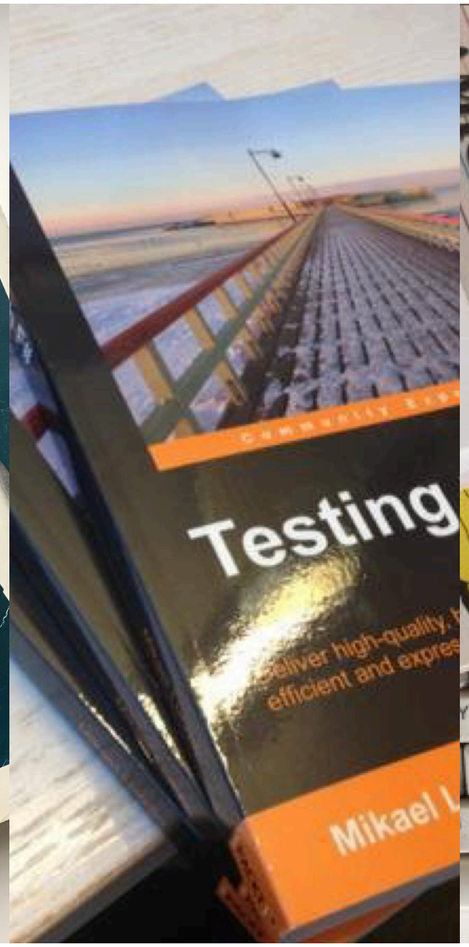
TDD

Workshop part 1

2017-03-30 | Valtech | Mikael Lundin | valte.ch/TDDWorkshop1

Who am I?


TURBO PASCAL
5.5/6.0
Stefan Njord
Lars Sandmark
Turbovägen
till Pascal
– programmering i Turbo Pascal 5.5/6.0
Studentlitteratur


Testing


AMIGA BASIC
basic
programmering
i

Schedule

Day 1

— 09-10 : Introduction to TDD

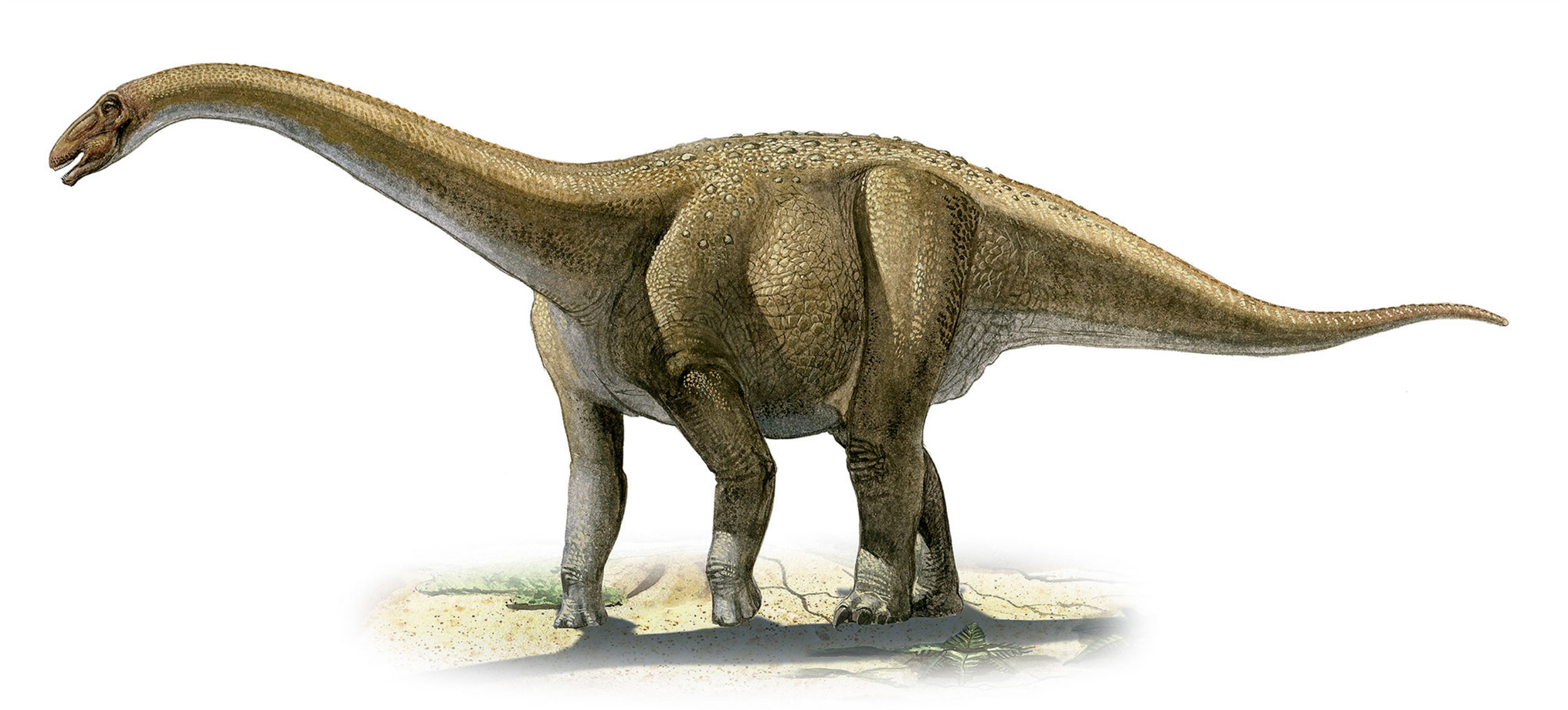
— 10-12 : Exercises

— 12-13 : Lunch

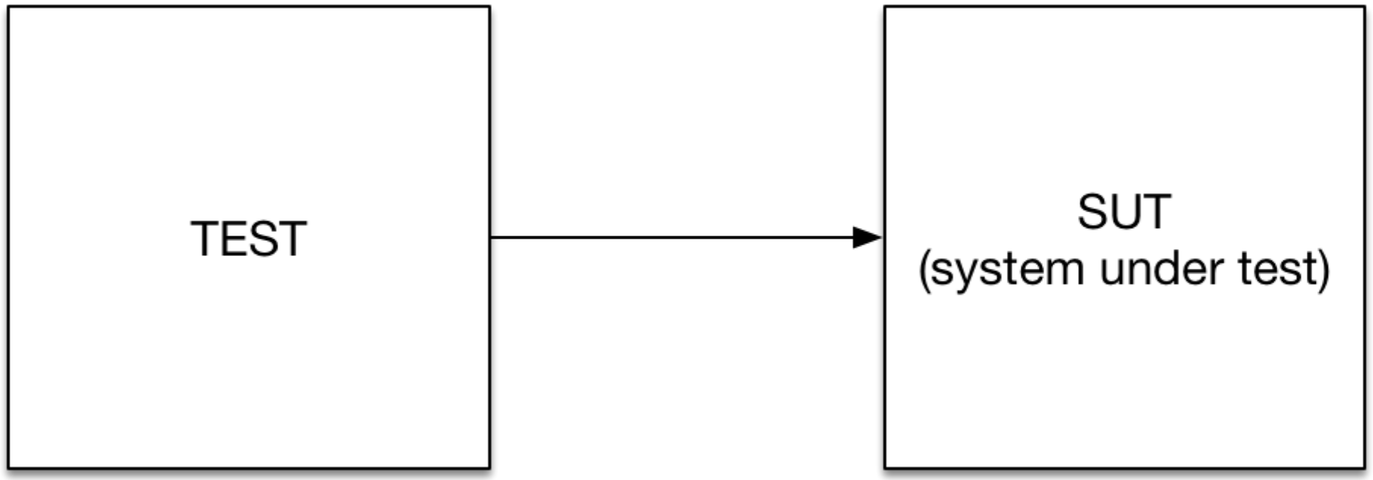
— 13-14 : Test-Driven Design

— 14-16 : Exercises

Thesaurus



Thesaurus



What is TDD?

In test-driven development, each new feature begins with writing a test.

— Wikipedia

bollocks!

What is TDD?

My definition

In TDD you use tests to specify, explore and make examples of the system under test.

— Mikael Lundin

TDD is not testing

Testing is the creative and exploring part of validating that a feature is solving a particular business problem.

— Something James Bach could've said

TDD is examples

```
// SUT
public int Add(int a, int b)
{
  return a + b;
}

// Test
public void Adding_one_and_two_returns_tree()
{
  Assert.AreEqual(3, Add(1, 2));
}
```

TDD is specifications

```
// SUT
public int Div(int a, int b)
{
  return a / b;
}

// Test
public void Arg_b_cannot_be_zero()
{
  Assert.Throws(() => Div(1, 0));
}
```

TDD is exploration

```
// SUT
public int Add(int a, int b)
{
  return a + b;
}

// Test
public void Exceeding_intMaxValue_will_cause_an_error()
{
  Assert.Throws(() => Add(int.MaxValue, 1));
}
```

Why *NOT* TDD?

A devils advocate approach.

1. It takes too long.
2. It creates a lot of waste.
3. Time spent on the wrong things.

Why TDD?

The possiblist approach.

1. Reduce feedback loop
2. Create predictability
3. Higher quality code and features

Always TDD!

The purist approach.

1. Always write your test first
2. Your code shall be 100% covered
3. Code without tests is legacy

Unit testing

— Identify the Unit

— Test in isolation

— Assert one thing

Identify the Unit

```
public class MovieRepository : IMovieRepository
{
  private int numberOfMovies;

  public int NumberOfMovies
  {
    get { return this.numberOfMovies; }
  }

  public Movie AddMovie(Movie movie)
  {
    // SUT code
  }
}
```

Test in isolation

The unit test is only allowed to operate in memory, *never do*

— Database calls

— HTTP calls

— Hard drive access

— COM interop

Assert only one thing

```
[Test]
public void AddMovie_Works()
{
  // arrange
  var movieRepository = new MovieRepository();

  // act
  var result = movieRepository.AddMovie(new Movie("Sgt. Kabukiman NYPD"));

  // assert
  Assert.AreEqual(1, movieRepository.NumberOfMovies);
  Assert.GreaterThan(0, result.ID);
  Assert.AreEqual("Sgt. Kabukiman NYPD", result.Title);
}
```

Unit Test: Example

```
[Test]
public void RomanNumeral_FromValue_should_return_5_when_value_is_V()
{
  // arrange
  var romanNumeral = new RomanNumeral();

  // act
  var result = romanNumeral.FromValue("V");

  // assert
  Assert.AreEqual(5, result);
}
```

Test naming

A good test name,
communicates the *unit of test*,
what result was expected
from the input that was given

Bad naming

```
RomanNumeral_test
UserRepository_GetUser_works
```

Test Frameworks

— NUnit

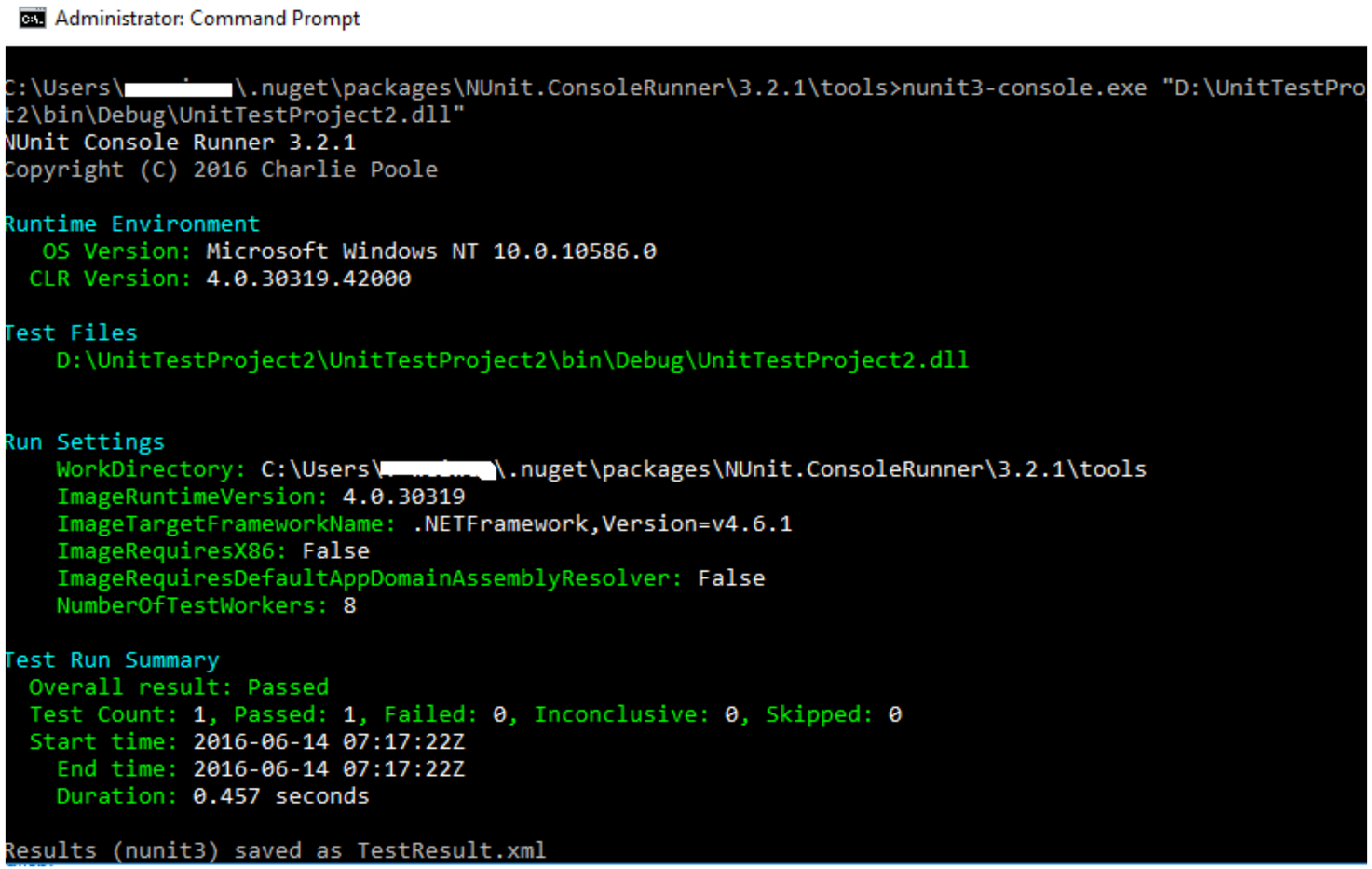
— xUnit

— MSUnit *(please don't)*

Test Runners

- Resharper
- NCrunch
- TestDriven.NET
- Visual Studio Test Explorer

Command Line



Build Server

#	Results	Artifacts	Changes	Started	Duration
#2120	Tests passed: 2	None	Maxim Podkolzine (1)	15 Feb 12 15:27	17m:26s
#2118	Tests failed: 2 (2 new), passed: 6360, ignored: 127	None	Changes (2)	15 Feb 12 15:19	2h:25m
#2116	Tests failed: 2 (2 new), passed: 7098, ignored: 35	None	Changes (2)	15 Feb 12 13:05	2h:33m
#2115	Tests passed: 7618, ignored: 35	None	Changes (7)	14 Feb 12 23:51	2h:49m
#2114	Tests passed: 2655, ignored: 24	None	Changes (4)	14 Feb 12 22:19	1h:31m
#2112	Tests failed: 2 (2 new), passed: 7098, ignored: 35	None	Changes (4)	14 Feb 12 21:19	2h:33m
#2110	Tests passed: 8181, ignored: 35	None	Changes (2)	14 Feb 12 19:30	2h:48m
#2109	Tests failed: 1 (1 new), passed: 8180, ignored: 35	None	Eugene Petrenko (1)	14 Feb 12 19:14	6h:51m
#2108	Tests passed: 7098, ignored: 35	None	Changes (3)	14 Feb 12 18:50	2h:28m
#2107	Tests passed: 7102, ignored: 35	None	Changes (4)	14 Feb 12 17:18	2h:43m
#2106	Tests failed: 1 (1 new), passed: 7096, ignored: 35	None	nikita.skvortsov (1)	14 Feb 12 16:37	2h:46m
#2105	Tests passed: 2594, ignored: 24	None	Changes (3)	14 Feb 12 15:16	1h:19m
#2102	Tests passed: 2594, ignored: 24	None	Changes (2)	14 Feb 12 13:45	1h:18m

Exercises!

Exercise 1:1

Implement test *as an example* that will verify that 11 becomes XI

```
[Test]
public void RomanNumeral_should_transform_11_to_numeral_XI()
{
    // arrange
    // act
    // assert
}
```

Exercise 1:2

Implement test *as a specification* on what to expect when input value is a negative number.

Exercise 1:3

Data driven testing could be of use here. Turn your test into a `[TestCase]` and run all test cases that you find are needed.

```
[TestCase(5, "V")]
[TestCase(11, "XI")]
public void RomanNumeral(int value, string expectedResult)
{
   //..
}
```

Exercise 1:4

Discuss: What tests are needed to completely cover the functionality?

— Tests as Examples

— Tests as Specifications

— Tests as Exploratory

Did you find the bugs?