Luxoft
A DXC Technology Company

{Software Architecture: as code}

Diagrams & Qualities as code

About me

Short bio

- I am a technical team lead at Luxoft a DXC Technology Company.
- As core value I am a lifelong learner with focus on progress (not on perfection) and following the split-apply-combine strategy approach.
- I have more than 18 years of experience in Java Enterprise complex projects. I started with an ERP project then I have gone through e-governments projects for 10 years and for the last 7 years I have been here at Luxoft working on banking projects.
- As part of my work, I often put on the solution architect hat and dive into architectural thinking, working with team members and peers on frameworks, design patterns, best practices in order to build the bridge between the business change/request and the technology solution and outline each of the phases and requirements required to make that solution work.

Software Architecture

Some thoughts

- (un)fortunately, there is no consensus about the definition, so I may share some thoughts
- “Architecture is about the important stuff. Whatever that is.” –Ralph Johnson
- Architecture is a way of thinking, it is a mindset, to solve something, to tell in a simple way an intricate story, to build up a model, to make it easy to transmit to the others what we are intending/proposing in order to solve our problem. Software architecture deals with abstraction, with decomposition and composition, with relationships & interoperability, with style.

Software diagrams as code

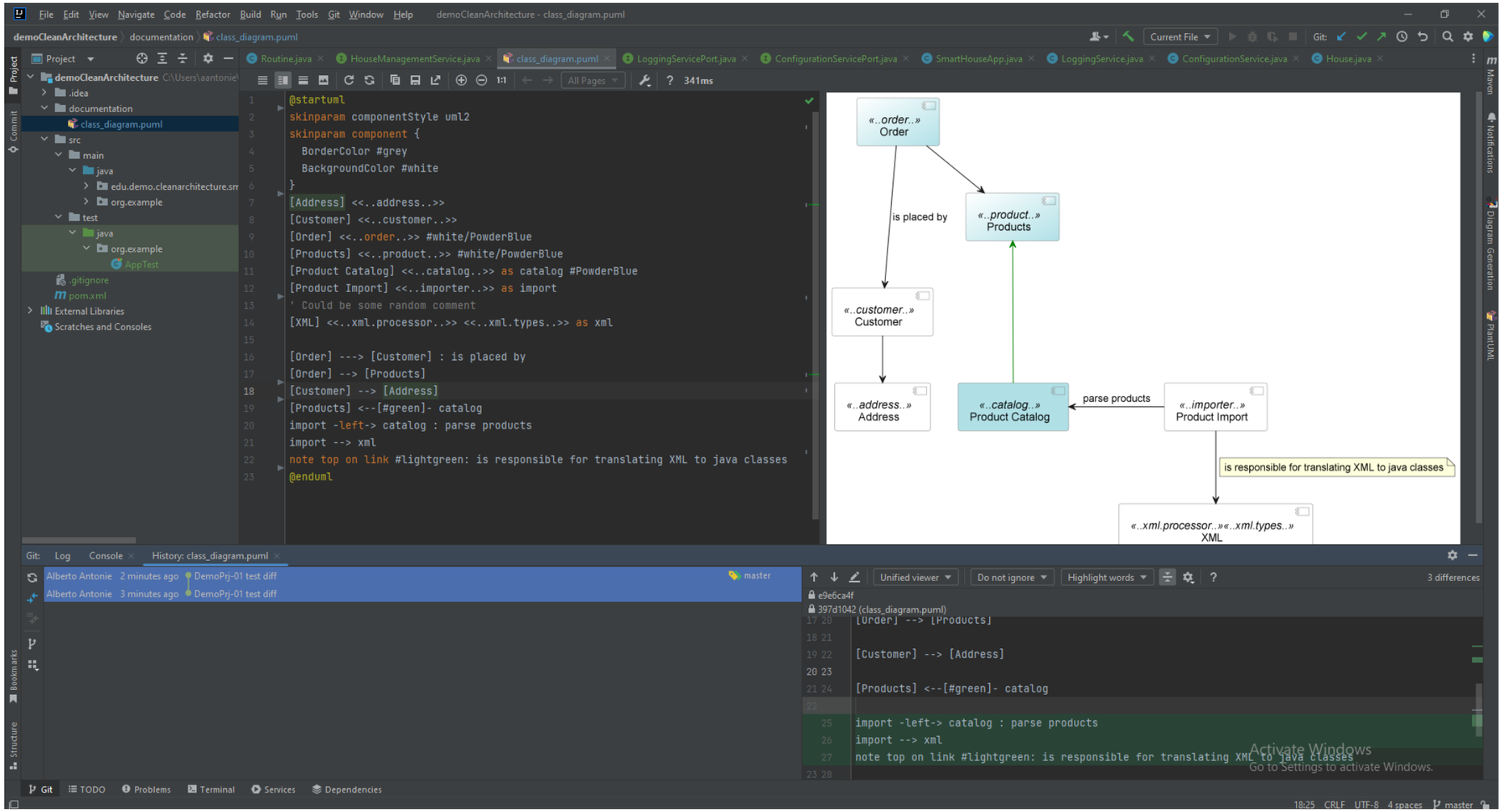
- As software engineers, when addressing software architecture, we all have to create architecture diagrams to convey our ideas with other fellow engineers and stakeholders.
- These days we use everything as code approach for documentation, infrastructure provisioning, CI/CD templates, and many other things.
- Creating architecture diagrams has now the support for 'as code' approach.

Software diagrams as code

Use case

- *Diagram as Code* allows us to track the architecture diagram changes in any version control system.
- **PlantUML** is used to draw UML diagrams, using a simple and human readable text description.
- We can use this tool for visual documentation purposes as it has in-built support to a class, object, sequence, activity, and use case diagrams. It helps us to create a visual representation of the abstract ideas/concepts of the system’s design.
- The PlantUML is compatible with all java-based IDEs: IntelliJ, Eclipse
- The beauty of this approach is that it enables software engineers to use the same tools as for software development thus benefiting also of version control for clear and easy tracing architecture design changes over time.

Software diagrams as code



Software Architecture as code

- “Truth can only be found in one place: the code.” –Robert C. Martin, Clean Code: A Handbook of Agile Software Craftsmanship
- Nowadays software architecture is not only a professional specialization anymore but increasingly it is more of a function in software engineering.
- One of the perils of any software solution is the software degradation/software entropy, understood as insufficient care in maintaining the coherence with product design and established design principle.
- Software architecture can and must enforce architecture fitness test which validates architecture design over time and provides continuous feedback for architectural conformance and informs the development process as it happens rather than after the fact. Archunit framework offers the capabilities for software architecture fitness testing.
- “ArchUnit is a free, simple and extensible library for checking the architecture of your Java code using any plain Java unit test framework. That is, ArchUnit can check dependencies between packages and classes, layers and slices, check for cyclic dependencies and more.” - https://www.archunit.org/

Software Architecture as code

Architectural style conformity test

Hexagonal architectural style

```java
@AnalyzeClasses(packages = {
        "edu.demo.cleanarchitecture.smarthouse"
})
public class HexagonalArchitectureStyleTest {

    @ArchTest
    static final ArchRule onion_architecture_is_respected = onionArchitecture()
            .domainModels("..domain.model..")
            .domainServices("..domain.service..")
            .applicationServices("..application..")
            .adapter("config", "..framework.api.adapters.config..")
            .adapter("instrumentation", "..framework.api.adapters.instrumentation..")
            .adapter("instrumentation.web", "..framework.api.adapters.instrumentation.web..")
            .adapter("logging", "..framework.spi.adapters.logging..")
            .adapter("repository", "..framework.spi.adapters.repository..");

    @ArchTest
    SliceRule adapters_Should_Not_Depend_On_Each_Other =
            SlicesRuleDefinition.slices() Creator
                    .matching("..framework.(**)..") GivenSlices
                    .should().notDependOnEachOther() SliceRule
                    .ignoreDependency(nameMatching(".*repositories.*"),nameMatching(".*entities.*"));
```

Layered architectural style

DashboardService.java

```java
@AnalyzeClasses(packages = {
        "edu.demo.cleanarchitecture.smarthouse.framework.api.adapters"
})
public class LayeredMVCArchitectureStyleTest {

    @ArchTest
    static ArchRule layered_MVC_Architecture = layeredArchitecture().consideringAllDependencies()
                    .layer("model").definedBy("..instrumentation.web.model..")
                    .layer("view").definedBy("..instrumentation.web.view..")
                    .layer("controller").definedBy("..instrumentation.web.controller..")
                    .layer("service").definedBy("..instrumentation..")
                    .whereLayer("model").mayOnlyBeAccessedByLayers("controller")
                    .whereLayer("view").mayOnlyBeAccessedByLayers("controller")
                    .whereLayer("service").mayOnlyBeAccessedByLayers("controller");
```

LayeredDAOArchitectureStyleTest.java

```java
@AnalyzeClasses(packages = {
        "edu.demo.cleanarchitecture.smarthouse.framework.spi.adapters.repository"
})
public class LayeredDAOArchitectureStyleTest {

    @ArchTest
    static ArchRule layered_DAO_Architecture = layeredArchitecture().consideringAllDependencies()
                    .layer("entity").definedBy("..db.entities..")
                    .layer("dao").definedBy("..db.repositories..")
                    .layer("service").definedBy("..")

                    .whereLayer("entity").mayOnlyBeAccessedByLayers("dao", "service")
                    .whereLayer("dao").mayOnlyBeAccessedByLayers("service");

    @ArchTest
    static ArchRule entities_must_be_suffixed_in_correct_package =
```

Software Architecture as code

Project development practices compliance test

```java
public class ProjectDevelopmentPracticesTest {

    @ArchTest
    ArchRule classes_should_not_throw_generic_exceptions = NO_CLASSES_SHOULD_THROW_GENERIC_EXCEPTIONS;

    @ArchTest
    ArchRule classes_should_not_use_java_util_logging = NO_CLASSES_SHOULD_USE_JAVA_UTIL_LOGGING;

    /*(that is, the System.out, System.err, and printStackTrace methods: use a logging library instead)*/
    @ArchTest
    ArchRule classes_should_not_use_standard_console_printing = NO_CLASSES_SHOULD_ACCESS_STANDARD_STREAMS;

    /*(use this when you want the team use the modern java.time API instead)*/
    @ArchTest
    ArchRule classes_should_not_use_joda_time = NO_CLASSES_SHOULD_USE_JODATIME;

    @ArchTest
    SliceRule classes_must_be_free_of_cycles = SlicesRuleDefinition.slices()
            .matching( packageIdentifier: "..(**)..").should().beFreeOfCycles();

    @ArchTest
    static ArchRule util_Classes_Must_Have_Only_Private_Constructor = classes() GivenClasses
            .that().haveSimpleNameEndingWith( s: "Util").or().haveSimpleNameEndingWith( s: "Utils") GivenClassesConjunction
            .should() ClassesShould
            .haveOnlyPrivateConstructors() ClassesShouldConjunction
            .allowEmptyShould( b: true);
    @ArchTest
    static ArchRule all_Classes_Must_Not_Have_Public_Fields = fields().that() FieldsThat<capture of ? extends GivenFieldsConjunction>
            .areDeclaredInClassesThat().areNotEnums().and().areDeclaredInClassesThat().haveNameNotMatching( s: ".*Test") ca
            .should().bePrivate() capture of ?
            .orShould().beProtected() capture of ?
            .andShould().notBePublic();
    @ArchIgnore
    @ArchTest
```

Software Architecture as code

PlantUML diagram class dependencies test

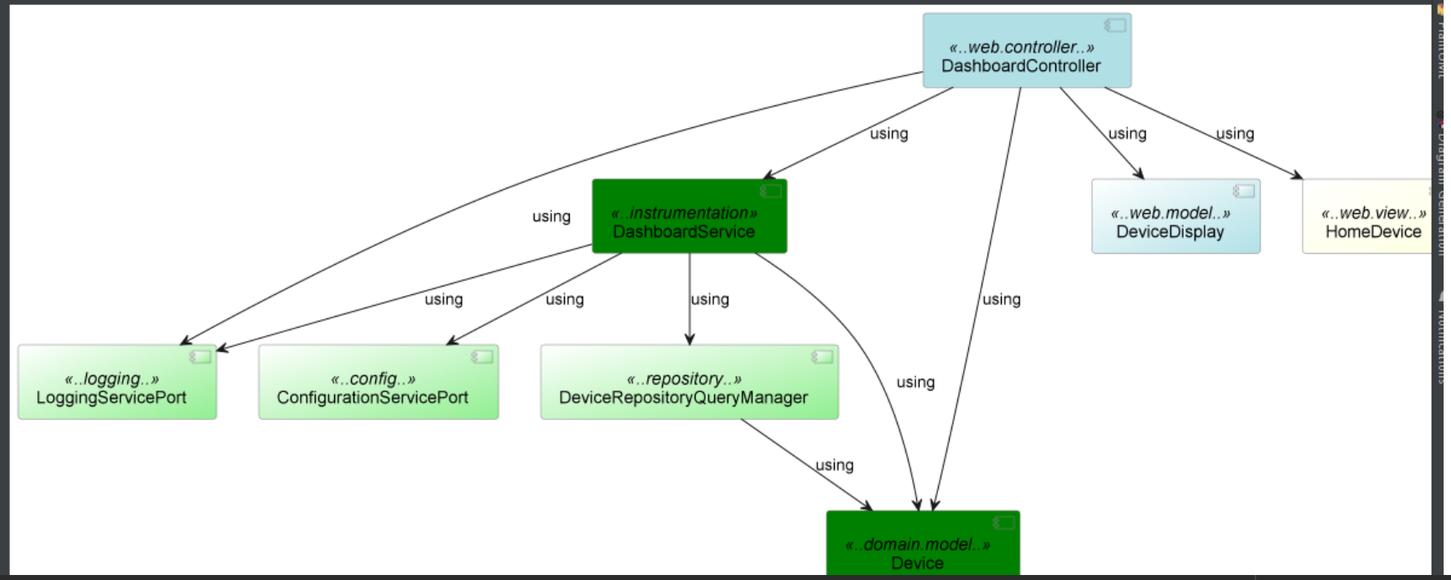
```
  BackgroundColor #white
}

[ConfigurationServicePort] <<..config..>> #white/LightGreen
[LoggingServicePort] <<..logging..>> #white/LightGreen
[DeviceRepositoryQueryManager] <<..repository..>> #white/Li
[DashboardService] <<..instrumentation>> #Green
[DashboardController] <<..web.controller..>> #PowderBlue
[Device] <<..domain.model..>> #Green
[DeviceDisplay] <<..web.model..>> #white/PowderBlue
[HomeDevice] <<..web.view..>>  #white/LightYellow

[DashboardController] --> [LoggingServicePort] : using
[DashboardController] --> [DashboardService] : using
[DashboardController] --> [DeviceDisplay] : using
[DashboardController] --> [HomeDevice] : using
[DashboardController] --> [Device] : using

[DashboardService] --> [Device] : using
[DashboardService] --> [ConfigurationServicePort] : using
[DashboardService] --> [LoggingServicePort] : using
[DashboardService] --> [DeviceRepositoryQueryManager] : us
[DeviceRepositoryQueryManager] --> [Device] : using


@enduml
```




```
public class DashboardWebControllerClassDependencyUMLTest {

    1 usage
    private final JavaClasses classes = new ClassFileImporter().importPackages("edu.demo.cleanarchitecture.smarthouse");
    1 usage
    private final URL plantUmlDiagram = DashboardWebControllerClassDependencyUMLTest.class.getResource( s: "/edu/demo/cleanarchitecture/smarthouse/

    @Test
    public void classes_should_adhere_to_dashboard_web_controller_considering_only_dependencies_in_diagram() {
        classes().that().doNotBelongToAnyOf(HouseManagementService.class) GivenClassesConjunction
                .should(adhereToPlantUmlDiagram(plantUmlDiagram, consideringOnlyDependenciesInDiagram())
                        .ignoreDependenciesWithTarget(GET_PACKAGE_NAME.is(PackageMatchers.of( ...packageIdentifiers: "", "java.."))
                                .as( description: "that is part of JDK"))
                ) ClassesShouldConjunction
                .check(classes);
    }
}
```

Luxoft
A DXC Technology Company

Software Architecture as code

Closing thoughts

- Having software architecture models created using code so using the same tools as for software development, they are also versionable alongside your codebase
- This kind of tooling enables what I would call Mob design, diagrams can be automatically generated and refreshed as you type. Using Plantuml and a large shared screen, it is very convenient for a team to brainstorm and create/visualize several architecture scenarios
- The diagrams and documentation as code provides support for a more accurate, up-to-date, even daring to living software architecture diagrams that actually reflect the code
- ArchUnit as software architecture fitness testing provides active capabilities for verifying predefined application architecture characteristics and architectural constraints as it is written and runs as a unit test that gives developers and application architects fast feedback on their work.
- It also allows a team to define its own best practices and architecture rules and having architectural decisions documented with a reason as code in repository
- The presented tools can address the Model-code gap concern providing the support to guard/keep code and model in synch through the architecture-fitness tests and feedback to diagrams compliance

Software Architecture as code

Resources

- https://github.com/antoniealberto/demoCleanArchitecture
- https://www.archunit.org
- https://plantuml.com/
- https://github.com/TNG/ArchUnit-Examples
- https://beza1e1.tuxen.de/definitions_software_architecture.html
- https://resources.sei.cmu.edu/asset_files/Presentation/2015_017_101_438772.pdf
- https://blogs.oracle.com/javamagazine/post/unit-test-your-architecture-with-archunit
- https://medium.com/ing-blog/architecture-and-design-unit-testing-using-archunit-f3ba6b5ae03b

Thank you!

Q&A