

STOP

UNIT

TESTING

STOP

UNIT

TESTING*

* Sort of...

STOP

UNIT

TESTING*

* Sort of...



Alec Wojciechowski

Senior Consultant

ILM Professional Services

alec.wojciechowski@ilmservice.com

@wojonet (private)

Also on LinkedIn

Who/What is this for?

- Challenging developer dogma
- Prioritizing developer's (or QA's!) time
- Catching the 'real' bugs vs. those that we assume might occur
- Cutting out unnecessary complexity from our applications
- Helping us understand our own code better

Who/What is this not for?

- An excuse not to do testing at all
 - (Even unit testing!)
- TDD (maybe?)
- Crazy people who love to write tests

4 ways to abuse unit tests

- Test things that don't need testing
- Test assumptions, not actualities
- Use tests as a crutch (unit tests as an anti-pattern)
- Test things that are low risk (sort of a repeat?)

Unit Testing

Testing individual units of work. If our code is separated into atomic pieces that are sufficiently isolated from one another, we should be able to test the functionality of any single piece apart from any other, hence we get the term 'unit test'

Test things that don't need testing

Testing individual units of work. If our code is separated into atomic pieces that are sufficiently isolated from one another, we should be able to test the functionality of any single piece apart from any other, hence we get the term 'unit test'

Code

```
0 references | 0 changes | 0 authors, 0 changes  
public int Add(int a, int b)  
{  
    return a + b;  
}
```

Unit Tests

```
[TestCase(5, 2, 7)]  
[TestCase(5, -2, 3)]  
[TestCase(-5, -2, -7)]  
[TestCase(0, 0, 0)]  
0 references | 0 changes | 0 authors, 0 changes  
public void TestAddition(int num1, int num2, int expected)  
{  
    Program p = new Program();  
    int actual = p.Add(num1, num2);  
  
    Assert.AreEqual(expected, actual);  
}
```


YAAAaaaaaaayyyy.....?????

NAILED IT. You've successfully unit tested your addition function, but there's an empty feeling inside like you basically accomplished nothing. Life seems meaningless.

**IT'S NOT YOU -
IT'S YOUR UNIT TEST**

Another (Less Trivial) Example

```
public class MockPersonRepository : IPersonRepository
{
    public static List<Person> _thePeople = new List<Person>()
    {
        new Person() { Id = 1, FirstName = "Jerry", LastName = "Carpetsen", Address = "123 Fake St.", Age = 33, Income = 3200M },
        new Person() { Id = 2, FirstName = "Frank", LastName = "Donohue", Address = "8293 Frankfurter Dr.", Age = 75, Income = 300200M },
        new Person() { Id = 3, FirstName = "Flip", LastName = "Dudface", Address = "891 Crank St.", Age = 22, Income = 200M },
        new Person() { Id = 4, FirstName = "Tim", LastName = "Blankenship", Address = "1800 Penn Ave.", Age = 54, Income = 75000M },
        new Person() { Id = 5, FirstName = "Zip", LastName = "Valentine", Address = "918 Park St.", Age = 11, Income = 1200M }
    };

    0 references | alec, 10 hours ago | 1 author, 1 change
    public void AddPerson(Person person)
    {
        _thePeople.Add(person);
    }

    0 references | alec, 10 hours ago | 1 author, 1 change
    public void DeletePerson(int id)
    {
        _thePeople.RemoveAll(x => x.Id == id);
    }

    0 references | alec, 10 hours ago | 1 author, 1 change
    public void EditPerson(int id, Person person)
    {
        DeletePerson(id);
        AddPerson(person);
    }

    0 references | alec, 10 hours ago | 1 author, 1 change
    public List<Person> GetPeople()
    {
        return _thePeople;
    }

    0 references | alec, 10 hours ago | 1 author, 1 change
    public Person GetPerson(int id)
    {
        return _thePeople.FirstOrDefault(x => x.Id == id);
    }
}
```

Another (Less Trivial) Example (Cont.)

```
public class PersonController : ApiController
{
    IPersonRepository _repo;

    0 references | wojonet, 9 hours ago | 1 author, 1 change
    public PersonController(IPersonRepository r)
    {
        _repo = r;
    }

    0 references | wojonet, 9 hours ago | 2 authors, 3 changes
    public HttpResponseMessage GetPeople()
    {
        try
        {
            string jsonResult = JsonConvert.SerializeObject(_repo.GetPeople());
            HttpResponseMessage response = Request.CreateResponse(HttpStatusCode.OK);
            response.Content = new StringContent(jsonResult, Encoding.UTF8, "application/json");
            return response;
        }
        catch
        {
            return Request.CreateResponse(HttpStatusCode.InternalServerError);
        }
    }

    0 references | wojonet, 9 hours ago | 1 author, 1 change
    public HttpResponseMessage GetPerson(int id)
    {
        try
        {
            Person thePerson = _repo.GetPerson(id);
            HttpResponseMessage response = Request.CreateResponse(HttpStatusCode.NotFound);
            if (thePerson != null)
            {
                response = Request.CreateResponse(HttpStatusCode.OK);
                string jsonResult = JsonConvert.SerializeObject(thePerson);
                response.Content = new StringContent(jsonResult, Encoding.UTF8, "application/json");
            }
            return response;
        }
        catch
        {
            return Request.CreateResponse(HttpStatusCode.InternalServerError);
        }
    }
}
```

```
0 references | wojonet, 9 hours ago | 1 author, 1 change
public HttpResponseMessage PostPerson(Person person)
{
    try
    {
        _repo.AddPerson(person);
        return Request.CreateResponse(HttpStatusCode.Created);
    }
    catch
    {
        return Request.CreateResponse(HttpStatusCode.InternalServerError);
    }
}
```

```
0 references | wojonet, 9 hours ago | 1 author, 1 change
public HttpResponseMessage PutPerson(int id, Person person)
{
    try
    {
        _repo.EditPerson(id, person);
        return Request.CreateResponse(HttpStatusCode.NoContent);
    }
    catch
    {
        return Request.CreateResponse(HttpStatusCode.InternalServerError);
    }
}
```

```
0 references | wojonet, 9 hours ago | 1 author, 1 change
public HttpResponseMessage DeletePerson(int id)
{
    try
    {
        _repo.DeletePerson(id);
        return Request.CreateResponse(HttpStatusCode.NoContent);
    }
    catch
    {
        return Request.CreateResponse(HttpStatusCode.InternalServerError);
    }
}
```

Another (Less Trivial) Example (Cont.)

```
[TestFixture]
0 references | wojonet, 9 hours ago | 1 author, 1 change
public class PersonTests
{
    private static IPersonRepository _mockRepo;

    [TestFixtureSetUp]
    0 references | 0 changes | 0 authors, 0 changes
    public void StartUp()
    {
        _mockRepo = new MockPersonRepository();
    }

    [TestCase(5)]
    0 references | 0 changes | 0 authors, 0 changes
    public void TestGetPeople(int expectedPeople)
    {
        var numberOfPeople = _mockRepo.GetPeople().Count();
        Assert.AreEqual(expectedPeople, numberOfPeople);
    }

    [TestCase(2, "Frank")]
    0 references | 0 changes | 0 authors, 0 changes
    public void TestGetPerson(int personId, string expectedFirstName)
    {
        var aPerson = _mockRepo.GetPerson(personId);
        Assert.AreEqual(aPerson.FirstName, expectedFirstName);
    }

    [TestCase]
    0 references | 0 changes | 0 authors, 0 changes
    public void TestAddPerson()
    {
        Person newPerson = new Person()
        {
            FirstName = "Bronie",
            LastName = "Johnson",
            Address = "298 Oak Lane",
            Age = 37,
            Id = 7,
            Income = 60192M
        };

        _mockRepo.AddPerson(newPerson);

        var verifiedPerson = _mockRepo.GetPerson(7);
        Assert.NotNull(verifiedPerson);
        Assert.AreEqual(verifiedPerson.Id, 7);
    }
}
```

```
[TestCase]
0 references | 0 changes | 0 authors, 0 changes
public void TestEditPerson()
{
    Person newPerson = new Person()
    {
        FirstName = "Blanco",
        LastName = "Johnson",
        Address = "298 Oak Lane",
        Age = 37,
        Id = 7,
        Income = 60192M
    };

    _mockRepo.EditPerson(7, newPerson);
    var verifiedPerson = _mockRepo.GetPerson(7);
    Assert.NotNull(verifiedPerson);
    Assert.AreEqual(verifiedPerson.FirstName, "Blanco");
}

[TestCase]
0 references | 0 changes | 0 authors, 0 changes
public void TestDeletePerson()
{
    _mockRepo.DeletePerson(2);
    var verifiedPerson = _mockRepo.GetPerson(2);
    Assert.IsNull(verifiedPerson);
}
}
```

Wait...what did we test?

- Does our ASP.NET Web API controller work?
- Does our fake repository work?
- Will our tests pass?
- Have we seen this before?

An even less trivial example

```
1725 // Sorts the elements in a section of two arrays based on the keys in the
1726 // first array. Elements in the keys array specify the sort keys for
1727 // corresponding elements in the items array. The sort compares the
1728 // keys to each other using the given IComparer interface. If
1729 // comparer is null, the elements are compared to each other using
1730 // the IComparable interface, which in that case must be implemented
1731 // by all elements of the given section of the keys array.
1732 //
1733 [System.Security.SecuritySafeCritical] // auto-generated
1734 [ReliabilityContract(Consistency.MayCorruptInstance, Cer.MayFail)]
1735 public static void Sort(Array keys, Array items, int index, int length, IComparer comparer) {
1736     if (keys==null)
1737         throw new ArgumentNullException("keys");
1738     if (keys.Rank != 1 || (items != null && items.Rank != 1))
1739         throw new RankException(Environment.GetResourceString("Rank_MultiDimNotSupported"));
1740     if (items != null && keys.GetLowerBound(0) != items.GetLowerBound(0))
1741         throw new ArgumentException(Environment.GetResourceString("Arg_LowerBoundsMustMatch"));
1742     if (index < keys.GetLowerBound(0) || length < 0)
1743         throw new ArgumentOutOfRangeException((length<0 ? "length" : "index"), Environment.GetResourceString("ArgumentOutOfRange_NeedNonNegNum"));
1744     if (keys.Length - (index - keys.GetLowerBound(0)) < length || (items != null && (index - items.GetLowerBound(0)) > items.Length - length))
1745         throw new ArgumentException(Environment.GetResourceString("Argument_InvalidOffLen"));
1746
1747     Contract.EndContractBlock();
1748
1749     if (length > 1) {
1750         if (comparer == Comparer.Default || comparer == null) {
1751             bool r = TryS2Sort(keys, items, index, index + length - 1);
1752             if (r)
1753                 return;
1754         }
1755
1756         Object[] objKeys = keys as Object[];
1757         Object[] objItems = null;
1758         if (objKeys != null)
1759             objItems = items as Object[];
1760         if (objKeys != null && (items==null || objItems != null)) {
1761             SorterObjectArray sorter = new SorterObjectArray(objKeys, objItems, comparer);
1762             sorter.Sort(index, length);
1763         }
1764         else {
1765             SorterGenericArray sorter = new SorterGenericArray(keys, items, comparer);
1766             sorter.Sort(index, length);
1767         }
1768     }
1769 }
```

So....what's the point?

- Maybe we shouldn't write tests for things that are pretty straight-forward. This seems fairly obvious? What about code coverage?

“Measuring programming progress by lines of code is like measuring aircraft building progress by weight.”

- Bill Gates

“Measuring programming progress by lines of code is like measuring aircraft building progress by weight.”

- Bill Gates

“Measuring software quality by % code coverage is like measuring aircraft building progress by surfaces painted”

- Me

Test assumptions, not actualities

What assumptions about the input and output do we make as developers?

We're mind readers, right?

Has this happened to you?

Your vendor provides the following (No schema!):

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <course>
3   <students>
4     <student>
5       <firstname>George</firstname>
6       <lastname>Franklin</lastname>
7       <age>6</age>
8       <gradelevel>1</gradelevel>
9     </student>
10    <student>
11      <firstname>Jimmy</firstname>
12      <lastname>Doolittle</lastname>
13      <age>10</age>
14      <gradelevel>5</gradelevel>
15    </student>
16    <student>
17      <firstname>Beth</firstname>
18      <lastname>Robinson</lastname>
19      <age>8</age>
20      <gradelevel>4</gradelevel>
21    </student>
22  </students>
23 </course>
```

What could possibly go wrong?

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <course>
3   <students>
4     <student>
5       <firstname>George</firstname>
6       <lastname>Franklin</lastname>
7       <age>6</age>
8       <gradelevel>1</gradelevel>
9     </student>
10    <student>
11      <firstname>Jimmy</firstname>
12      <lastname>Doolittle</lastname>
13      <age>10</age>
14      <gradelevel>5</gradelevel>
15    </student>
16    <student>
17      <firstname>Beth</firstname>
18      <lastname>Robinson</lastname>
19      <age>8</age>
20      <gradelevel>4</gradelevel>
21    </student>
22    <student>
23      <firstname>Suzanne</firstname>
24      <lastname>Quincy</lastname>
25      <age>6</age>
26      <gradelevel>K</gradelevel>
27    </student>
28  </students>
29 </course>
```

So....what's the point?

- Maybe unit tests aren't the answer in this case....
- Can we do better?

ENTER....Integration tests!

Wait, but...

Aren't integration tests "hard to do?"..

Unit tests can create an “anti-pattern”

AKA - “Uhh, I’m not sure if this code works, but the unit tests pass, so.....”

What does that code do?

- Turns out it's actually a game that generates output that is another progression of the game...
- What's the point....?

Test things that are low risk

As developers we sometimes forget that unit tests are about risk management, not about making the tiny little circles green.

Think aviation.

What is risk management?

Simply put, it's an equation:

RISK CHANCE * RISK IMPACT = TOTAL RISK

So when should when unit test?

Possible Criteria:

- Solution is complex
- We can't get any closer than our mocked up data
- This is mission critical stuff
- As a 'proof' to other developers?

Thank you!

Any questions....

Comments?