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

Martin Fowler

Unit tests en Xamarin

Alexander Llanes

Xamarinuy

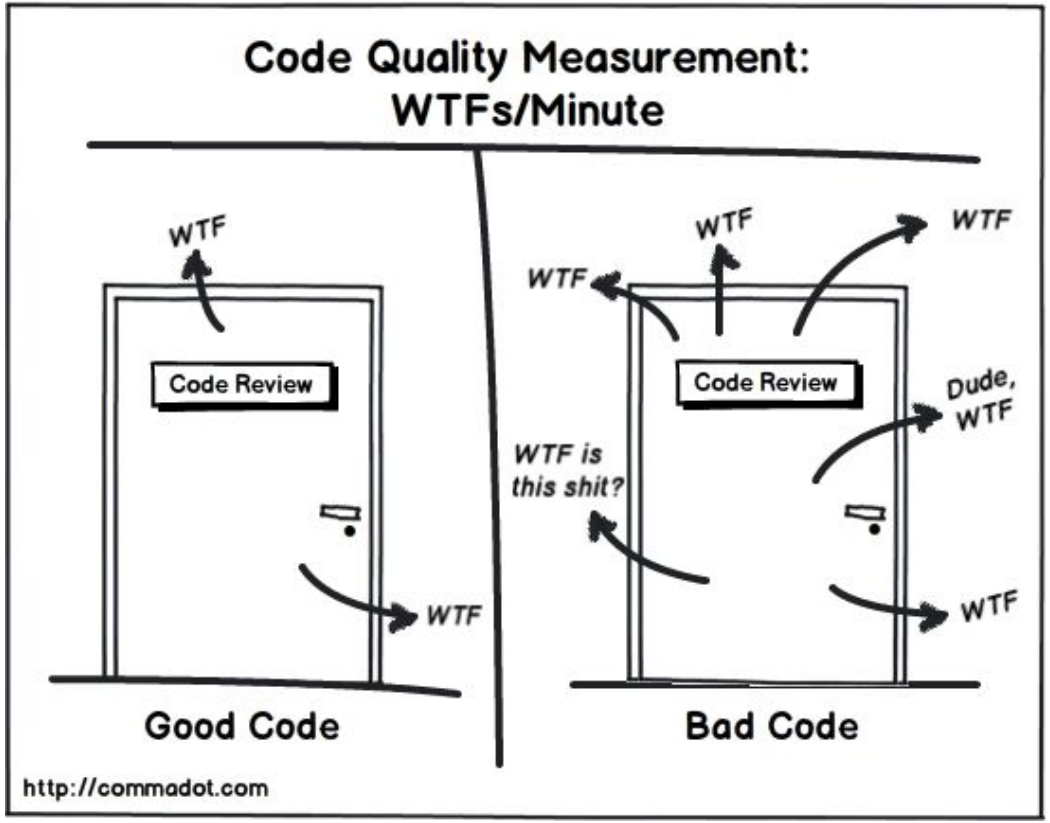
alex.llanes@gmail.com

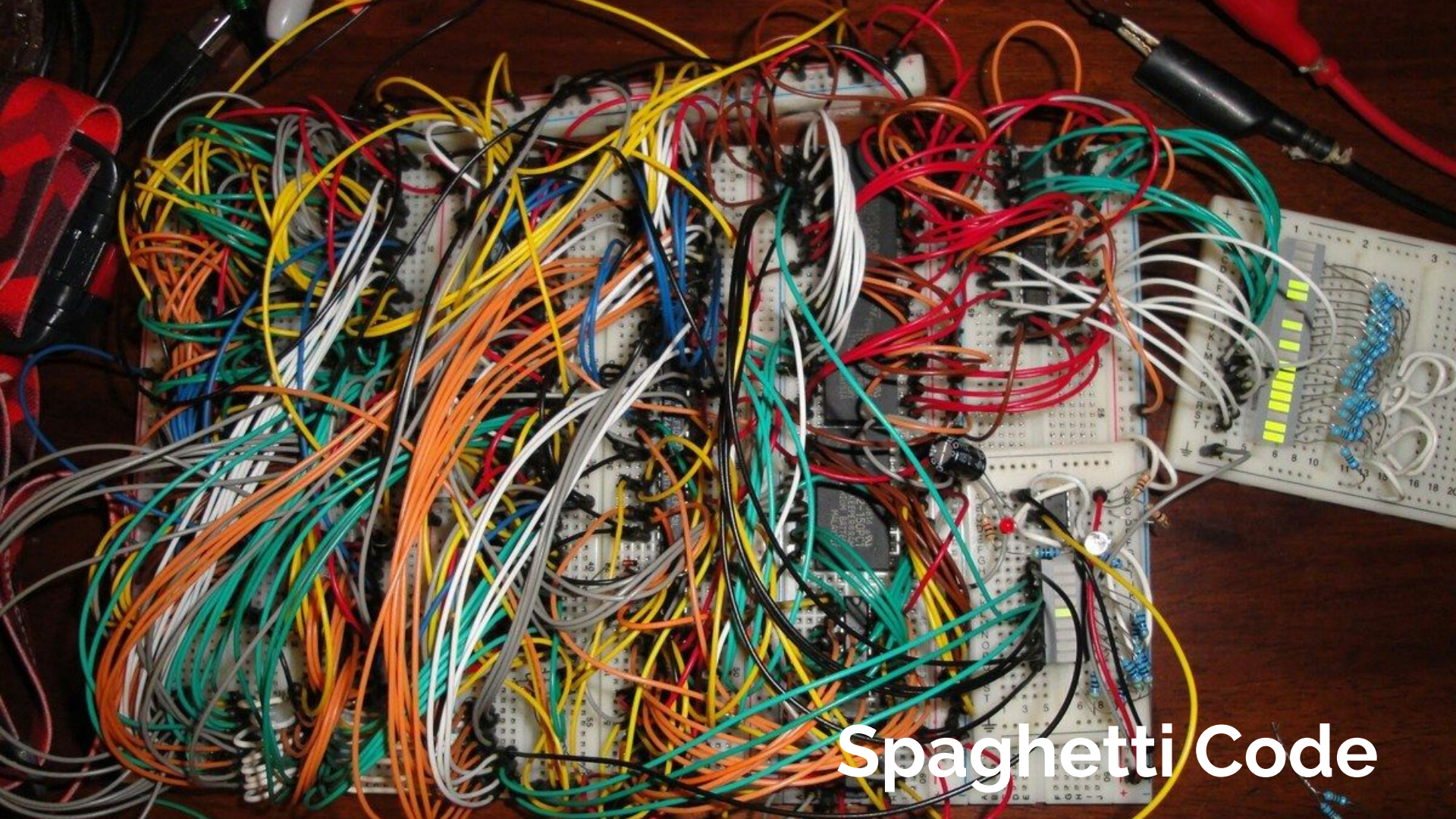
allanes87

Linkedin

The beginning

Which door represents your code?



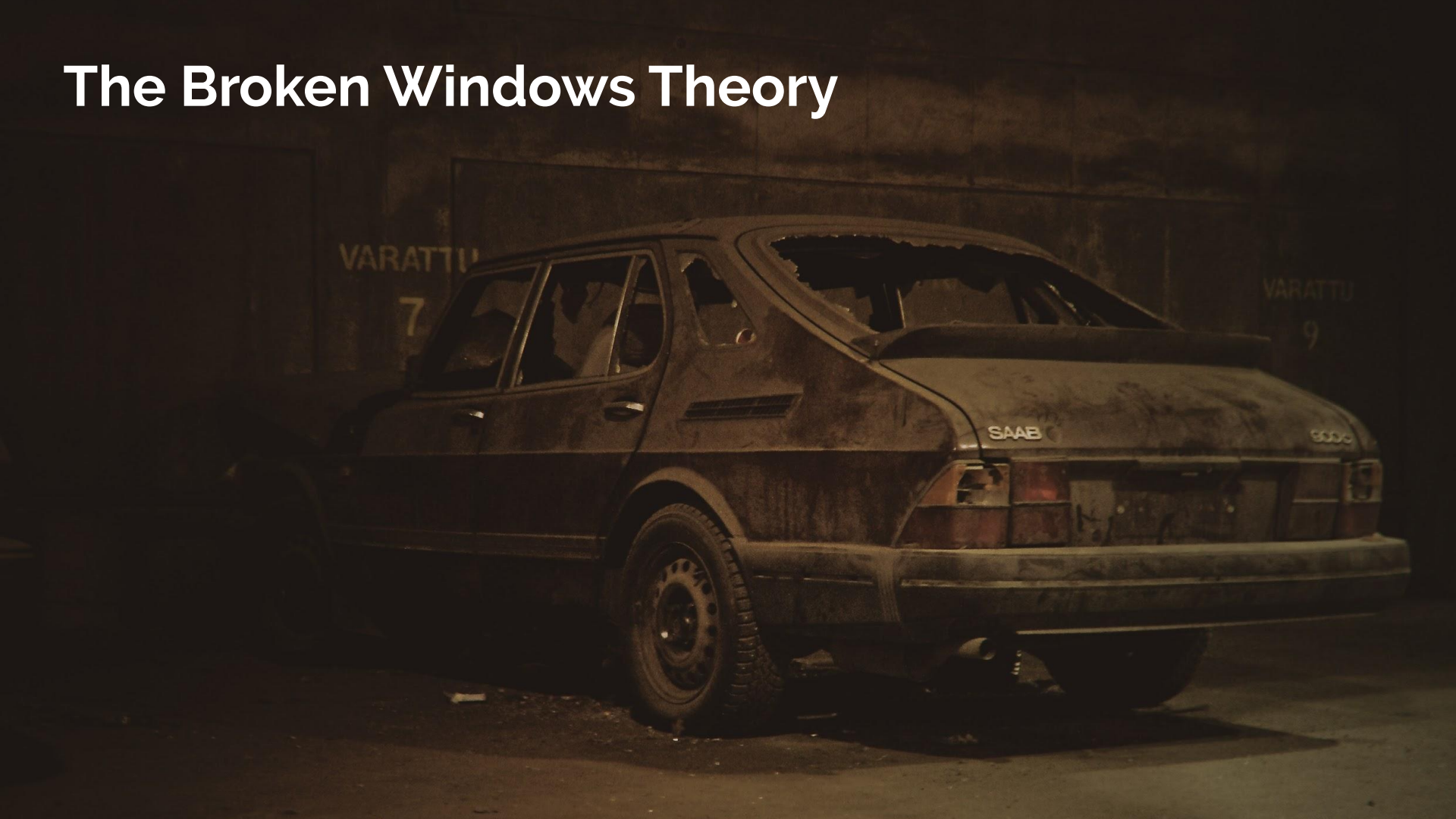

Spaghetti Code

Clean Code is code, that is easy to understand and easy to change

Boy scout rule

Leave your code better than you found it.

The Broken Windows Theory



Keys

Meaningful Names

DRY

Functions

Comments

Dependency inversion

Code review

SOLID

Pair programming

- Use intention revealing **names**
- A **function** shouldn't have more than 3 arguments
- **function** should do one thing, do it well and should do it only
- 20 lines per function (**small functions**)
- **Methods** should have **verb** or verb phrase names like postPayment
- Correct naming can prevent **comments**
- Do not leave **commented** code
- **Classes** and objects should have noun or noun phrase names like Customer
- Don't repeat yourself (**DRY**)

- Don’t repeat yourself (**DRY**)

GenerateTripList

```
var newTrip = new TripModel()
{
    TripID = passenger.TripID,
    TripPassengerID = passenger.ID,
    ApprovalProfileID = tripApproval.ApprovalProfileID,
    TripPassengerName = string.Format("{0} {1}", profile.FirstName, profile.LastName).Trim(),
    SkyD = passenger.SuperPNR,
    HasCarReservations = passenger.HasCarReservations,
    HasFlightReservations = passenger.HasFlightReservations,
    HasHotelReservations = passenger.HasHotelReservations,
    HasMiscReservations = passenger.HasMiscReservations,
    ReasonId = tripReason.ID,
    Reason = tripReason.Reason,
    CostCenterId = (costCenter != null) ? costCenter.ID : 0,
    CostCenter = (costCenter != null) ? costCenter.Name : string.Empty,
    GradeId = (grade != null) ? grade.ID : 0,
    Grade = (grade != null) ? grade.Name : string.Empty,
    DepartmentId = (department != null) ? department.ID : 0,
    Department = (department != null) ? department.Name : string.Empty,
    TripStatus = tripStatus,
    TotalCost = GetPrice((double)tripSaving.TotalTripCost),
    SaveUpToCost = GetPrice((double)tripSaving.TotalTripSaving),
    ReferenceCode = trip.ReferenceCode,
    TripDescription = trip.Name,
    EmployeeNumber = profile.EmployeeNumber,
    IsMultiDestination = passenger.HasFlightReservations && flightReservations.Any(f => f.JourneyTypeID ==
    User = new TripUser()
    {
        Name = string.Format("{0} {1}", profile.FirstName, profile.LastName).Trim(),
        ProfileImage = string.IsNullOrEmpty(profile.Picture) ? null : url + profile.Picture,
    },
    ApprovalRequestDate = dashboardTrip.ApprovalRequestDate
};
```

GenerateTripModel

```
var newTrip = new TripModel()
{
    TripID = passenger.TripID,
    TripPassengerID = passenger.ID,
    ApprovalProfileID = tripApproval.ApprovalProfileID,
    TripPassengerName = string.Format("{0} {1}", profile.FirstName, profile.LastName).Trim(),
    SkyD = passenger.SuperPNR,
    HasCarReservations = passenger.HasCarReservations,
    HasFlightReservations = passenger.HasFlightReservations,
    HasHotelReservations = passenger.HasHotelReservations,
    HasMiscReservations = passenger.HasMiscReservations,
    ReasonId = tripReason.ID,
    Reason = tripReason.Reason,
    CostCenterId = (costCenter != null) ? costCenter.ID : 0,
    CostCenter = (costCenter != null) ? costCenter.Name : string.Empty,
    GradeId = (grade != null) ? grade.ID : 0,
    Grade = (grade != null) ? grade.Name : string.Empty,
    DepartmentId = (department != null) ? department.ID : 0,
    Department = (department != null) ? department.Name : string.Empty,
    TripStatus = tripStatus,
    TotalCost = (tripSaving != null) ? GetPrice((double)tripSaving.TotalTripCost) : GetPrice(0),
    SaveUpToCost = (tripSaving != null) ? GetPrice((double)tripSaving.TotalTripSaving) : GetPrice(0),
    ReferenceCode = trip.ReferenceCode,
    TripDescription = trip.Name,
    EmployeeNumber = profile.EmployeeNumber,
    IsMultiDestination = passenger.HasFlightReservations && flightReservations.Any(f => f.JourneyTypeID ==
    User = new TripUser()
    {
        Name = string.Format("{0} {1}", profile.FirstName, profile.LastName).Trim(),
        ProfileImage = string.IsNullOrEmpty(profile.Picture) ? null : url + profile.Picture,
    },
};
```

- Use intention revealing **names**

```
int d; // elapsed time in days.
```

```
int elapsedTimeInDay;
```

- A **function** shouldn't have more than 3 arguments

```
0 references | 0 changes | 0 authors, 0 changes
public void SaveToFile(string fileName, string fistName, string lastName, string document, string phoneNumber) { }

0 references | 0 changes | 0 authors, 0 changes
public void SaveToFile(string fileName, User person) { }
```

- Correct naming can prevent **comments**

```
/// <summary>
/// Gets or sets the origin airport code.
/// </summary>
/// <value>
/// The origin airport code.
/// </value>
public string OriginAirportCode { get; set; }

/// <summary>
/// Gets or sets the destination airport code.
/// </summary>
/// <value>
/// The destination airport code.
/// </value>
public string DestinationAirportCode { get; set; }
```


A Lannister always...
Pays his technical debt

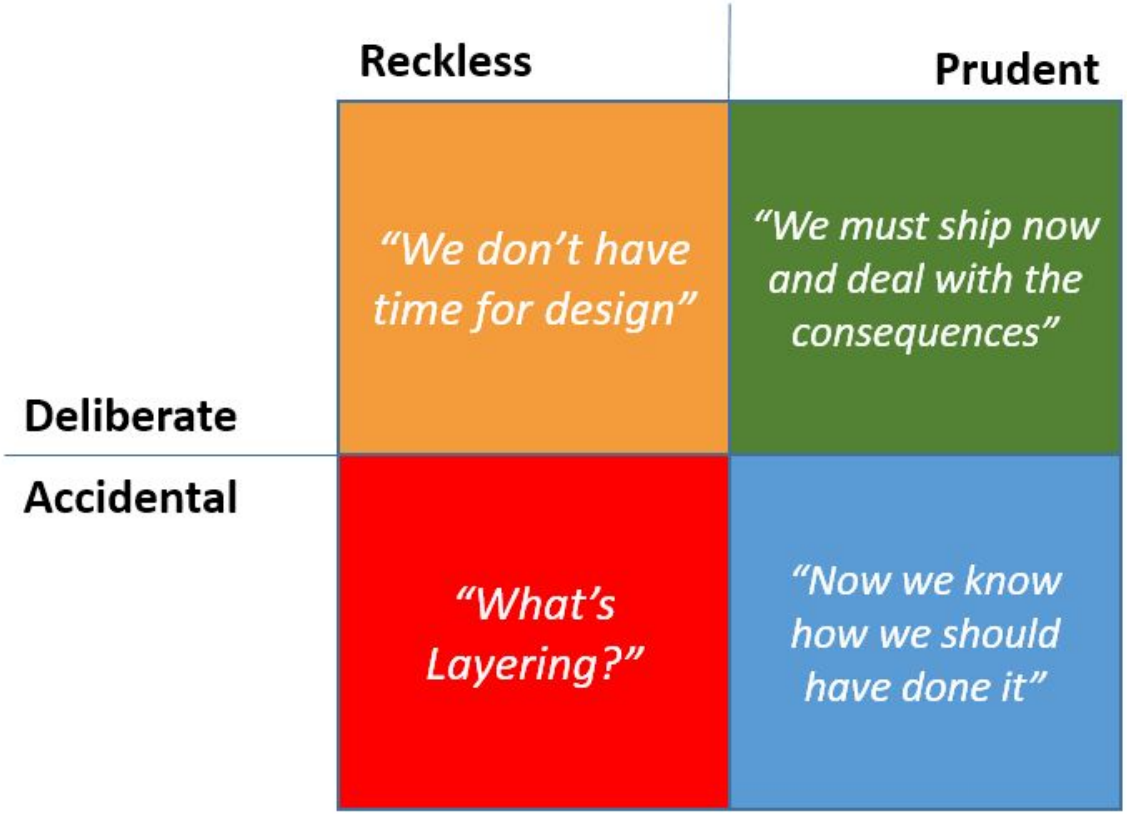
Technical Debt

Customer's view

Developer's view

“Always code as if the guy who ends up maintaining your code will be a violent psychopath who knows where you live”

Technical Debt Quadrants



Refactor

fear of Refactor

Code for humans

Apply Clean Code

Pay your debts

Pair Programming

Continuous Refactor

Code Review

UnitTest

Unit Test

Xamarin.Forms

Why MVVM?

MVVM benefits

The benefits of using the MVVM pattern are as follows:

- If there's an existing model implementation that encapsulates existing business logic, it can be difficult or risky to change it. In this scenario, the view model acts as an adapter for the model classes and enables you to avoid making any major changes to the model code.
- Developers can create unit tests for the view model and the model, without using the view. The unit tests for the view model can exercise exactly the same functionality as used by the view.
- The app UI can be redesigned without touching the code, provided that the view is implemented entirely in XAML. Therefore, a new version of the view should work with the existing view model.
- Designers and developers can work independently and concurrently on their components during the development process. Designers can focus on the view, while developers can work on the view model and model components.

Why UnitTest?

```
[Fact]
0 references
public void NotificationFormattedDate_MinutesAgo_ShouldBeSuccess()
{
    var notificationTrip = new NotificationTrip()
    {
        TripId = 12,
        Id = 7,
        SkyD = "QXDK5URH",
        PassengerId = 12,
        NotificationType = 0,
        NotificationDate = DateTime.Now.AddMinutes(-3),
        AgentId = 0,
        IsRead = false,
        IsSystem = false,
        TravellerName = "Alex Llanes",
        RequestingProfileTripUser = TripUserStub.GetUserWithImage()
    };

    notificationTrip.NotificationFormattedDate.Should().Contain("3 minutes ago");
}
```

```
public string NotificationFormattedDate
{
    get
    {
        if (NotificationDate.Day == DateTime.Now.Day)
        {
            if (DateTime.Now.Hour > NotificationDate.Hour)
            {
                return string.Format(AppResources.Notifications_HoursAgoLabel,
                    $"{DateTime.Now.Hour - NotificationDate.Hour}");
            }
            else if (DateTime.Now.Minute > NotificationDate.Minute)
            {
                return string.Format(AppResources.Notifications_MinutesAgoLabel,
                    $"{DateTime.Now.Minute - NotificationDate.Minute}");
            }
            else
            {
                return string.Format(AppResources.Notifications_SecondsAgoLabel,
                    $"{DateTime.Now.Second - NotificationDate.Second}");
            }
        }
        else if (NotificationDate.Day == DateTime.Now.AddDays(-1).Day)
        {
            return AppResources.Notifications_YesterdayLabel;
        }
        else
        {
            return DateTimeUtil.GetDateToddMMMMyyyyHHmmUTC(NotificationDate);
        }
    }
}
```

Why UnitTest?

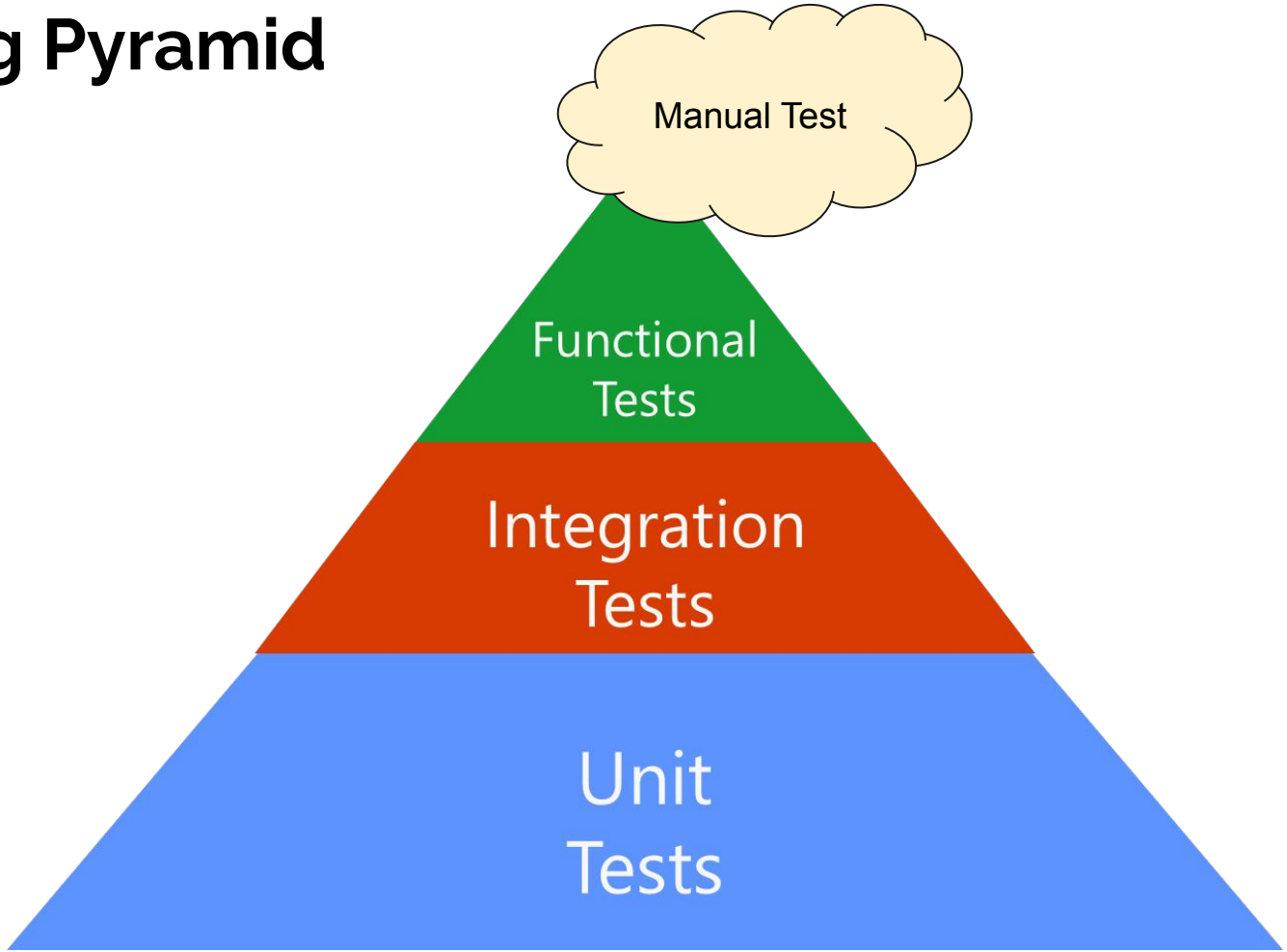
```
[Fact]
✓ | 0 references
public void NotificationFormattedDate_MinutesAgo_ShouldBeSuccess()
{
    var notificationTrip = new NotificationTrip()
    {
        TripId = 12,
        Id = 7,
        SkyD = "QXDK5URH",
        PassengerId = 12,
        NotificationType = 0,
        NotificationDate = DateTime.Now.AddMinutes(-3),
        AgentId = 0,
        IsRead = false,
        IsSystem = false,
        TravellerName = "Alex Llanes",
        RequestingProfileTripUser = TripUserStub.GetUserWithImage()
    };

    notificationTrip.NotificationFormattedDate.Should().Contain("3 minutes ago");
}
```

```
4 references | ✓ 4/4 passing
public string NotificationFormattedDate
{
    get
    {
        var dateTimeNow = DateTime.Now;
        TimeSpan timeSpan = dateTimeNow.Subtract(NotificationDate);

        string result;
        switch (timeSpan.Days)
        {
            case int days when days > 1:
                result = string.Format(AppResources.Notifications_DaysAgoLabel,
                    $"{timeSpan.Days}");
                break;

            case int days when days == 1:
                result = AppResources.Notifications_YesterdayLabel;
                break;

            case int days when days == 0:
                result = GetDateTimeDiffOnSameDay(timeSpan);
                break;

            default:
                result = DateTimeUtil.GetDateToddMMMMyyyyHHmmUTC(NotificationDate);
                break;
        }

        return result;
    }
}
```

Terminology

Little concepts to be on the same page

Testing Pyramid



Arrange
Act
Assert

```
[Test]
✅ | 0 references
public void EmailEmpty_ShouldBeInvalid()
{
    //Arrange
    _AuthenticationService
        .Setup(x => x.DoLogin(It.IsAny<string>(), It.IsAny<string>()))
        .ReturnsAsync(true);

    var loginViewModel = new LoginViewModel(_dialogService.Object,
        _navigationService.Object,
        _loggerService.Object,
        _AuthenticationService.Object);

    loginViewModel.Email.Value = string.Empty;
    loginViewModel.Password.Value = "Passw0rd";

    //Act
    loginViewModel.DoLoginCommand.Execute(null);


    //Assert
    loginViewModel.Email.IsValid.Should().BeFalse();
}
```

NUnit vs XUnit

	NUnit	xUnit.net
Supported Platforms	UWP, Desktop, Windows Phone, Xamarin Android, Xamarin iOS, ASP.net	UWP, Desktop, Windows Phone, Xamarin Android, Xamarin iOS, ASP.net
.Net Core Support	YES	YES
IDE Tools Support	Visual Studio, Visual Studio Code, Resharper	Visual Studio, Visual Studio Code, Resharper
CI Tools Support	TeamCity, VSTS, MSBuild, CuiseControl.net, Bamboo, Jenkins	TeamCity, VSTS, MSBuild, CuiseControl.net, Bamboo, Jenkins
Parallel Execution	YES	YES
Execution Isolation Level	Per test **class**	Per test **method**
Extensible Test Attributes	NO (**Test** and **TestCase** attributes are sealed)	YES (**Theory** and **Fact** attributes are extensible)

XUnit

```
public class CalculatorTests
{
    [Fact]
    public void CanAdd()
    {
        var calculator = new Calculator();

        int value1 = 1;
        int value2 = 2;

        var result = calculator.Add(value1, value2);

        Assert.Equal(3, result);
    }
}
```

```
[Theory]
[InlineData(1, 2, 3)]
[InlineData(-4, -6, -10)]
[InlineData(-2, 2, 0)]
[InlineData(int.MinValue, -1, int.MaxValue)]
public void CanAddTheory(int value1, int value2, int expected)
{
    var calculator = new Calculator();

    var result = calculator.Add(value1, value2);

    Assert.Equal(expected, result);
}
```

NUnit

```
[TestFixture]
0 references
public class LoginViewModelTest
{
    private readonly Mock<INavigationService> _navigation
    private readonly Mock<IDialogService> _dialogService
    private readonly Mock<ILoggerService> _loggerService
    private readonly Mock<IAuthenticationService> _Authen

    [SetUp]
    0 references
    public void Init()
    {
        Xamarin.Forms.Mocks.MockForms.Init();
    }
```

```
[TestCase("alex.llanes@gmail.com")]
[TestCase("Hello.world@gmail.com")]
0 references
public void DoLogin_ShouldBeNavigate(string email)
{
    //Arrange
    _AuthenticationService
        .Setup(x => x.DoLogin(It.IsAny<string>(), It.IsAny<string>()))
        .ReturnsAsync(true);

    _navigationService
        .Setup(x => x.NavigateToAsync<MainViewModel>())
        .Verifiable();

    var loginViewModel = new LoginViewModel(_dialogService.Object,
        _navigationService.Object,
        _loggerService.Object,
        _AuthenticationService.Object);

    loginViewModel.Email.Value = email;
    loginViewModel.Password.Value = "Passw0rd";
```

- Mocks
- Stubs
- Fake

Setup

Change our project to use UnitTest

- .NetStandard

ViewModelBase

```
public abstract class ViewModelBase : INotifyPropertyChanged
{
    protected readonly IDialogService DialogService;
    protected readonly INavigationService NavigationService;

    public event PropertyChangedEventHandler PropertyChanged;

    2 references
    public bool Initialized { get; set; } = false;
    0 references
    public bool IsBusy { get; set; }

    0 references
    public ViewModelBase()
    {
        DialogService = Locator.Instance.Resolve<IDialogService>();
        NavigationService = Locator.Instance.Resolve<INavigationService>();
    }
```

```
public abstract class ViewModelBase : INotifyPropertyChanged
{
    protected readonly IDialogService DialogService;
    protected readonly INavigationService NavigationService;

    public event PropertyChangedEventHandler PropertyChanged;

    2 references
    public bool Initialized { get; set; } = false;
    0 references
    public bool IsBusy { get; set; }

    0 references
    public ViewModelBase(IDialogService dialogService,
        INavigationService navigationService)
    {
        DialogService = dialogService;
        NavigationService = navigationService;
    }
```

You MUST call Xamarin.Forms.Init()


```
[SetUp]
public void SetUp()
{
    Xamarin.Forms.Mocks.MockForms.Init();
}
```

Dependency Service

```
var fileName = "MyFileName";

var path = DependencyService.Get<IFileHelper>()
    .GetLocalFilePath(fileName);
```

```
LoadApplication(new App((builder) =>
{
    builder.RegisterType<FileHelper>().As<IFileHelper>().SingleInstance();
}));
```

```
public void RegisterDependencies(Action<ContainerBuilder> registerPlatformDepedencies = null)
{
    _containerBuilder = new ContainerBuilder();

    // Register ViewModels
    _containerBuilder.RegisterType<TodoViewModel>();
    _containerBuilder.RegisterType<MainViewModel>();
    _containerBuilder.RegisterType<MySummaryExpensesViewModel>();
    _containerBuilder.RegisterType<NewEntryViewModel>();

    // Register Services. For singleton pattern use .SingleInstance();
    _containerBuilder.RegisterType<NavigationService>().As<INavigationService>().SingleInstance();
    _containerBuilder.RegisterType<DialogService>().As<IDialogService>();
    _containerBuilder.RegisterType<RequestProvider>().As<IRequestProvider>();
    _containerBuilder.RegisterType<LoggerService>().As<ILoggerService>();

    if (registerPlatformDepedencies != null)
        registerPlatformDepedencies.Invoke(_containerBuilder);
```

Nuget Plugins

```
//Registrar en AutoFac - ViewModelLocator.cs
builder.RegisterInstance(Plugin.Media.CrossMedia.Current);

//Para usar inyectarlo directo en el constructor
public ProfileService(IMedia media){
}
```

Xamarin Essential

Xamarin.Essentials.Interfaces 1.3.0

Unofficial automatically-generated interfaces for the cross platform APIs exposed by Xamarin.Essentials.
Suitable for mocking and for applications using dependency injection.

```
using Xamarin.Essentials.Implementation;
using Xamarin.Essentials.Interfaces;

builder.Register<IAccelerometer, AccelerometerImplementation>();
builder.Register<IBattery, BatteryImplementation>();
```

```
var mockGeo = new Mock<IGeocoding>();
mockGeo.Setup(x => x.GetPlacemarksAsync(It.IsAny<double>(), It.IsAny<double>()))
       .ReturnsAsync(Enumerable.Empty<Placemark>());
// etc.
```

Tips

One Bug == One Test

Try to test one thing with each unit test

```
    ReservationAdditionalDetailNavigationData navigationData = new Reserva
    {
        Reservation = flightReservation,
        IsApprovalFlow = true,
        Passenger = TripUserStub.GetUser()
    };

    await reservationAdditionalDetailViewModel.InitializeAsync(navigationD
    priceAnalysisModel.Cards = priceAnalysisCardList;
    priceAnalysisModel.ProgressBarItems = progressBarItemList;

    Assert.Equal(flightReservation.BudgetList != null && flightReservation
    Assert.Equal(flightReservation.GetDepartureDate, reservationAdditional
    Assert.Equal(flightReservation.Name, reservationAdditionalDetailViewMo
    Assert.Equal(navigationData.IsApprovalFlow, reservationAdditionalDetai
    Assert.Equal(Color.FromHex("#00BAF6"), reservationAdditionalDetailView
    Assert.Null(reservationAdditionalDetailViewModel.Traveller);
    Assert.True(ReservationStub.EqualFlightReservation(flightReservation,
    Assert.True(ApprovalMeterStub.EqualsApprovalMeter(flightReservation.Ap
    Assert.True(reservationAdditionalDetailViewModel.IsFlightReservation);
    Assert.False(reservationAdditionalDetailViewModel.IsHotelReservation);
    Assert.False(reservationAdditionalDetailViewModel.IsCarReservation);
    Assert.True(reservationAdditionalDetailViewModel.HasPriceAnalysis);
    Assert.Null(reservationAdditionalDetailViewModel.Traveller);
    Assert.False(reservationAdditionalDetailViewModel.MessageSelected);

    //Price analysis
    Assert.Equal(reservationAdditionalDetailViewModel.PriceAnalysisModel.T
    Assert.Equal(reservationAdditionalDetailViewModel.PriceAnalysisModel.R
    Assert.True(PriceAnalysisStub.EqualPriceAnalysisModel(reservationAddit
}
```

Moq

```
private readonly Mock<INavigationService> _navigationService = new Mock<INavigationService>();
private readonly Mock<IDialogService> _dialogService = new Mock<IDialogService>();
private readonly Mock<ILoggerService> _loggerService = new Mock<ILoggerService>();


//Arrange
_AuthenticationService
    .Setup(x => x.DoLogin(It.IsAny<string>(), It.IsAny<string>()))
    .ReturnsAsync(true);

//Assert
_navigationService.Verify(x => x.NavigateToAsync<MainViewModel>());

//Assert
_navigationService.VerifyNoOtherCalls();


_navigationService.Verify((s) => s.NavigateToAsync<TripConfirmViewModel>(It.IsAny<Trip>()));
```

Moq

Times

```
_navigationService.Verify(n => n.RemoveBackStackAsync(), Times.Exactly(2));

_dialogService.Verify(n => n.ShowAlertAsync(It.IsAny<string>(), It.IsAny<string>(),
    It.IsAny<string>()), Times.Once);

_navigationService.Verify(n => n.RemoveBackStackAsync(), Times.Never);
```

Navigation with properties

```
_navigationService.Verify(s => s.NavigateToPopupAsync<ResultPopUpViewModel>
    (It.Is<Core.Models.PopUpResult>(t => t.ImageSource == "WarningIcon"), It.IsAny<bool>()));
```

FluentAssertions

```
Customer actual = new Customer { Id = 1, Name = "John" };
Customer expected = new Customer { Id = 1, Name = "John" };

Assert.Equal(expected, actual); // The test will fail here
```

```
Customer customer1 = new Customer { Id = 1, Name = "John" };
Customer customer2 = new Customer { Id = 1, Name = "John" };

customer1.ShouldBeEquivalentTo(customer2);
```

Let’s code

Thank you!