Strategy and Automation for Ensuring High Code Quality

—

JavaCRO 2022, Autumn
10.10.2022

IBM iX

Your presenter

Daniel Strmečki
Director Digital Platforms –
daniel.strmecki@ibmix.hr
https://www.linkedin.com/in/strmecki/

Personality: stubborn, ambitious, perfectionist
Appreciates: trust, transparency, integrity, quality
Loves: well-defined processes, clear documentation
Interests: sports, cool tech

Did you know...

... that IBM iX is one of the world's largest digital agencies –

... and is considered the global leader in customer experience?

Figure 1: Magic Quadrant for CRM and Customer Experience Implementation Services

IBM iX global:

57 studios, 17.000 people

NORTH AMERICA

CANADA
Montreal
Toronto

UNITED STATES
Atlanta
Austin
Cambridge
Chicago
Columbus
New York City
San Francisco
Washington D.C.

LATIN AMERICA

ARGENTINA
Buenos Aires

CHILE
Santiago

COLOMBIA
Bogota

MEXICO
Mexico City

EUROPE

AUSTRIA
Graz
Vienna
Wels

BELGIUM
Brussels

CROATIA
Varazdin
Zagreb
Osijek

CZECH REPUBLIC
Prague

DENMARK
Copenhagen

FINLAND
Helsinki

FRANCE
Paris

GERMANY
Berlin
Dusseldorf
Hamburg
Munich
Ehningen

ITALY
Milan

NETHERLANDS
Amsterdam
Groningen

POLAND
Warsaw

RUSSIA
Moscow

SPAIN
Barcelona
Madrid

SWEDEN
Malmo
Stockholm

SWITZERLAND
Zurich

UNITED KINGDOM
London
Hursley

MIDDLE EAST & AFRICA

SOUTH AFRICA
Johannesburg

TURKEY
Istanbul

UNITED ARAB EMIRATES
Dubai

ASIA PACIFIC

AUSTRALIA
Melbourne
Sydney

CHINA
Beijing
Dalian
Hong Kong
Shanghai

INDIA
Bangalore
Mumbai

JAPAN
Tokyo

NEW ZEALAND
Auckland

SINGAPORE
Singapore

SOUTH KOREA
Seoul

TAIWAN
Taipei

What we do?

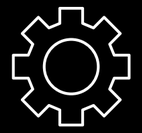
Reinventing business, improving experience, inspiring people.

We advise

Strategic consulting with in-depth technology and industry expertise

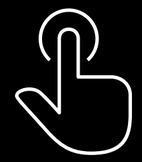
We create

Experience design and communication with an impact



We build

Engineering & development of platforms, services & products



We operate

Agile management, workflows and implementation

Transformation of the entire digital customer lifecycle

IBM iX developed a new digital ecosystem for the European market based on Adobe that spans the complete customer journey – including a website for marketing and sales, a mobile-optimised Car Configurator and a new myHyundai customer portal.

→ www.hyundai.com/eu
→ www.hyundai.com/it

Our agenda

1) Introduction

2) Quality approach

3) Coding guidelines

4) Onboarding experience

5) Ensuring code quality
   - Static code analysis
   - Clean architecture
   - Custom Sonar rules

6) Q&A

Introduction
Why do we need standardization and code quality checks?

What did we want to achieve?

1) Ensure high perceived quality for our customers

2) Ensure high code quality for our developers

3) Stop repeating the same mistakes on multiple projects

4) Align on best practices across projects and teams

5) New-joiner should learn from our previous experience

Any fool can write code that a computer can understand. Good programmers write code that humans can understand

by Martin Fowler

Quality approach
How to establish a company-wide quality assurance approach?

Quality approach

A companywide quality assurance approach is established within IBM iX to ensure high levels of quality.

In short, this approach provides guidelines, processes and best practices for our projects which guides the whole project team on how to achieve high quality.

Holistic team approach

Entire scrum team is responsible for Quality

Team contains members with testing and test automation experience

Tester contributes throughout the whole user story lifecycle

Testing is part of development

Testing is not a phase in agile projects

Development = test & code

Team plans test activities

Test (automation) strategy is created by the team

High level of automation

Integration of CI-Tools, such as Jenkins

Quick (daily) feedback and fast execution of checks. Minimize manual “checking” efforts

High level of automation allows exploratory testing

Test automation strategy

Following tests are to be integrated into the Continuous Integration and when to run

01

Static Code

On every pull-request

02

Unit Test

During build time

03

Integration Test

Nightly

04

Automated Regression Tests (E2E, UI)

Nightly

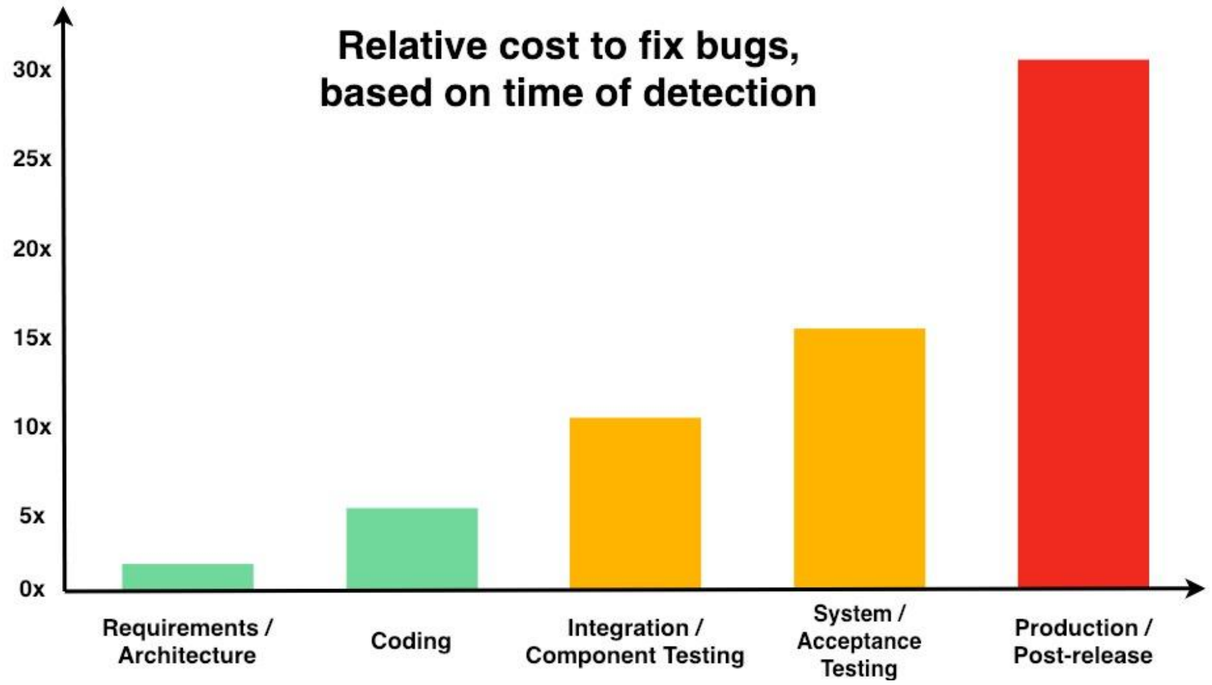
05

Performance & Security Tests

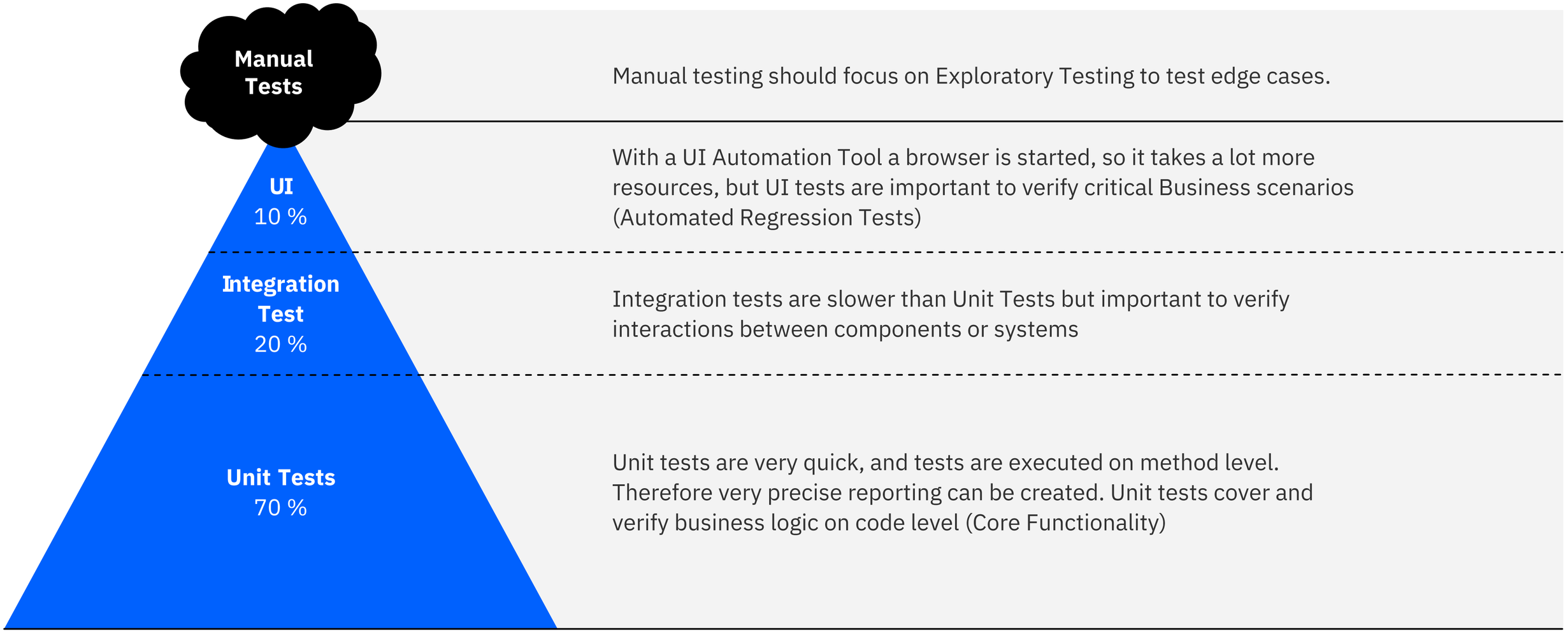
Nightly/Weekly

Benefits of automation

- Releasing quicker due to faster feedback
- Saving money due to automating repetitive testing
- Reducing project costs by automating manual work
- Avoiding cost of bugs



Test distribution



Quality gates

With the two dimensions of quality gates we use a simple approach to ensure a continuously increasing quality for our projects. One deals with source code quality and the other is more focussing on the bugs created by the customer.

Both quality gates need to pass

Code quality is important for overall software quality. And quality impacts how safe, secure, and reliable the codebase is.

We have defined Minimum Quality Gates which should be followed by each of our projects.
SonarQube is being used for Source Code Analysis.
Aim is to have a clear code in order to reduce technical debt.

We want to delivery a bug free solution. Critical verified bugs created by the customer is a sign of weak quality.

Customer satisfaction is defined by how many bugs are found by the customer.
Normally, this needs to be defined with the customer.
Objective of these KPIs are to reduce critical bugs found by customers.

Zero bug policy

The zero bug policy enables the project team to have 0 bugs in the backlog at a certain point in time. As soon as a bug is being discovered, it is immediately fixed by the team and covered with automated tests, so the same bug can never appear again. Bugs are not tolerated.

The idea behind this policy is, that you do not have a backlog of open bugs at all, following the ideal to have as little technical debt as possible. This means that when a bug is raised, we either commit to fixing it right away or decide together to close it as a “Won’t Fix”. By doing this, we bring down technical debt to a minimum and consequently maximize efficiency and adaptability.

All bugs take priority over all new feature development or improvements. Either the issue is a bug and therefore takes priority, or, as is more often the case, the issue can be reclassified as an improvement or even a new feature and can be prioritized in the backlog for future iterations. By doing this, no new functionality can be hidden in bugs, diluting Velocity and Iteration Planning outcomes

Coding guidelines

How can we ensure that our code looks like it was written by IBM iX (on all projects)?

Value of documentation

Agile value misconception

- Working software over comprehensive documentation

```
try (ResourceResolver resourceResolver = getServiceResourceResolver()) {
    // Use resource resolver here
} catch (RuntimeException e) {
    log.error("Error creating base names", ex);
}
```

Documenting best practices

- Make notes on the good practices that you see on different projects
- We always refer to previous projects and lessons learned

```
@PostConstruct
public void init() {
    try {
        this.homePage = Optional.ofNullable(getAbsoluteParent())
                                .orElse(this.currentPage);
    } catch (final RuntimeException e) {
        log.error("Exception in post construct", e);
    }
}
```

Coding guidelines

Notes can become coding guidelines:

- Anyone can and should contribute
- Use them to align with all developers
- Use them to coach junior developers

Some benefits:

- Align codebases across projects
- Learn on mistakes others made
- Competitive advantage trough quality KPIs

Sling Models vs *WCMUsePojo* (Sonar rule: ibmix-aem:AvoidWcmUsePojoClass)

Many Sling projects want to be able to create model objects - POJOs which are automatically mapped from Sling objects, typically resources, but also request objects. Often these POJOs also use OSGi services and other common AEM objects, that can then be easily injected in Sling Models.

Although Java Use Provider (*WCMUsePojo* classes) provided through bundles are faster to initialize and execute than Sling Models for similar code, the Sling Models Use Provider provides the following advantages:

- Entirely annotation driven - "pure" POJOs
- Easy to extend from other Sling Models
- Code is reusable
- Simple setup for unit testing

Thus, Sling Models should always be preferred over *WCMUsePojo* classes.

Use Interfaces for component models

All sling models which are used in AEM components should implement an interface which defines all the getter methods for the model.

This is best practice provided by Adobe and implemented in the AEM Core Components. In the model itself the resourcetype of the component has to be set in the Model annotation.

For example, if a component which contains two textfields (text1 and text2) should be implemented. Then the interface might look like this:

MyComponent.java

```
import org.osgi.annotation.versioning.ConsumerType;

@ConsumerType
public interface MyComponent  {

    String getText1();

    String getText2();

}
```

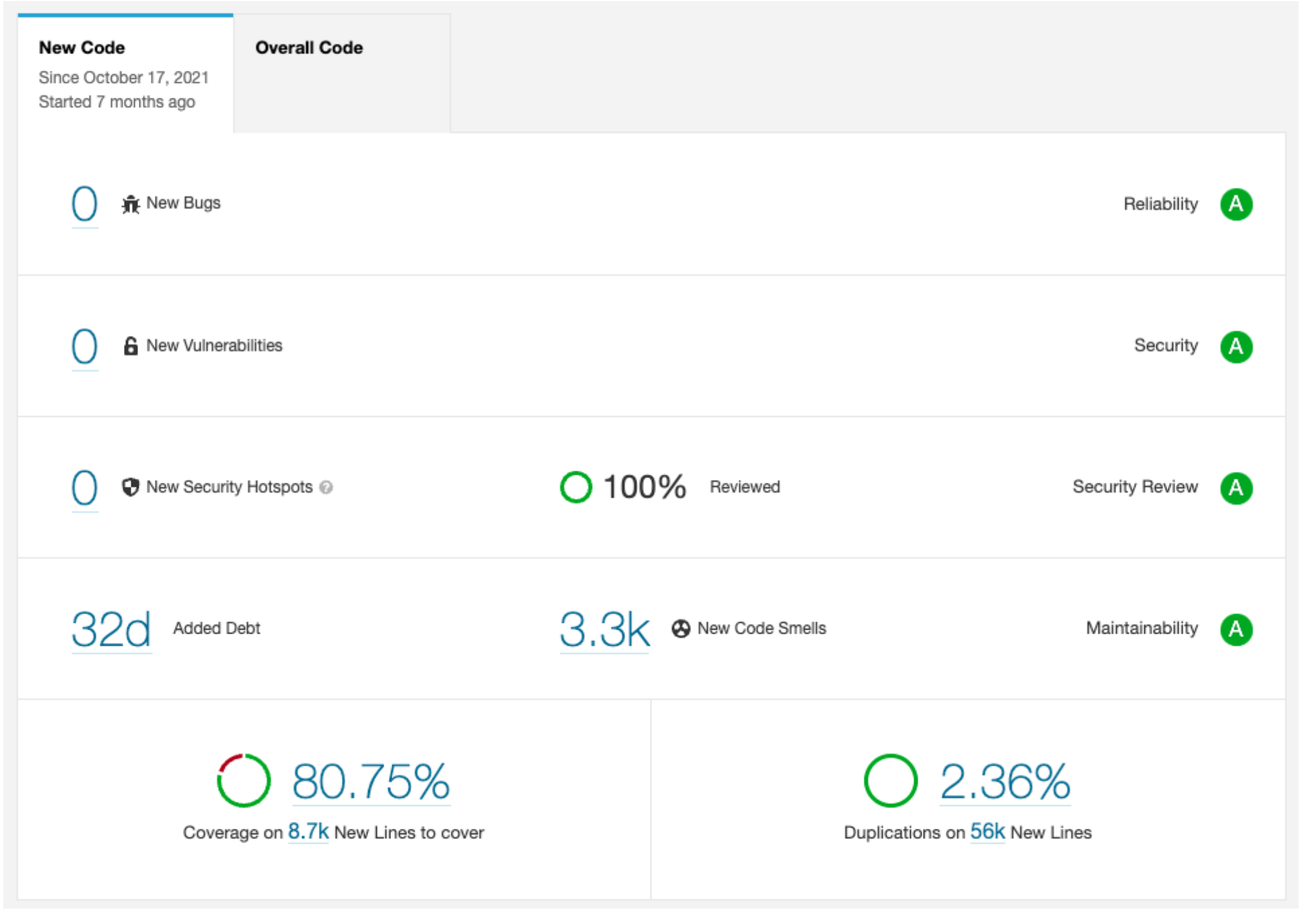
Static code analysis

SonarQube free version helped us with:

- Reporting code coverage
- Informing about clean coding practices
- Detecting Java-specific issues

But it didn’t solve:

- Ensuring high coverage & zero bugs policy
- Enforcing clean architecture
- Detecting framework-specific issues



Onboarding experience

How can we ensure that we pass our experience to new talents joining the company?

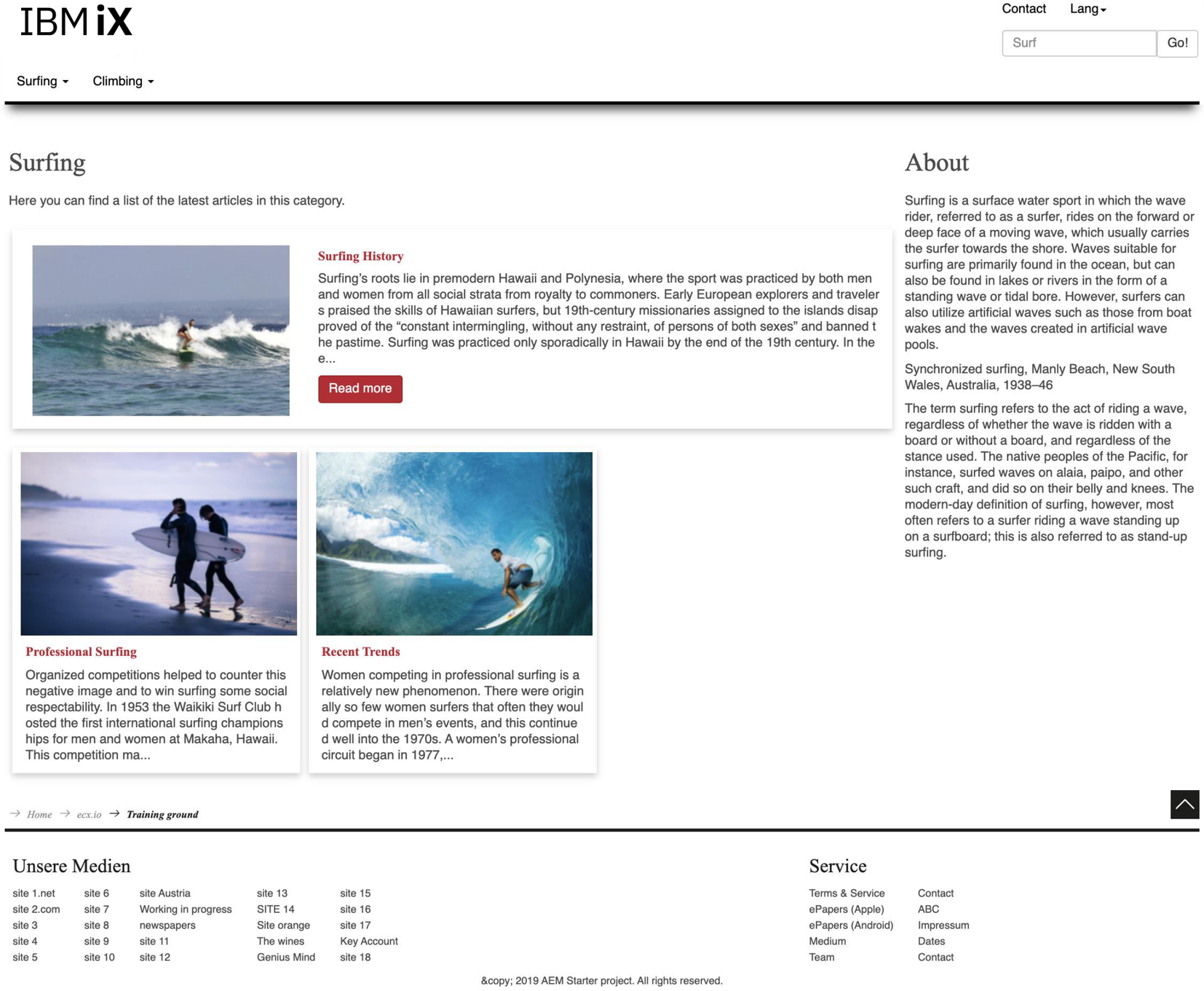
Simulation Project

Based on a real project

- Using latest Java version
- Using latest framework version

Examples for developing

- Custom components
- OSGi services
- Sling servlets
- Integration tests



- Starter 2.0
  - 1. Getting started
  - 2. Project introduction
  - 3. Development environment
  - 4. Project backlog
    - AEMSP-001: Article text
    - AEMSP-002: Article headlines
    - **AEMSP-003: Share module**
    - AEMSP-004: Image text component
    - AEMSP-005: Article language switch
    - AEMSP-006: Article event download
    - AEMSP-010: Breadcrumbs
    - AEMSP-011: Editable footer
    - AEMSP-012: Contact form
    - AEMSP-020: Date picker
    - AEMSP-030: Image slider
    - AEMSP-031: Featured article
    - AEMSP-032: Video module
    - AEMSP-033: News feed
    - AEMSP-034: Upcoming events
    - AEMSP-035: Quote component
    - AEMSP-036: Workflows
  - 5. Development tips
  - 6. Internal development
  - Project Backlog - FE
- Planning
- Topic owners

Acceptance criteria

Share module component contains

- **Go back** button
- **Print** and **Send via email** buttons
- **Share buttons** for **Facebook**, **Twitter** and **LinkedIn**

Functionality

- Component is added on an Article page by default (when a new article page is created)
  - Component can only be added to the main content area, not in the sidebar
- Clicking on Facebook, Twitter or LinkedIn buttons opens their standard share dialog provided by the API of the social media site
  - When clicked, always share the published URL of the article (even on author instance)
- Send via email opens your default mail client with subject set to title of the article page
- Go back returns to the previous page using browser history
- Print button opens the default print dialog box, which lets the user select native printing options

Self-explanatory tickets / requirements

Constraints

- Preview of Facebook share dialog is not available when you are trying to share your article link. It is usual behavior since your site is not publicly available, it is served on your local machine.
- Although some constraint exists, your solution should contain a real link to your article. When testing don't forget to remove the "editor.html" part of your URL.

Hints

- Create a Sling Model for your new component
  - Reuse the existing *LinkService* to generate the article external link
  - Follow guidelines and best practices for working with Sling Models
- Write a Unit/Integration test for the backend code using AEM Mocks
  - Refer to existing tests as examples
  - Follow guidelines and best practices for backend testing in AEM
- Use i18 translator for translating hardcoded labels from English to German

Hints pointing to guidelines and best practices

Design and layout

Provided design and front-end code

References for mentors

Diff Commits

Find text in diff and context lines

- core/src
  - main/java/io/ecx/aem/aemsp/core
    - models
      - event/dates
        - EventDatesExporterModel.java
        - EventDetailsModel.java
      - ArticlesListModel.java
    - records
      - ArticleItem.java
    - utils
      - Constants.java
  - test/java/io/ecx/aem/aemsp
    - context/services
      - ServicesImpl.java
      - ServicesMock.java
    - core
      - models
        - ArticlesListModelTest.java
        - LanguagesModelTest.java
        - NavigationModelTest.java
        - SearchResultsModelTest.java
      - services/impl
        - ImageFinderServiceImplTest.java
- ui.apps/src/main/content/jcr_root/apps/aemsp
  - components
    - content

core / src / main / java / io / ecx / aem / aemsp / core / models / event / dates / **EventDatesExporterModel.java** ADDED

Blame

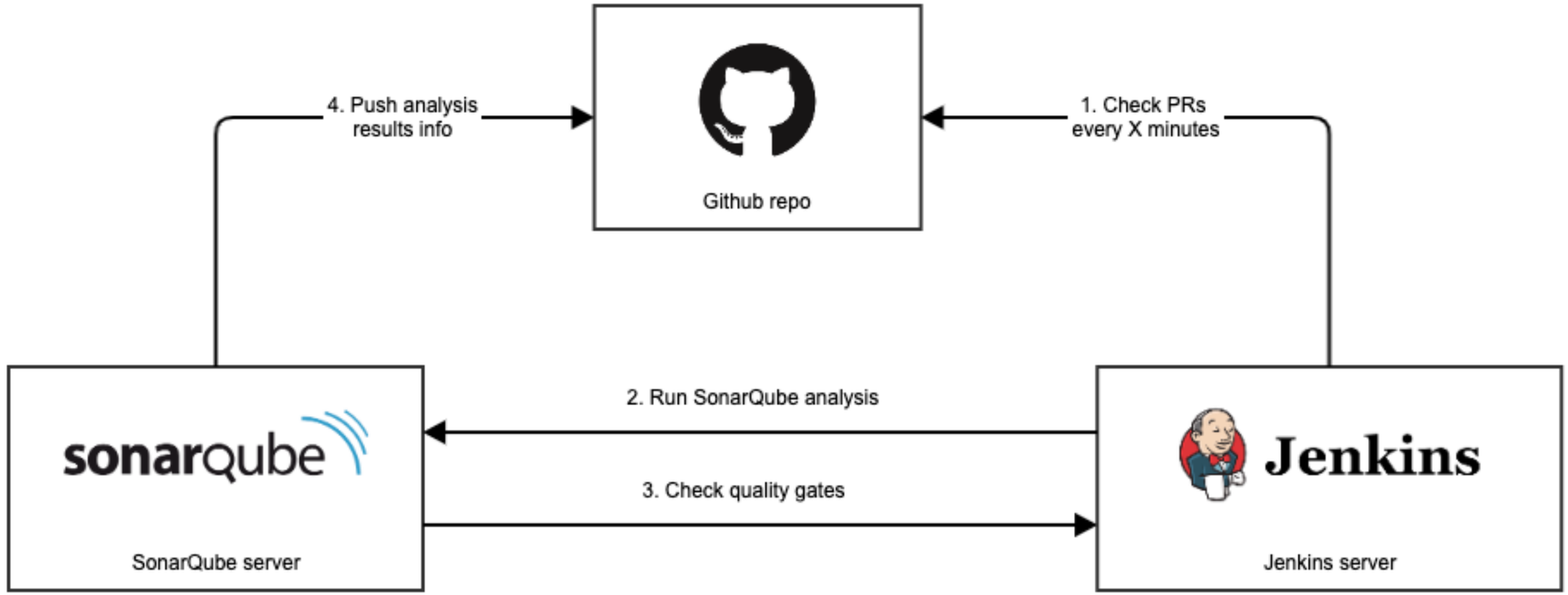
```
1 + package io.ecx.aem.aemsp.core.models.event.dates;
2 +
3 + import java.util.List;
4 + import java.util.Objects;
5 + import java.util.stream.Collectors;
6 +
7 + import javax.annotation.PostConstruct;
8 +
9 + import org.apache.commons.collections4.IteratorUtils;
10 + import org.apache.commons.lang3.StringUtils;
11 + import org.apache.sling.api.SlingHttpServletRequest;
12 + import org.apache.sling.models.annotations.DefaultInjectionStrategy;
13 + import org.apache.sling.models.annotations.Exporter;
14 + import org.apache.sling.models.annotations.Model;
15 +
16 + import com.adobe.acs.commons.models.injectors.annotation.AemObject;
17 + import com.day.cq.wcm.api.Page;
18 + import com.fasterxml.jackson.annotation.JsonProperty;
19 +
20 + import io.ecx.aem.aemsp.core.utils.Constants;
21 + import lombok.extern.slf4j.Slf4j;
22 +
23 + @Slf4j
24 + @Model(adaptables = SlingHttpServletRequest.class, defaultInjectionStrategy = DefaultInjectionStrategy.OPTIONAL, resourceType = "aemsp/components/
25 + @Exporter(name = "jackson", selector = "eventdates", extensions = "json")
26 + public class EventDatesExporterModel {
27 +
28 +     @AemObject
29 +     private Page currentPage;
30 +
31 +     private List<EventDetailsModel> eventDates;
32 +
33 +     @PostConstruct
34 +     public void init() {
35 +         try {
36 +             final List<Page> childPages = IteratorUtils.toList(this.currentPage.listChildren());
37 +             this.eventDates = childPages
38 +                                 .stream()
39 +                                 .map(Page::getContentResource)
40 +                                 .map(resource -> resource.getChild(Constants.NN_EVENT_DETAILS))
```

Ensuring code quality

How can we ensure that developers actually do fix Sonar issues and do apply coding guidelines?

Pull request decoration

- The SonarQube PR decoration can be integrated directly into mostly any CI/CD workflow
- Every time a new Pull Request is opened, the analysis can be executed automatically
- No merging allowed until the analysis is completed and quality gates passed



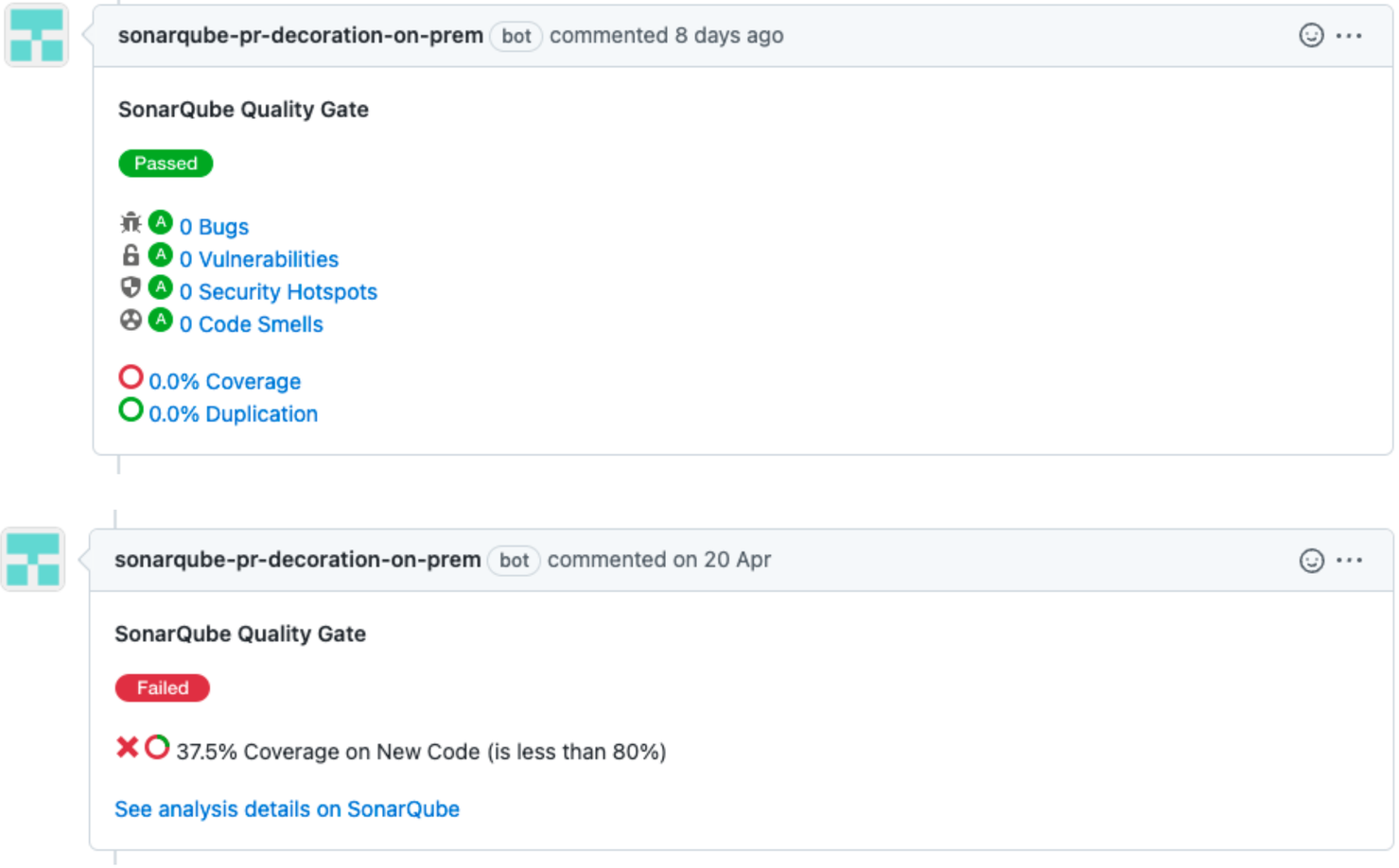
Ensuring high coverage & zero bugs policy

Pull-request decoration:

- Performing code analysis on the changes introduced
- Reporting results on you code collaboration tool
- Disabling merge in gate fail

With SonarQube Developer plan:

- Branch analysis + CI job
- Not free, paid licence



Enforcing clean architecture

ArchUnit testing library:

- Checking the architecture of your Java code using standard Junit tests and reflection
  - https://www.archunit.org/

It can check and enforce:

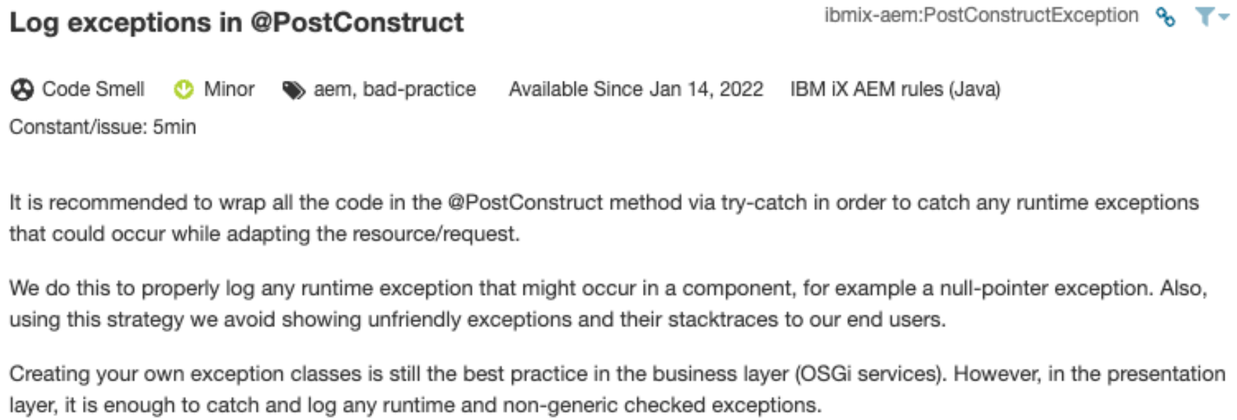
- Naming conventions
- Types and annotations used in packages
- Dependencies between packages, classes, layers and slices,
- Cyclic dependencies

```
classes()
  .that().resideInAPackage(SERVICES_PACKAGE)
  .should().beInterfaces();

classes()
  .that().resideInAPackage(SERVICES_PACKAGE)
  .should().haveSimpleNameEndingWith( s: "Service");

classes()
  .that().resideInAPackage(SERVICES_IMPL_PACKAGE)
  .should().beAnnotatedWith(Component.class);

noClasses()
  .that()
  .resideInAnyPackage(COMPONENTS_PACKAGE, MODELS_PACKAGE)
  .should()
  .dependOnClassesThat()
  .resideInAPackage(SERVLETS_PACKAGE);
```

Detecting framework-specific issues

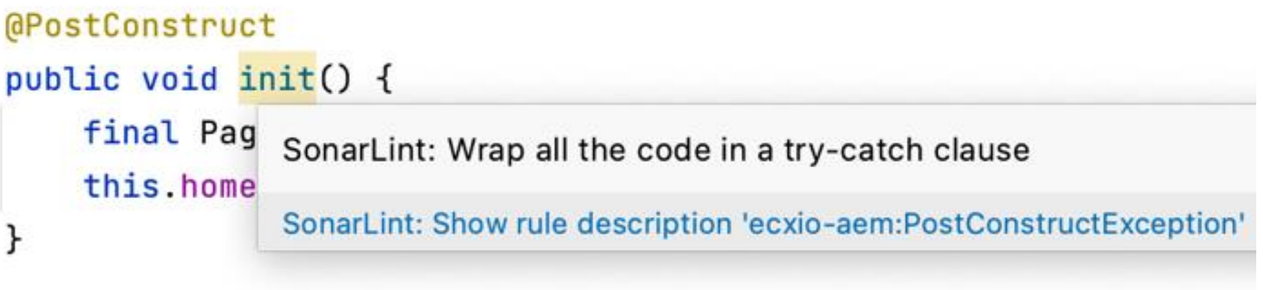
Framework-specifics:

- We use enterprise Java frameworks like Adobe Experience Manager and SAP CX that are not supported OOTB by Sonar rules
- Therefore, we had to write our own rules
  - https://github.com/IBM/ibm-ix-aem-sonarqube-plugin



Custom SonarQube rules:

- Coding guidelines can be automated using a set of custom SonarQube rules
- If it is not automated, it probably won't be used by developers



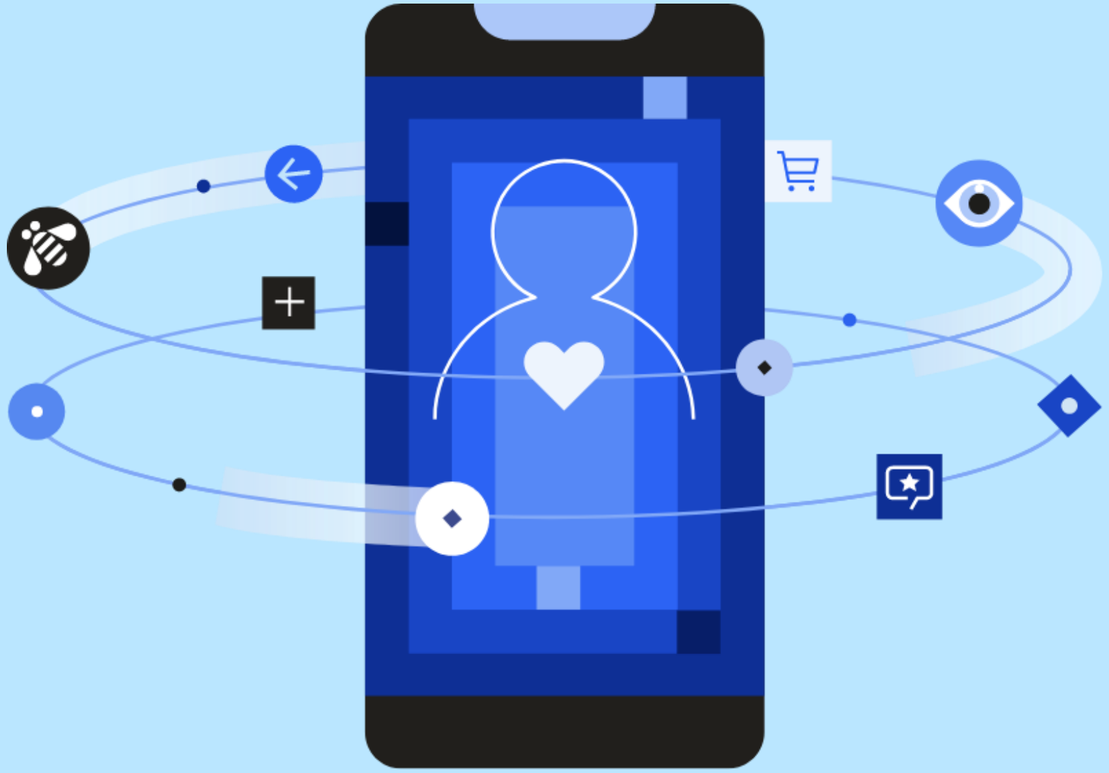
Automated checks, together with pull-request decoration feature, helps us ensure that guidelines and best practiced really do get applied by developers across projects.

Next up Developing Custom SonarQube Rules for Java

At 14:30 in Hall B

Martin Gluhak
Java/AEM Developer
martin.gluhak@ibmix.hr

Visit us @ our booth &
get a chance to win a
ticket for the JavaCro
conference 2023!



IBM iX