list of required behaviors, defined from the product side

03 WHAT EKUMEN DELIVERS

- Definition and implementation of a mission management layer
- Implementation of basic behaviors in the new architecture, including fallback and error-recovery
- A testing strategy, implemented

SNAPSHOT

DURATION:

Depends on the initial architecture and how many new behaviors are required. As a reference, a **3-month** engagement: **~1 month** analyzing the codebase and planning the testing approach (or planning/ laying the foundation for a mission management layer); 1 to 2 months on simulation integration, core test-case development and infrastructure, plus implementing the scoped behaviors

CORE STACK:

C++ · Python · ROS 2 · BehaviorTree.CPP · BehaviorTree ROS · SMACH · SMACC2 · gTest · gMock · Gazebo · Isaac Sim

EKUMEN
A GRID DYNAMICS COMPANY

Hardware and Middleware

Hardware Fit Check

Pain point:
sim-to-real gap, production readiness.

THE SITUATION

All robot types. All industries.

01 WHAT WE TYPICALLY SEE

→ Software runs fine in development, and each part works in isolation, but the system as a whole doesn't hit real-time performance targets on the target hardware.

02 BEFORE WE START

→ Target compute platform (x86 vs ARM)
→ Compute constraints: CPU, RAM, disk I/O, etc.

03 WHAT EKUMEN DELIVERS

→ Lambkin our internal, OSS, commercially-friendly benchmarking tool for localization packages (soon to include profiling too)
→ General profiling and micro-benchmarking of the SBOM

SNAPSHOT

DURATION:

Starting at
2 weeks
without custom platforms;
1 to 3 months
with custom platforms and optimizations

CORE STACK:

Lambkin | Perf | Google Benchmark

Middleware Match

Pain point:
scaling architecture.

THE SITUATION

All robot types (AGVs, AMRs, car-like, drones, quadrupeds, humanoids, robotic arms, and more). All industries.

01 WHAT WE TYPICALLY SEE

- Resource consumption issues
- System reliability issues

02 BEFORE WE START

- A specific compute platform
- A specific networking configuration
- A specific transport use case to address

03 WHAT EKUMEN DELIVERS

- Benchmarking, profiling, and improvement of the current DDS vendor selection, or a switch to another vendor (mirrors work current Ekumen engineers did at iRobot: github.com/irobot-ros/ros2-performance)

SNAPSHOT

DURATION:

2 weeks to 3 months

depending on complexity and scope (most engagements land around 1 month)

CORE STACK:

C++ | Python | ROS 2 | vendor-specific DDS tooling

EKUMEN
A GRID DYNAMICS COMPANY

Perception, Localization & Navigation

EKUMEN
A GRID DYNAMICS COMPANY

Localization Core

Pain point:
sim-to-real gap, field failures.

THE SITUATION

Ground or aerial mobile robots (vacuum cleaners, car-like, AGVs, AMRs, UGVs, drones, forklifts, quadrupeds, humanoids). Logistics, manufacturing, cleaning, delivery, surveillance, health, agriculture, automotive, forestry, mining.

01 WHAT WE TYPICALLY SEE

- You already have a localization solution, but it needs benchmarking, profiling, or improvement — or
- You're starting from scratch and need a localization solution researched and built

02 BEFORE WE START

- Sensor package: cameras, lidars, IMUs, wheel encoders, GPS, etc.
- Compute platform: x86 vs ARM
- Compute budget: CPU, RAM, disk I/O, etc.

03 WHAT EKUMEN DELIVERS

- Lambkin our internal, OSS, commercially-friendly benchmarking tool for localization packages
- Beluga (when it applies): our internal, OSS, commercially-friendly global localization framework

SNAPSHOT

DURATION:

2 weeks to 3 months
for benchmarking/ optimization;

1 to 3 months
for a baseline implementation

CORE STACK:

Python · C++ · ROS 2 · Cmake · Ethernet · EtherCAT · USB · Linux · Perf · Evo · Lambkin · Beluga

EKUMEN
A GRID DYNAMICS COMPANY

Nav Stack Sprint

Pain point:
sim-to-real gap, field failures.

THE SITUATION

Mobile robots (indoor or outdoor) that need autonomous navigation in real environments. Any industry.

01 WHAT WE TYPICALLY SEE

- Navigation works in simulation/demos but fails or is unpredictable once deployed (obstacles, dynamic environments, sensor noise the simulator didn't capture)
- The nav stack was tuned in simulation but doesn't transfer to the robot's real dynamics, kinematics, or sensors
- No systematic way to validate navigation before a field deployment, problems surface in front of the customer instead

02 BEFORE WE START

- A robot capable of running ROS 2 (or the target framework), with basic drivers already in place
- Access to the target environment, or a representative one, for testing
- A sensor suite installed and functioning (LiDAR, cameras, IMU, or similar)

03 WHAT EKUMEN DELIVERS

- We design and tune a navigation stack (localization, mapping, path planning, obstacle avoidance) fit to your robot
- We validate performance against real-world conditions using structured test scenarios
- Tuning guidelines and documentation so your team can adjust the stack going forward

SNAPSHOT

DURATION:

1 to 3 months
depending on scope;

up to 6 months
for full design, tuning, and field validation

CORE STACK:

ROS 2 | Nav2

localization and mapping solutions

Sensor drivers

EKUMEN
A GRID DYNAMICS COMPANY

Simulation & Testing

Sim Loop

Pain point:
no fast feedback loop for regression testing: slow, manual QA blocks both human and AI-driven development.

THE SITUATION

All robot types. All industries.

01 WHAT WE TYPICALLY SEE

- Neither developers nor AI coding agents can validate system-level behavior while iterating on features and fixes
- Regressions are only caught during long, costly rounds of manual testing
- AI agents have no way to close the loop during development, with no simulation to test against automatically

02 BEFORE WE START

- An existing simulation of the robot on a platform (Gazebo, Isaac Sim, or similar)
- A defined environment and scope
- Available infrastructure to run simulations in CI for regression testing

03 WHAT EKUMEN DELIVERS

- We take your existing robot simulation and optimize it for fast, resource-efficient use in a dev loop
- We instrument it and integrate it with your CI/CD pipeline for automated regression testing
- We add AI agent integration, so agents can run closed-loop development against the simulation

SNAPSHOT

DURATION:
3 months
(designing and implementing the testing framework)

CORE STACK:

C++ · Python · ROS · ROS 2 · Docker · GitHub Actions/ GitLab CI/Jenkins · Gazebo · Isaac Sim

EKUMEN
A GRID DYNAMICS COMPANY

Fleet & Field Operations

EKUMEN
A GRID DYNAMICS COMPANY

Fleet Sync

Pain point:
scaling from pilot to fleet.

THE SITUATION

Ground or aerial mobile robots. Logistics, cleaning, delivery, surveillance, health, agriculture, automotive, forestry, mining.

01 WHAT WE TYPICALLY SEE

- Robots operate as isolated systems, with no way to assign, track, or reroute tasks across the fleet
- Task dispatch is manual and doesn't scale as the fleet grows

02 BEFORE WE START

- Robots that already support (or can be adapted to) VDA 5050 or Open-RMF — if not, that's the Fleet Uplink case, not this one.
- A defined site/environment where the robots operate
- Robots capable of autonomous navigation and localization between points
- An API on the robots for task execution (navigation, battery, pickup, charging)
- A known compute footprint per robot

03 WHAT EKUMEN DELIVERS

- We implement a fleet management layer scoped to robot tracking and simple navigation tasks, using Open-RMF
- Analysis and implementation for secure infrastructure
- A basic dashboard for fleet operators
- Next step (beyond this engagement): custom task development and process integration

SNAPSHOT

DURATION:

3 months

CORE STACK:

Open-RMF · VDA 5050 · MQTT · Docker · TypeScript · Python · Rust · Go

TEAM:

2 people:

1 senior engineer leading the integration,

1 junior engineer for QA testing

EKUMEN
A GRID DYNAMICS COMPANY

Fleet Uplink

Pain point:
the robot can't join a fleet management system because it doesn't speak the protocol required.

THE SITUATION

Same robot types and industries as Fleet Sync.

01 WHAT WE TYPICALLY SEE

- The robot doesn't speak the protocol a fleet management system needs (e.g., missing VDA 5050 or Open-RMF)
- Even when a fleet management system exists, the robot can't be dispatched or tracked because of this gap

02 BEFORE WE START

- The fleet manager needs an API to dispatch tasks and retrieve robot status

03 WHAT EKUMEN DELIVERS

- We implement the missing protocol layer directly on the robot (VDA 5050, Open-RMF, or whatever's required), so it integrates without custom one-off work

SNAPSHOT

DURATION:
3 months

CORE STACK:
C++ · Docker · Python · MQTT · DDS

EKUMEN
A GRID DYNAMICS COMPANY

Remote Cockpit

Pain point:
production readiness, operational stability.

Ground or aerial mobile robots. Logistics, cleaning, delivery, surveillance, health, agriculture, automotive, forestry, mining.
Bundles four related upgrades to a teleoperation setup, each can be scoped independently.

MODULE A: WEB-BASED NAVIGATION & SLAM MONITORING

Duration: 8–12 weeks

- **Problem:** Client has Nav2/SLAM configured, but the only way to send goals and monitor mapping is RViz on a dedicated workstation. They need it from any browser.
- **Delivers:** Click-to-navigate from a browser; real-time map updates with path planning and costmap overlays; navigation status feedback and full integration testing.

MODULE B: MULTI-OPERATOR SESSION MANAGEMENT

Duration: 2–4 weeks

- **Problem:** Multiple operators drive the same robot at different times with no coordination, conflicting commands or silent takeovers.
- **Delivers:** A session manager with ping-based heartbeats, a temporal assignment table, visual operator-status indicators, lock/unlock and graceful handoff.

MODULE C: DESKTOP APP PACKAGING (ELECTRON)

Duration: 3–6 weeks

- **Problem:** Client needs the teleop interface as a native installable app across Windows/macOS/Linux — not a browser tab — with offline capability and auto-updates.
- **Delivers:** An Electron wrapper with platform-specific installers (.msi, .dmg, .AppImage/.deb), auto-update, system tray, configurable connection profiles.

MODULE D: FULL-APPLICATION GAMEPAD NAVIGATION

Duration: 2–4 weeks

- **Problem:** Operators must switch to mouse/keyboard for UI actions, breaking focus during critical moments.
- **Delivers:** A gamepad navigation layer mapping buttons/D-pad to all UI actions, so operators never leave the controller.

BEFORE WE START

- A ROS 2 robot with navigation and localization already configured
- An existing teleoperation stack on the robot
- An existing deployment solution to hook into our deployment lifecycle

CORE STACK:

JavaScript · TypeScript · C++ · Python · ROS · ROS 2 · WebSockets · HTTP · Database · Docker

Health Check

Pain point:
field failures, excessive manual intervention.

THE SITUATION

Robots deployed in the field, especially where physical access is costly or slow. Any industry.

01 WHAT WE TYPICALLY SEE

- When a robot fails in the field, there's no way to know what happened without physically accessing it or digging through raw logs
- Problems are discovered only when the robot stops working, with no early warning
- Diagnosing an issue depends on tribal knowledge, most failures escalate to the same senior engineers regardless of what broke

02 BEFORE WE START

- Robot software running ROS 2 (or comparable middleware) with access to internal state/topics
- A defined list of critical subsystems or failure modes to prioritize
- Existing infrastructure to host the dashboard and alerting, or a lightweight hosted option

03 WHAT EKUMEN DELIVERS

- We implement a diagnostic system that continuously monitors hardware/software subsystems and reports health status
- We define alerting for early warning signs, not just failure states, so issues get caught before downtime
- A dashboard so field or support teams can triage issues themselves, without escalating

SNAPSHOT

DURATION:
3 months

CORE STACK:

- ROS 2 diagnostics stack (diagnostic_updater, diagnostic_aggregator)
- Prometheus/Grafana
- MQTT for telemetry

EKUMEN
A GRID DYNAMICS COMPANY

Observability

Black Box

Pain point:
observability,
key-person risk.

THE SITUATION

Any industry.

01 WHAT WE TYPICALLY SEE

- A user performs a critical action and gets a 500 error, or a 15-second load time, with no way to see why
- An autonomous vehicle raises an error during a lane-change maneuver, and there's no information about the different sensors from the vehicle to understand what happened or improve its performance
- A memory leak is slowly consuming RAM, or a DB connection pool is quietly filling up, with nothing flagging it
- The application logs an error like NullReferenceException at line 42, with no trace of what led to it

02 BEFORE WE START

- A production-grade system written in .NET, C++, Elixir, Go, Java, JavaScript, TypeScript, Kotlin, PHP, Python, Ruby, or Rust
- Infrastructure already in place (cloud provider, Kubernetes, bare metal, etc.)

03 WHAT EKUMEN DELIVERS

- We implement observability with OpenTelemetry: tracing, logging, and metrics
- We deploy Prometheus, Grafana, Jaeger, Loki, and/or similar, or cloud-provided aggregation tools (AWS, GCP)

SNAPSHOT

DURATION:

Roughly
3 months,
1 month
infra setup,
2 months
service implementation
+ dashboard configuration.

CORE STACK:

C++ · Go · Java · Kotlin · Python · Ruby · Rust · OpenTelemetry · Loki/ELK · Prometheus/Grafana · Jaeger

EKUMEN
A GRID DYNAMICS COMPANY

EKUMEN
A GRID DYNAMICS COMPANY

ekumenlabs.com

ekumenlabs.com/research