Architecture Review

Independent Evaluation of Architecture, Maintainability and Structural Risks

An anonymized excerpt from a real architecture validation report.

Conducted by Nicolai Wolko - WBK Consulting AG - Heise Author - Switzerland

Representative Example (2025)

Table of Contents

Table of Contents 2

Executive Summary 3

- Scope and Limitations 3
- Overall Architecture Rating 3
- Key Recommendations 4

Project Context 5

- Application Scope and Business Overview 5
- Key Metrics 5
- Primary Objectives 5

Architecture Assessment 6

- Layer Structure and Dependency Analysis 6
- Maintainability and Code Quality Observations 7
- Module Boundaries and Shared Libraries 8

Risk Evaluation 8

- Technical Debt Classification 8
- Dependency Cycles and Coupling Analysis 9
- Test Coverage and Validation Maturity 10

Key Findings and Refactoring Recommendations 10

- Separation of Concern Violations 11
- Missing State Management 13
- Very high coupling 14

Project Context

Application Scope and Business Overview

Example Company, based in Switzerland, is a proptech company providing a specialized SaaS solution.

Current development team stated the architectural design of the application is broken and out of date. Time to production of feature tickets has increased dramatically - quality of application is decreasing. There are bugs in productive, which the developer team cannot address.

Goal of this review was to evaluate the architectural health, maintainability and modular consistency of the codebase to provide actionable recommendations for refactoring and governance.

Key Metrics

Metric	Value	Comment
Lines of Code (LOC)	~ 85’000	Core + Frontend Modules + Shared Libraries
Angular Version	12	Out of date
Team Size	7 Developers	Distributed, last initial developer quit 3 months ago
Primary Stack	Angular, Typescript	
Last Audit	None	No former architecture validation before

Primary Objectives

1. Determine how closely the current structure aligns with intended architectural boundaries.
2. Identify design and implementation factors that negatively affect productivity and long-term stability.
3. Quantify the scope of structural problems and provide an evidence-based roadmap for refactoring.

Separation of Concern Violations

Observation

The reviewed Angular codebase shows extensive business logic within UI components.

Several core workflows (validation, data transformation and orchestration between services) are implemented directly in component classes rather than delegated to dedicated services or domain layers.

A base class named ExampleProductPageComponent centralizes common logic such as data loading, state tracking, and utility methods.

All major page components inherit from this class, resulting in:

- Strong coupling between components and shared utilities
- Implicit dependencies on internal component state
- Deep inheritance hierarchies (up to 3 levels)
- Component files exceeding 2 000 - 3 000 LOC

This pattern leads to tight coupling between UI and business logic, making components difficult to test, extend, and refactor.

Refactoring effort increases non-linearly with feature growth, as logic is repeated or overwritten in subclasses.

Root Cause

It appears likely that the initial development was conducted by a team with limited architectural experience or insufficient familiarity with Angular’s design principles. Foundational architectural decisions were made early without long-term maintainability in mind. Over subsequent iterations, systematic refactoring and technical debt reduction were neglected, resulting in accumulated structural erosion.

This aspect is critical when evaluating whether the application should be refactored or reimplemented.

If the underlying causes - missing architectural ownership and lack of clean design principles - are not addressed in a potential rewrite, the new system is expected to reproduce the same structural deficiencies within a short time frame.

This architectural drift has also reduced feature delivery speed and created uncertainty in planning, as technical issues increasingly dominate sprint goals.