JavaCro'19

SV GROUP

Keeping your architecture clean with ArchUnit

Alen Kosanović
alen.kosanovic@svgroup.hr
Umag, May 13th 2019

Software architecture during inception

A well suited architecture is picked

Coding rules are defined

These rules are written in a document

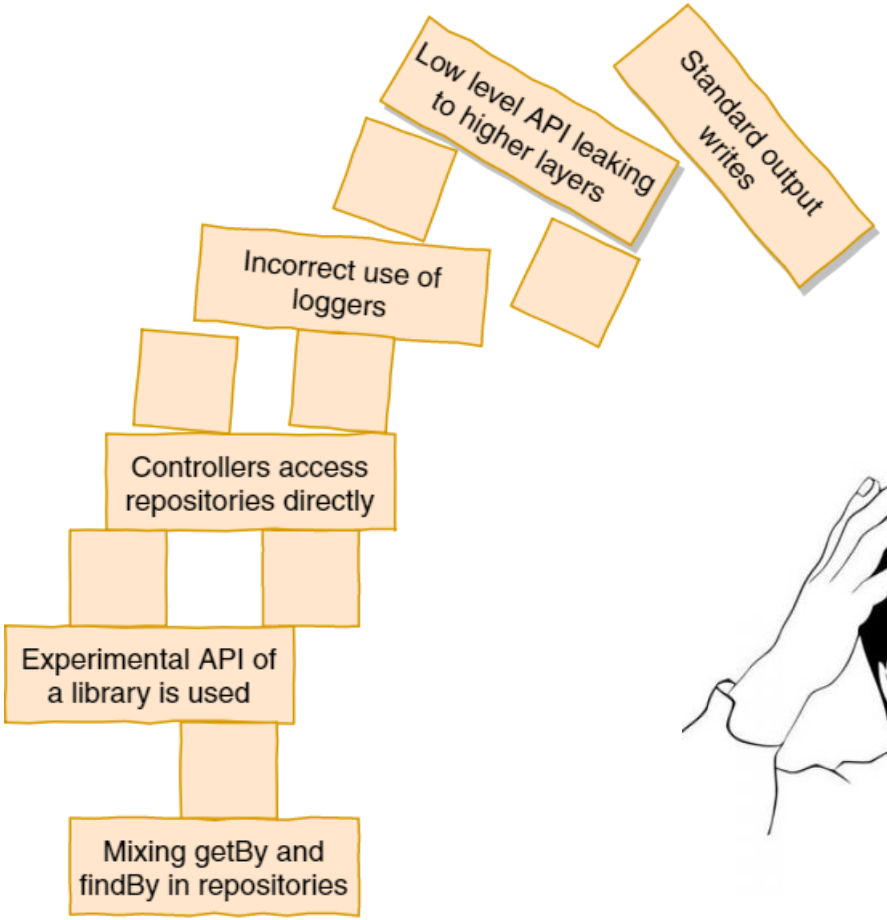
They are also presented to all the developers

Code reviewing will also be performed

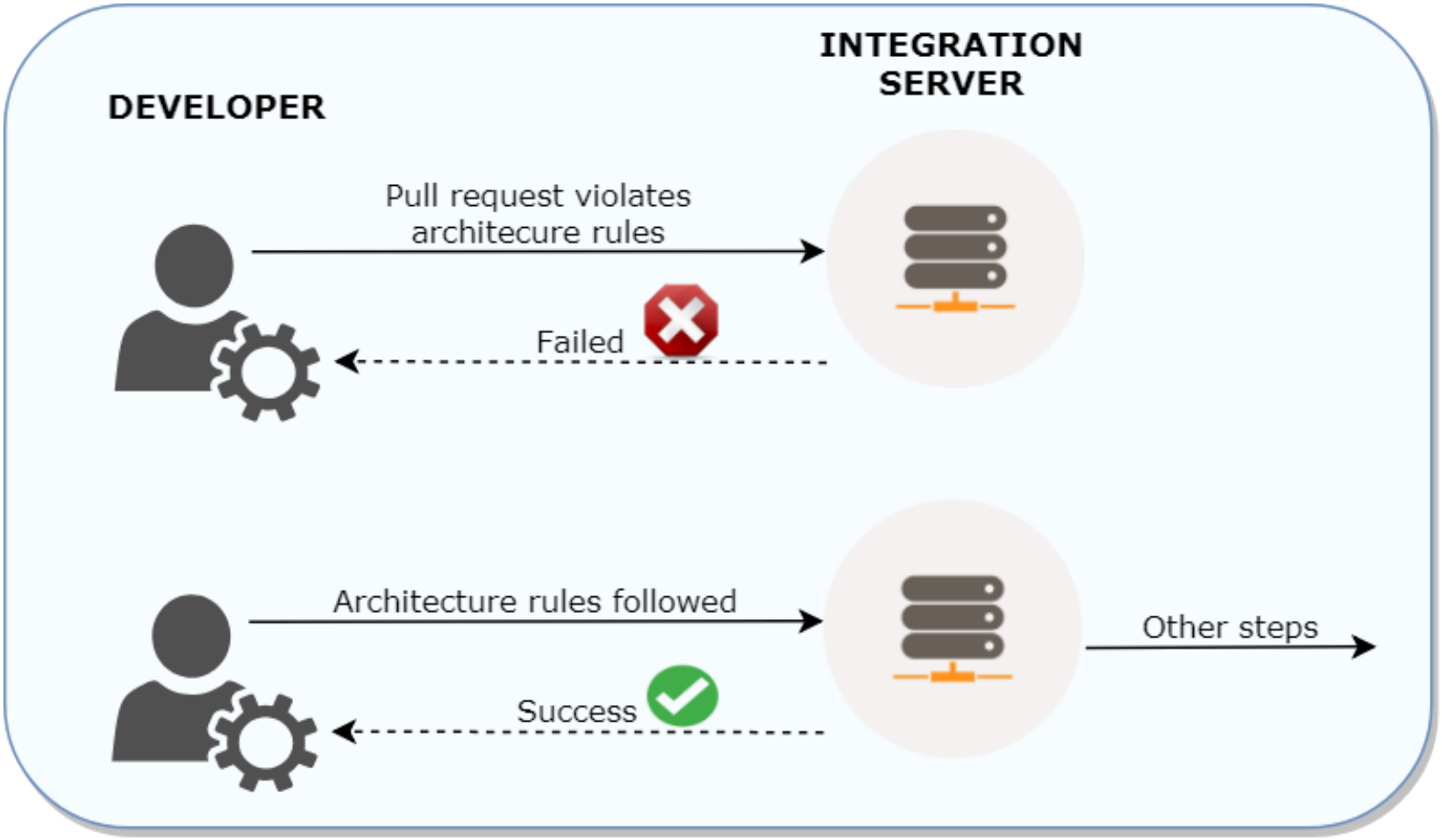
What are the challenges?

- New developers
- Old developers
- Documents are not read, well understood or are forgotten
- Documents can get out-of-date
- Peer reviewing culture not fully adopted
- Approaching deadlines

Software architecture in reality



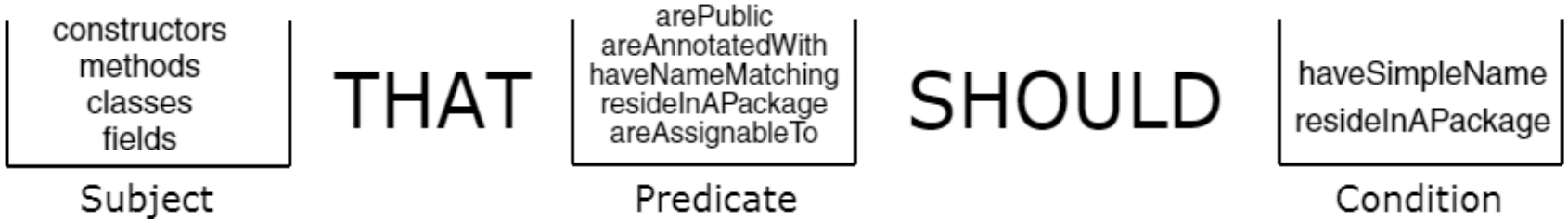
Codifying architecture rules



ArchUnit tests

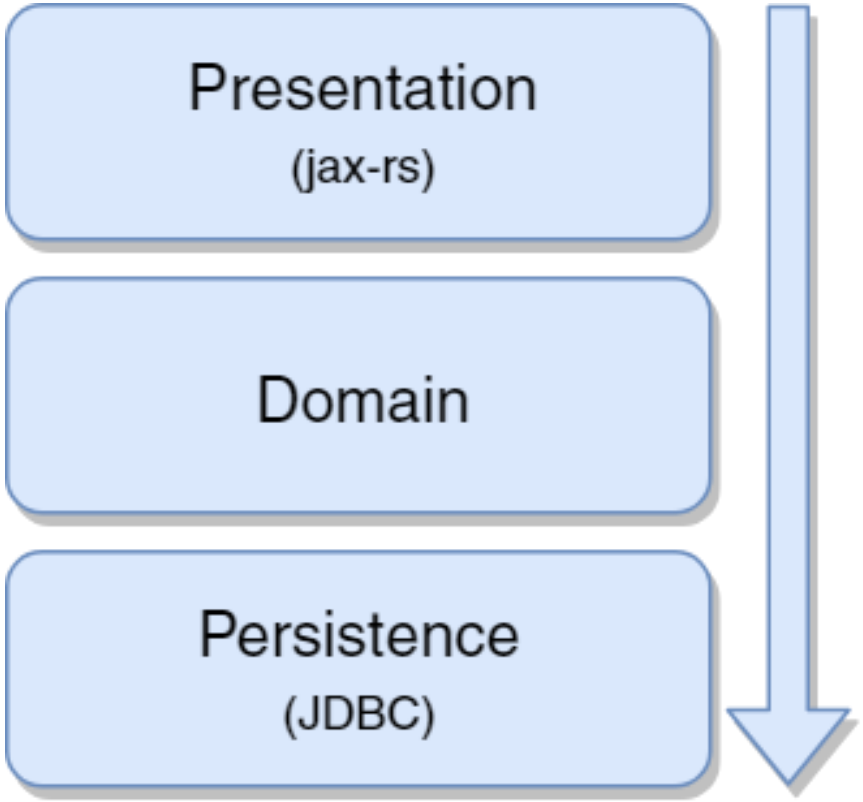
```
@Test
public void repositoriesHaveSuffix() {
    classes() GivenClasses
            .that().areNotAnnotatedWith(Repository.class)
            .should().haveSimpleNameEndingWith( suffix: "Repository")
            .check(projectClasses);
}
```

Most of arch unit syntax comes down to ...



... and you can use logical operators AND, OR, NOT.

The *demo* architecture – Blogging REST API



Scenario:

- REST API for a blog app
- Support for blog posts was added while maintaining architecture rules
- While adding support for comments, several rules were violated

Basic rules

1. Only *domain layer* can depend on *persistence layer*
2. Only *presentation layer* should depend on *domain layer*
3. No layers should depend on *presentation layer*
4. Only *persistence layer* can access java.sql
5. Repository methods should only be called find/save/delete

ArchUnit features

- Tests are written in plain Java
- Intuitive, expressive DSL syntax
- Descriptive failing messages
- Support for all testing frameworks
- Extended support for JUnit 4 and JUnit 5
  - Imported classes are cached
- Support for architectures: Layered, Hexagonal (planned)
- PlantUML Component Diagrams as rules
- General coding rules:
  - Using standard streams for logging
  - Throwing generic exceptions
  - Using Joda

More rules

1. Communication between layers should be done only between interfaces
2. Define rules for proper logging
3. Define naming rules for Loggers
4. Java.time should be prefered to java.util.Date

Final notes

- Not a replacement for static analysis tools
- Not a replacement for peer reviewing
- Still in initial development (major version still not released)
- Open source
  - *“Good start for contributing”* issues

```
@Test
public void questions() {
    people().that().haveQuestions()
            .should().ask()
            .check(audience);
}
```

For more information

- Source code available at https://github.com/kosani/ArchUnit-javacro19
- Illustrations were made in draw.io