Escuela de Ingeniería Informática UNIOVI

Architecture Modernization

A Practical Approach

Javier Menéndez Osendi, Fernando Sutil Fernández, Pablo Roces Pérez, Sergio Riesco Collar

11-4-2025

Contenido

Introduction to Architecture Modernization

Software systems age. As businesses grow, new technologies emerge, and customer needs shift, old systems can become barriers. They slow down development, raise costs, or block new opportunities. Architecture modernization tackles this problem. It updates systems to meet current demands while preparing for future goals. Nick Tune and Jean-Georges Perrin, in their Software Engineering Radio podcast, share insights on how to approach this process. They describe modernization as a mix of technical and social efforts, driven by business needs, not just code fixes.

Modernization isn't a single step or a rigid plan. It's a set of choices. Teams might replace outdated tools, rethink how data flows, or adjust team structures. The goal is clear: remove obstacles that stop a business from moving forward. For example, a legacy system might make adding a simple API costly, forcing a company to say no to a valuable partner. Modernization solves this by aligning the system with what the business wants—whether that's entering new markets or cutting support costs. Tune and Perrin stress that success depends on understanding why modernization matters and balancing it with other priorities.

The podcast highlights four key areas. First, the initial approach focuses on assessing what's broken and why it matters to the business. Second, data engineering plays a big role, as outdated data practices can cripple progress. Third, balancing modernization with business priorities ensures teams don't neglect new features for the sake of upgrades. Finally, risks like incomplete migrations need careful planning to avoid worse problems than before. Each area requires teams to think beyond code to people, processes, and goals.

Why Data Matters

Data often drives modernization. Perrin explains that old data systems, like on-premises warehouses, struggle in the cloud era. Moving them without rethinking their design leads to poor results. For instance, a system optimized for SQL Server won't thrive on Redshift without changes. Regulations, like GDPR, also push companies to update how they handle data. Modern approaches, like data mesh, help by treating data as a product, making it easier to manage and use.

Balancing Priorities

Modernization competes with building new features. Tune warns that teams often favor features because they're visible and rewarded. To counter this, leaders must agree on modernization's value—whether it's saving money or enabling growth. Perrin adds that in data projects, measuring success is tough. Teams should set

clear goals, like better compliance, and track them to prove progress. Both suggest creating multiple plans to show trade-offs, helping everyone see what's at stake.

Managing Risks

Things can go wrong. Tune notes that many projects stall when priorities shift, leaving systems half-modernized and more complex. Another risk is lacking skills—teams may not know how to break apart tightly coupled code. Perrin emphasizes delivering small wins to keep momentum. Planning helps, but discipline matters more. Teams must stick to their goals, even when tempted by quick feature requests.

Conclusion

Architecture modernization isn't just about new tech. It's about making systems serve the business better. Tune and Perrin offer a grounded view: start with a clear reason, involve data, align priorities, and plan for risks. Done right, modernization doesn't just fix old problems—it opens new possibilities.