GlobalLogic®

Architecture Compliance Using ArchUnit

by Ashutosh Gupta, Sr. Solution Architect, GlobalLogic

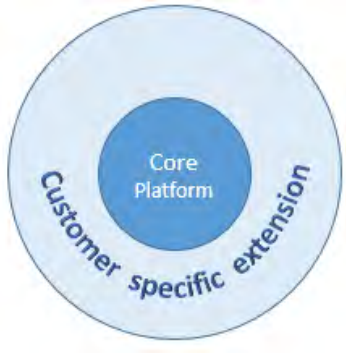
Contents

Why Automate Architecture Compliance?

I was once involved in a project where a services team customized a core product for different customers.

Over time, the team began overriding the service and BPM layers written in the core product instead of using an adapter to customize the functionality.

The team's motivation was to deliver the project swiftly, but there was a tradeoff between delivery time and maintainability.



Using customer-specific extensions to customize core functionality

Overriding the core business services layer lead to instability over time and a nightmare for the team. If you need to upgrade a customer to the latest version of a core product, it would mean changing multiple functions in the classes in the service and BPM layers to match the project's functionality.

There are multiple solutions to this problem, such as making the service class final in the core project, rendering it impossible for service to override the class. Unfortunately, that is not practical when it comes to the domain. You may have an abstract service, which may have specialized implementations.

Imagine for a moment there was a possibility to automate the architectural decision that could govern and alert anyone overriding services or BPM layers in services.

It would force the services team to customize functionality using the base product's adapters. It would be a different story altogether. As a team, you can save a tremendous amount of time and money while increasing quality.

You make several architectural decisions during the lifecycle of a project. Architects need to ensure compliance with architectural decisions. Ensure compliance by performing checks manually or with the help of automation. Some compliance checks require manual interventions (e.g., peer code review). As a team grows, compliance through manual checks becomes cumbersome.

It is a good idea to automate the compliance check, wherever possible, right from the beginning of a project.

How automation can impact your project

When developers of various experience levels join your team—from a trainee just out of college to experienced experts—you can guarantee that everyone in the project respects the architectural decisions with an automation tool like ArchUnit.

Benefits of automating architectural compliance can include:

High Maintainability: When all team members are bound to adhere to architectural decisions, it ensures that everyone can quickly pinpoint the layers and segregate changes, resulting in high maintainability.

Architectural Quality: Enforcing the architectural quality within the CI/CD pipeline helps the team automatically adhere to architectural compliance, making governance easier. It would help prevent a decline in quality by publishing metrics that would otherwise be very hard to determine.

Time Savings: The process of reviewing architecture governance manually within the project is both time-consuming and error-prone. Automating the architecture governance rules helps achieve efficiency.

ArchUnit: An Architecture Compliance Framework

ArchUnit is an open source framework that uses a Java unit test framework such as JUnit or TestNG for automating compliance checks in projects developed in Java. ArchUnit can help enforce compliance including:

- Coding rules, eg.: No class should throw generic exceptions, or conventions such as Logger should be private, static final, etc.
- Naming conventions: e.g.: interfaces should end with the word “Interface.”
- Security: e.g.: the controller should always annotate a secured interface with @Secured spring.
- Compliance in layered and onion architecture, e.g.: the controller should not call a persistence layer directly.
- Checking cyclic dependencies.
- Adherence of classes to UML diagrams.

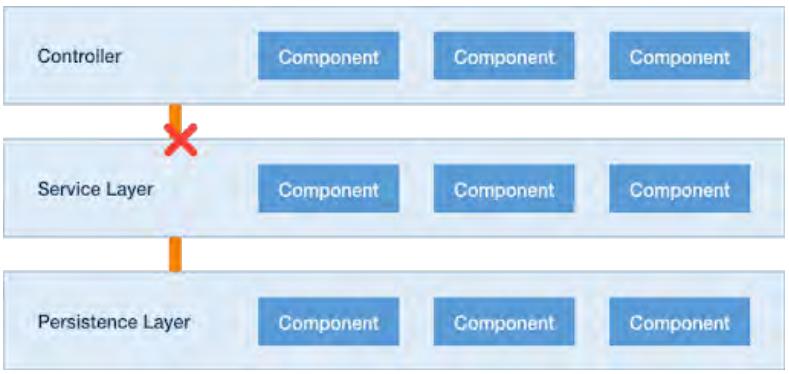
Detecting violations in layered architecture: an example

Consider an example where a new developer tries to use a controller to access the persistence layer instead of calling the service layer.

To utilize ArchUnit in your project, include ArchUnit core dependency in the project’s maven pom or Gradle. The following example adds the dependencies in Gradle.

build.gradle

```
dependencies {
    testImplementation
'com.tngtech.archunit:archunit:0.14.1'
}
```



Layered architecture showing violation

```
@AnalyzeClasses(packages = "com.mycompany.layers")
public class LayeredArchitectureTest {


        @ArchTest
        static final ArchRule layer_dependencies_are_respected =
        LayeredArchitecture()
              .layer("Controllers").definedBy("com.mycompany.layers.controller..")
              .layer("Services").definedBy("com.mycompany.layers.service..")
              .layer("Persistence").definedBy("com.mycompany.layers.persistence..")

              .whereLayer("Controllers").mayNotBeAccessedByAnyLayer()
              .whereLayer("Services").mayOnlyBeAccessedByLayers("Controllers")
              .whereLayer("Persistence").mayOnlyBeAccessedByLayers("Services");
}
```

The above test ensures that classes defined in the controller layer (i.e., defined in the controller package) are calling classes in services (i.e., defined in the services package), and classes in services are calling classes in the persistence layer (i.e., defined in the persistence package).

It also ensures that classes in the controller layer are not accessed by any other layer, ensuring everybody follows layered architecture.

- Through @AnalyzeClasses annotation, the developer provides a list of packages, separated by a comma, and scans it while executing the test.
- You use @ArchUnit annotation to specify classes that JUnit picks up automatically.
- ArchRule class defines the ArchUnit rule. You typically use a factory method within ArchRuleDefinition to define rules.
- LayeredArchitecture() is used to assert a typical layered architecture, e.g., which layers exist and which layer can access another.
- layer() Defines a new layer in layered architecture and where each layer resides in a specific package. The two dots represent any number of packages.
- whereLayer() and mayOnlyBeAccessedByLayers() define accessibility rules. E.g., in the above example, the rule checks that only the controller can access the service layer.
- Use mayNotBeAccessedByAnyLayer() to define rules that any specified layer cannot access a specific layer.
- ArchUnit provides many such APIs to define different rules for the architecture.

License

ArchUnit comes under Apache License 2.0. It uses:

ASM: An all-purpose Java bytecode manipulation and analysis framework. Use it to modify existing classes or dynamically generate classes directly in binary form.

Google Guava: A set of core Java libraries from Google that includes new collection types (such as multimap and multiset), immutable collections, a graph library, and utilities for concurrency, I/O, hashing, caching, primitives, strings, and more.

Conclusion

Automating architecture and coding rules through open source libraries like ArchUnit enforces the defined rules. Once you integrate automation with unit test cases within the build pipeline, the team is required to follow guidelines they might otherwise unknowingly violate.

References

- https://www.archunit.org/
- https://www.archunit.org/userguide/html/000_Index.html
- https://github.com/TNG/ArchUnit

About The Author

Ashutosh Gupta, Sr. Solution Architect, GlobalLogic is a technology enthusiast with 20 years of experience. He has worked with GlobalLogic India Pvt Limited for more than 13 years and has significant experience in the design and architecture of high-volume, load-balanced solutions in distributed applications. He is passionate about building solutions using cutting-edge open source technologies.

GlobalLogic®

GlobalLogic is a leader in digital product engineering. We help our clients design and build innovative products, platforms, and digital experiences for the modern world. By integrating strategic design, complex engineering, and vertical industry expertise,— we help our clients imagine what's possible and accelerate their transition into tomorrow's digital businesses. Headquartered in Silicon Valley, GlobalLogic operates design studios and engineering centers around the world, extending our deep expertise to customers in the communications, automotive, healthcare, technology, media and entertainment, manufacturing, and semiconductor industries.

www.globallogic.com