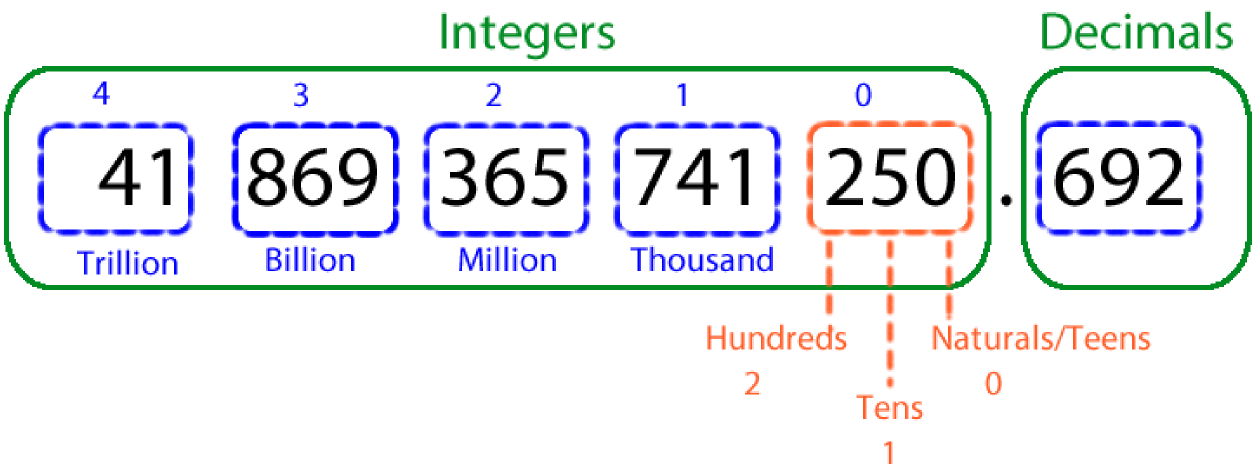
Development Approach

1. Logic

The main logic of the program is based on set of numbers and its positions.

1. Based on the dot, **two set of numbers** are created: Integers and decimals.
2. For the **integers**, sets of three numbers are created and based on their position the name of the scale is determined. The range of valid values is determined by the capacity of the Decimal[1] object in C#, which is a range of 0-28 digits.
   Min value: -9999999999999999999999999999
   Max value: 9999999999999999999999999999
3. For the **decimals**, taking into consideration that there is no money value after three decimals, the program accepts a maximum of two decimals.
   Min value: 0.0
   Max value: 0.99
4. For an individual **set of three numbers**, the value is determined based on the position of the number.
5. The word "and" has been added at the end of each scale (million, thousand, hundred, etc.)
6. The symbol "-" has been added to concatenate the composed tens.
   For example:
   23= twenty-three
   20= twenty



Logic Approach

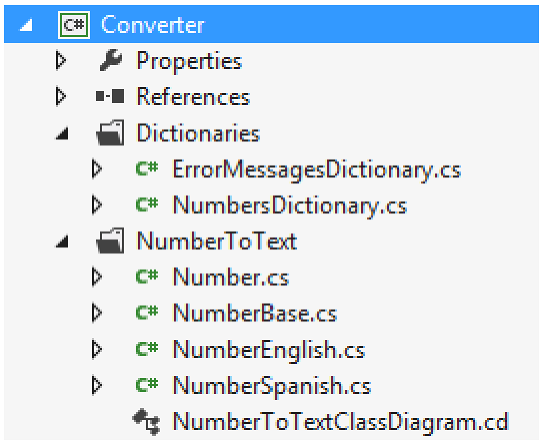
[1] http://msdn.microsoft.com/en-us/library/System.Decimal.aspx

2. Implementation approach

The source files for the program are on the "source code" folder. The projects are structured in the following way:

- Converter. Class library component with main logic to convert a decimal number to text.
- ConverterResources. Resource files for the numbers text.
- ConverterUI. Desktop application to test dynamically the converter program.
- ConverterUnitTesting. Unit Testing methods and data for the converter program.

Converter.dll


```
◢ Converter
   ▷ Properties
   ▷ References
   ◢ Dictionaries
      ▷ ErrorMessagesDictionary.cs
      ▷ NumbersDictionary.cs
   ◢ NumberToText
      ▷ Number.cs
      ▷ NumberBase.cs
      ▷ NumberEnglish.cs
      ▷ NumberSpanish.cs
        NumberToTextClassDiagram.cd
```

Class Library.
The converter functionality was built in a class library component so it could be used in a web page, web service, desktop, console application, or any other component by just making a reference.

The name of the library is "Converter" so in the future more classes related to conversions could be added; for example: conversions from imperial to metric units or grade Centigrade to Fahrenheit.

Grouping the classes into folders ("NumberToText") allows expanding the functionality to other conversions such as "Units" or "Temperature".

Factory Pattern

A factory was implemented in order to create a variety of types based on a fixed abstract definition.

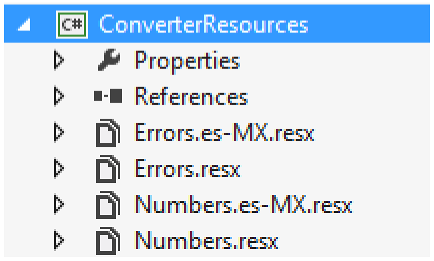
The objective is to set the framework to create numbers into text in different languages. The "NumberBase" class has the whole implementation to convert a decimal number into text; however, probably just some methods might need to be override in order to return the correct text without re-implementing the logic.

Although different languages act differently, for most of them the logic is the same one.

Static method.
Taking into consideration that the method to convert decimal numbers into text is stateless and we do not need to dispose any object, it is more convenient for the client class to call a static method. For example, the call would be: Number.ToText("152.36");

Decimal.
Working with decimals was chosen because this type is most commonly used for financial and monetary calculation which requires higher accuracy.
This type will determine the maximum value for the input number; which will be 28 significant digits.

ConverterResources.dll



Resource Files.

Nowadays, most of the applications need to be localized in order to expand de sales market to more clients over the world; therefore, it is important to consider the resource files in the architecture of the program.

The resource files are set by default in lower case letters; however they could be converted to upper case in the user interface.

Error Messages

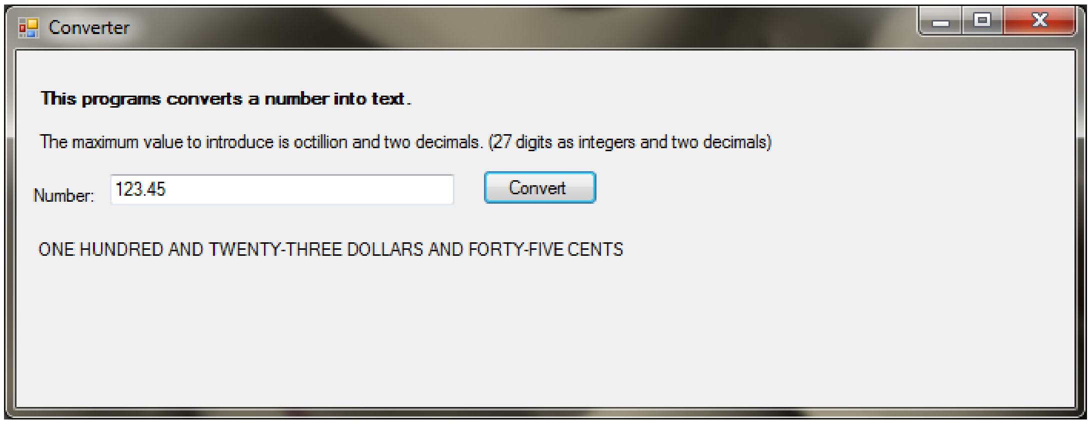
The exceptions are also enlisted in a separate class in order to provide a clear description of the error in the corresponding language.

ConverterUI Windows Application

The following desktop application is provided to test the cheque converter program in a dynamic way. In order to run the application download the folder "application" and follow these instructions:

1. Run the executable file **ConverterUI.exe**
2. Type a decimal number on the text field.
3. Click the "Convert" button.

Note: The ConverterUI.exe needs the following components to work: ConverterResources.dll, Converter.dll



This programs converts a number into text.

The maximum value to introduce is octillion and two decimals. (27 digits as integers and two decimals)

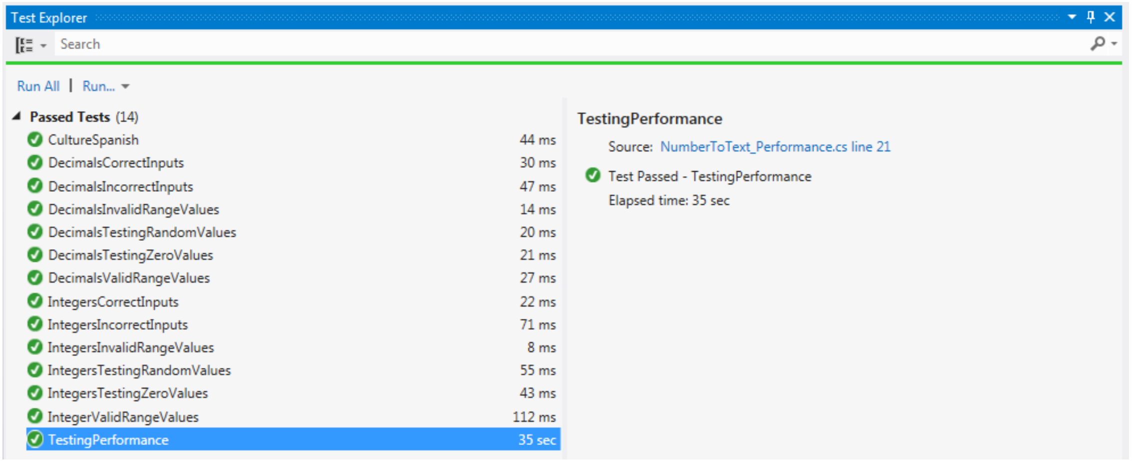
ONE HUNDRED AND TWENTY-THREE DOLLARS AND FORTY-FIVE CENTS

Converter desktop application

3. ConverterUnitTesting

The test data files that are used in the test methods could be found in the following folder “source code\ConverterUnitTesting\TestData\”, the methods cover the following scenarios (explained in detail on the Test Plan):

- Valid and invalid range values for integers and decimals.
- Valid and invalid input values for integers and decimals.
- Valid values for integers and decimals.
- Input values in other culture configuration (Spanish).



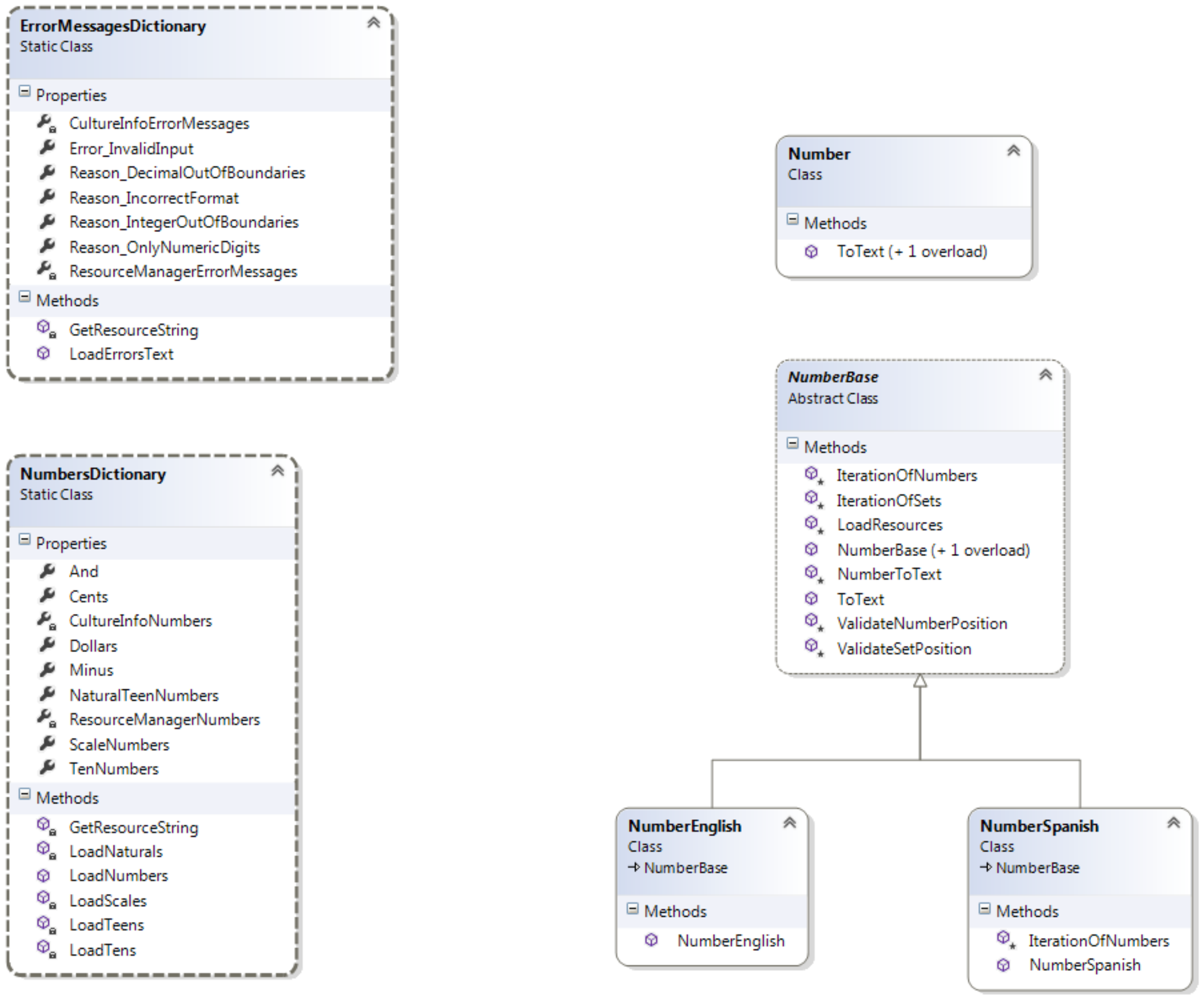
```
Passed Tests (14)
CultureSpanish                      44 ms
DecimalsCorrectInputs               30 ms
DecimalsIncorrectInputs             47 ms
DecimalsInvalidRangeValues          14 ms
DecimalsTestingRandomValues         20 ms
DecimalsTestingZeroValues           21 ms
DecimalsValidRangeValues            27 ms
IntegersCorrectInputs               22 ms
IntegersIncorrectInputs             71 ms
IntegersInvalidRangeValues           8 ms
IntegersTestingRandomValues         55 ms
IntegersTestingZeroValues           43 ms
IntegerValidRangeValues            112 ms
TestingPerformance                  35 sec

TestingPerformance
  Source: NumberToText_Performance.cs line 21
  Test Passed - TestingPerformance
  Elapsed time: 35 sec
```

Unit testing methods

4. Assumptions

- The program just accepts two decimal positions, considering that the maxim value for cents is represented in two digits.
  0.1 valid number in terms of money, representing ten cents
  0.01 valid number in terms of money, representing one cents
  00.001 invalid
- In terms of integers, a valid value should be a positive or negative number with 28 digits as maximum.
- English will be the default language for the cheque converter.

4. Class Diagram



Converter Class Diagram