Automation projects technical guidelines

Raul Garvizu, Silvia Valencia, Mauricio Cadima

April 2015

Jala oft

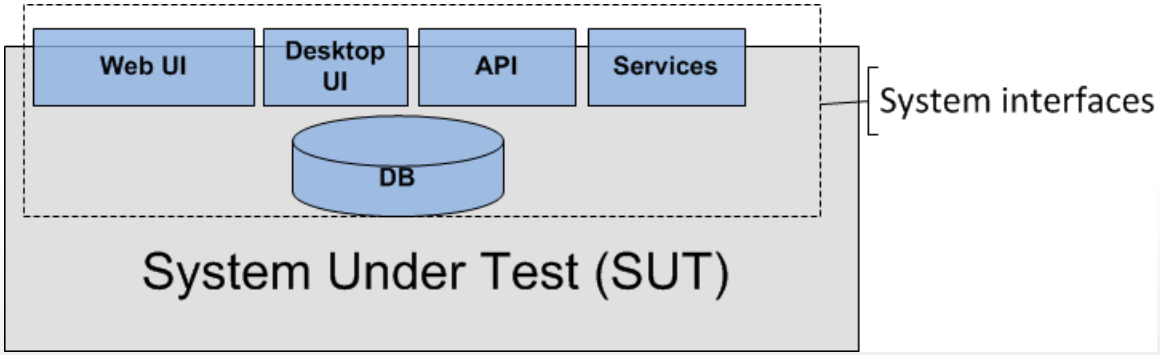
Agenda

- Architectural and design recommendations
  - Architecture recommendations
  - Doman Specific Language
  - Design Patterns
  - Object Oriented Design Principles
- Testing strategy and test definition best practices
  - Test design considerations
  - Automated Test Strategy
  - Test case Organization recommendations
- Development process best practices
  - Best development process practices like continuous integration, execution and others
  - Source Code management and branching model

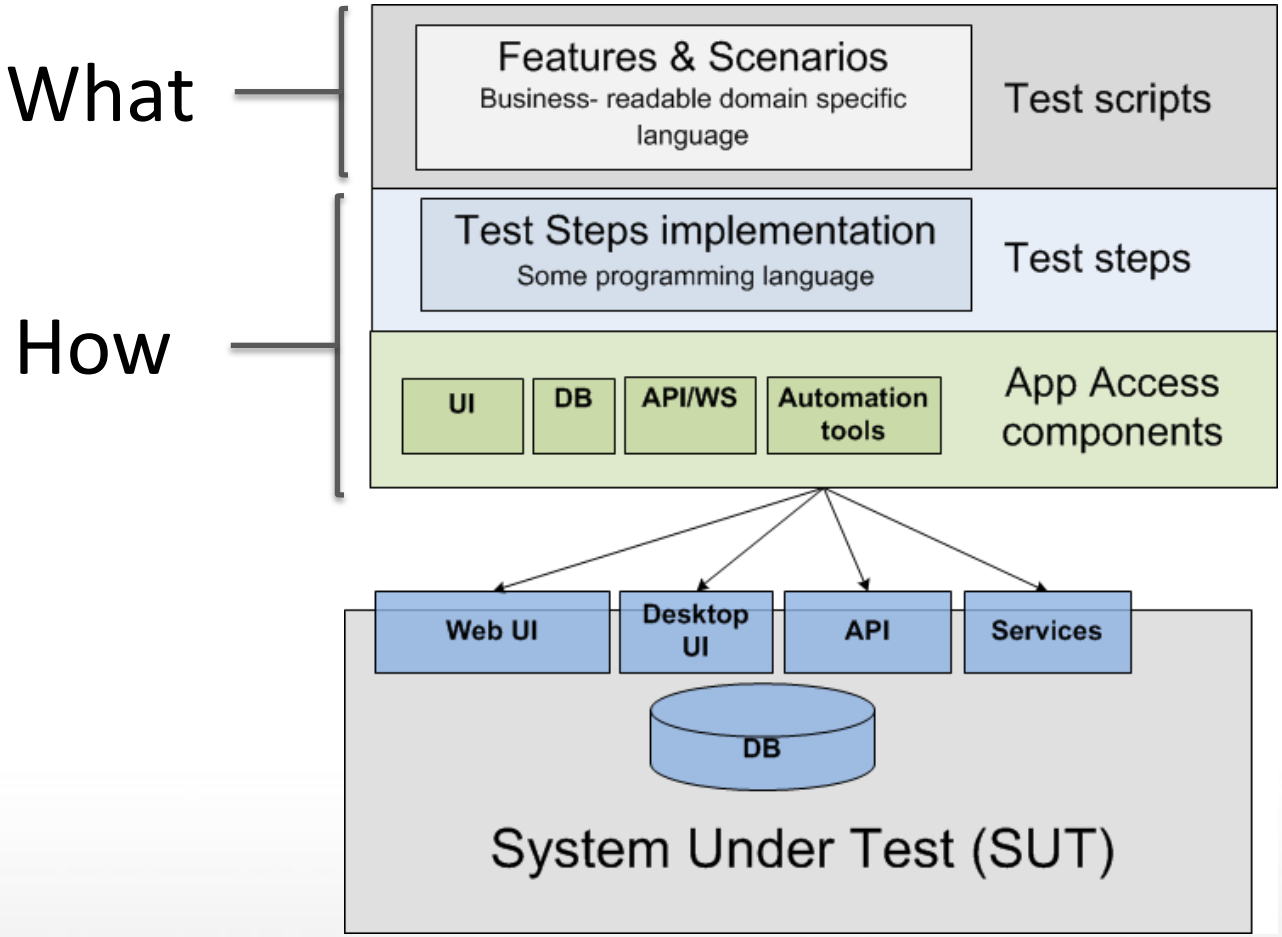
ARCHITECTURAL AND DESIGN RECOMMENDATIONS

SUT & System interfaces

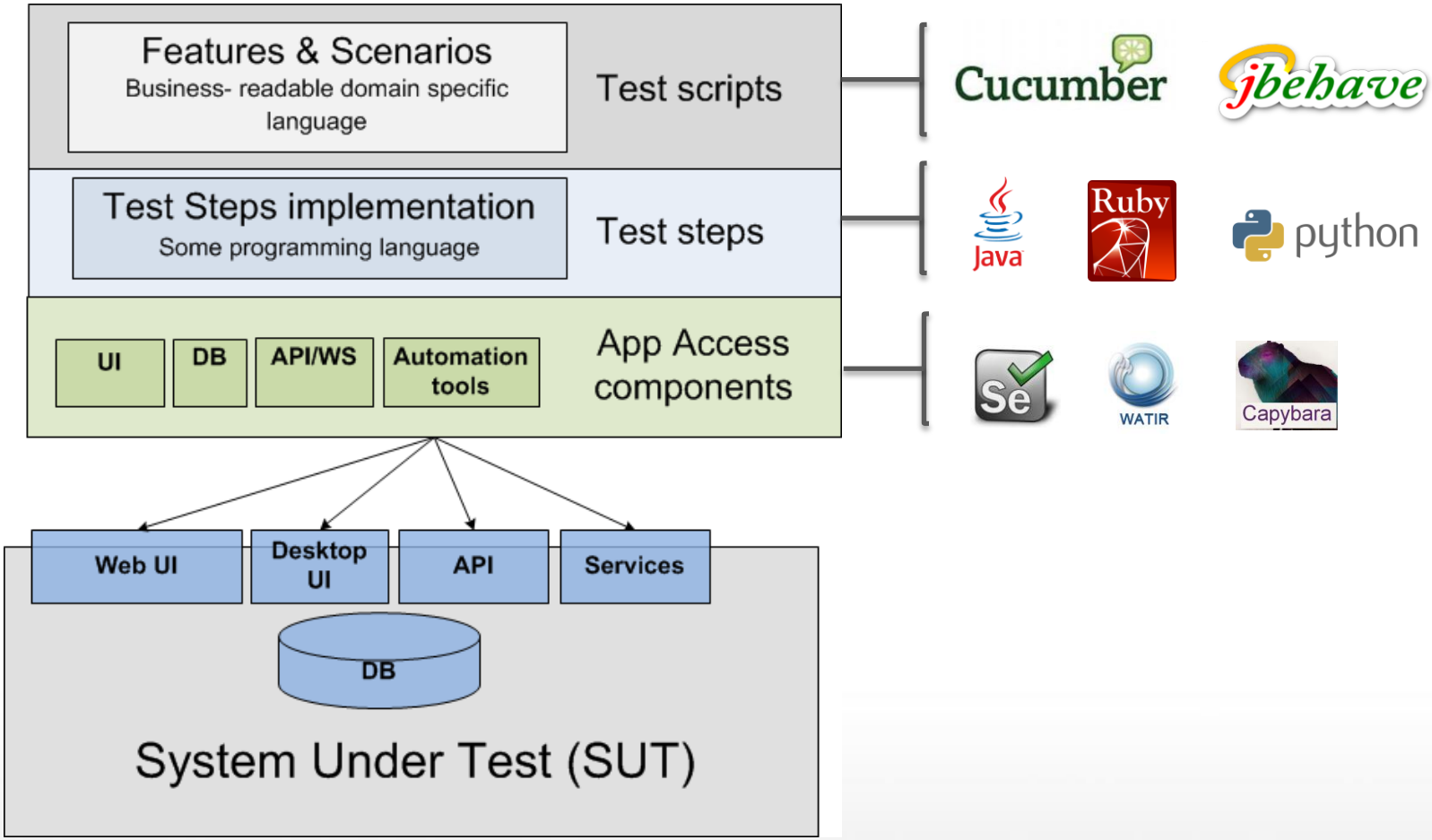
Be aware of the System Interfaces
You may need to interact with them for testing or validation purposes



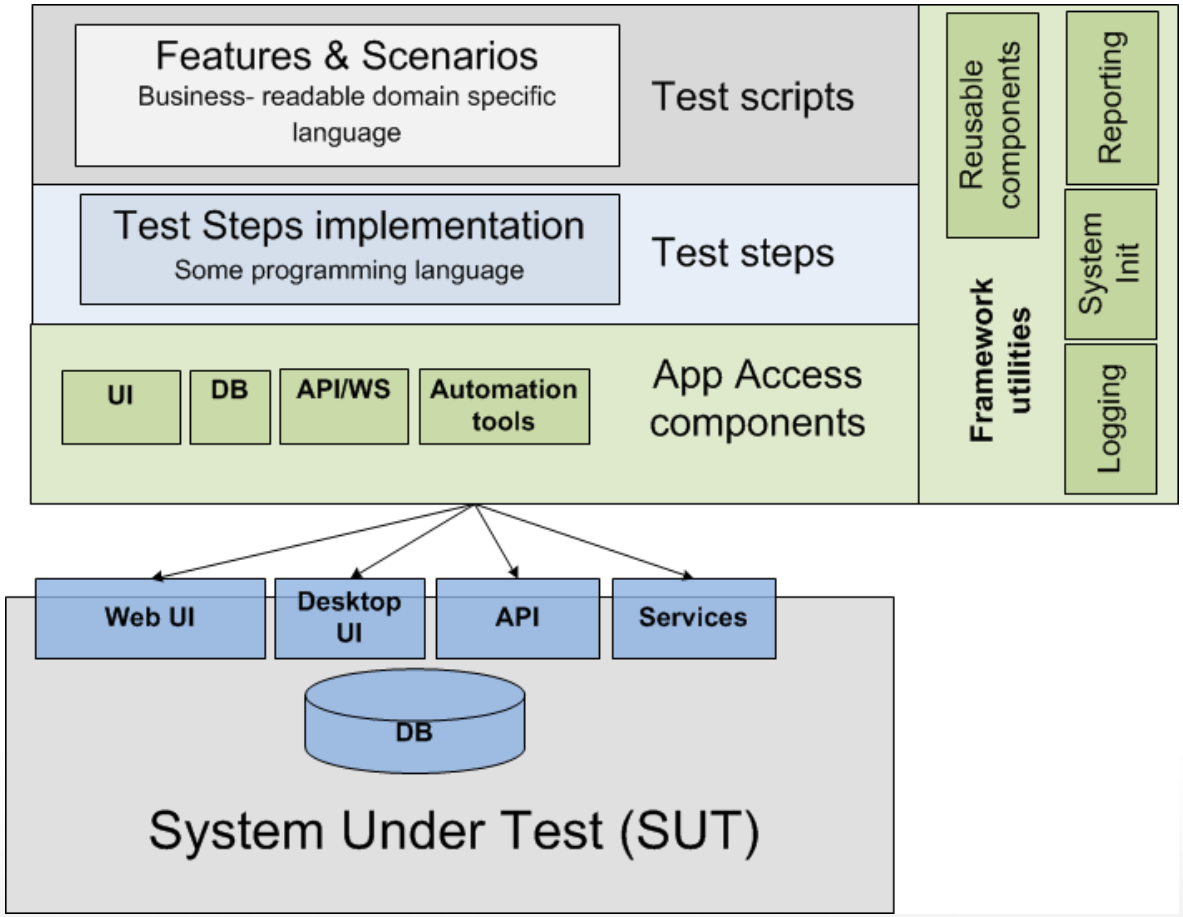
Architecture recommendations – SoC via Layers



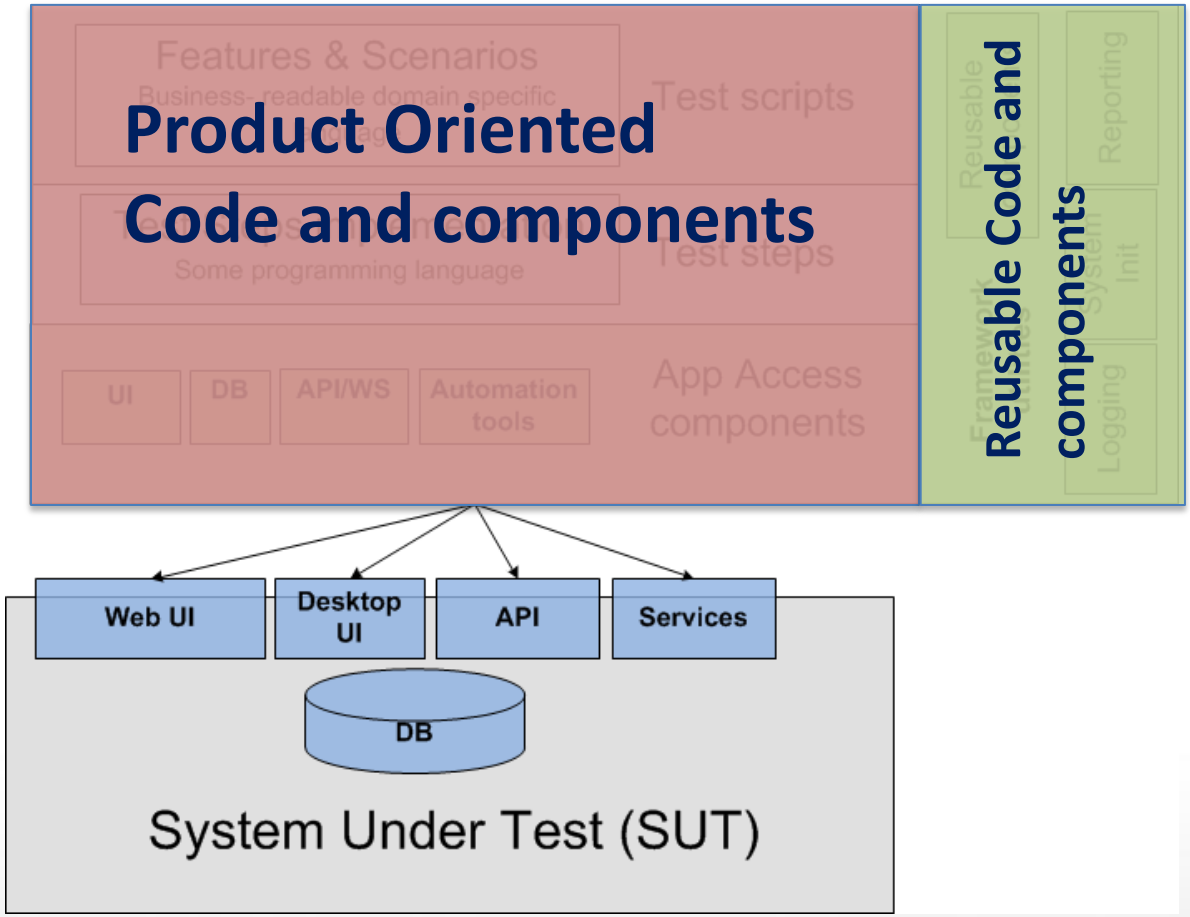
Architecture Recommendations – Tech Stack



Architecture Recommendations – Framework utilities

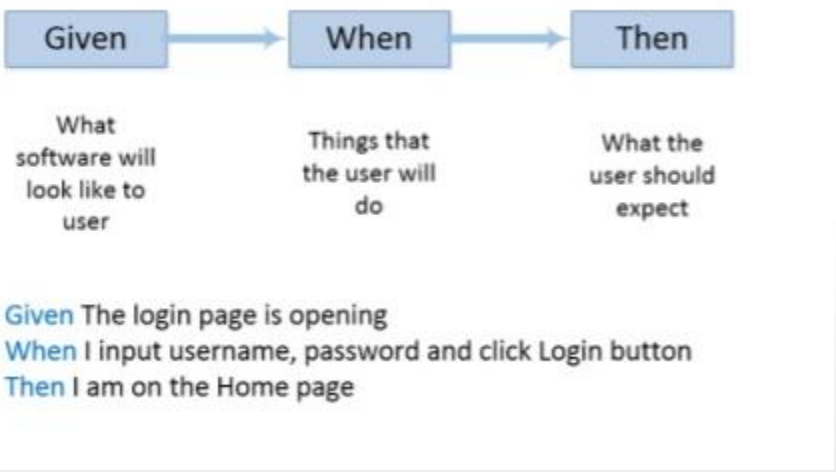


Architecture Recommendations – Framework utilities



Use Domain Specific Language, why?

- Simplifies communication
  - Reduces misunderstandings
  - Test expressed in a language understandable by business and technical people
  - Ensures that everyone is not only on the same page, but also using the same words
- Documents behavior
  - The tests document the behavior of the system and are also valuable as documentation of the product



Tips

- Fight for the correct adoption of BDD in the development life-cycle , i.e. push to use DSL to define user stories acceptance criteria

Domain Specific Language : Gherking

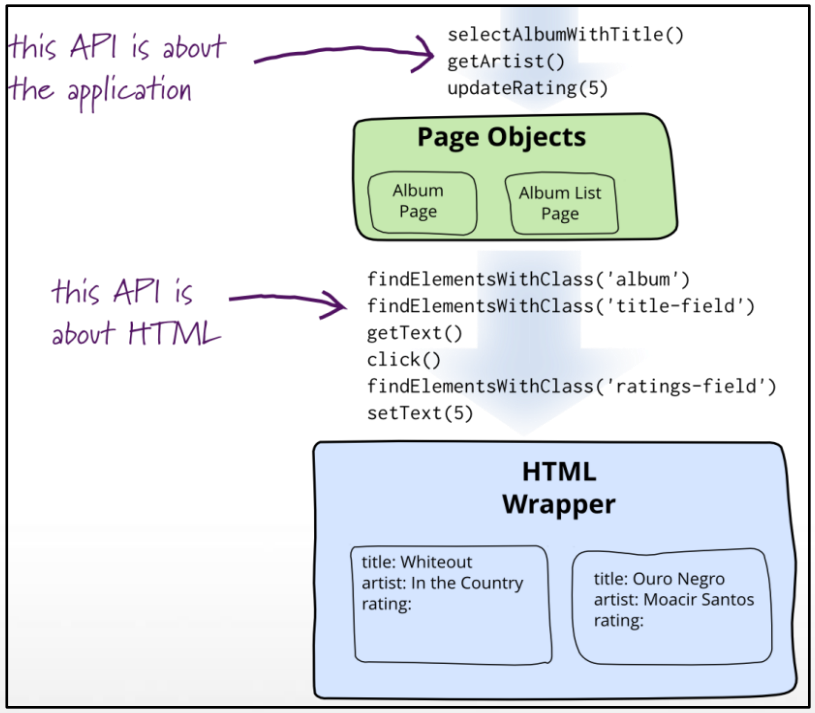
```
Feature: Some terse yet descriptive text of what is
desired
    Textual description of the business value of this
    feature
    Business rules that govern the scope of the feature
    Any additional information that will make the feature
    easier to understand

    Scenario: Some determinable business situation
        Given some precondition
        And some other precondition
        When some action by the actor
        And some other action
        And yet another action
        Then some testable outcome is achieved
        And something else we can check happens too
```

Recommended Design patterns

- Adopt Page Objects Pattern (Martin Fowler)

  Goal: Clean separation between test code and UI elements' interaction code



Ref: Martin Fowler site – Page Objects article

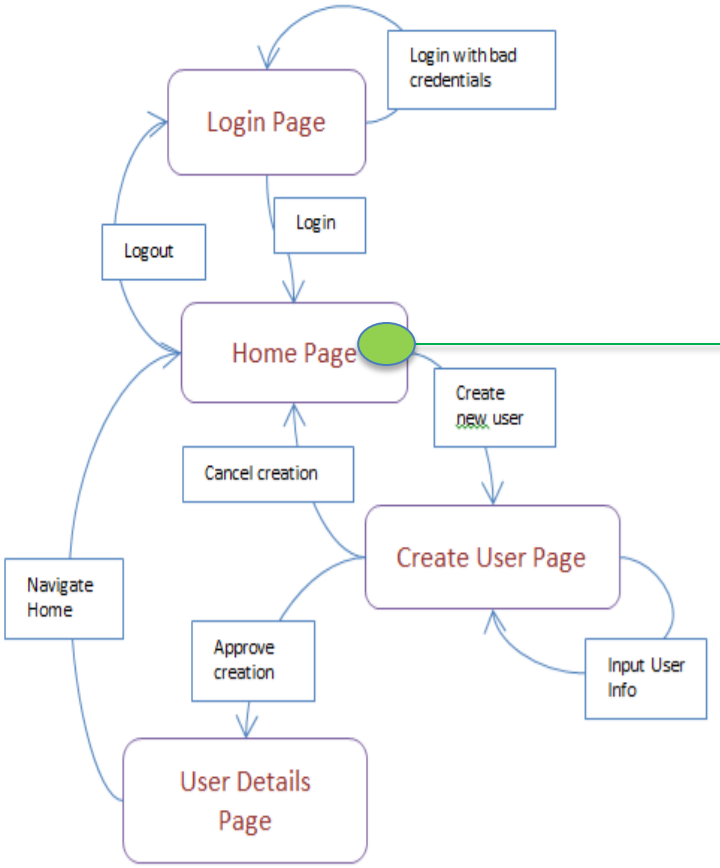
Benefits

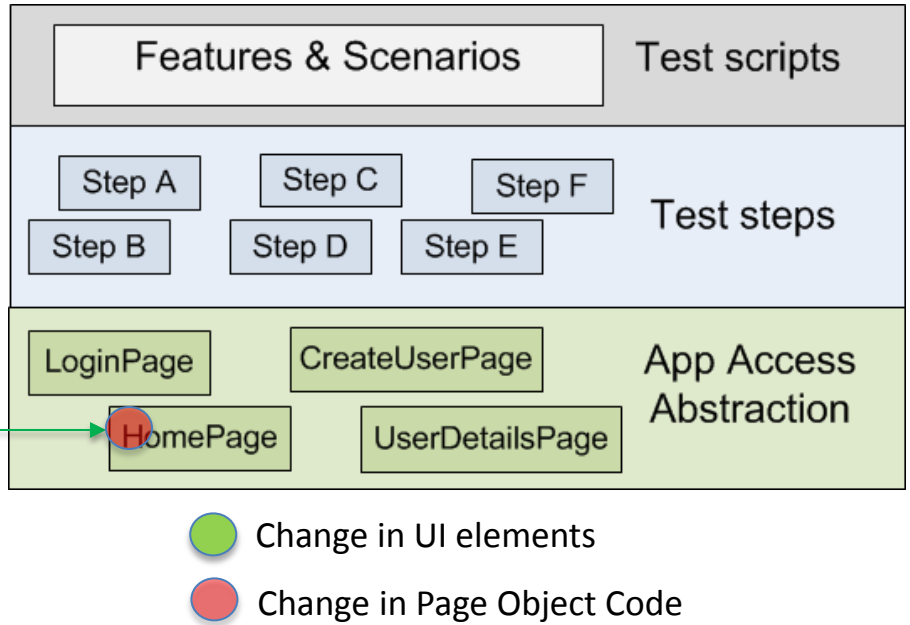
- Promotes code reuse and reduces duplication
- Improves test readability
- Increases test robustness
- Improves maintainability
- Improves SoC

Proposal: Design Patterns Adoption

Page Objects (Martin Fowler)

Goal: Clean separation between test code and UI elements' interaction code





Change in UI elements

Change in Page Object Code

Recommended Design patterns

- Page Factory

  Lets to automatically initialize elements defined in Page Object (binding – action)

```
public class LoginPage extends BasePage {

    @FindBy(how= How.ID, using ="email")
    @CacheLookup
    protected WebElement emailInput;

    @FindBy(how= How.ID, using ="password")
    @CacheLookup
    protected WebElement passwordInput;

    public LoginPage(){
        PageFactory.initElements(TestHooks.driver, this);
    }
```

FindsBy- annotation- automatically find elements on page.

CacheLookup- annotation– caches found element.

Benefits

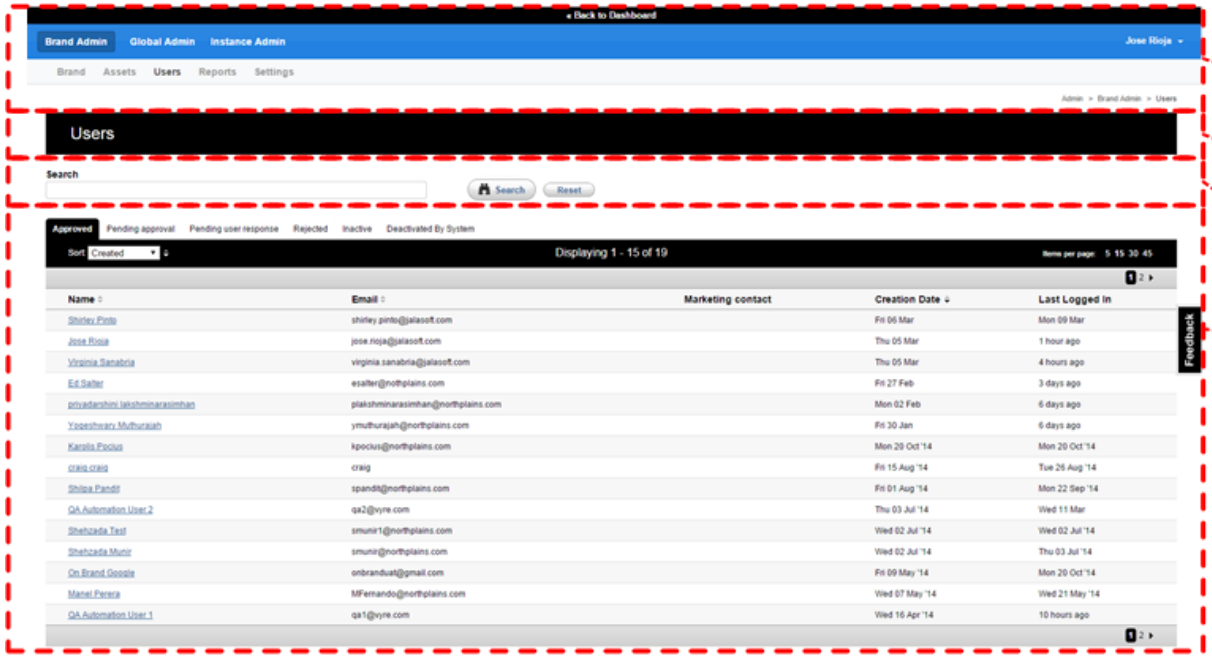
- Simpler code, improves readability
- Improves performance (caching)
- Reduces verbosity

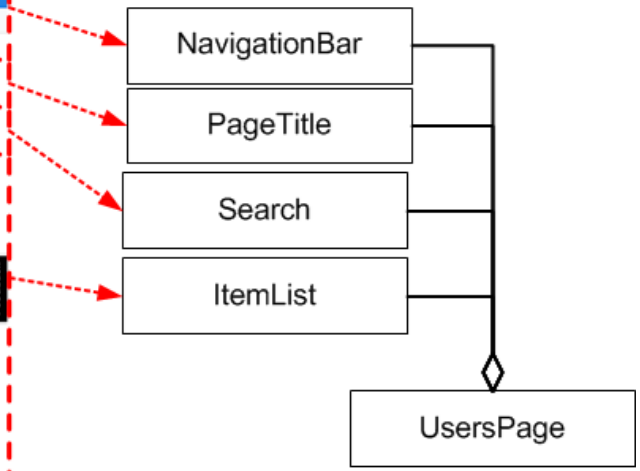
Ref: Selenium wiki – PageFactory article

Recommended Design patterns

- Composite Page Object

  Aggregates reused page objects/components in one external object





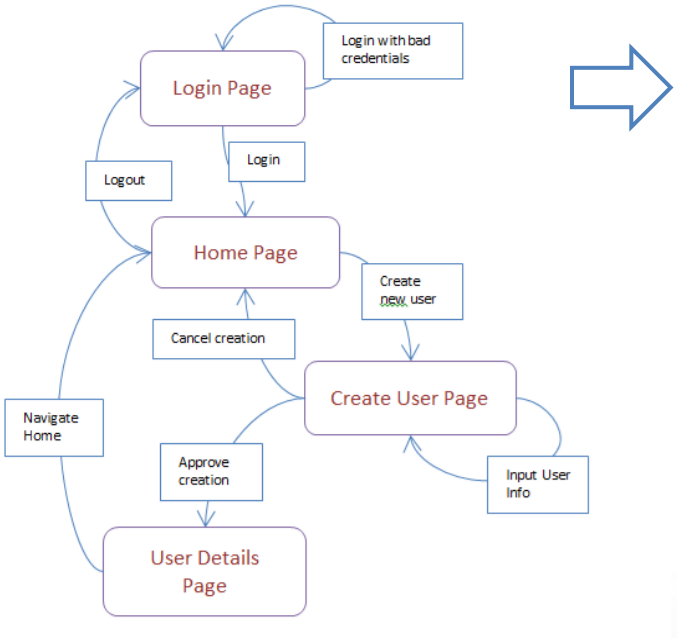
Benefits

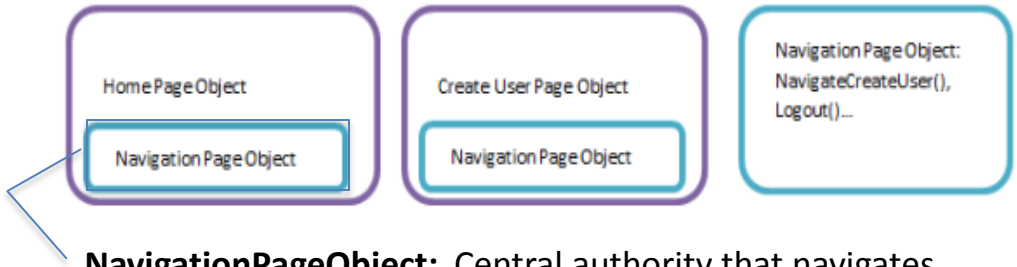
- Promotes code reuse and reduces duplication
- Improves maintainability
- Improves SoC and encapsulation

Recommended Design patterns

- Transporter/Navigator

  Centralizes the navigation control in the tested system according to the test requirements





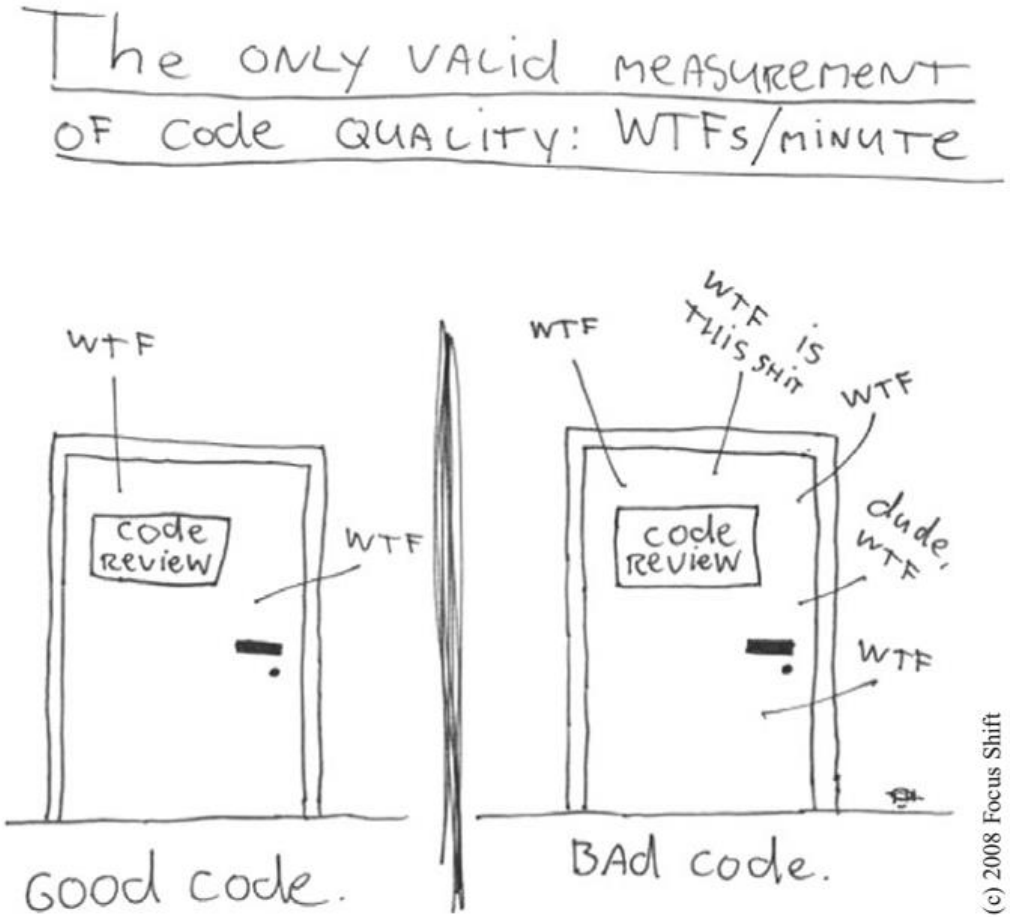
NavigationPageObject: Central authority that navigates through the application on behalf of tests.

Benefits

- Improves maintainability
- Improves SoC

Ref: [Design Patterns for Customer Testing]

Clean code



Reproduced with the kind permission of Thom Holwerda.
http://www.osnews.com/story/19266/WTFs_m

Object Oriented Design Principles

Single Responsibility Principle

A class should have one, and only one, reason to change.

Open Closed Principle

You should be able to extend a classes behavior, without modifying it.

Liskov Substitution Principle

Derived classes must be substitutable for their base classes.

Interface Segregation Principle

Make fine grained interfaces that are client specific.

Dependency Inversion Principle

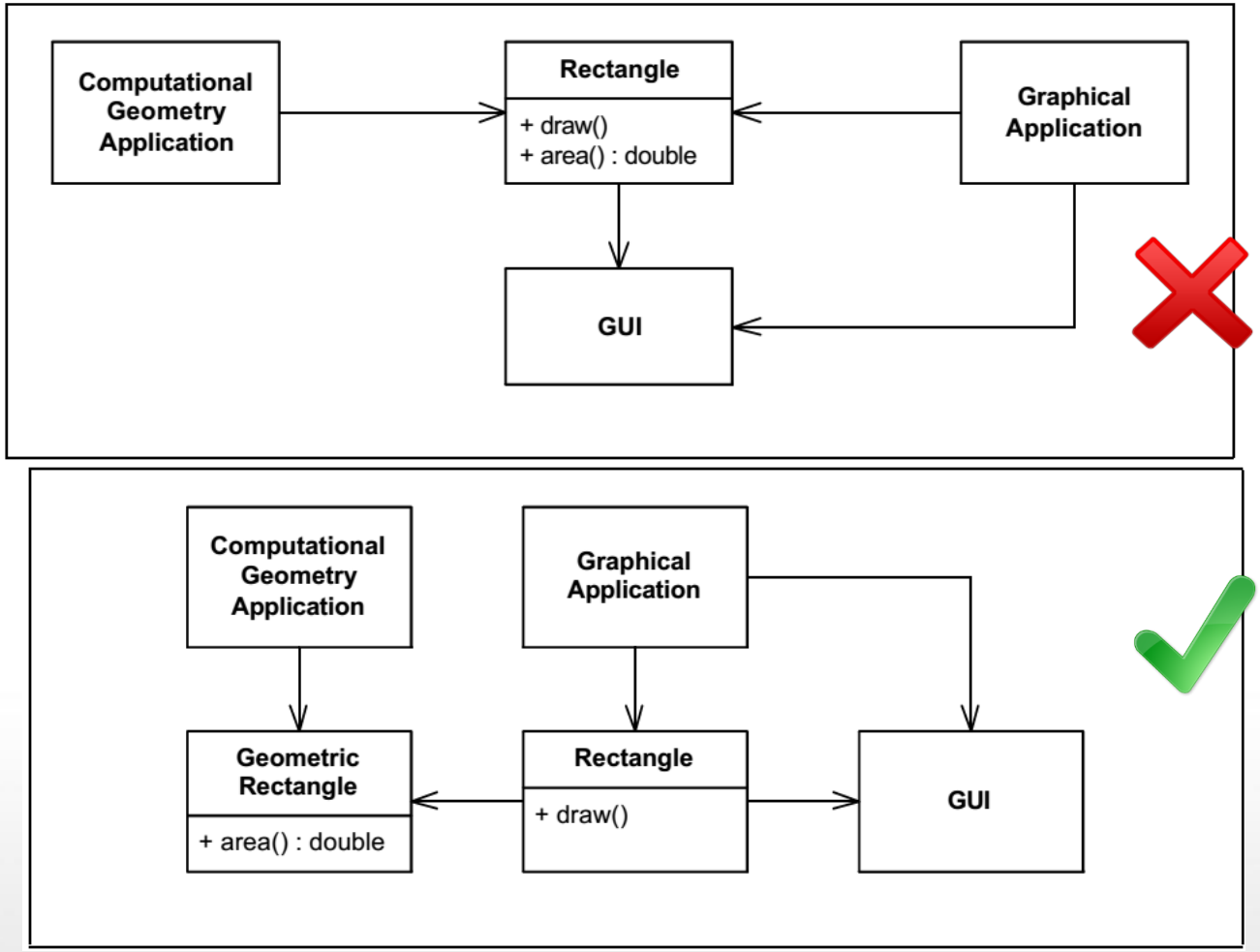
Depend on abstractions, not on concretions.

Clean, maintainable , readable , extendible, reusable code

Ref: The Principles of OOD
Robert C Martin,

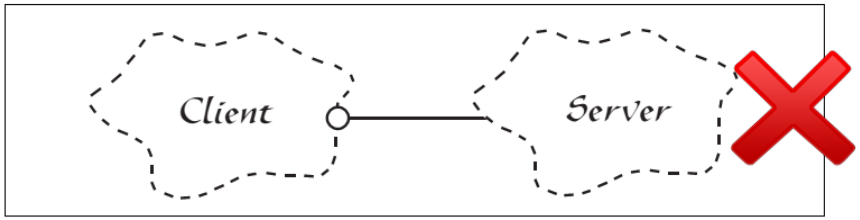
Single Responsibility Principle

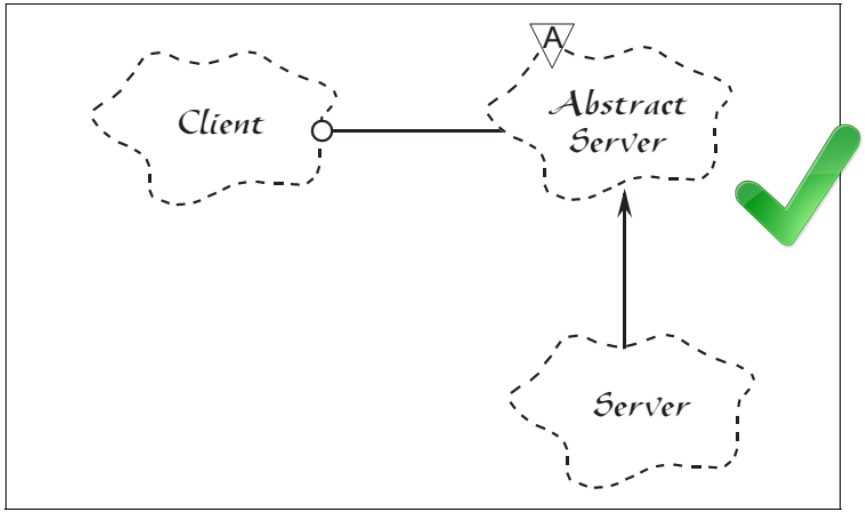
A class should have one, and only one, reason to change.

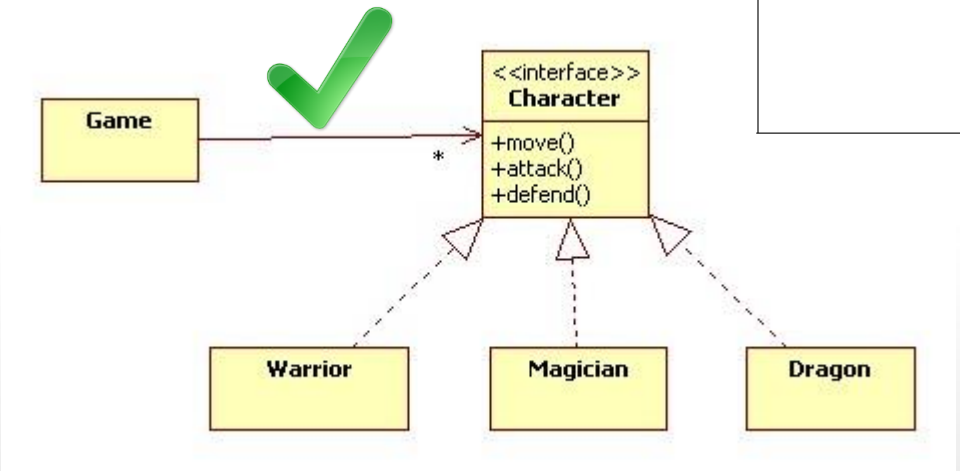


Open Closed Principle

"Software entities (classes, modules, methods) should be open for extension, but closed for modification"

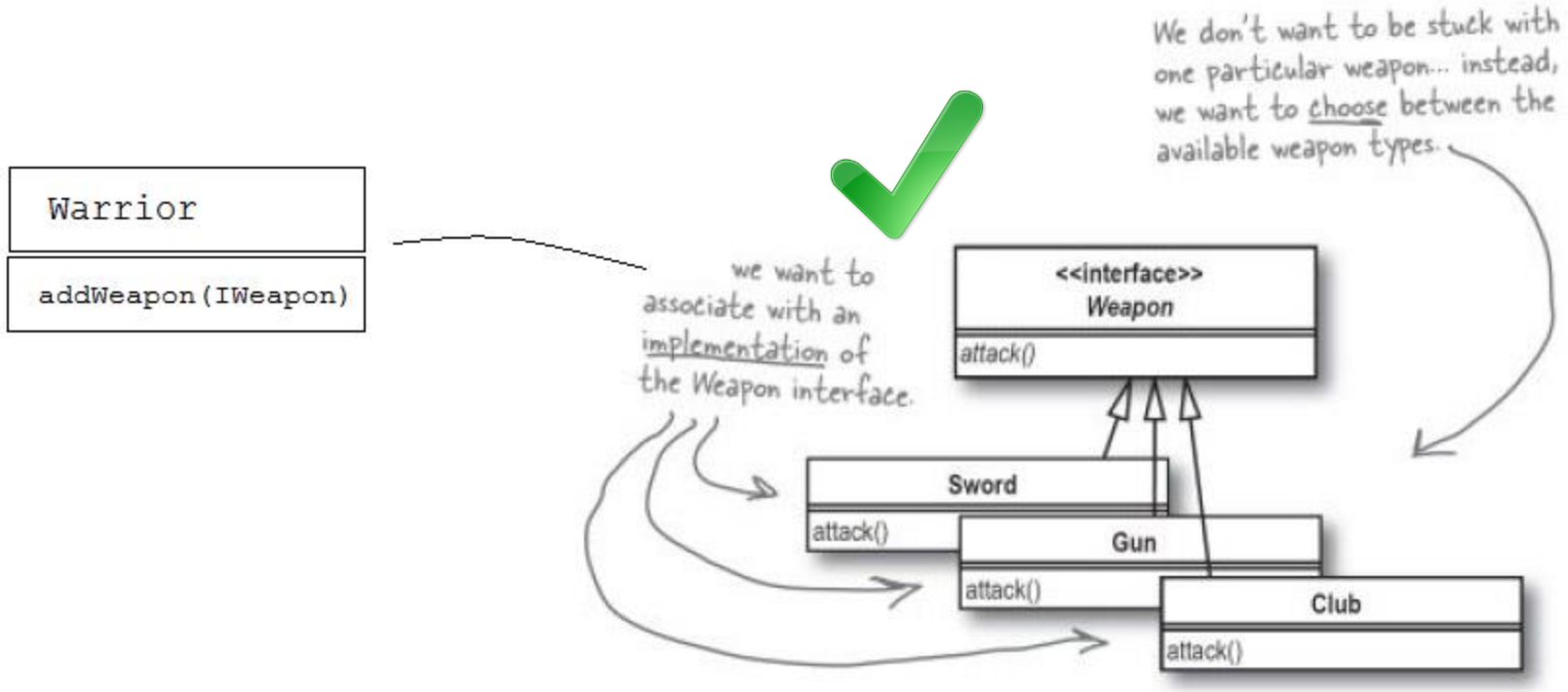






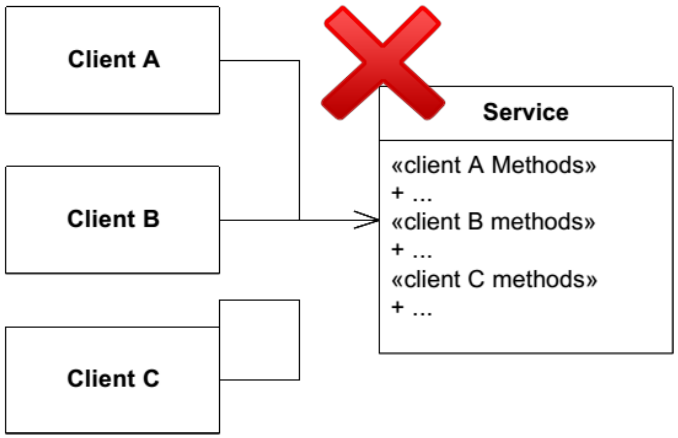
Liskov Substitution Principle

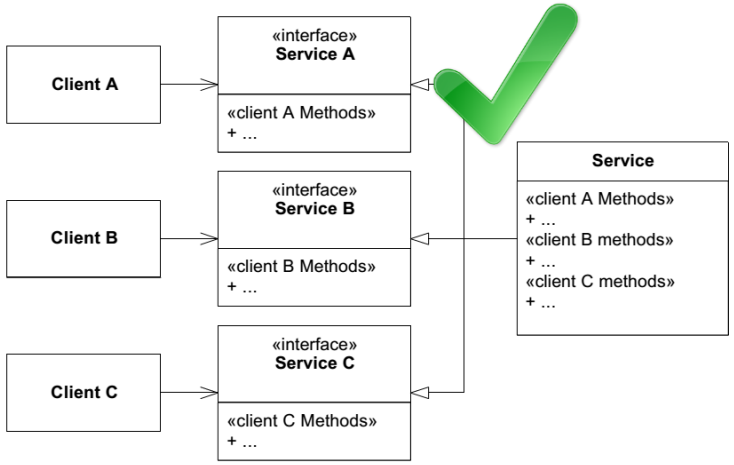
"Subtypes must be substitutable for their base types"



Interface Segregation Principle

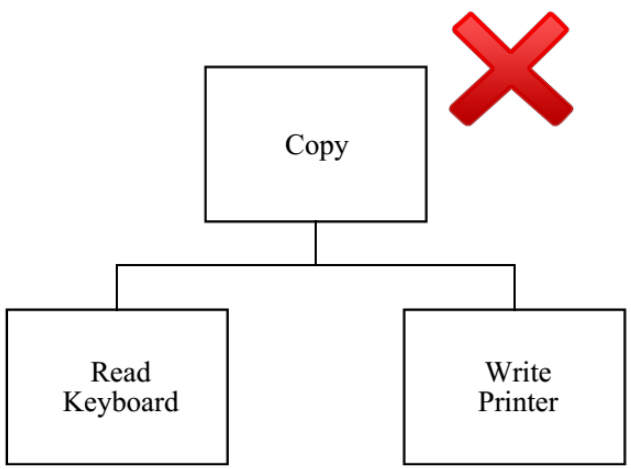
"Many client specific interfaces are better than one general purpose interface"
"Clients should not be forced on methods that they do not use"

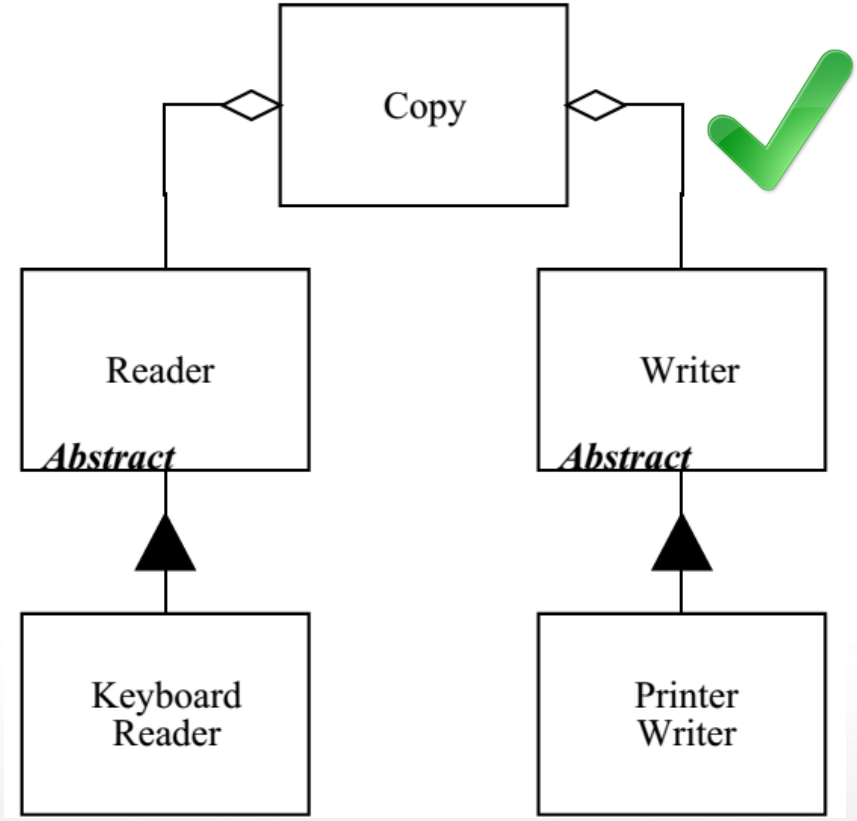




Dependency Inversion Principle

"Depend on abstractions. Do not depend on concretions"





Clean Code recommendations

- Code Review is a must
- Apply Naming Conventions & Coding Standards
- Enforce coding standards through check-in policies.

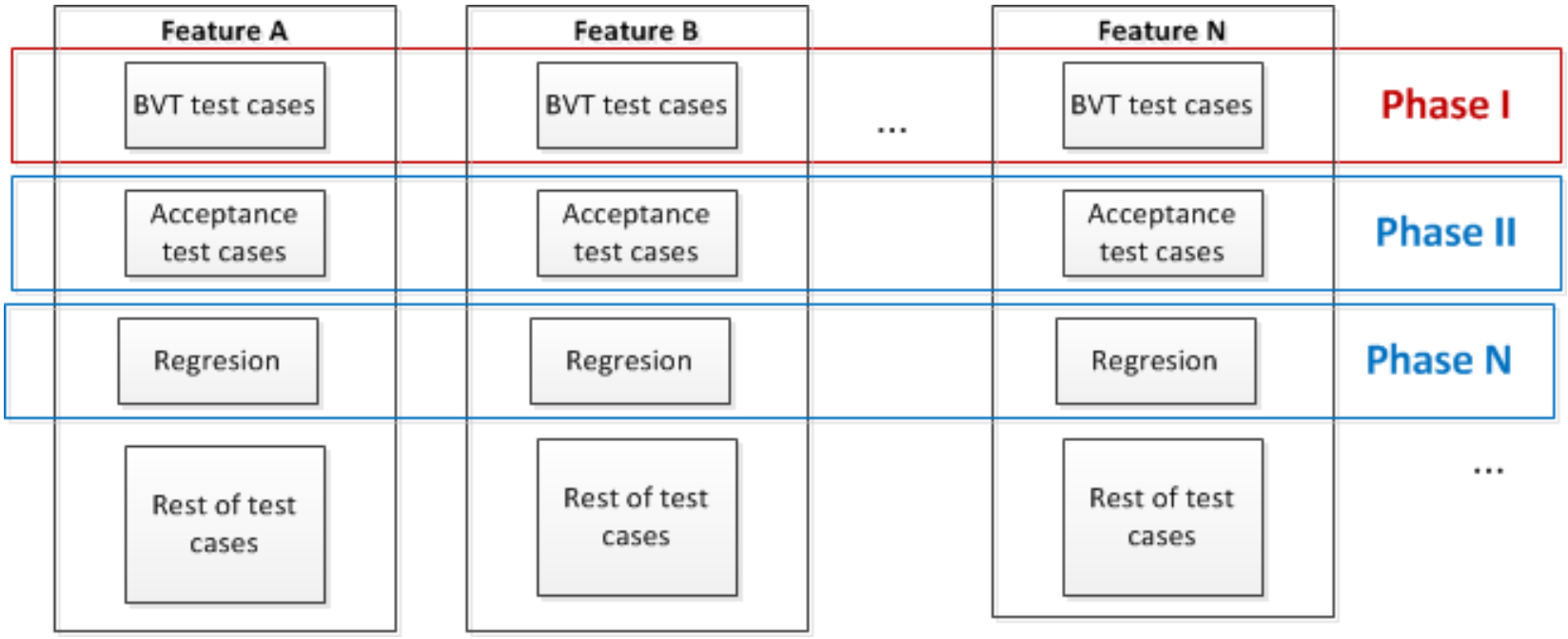
TESTING STRATEGY AND TEST DEFINITION BEST PRACTICES

Get agreements with development and stakeholders

- UI element IDs and naming conventions
- UI specific behaviors or signals for testing purposes
- How acceptance criteria will be defined and expressed
- How Internationalization support will be validated (i.e. Can dev and automation share language resource files?)

Establish consistency in approach which positively impacts efficiency.
And encourage full-team participation in overall product quality as a result of collective standards.

Automated Test Strategy



Test case Organization recommendations

- Test Case Organization: Tags by priority, features, areas, type of execution (BVT, Full Regression, etc.)

```
@billing @bicker @annoy
Feature: Verify billing
```

Gain test organization consistency and execution flexibility

Expand Test Design Strategies

- Location elements strategy
  - By Id, Name, CSS and/or XPath
- Avoid to use random sleeps , use explicit “wait” for elements
- Move some configuration information to external files.
  (i.e. paths, timeouts , user credentials, etc.)

More efficient test case creation and execution while reducing exposure to non-functional breakdowns (e.g. timeouts).

DEVELOPMENT PROCESS BEST PRACTICES

Streamline the Automation Process

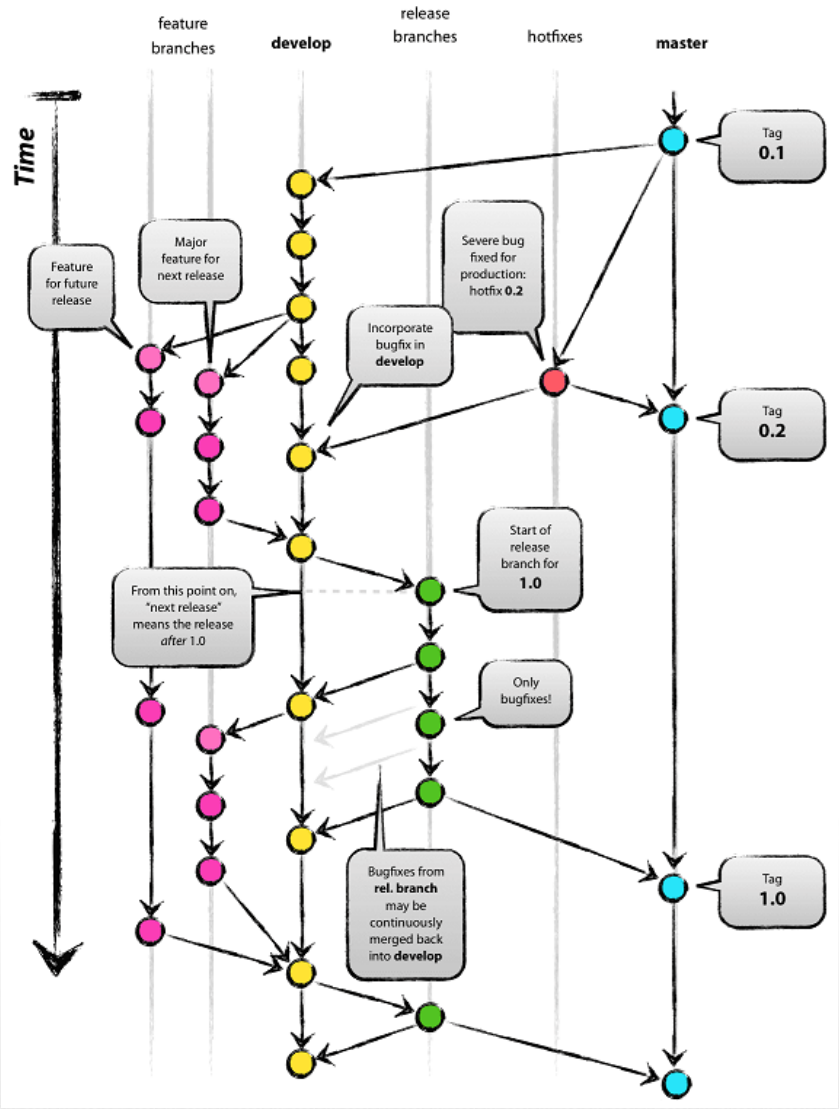
- Treat automation as you would any other software project:
  - Code Reviews
  - Adopt continuous integration principles/execution
  - Establish parallel execution for multi-browser support
  - Require unit testing and mocking techniques as part of test writing
  - Include automated performance and scalability testing

The value gained from automation typically is in direct proportion to the investment made to development process and good practices adoption, it can be incremental but it must be significant in its own right.

Code management and branching model is important

Adopt Gitflow as branching model

- Provides a robust framework for managing code in small and larger projects.
- Ensures consistency in code branching strategy and release management.
- Increased simplicity and reduce the risk of introducing bugs due bad code management.



Increased simplicity, consistency agility and robustness in code management through effective tools and elegant and easy to understand model.

Q&A?

Controversial Question

Should automated scalability, performance , security testing be part of the automation QA/Dev role?

My answer is

YES

References

- Martin Fowler site – Page Objects article *by* Martin Fowler site
- Selenium wiki – PageObjects article
- Selenium wiki – PageFactory article
- Design Patterns for Customer Testing *by* Misha Rybalov
- Simon Stewart blog
- Cucumber wiki
- Git SVN Comparison
- A successful Git branching model *by* Vincent Driessen
- Gitflow workflow *by* Atlassian
- The Cucumber Book – BDD for testers and developers *by* Matt Wynne and Aslak Hellesoy
- Selenium 2 Testing Tools *by* David Burns

Contact Information

Contact People: Mauricio Cadima, Silvia Valencia, Raul Garvizu

E-mail:

mauricio.cadima@jalasoft.com, silvia.valencia@jalasoft.com , raul.garvizu@jalasoft.com