HOW TO MAKE QUALITY ANDROID APPS

Hai Nguyen
Technical Lead – CBA Digital Solutions

@minhhai8507

What defines quality of an app?

- No(less) bugs
- Maintainability
- Scalability
- Performance
- Code smells
- Vulnerabilities
- …

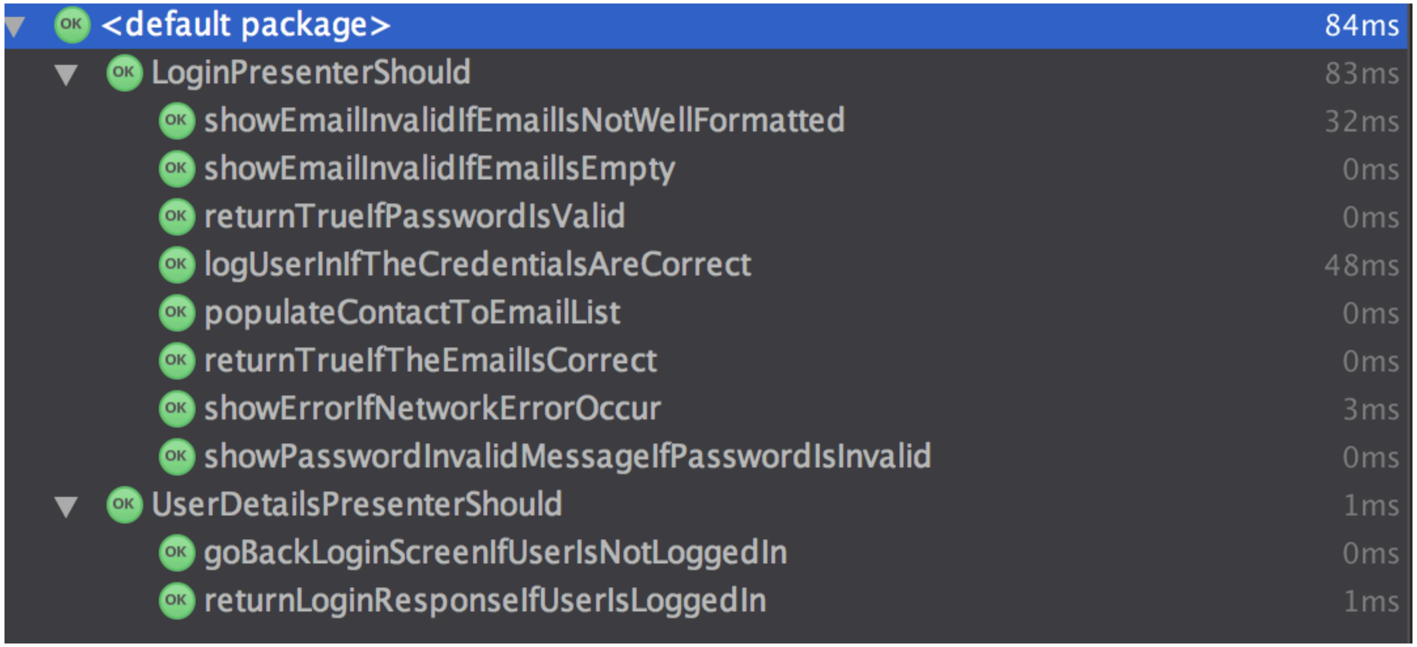
OUR GOAL IS TO WRITE BUG-FREE SOFTWARE. I'LL PAY A TEN-DOLLAR BONUS FOR EVERY BUG YOU FIND AND FIX.
S. Adams E-mail: SCOTTADAMS@AOL.COM
YAHOO!
WE'RE RICH
YES!!! YES!!! YES!!!
11/13 © 1995 United Feature Syndicate, Inc.(NYC)
I HOPE THIS DRIVES THE RIGHT BEHAVIOR.
I'M GONNA WRITE ME A NEW MINIVAN THIS AFTER-NOON!

How to improve the quality?

- Testing
  - Unit test
  - Automation test
- Architecture
- Static code analysis
- Code review
- Continuous integration
- Culture

Why Android so hard to write unit test?

- Android framework dependencies
- Tight coupling
- Slow start-up and execution time
- Run on real device or emulator



```
<default package>                                          84ms
  LoginPresenterShould                                     83ms
    showEmailInvalidIfEmailIsNotWellFormatted              32ms
    showEmailInvalidIfEmailIsEmpty                          0ms
    returnTrueIfPasswordIsValid                             0ms
    logUserInIfTheCredentialsAreCorrect                    48ms
    populateContactToEmailList                              0ms
    returnTrueIfTheEmailIsCorrect                           0ms
    showErrorIfNetworkErrorOccur                            3ms
    showPasswordInvalidMessageIfPasswordIsInvalid           0ms
  UserDetailsPresenterShould                                1ms
    goBackLoginScreenIfUserIsNotLoggedIn                    0ms
    returnLoginResponseIfUserIsLoggedIn                     1ms
```

CAN

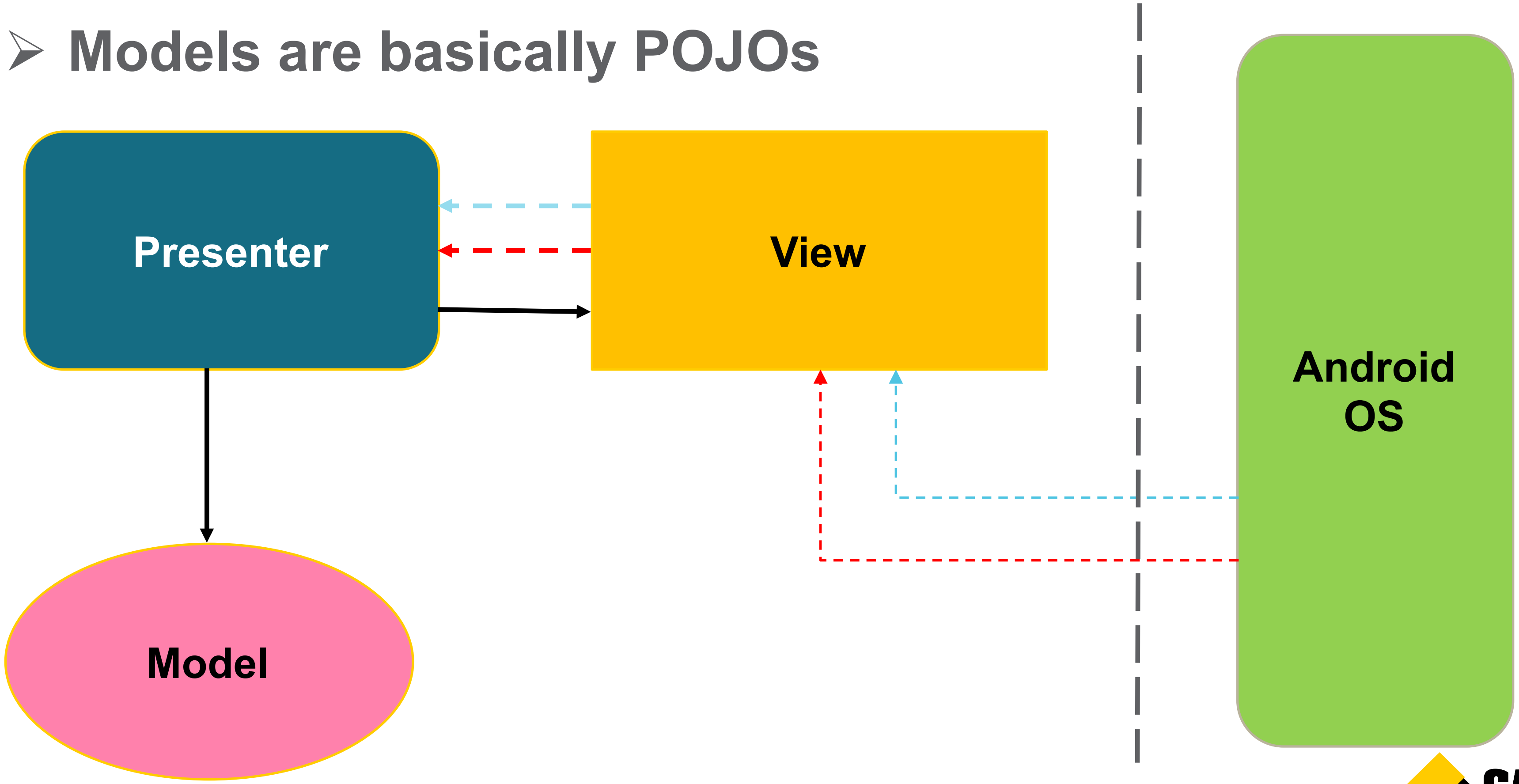
How to make unit test easier in Android

- Remove the dependency with Android framework
- Loose coupling, separation of concerns
- Dependency Injection
- S.O.L.I.D
- Architecture patterns
  - MVP
  - MVVM
  - Redux
  - Riblets
  - …

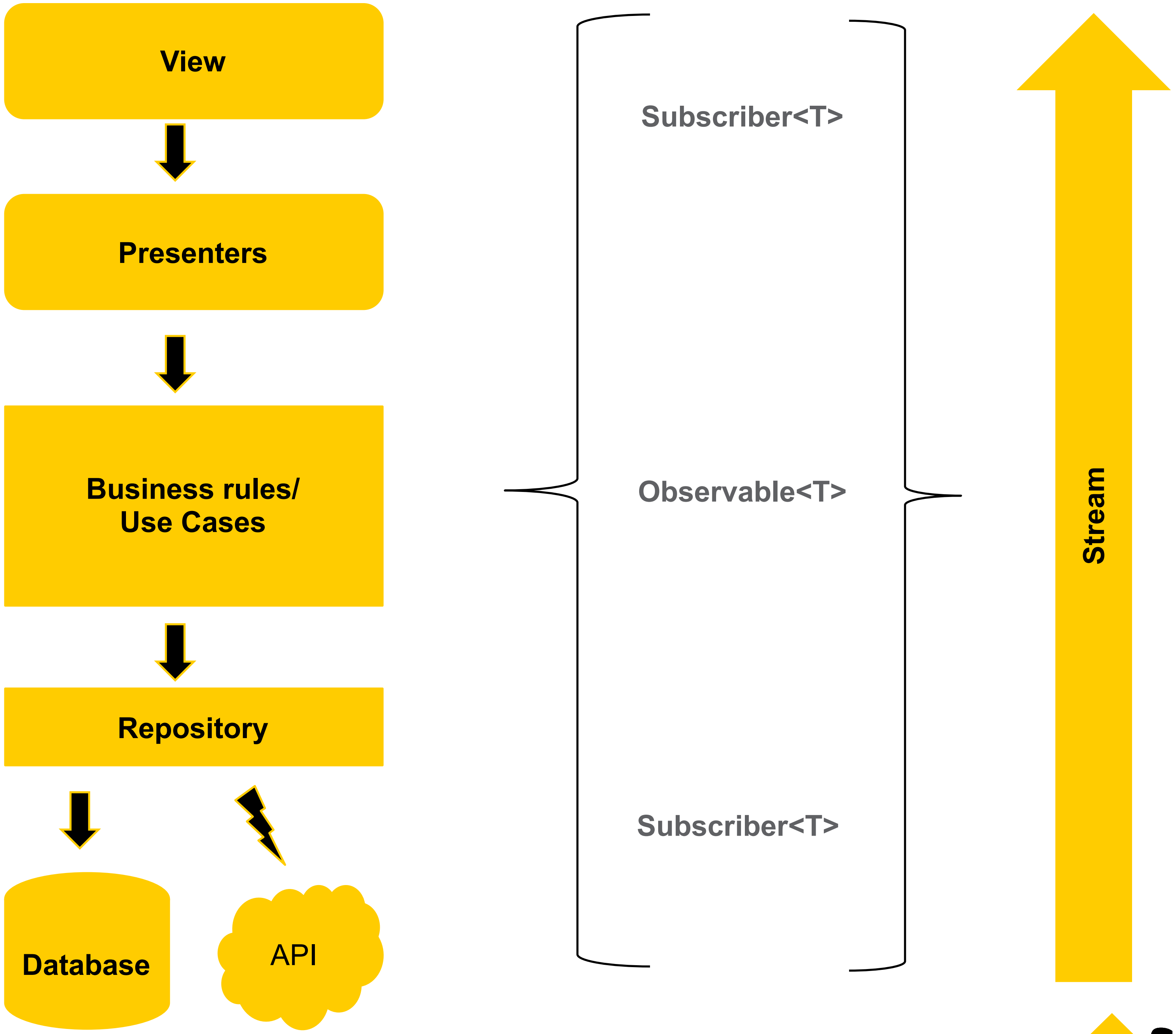
CAN

We choose MVP

- View as dump as possible
- All the logics are in Presenters
- Models are basically POJOs

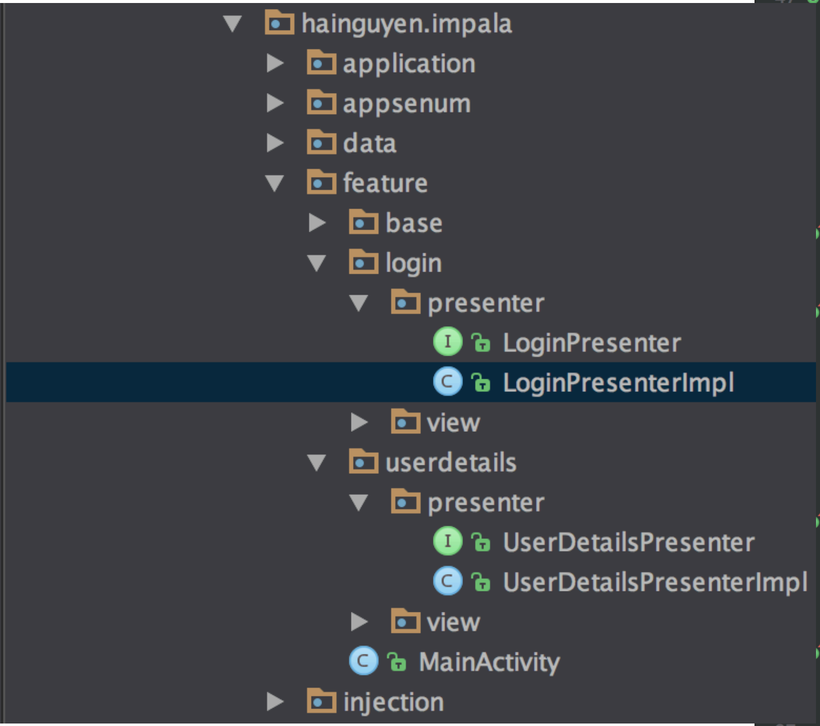


CAN

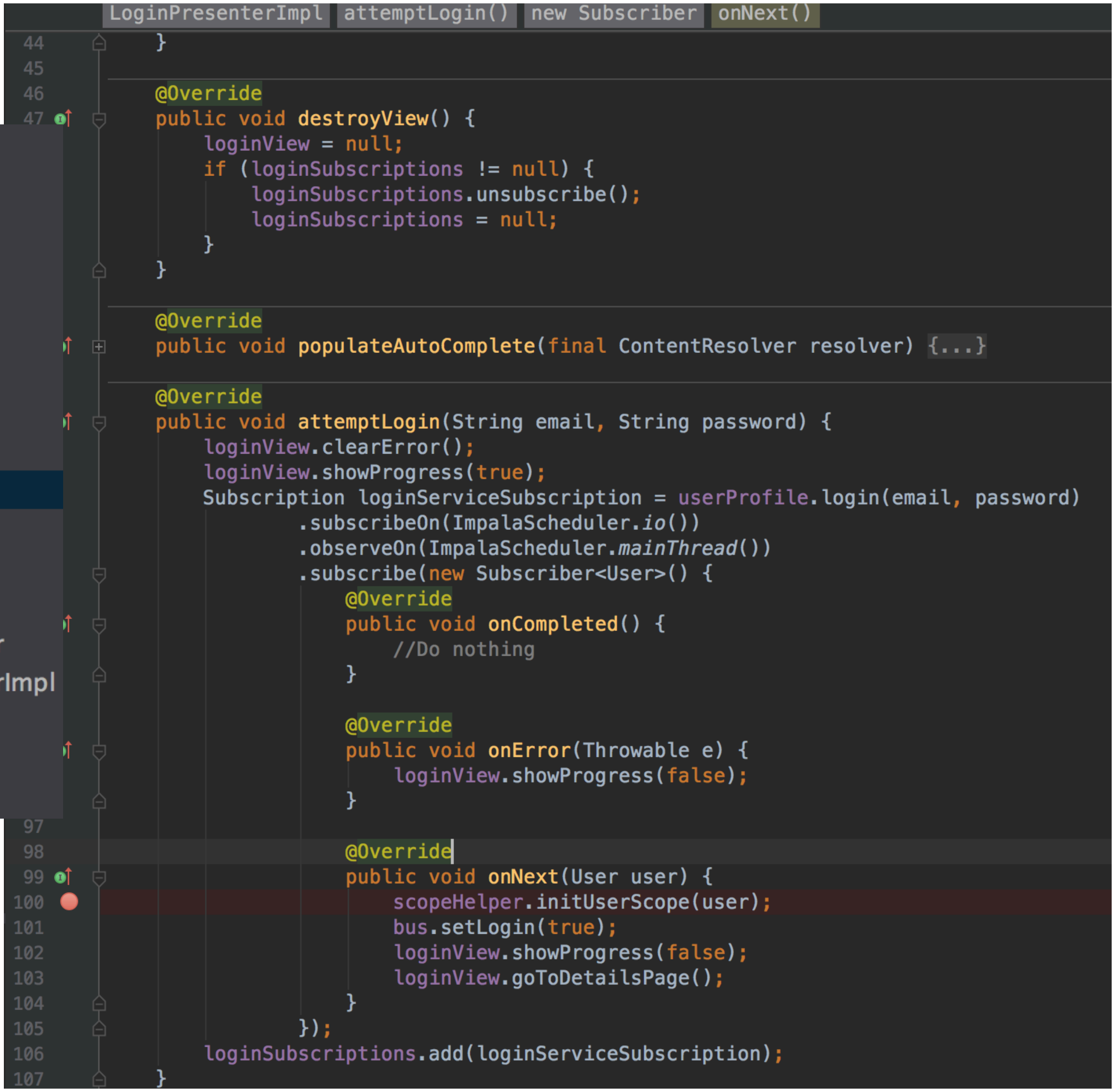

View
Presenters
Business rules/
Use Cases
Repository
Database
API
Subscriber<T>
Observable<T>
Subscriber<T>
Stream

CAN

How it looks like?



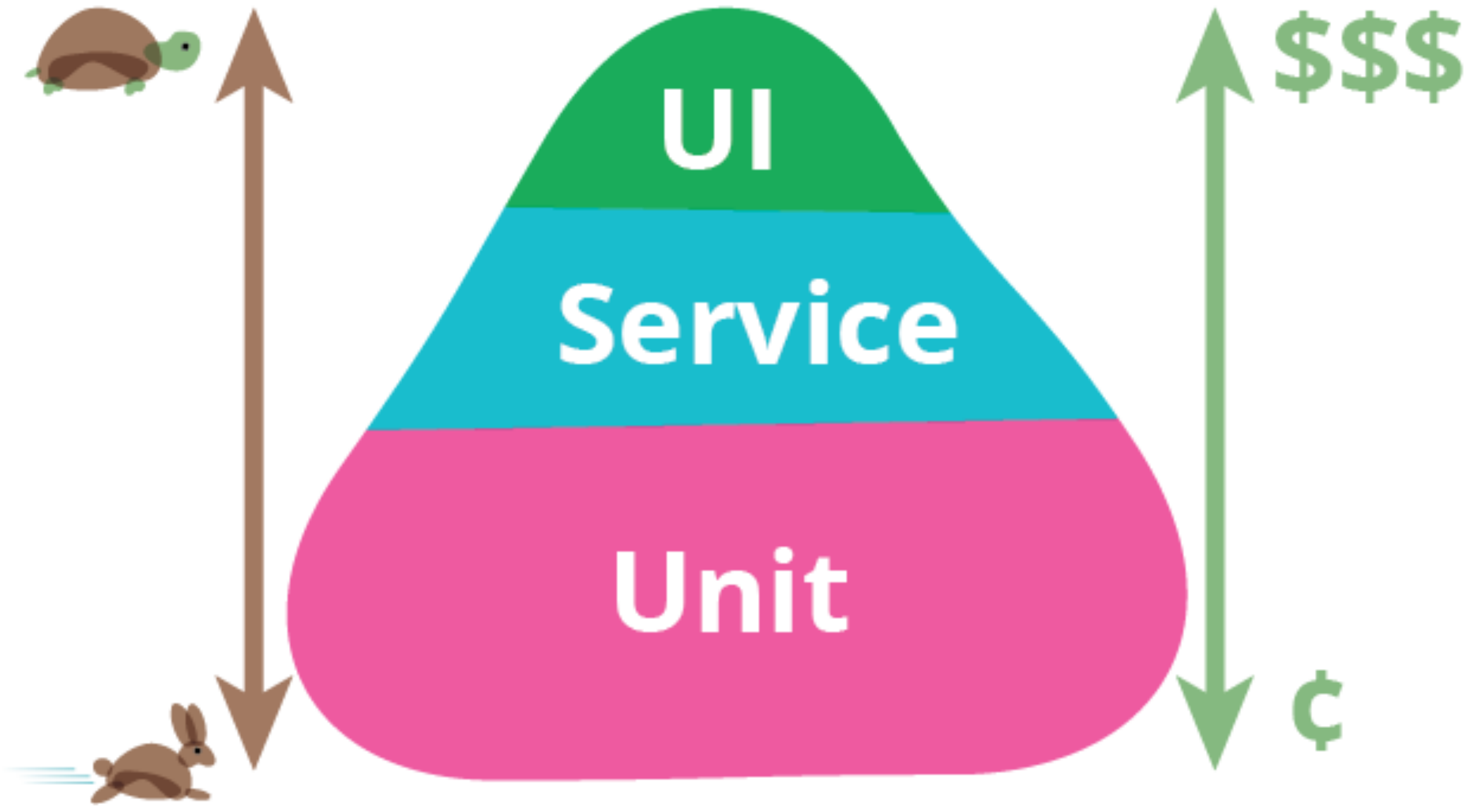
```
▼ hainguyen.impala
    ▶ application
    ▶ appsenum
    ▶ data
    ▼ feature
        ▶ base
        ▼ login
            ▼ presenter
                LoginPresenter
                LoginPresenterImpl
            ▶ view
        ▼ userdetails
            ▼ presenter
                UserDetailsPresenter
                UserDetailsPresenterImpl
            ▶ view
        MainActivity
    ▶ injection
```



LoginPresenterImpl | attemptLogin() | new Subscriber | onNext()

```java
    }

    @Override
    public void destroyView() {
        loginView = null;
        if (loginSubscriptions != null) {
            loginSubscriptions.unsubscribe();
            loginSubscriptions = null;
        }
    }

    @Override
    public void populateAutoComplete(final ContentResolver resolver) {...}

    @Override
    public void attemptLogin(String email, String password) {
        loginView.clearError();
        loginView.showProgress(true);
        Subscription loginServiceSubscription = userProfile.login(email, password)
                .subscribeOn(ImpalaScheduler.io())
                .observeOn(ImpalaScheduler.mainThread())
                .subscribe(new Subscriber<User>() {
                    @Override
                    public void onCompleted() {
                        //Do nothing
                    }

                    @Override
                    public void onError(Throwable e) {
                        loginView.showProgress(false);
                    }

                    @Override
                    public void onNext(User user) {
                        scopeHelper.initUserScope(user);
                        bus.setLogin(true);
                        loginView.showProgress(false);
                        loginView.goToDetailsPage();
                    }
                });
        loginSubscriptions.add(loginServiceSubscription);
    }
```

What else can we test?



Ref: https://martinfowler.com/bliki/TestPyramid.html

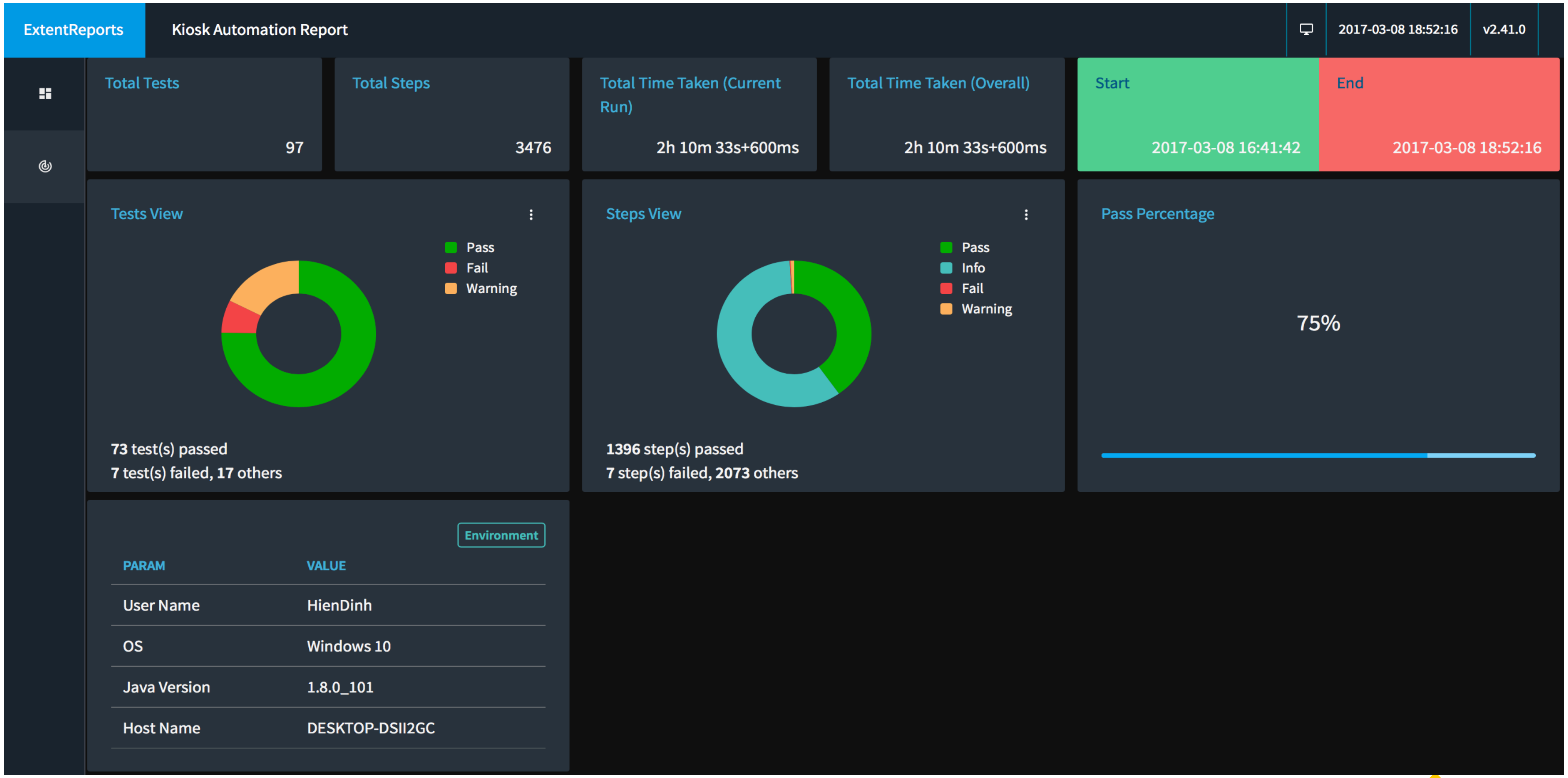
Automation test

- Automation frameworks
  - Appium for Android and iOS
  - Calabash for Android and iOS
  - MonkeyTalk for Android and iOS
  - Robotium for Android
  - Selendroid for Android
  - ….
- Automated
- Run on multiple devices
- Detects UI and functional bugs
- Requires coding skills

CAN

CBA Solution's Automation Framework

- Created our own automation framework bases on Appium
- Use extend reports



ExtentReports | Kiosk Automation Report | 2017-03-08 18:52:16 | v2.41.0

Total Tests	Total Steps	Total Time Taken (Current Run)	Total Time Taken (Overall)	Start	End
97	3476	2h 10m 33s+600ms	2h 10m 33s+600ms	2017-03-08 16:41:42	2017-03-08 18:52:16

Tests View

🟩 Pass
🟥 Fail
🟧 Warning

73 test(s) passed
7 test(s) failed, 17 others

Steps View

🟩 Pass
🟦 Info
🟥 Fail
🟧 Warning

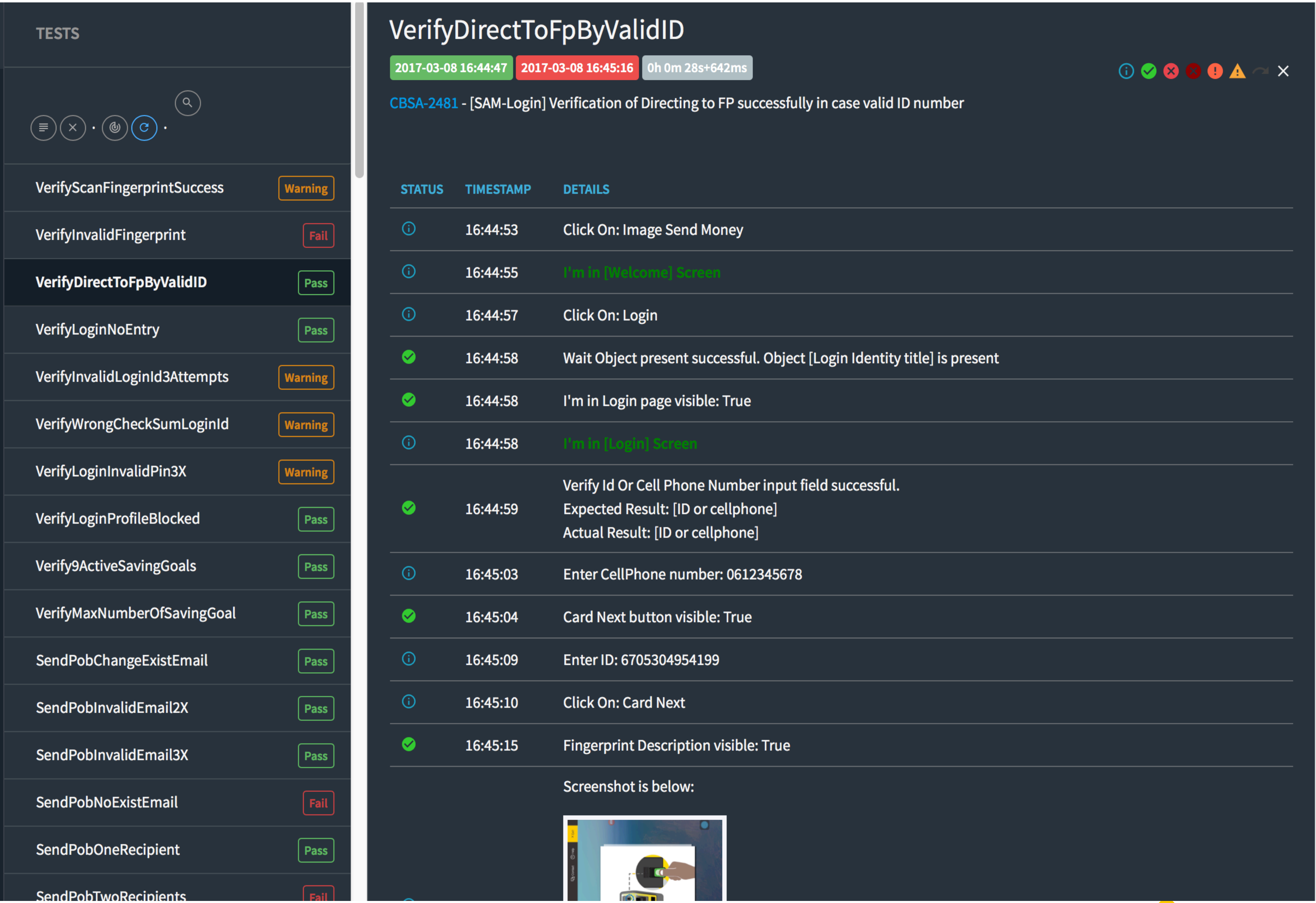
1396 step(s) passed
7 step(s) failed, 2073 others

Pass Percentage

75%

Environment

PARAM	VALUE
User Name	HienDinh
OS	Windows 10
Java Version	1.8.0_101
Host Name	DESKTOP-DSII2GC



TESTS

VerifyScanFingerprintSuccess	Warning
VerifyInvalidFingerprint	Fail
VerifyDirectToFpByValidID	Pass
VerifyLoginNoEntry	Pass
VerifyInvalidLoginId3Attempts	Warning
VerifyWrongCheckSumLoginId	Warning
VerifyLoginInvalidPin3X	Warning
VerifyLoginProfileBlocked	Pass
Verify9ActiveSavingGoals	Pass
VerifyMaxNumberOfSavingGoal	Pass
SendPobChangeExistEmail	Pass
SendPobInvalidEmail2X	Pass
SendPobInvalidEmail3X	Pass
SendPobNoExistEmail	Fail
SendPobOneRecipient	Pass
SendPobTwoRecipients	Fail

VerifyDirectToFpByValidID

2017-03-08 16:44:47 2017-03-08 16:45:16 0h 0m 28s+642ms

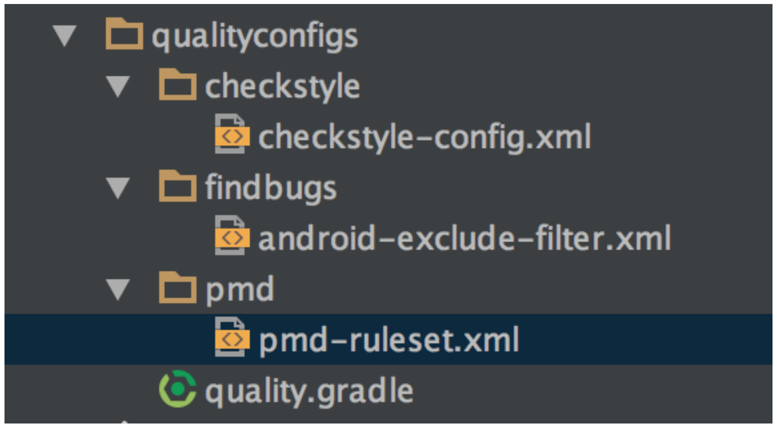
CBSA-2481 - [SAM-Login] Verification of Directing to FP successfully in case valid ID number

STATUS	TIMESTAMP	DETAILS
ⓘ	16:44:53	Click On: Image Send Money
ⓘ	16:44:55	I'm in [Welcome] Screen
ⓘ	16:44:57	Click On: Login
✓	16:44:58	Wait Object present successful. Object [Login Identity title] is present
✓	16:44:58	I'm in Login page visible: True
ⓘ	16:44:58	I'm in [Login] Screen
✓	16:44:59	Verify Id Or Cell Phone Number input field successful. Expected Result: [ID or cellphone] Actual Result: [ID or cellphone]
ⓘ	16:45:03	Enter CellPhone number: 0612345678
✓	16:45:04	Card Next button visible: True
ⓘ	16:45:09	Enter ID: 6705304954199
ⓘ	16:45:10	Click On: Card Next
✓	16:45:15	Fingerprint Description visible: True
		Screenshot is below:

CAN

Code smells, coding style...

- Detect bugs before the code get merged
  - SonarQube
  - Findbugs
  - PMD
- Coding convention
  - Style check



```
▼ qualityconfigs
    ▼ checkstyle
        checkstyle-config.xml
    ▼ findbugs
        android-exclude-filter.xml
    ▼ pmd
        pmd-ruleset.xml
    quality.gradle
```

Why we need static code analysis?

- Detecting potential bugs,
- Detecting introduced vulnerabilities,
- Pointing out duplicated code,
- Detecting design flaws,
- Enforcing coding standards,
- Detecting dead code,
- Measuring technical debt,
- Providing an overall view of the project health.

Style Check

Analyses pure source code without any major preprocessing (simple AST tree created by Checkstyle). As a consequence it is very fast.

- Naming conventions,
- Headers, imports,
- White spaces, formatting,
- Javadoc comments,
- Good practices, code conventions,
- Method parameters,
- Cyclomatic complexity,
- Any regexp.

Findbugs

Findbugs is a program which uses static analysis to look for bugs in Java code. Analyses byte code. Requires compilation. As a consequence, rules are able to access information not only about the currently analysed class..

- Possible bugs,
- Design flaws.
- Bad practices,
- Multithreaded correctness,
- Code vulnerabilities.

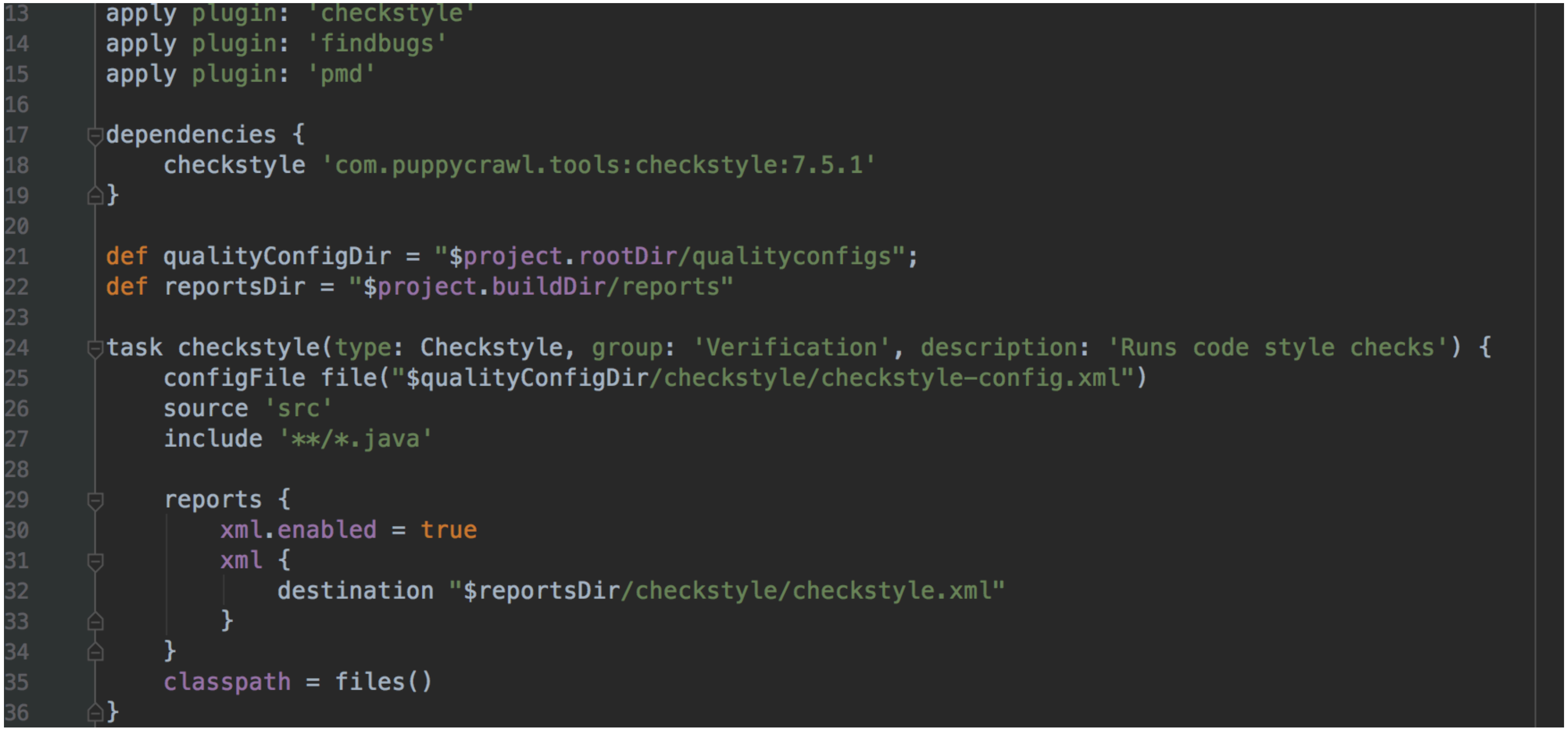
CAN

PMD

PMD is a source code analyzer. It finds common programming flaws like unused variables, empty catch blocks, unnecessary object creation, and so forth. Analyses AST ([Abstract Syntax Tree]) generated by JavaCC. Does not require compilation. More advanced checks can be implemented, but still limited to one class.

- Possible bugs,
- Dead code,
- Duplicated code,
- Cyclomatic complexity,
- Overcomplicated expressions,
- And in fact everything that Checkstyle is capable of.

Gradle task



```
apply plugin: 'checkstyle'
apply plugin: 'findbugs'
apply plugin: 'pmd'

dependencies {
    checkstyle 'com.puppycrawl.tools:checkstyle:7.5.1'
}

def qualityConfigDir = "$project.rootDir/qualityconfigs";
def reportsDir = "$project.buildDir/reports"

task checkstyle(type: Checkstyle, group: 'Verification', description: 'Runs code style checks') {
    configFile file("$qualityConfigDir/checkstyle/checkstyle-config.xml")
    source 'src'
    include '**/*.java'

    reports {
        xml.enabled = true
        xml {
            destination "$reportsDir/checkstyle/checkstyle.xml"
        }
    }
    classpath = files()
}
```

Fail fast

If bugs/defects are inevitable, how to detect them as soon as possible?

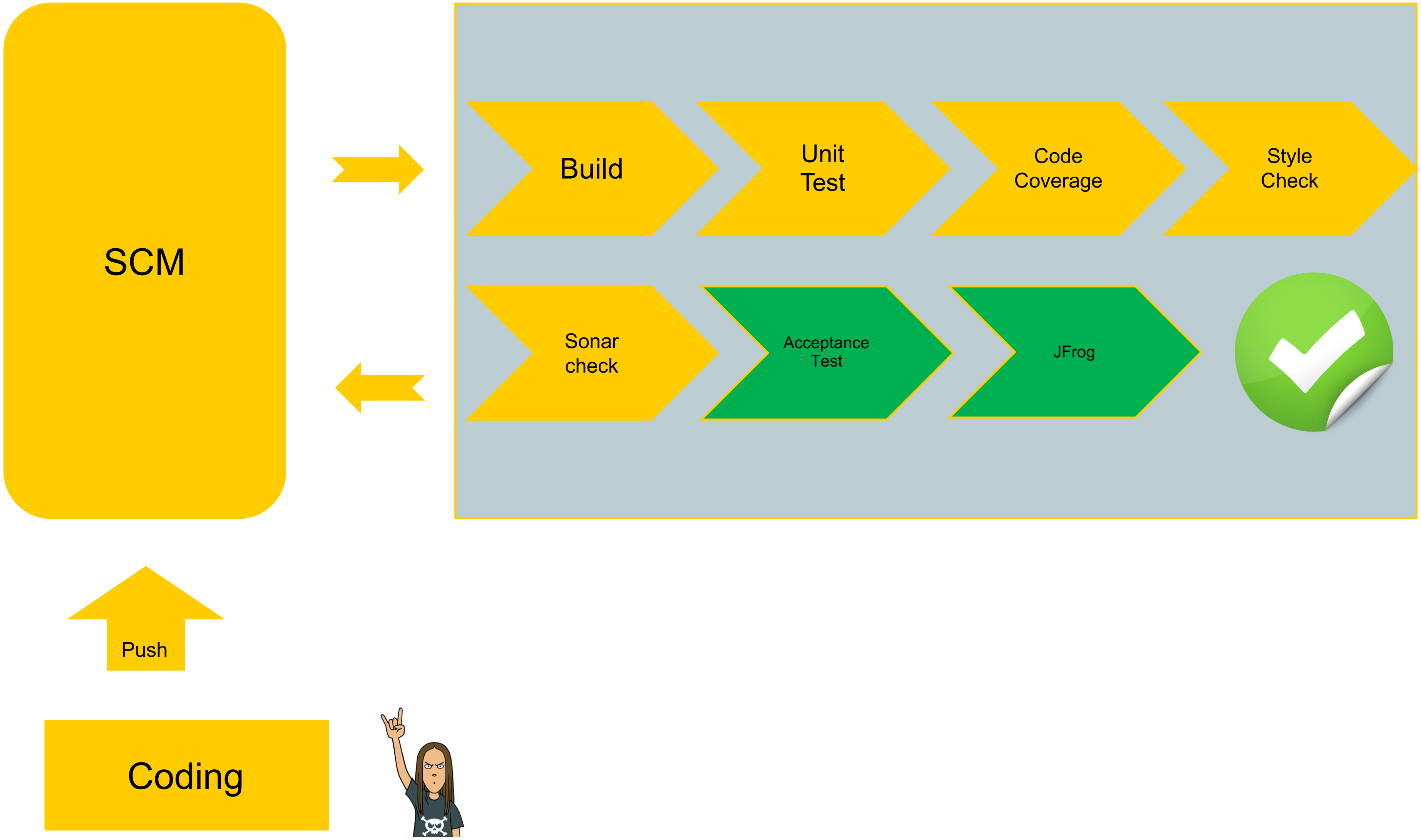
➔ *Continuous Integration is a software development practice where members of a team integrate their work frequently. Each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible. Many teams find that this approach leads to significantly reduced integration problems and allows a team to develop cohesive software more rapidly.*

Martin Fowler

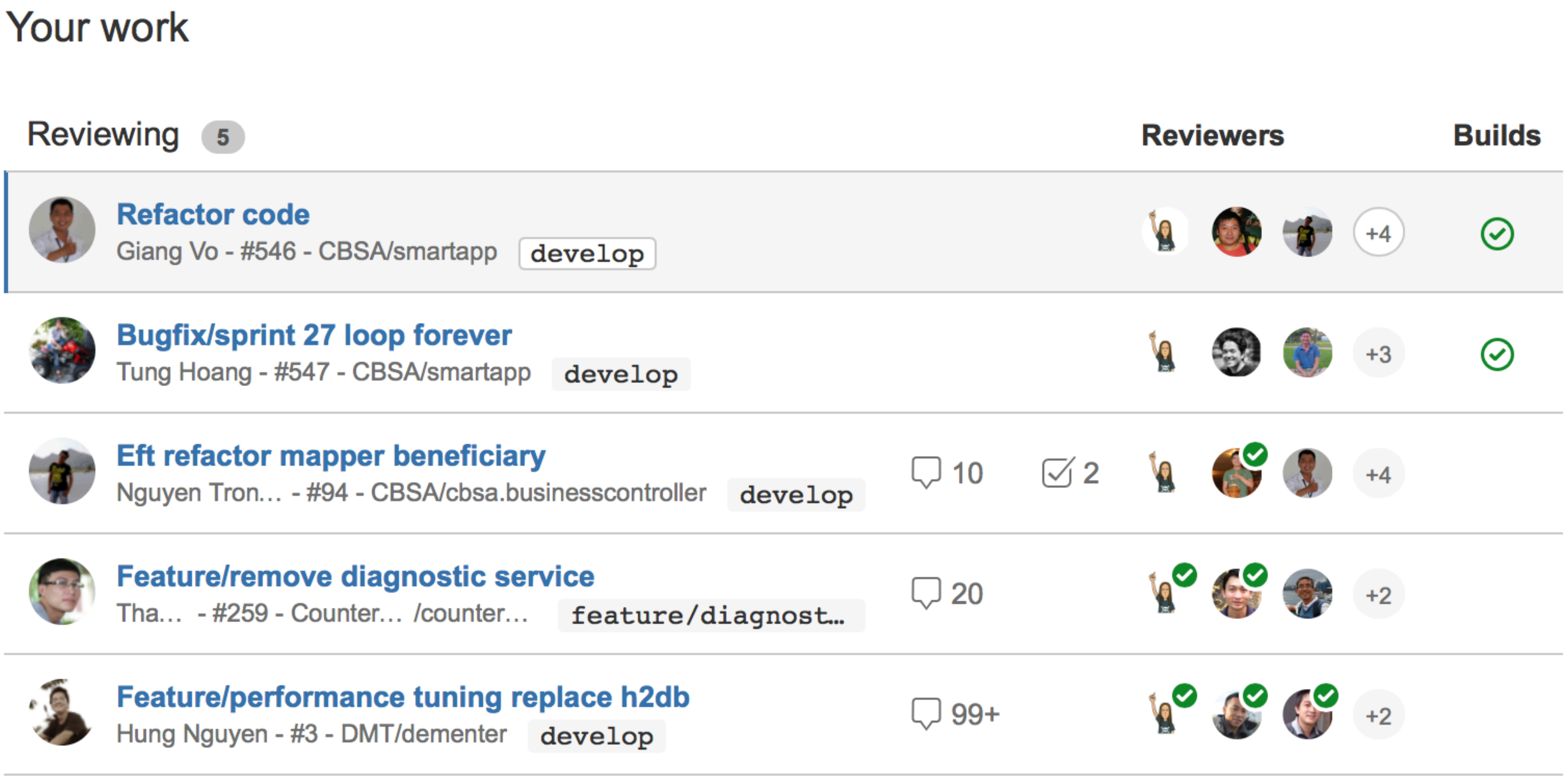
CI tools

- Jenkins
- TeamCity
- Travis CI
- Go CD
- Bamboo
- Circle CI
- Codeship
- …

A good build pipeline



Code review



Your work

Reviewing 5			Reviewers	Builds
Refactor code Giang Vo - #546 - CBSA/smartapp `develop`			+4	✓
Bugfix/sprint 27 loop forever Tung Hoang - #547 - CBSA/smartapp `develop`			+3	✓
Eft refactor mapper beneficiary Nguyen Tron… - #94 - CBSA/cbsa.businesscontroller `develop`	10	2	+4	
Feature/remove diagnostic service Tha… - #259 - Counter… /counter… `feature/diagnost…`	20		+2	
Feature/performance tuning replace h2db Hung Nguyen - #3 - DMT/dementer `develop`	99+		+2	

CAN

Engineering Guild

- Purpose:
  - Sharing knowledge
  - Engineering Practices
  - Innovation
- Values
  - Selflessness
  - Passion
  - Communication

- Innovative ideas from the guild
  - Hercules
  - Achilles – the mock service
  - Poseidon – GA is very simple

CAN

Recap

- Automate the testing as much as possible
- Choose a good architecture works for your team
- Use static analysis tools
- Fail fast
- Collaboration
- Build an engineering culture
- Sample code :https://github.com/hai-nguyen/Impala

CAN

THANK YOU

CAN