Requirement Driven Development and Test-Driven Development

Ken Pugh (ken@kenpugh.com)

Introduction

There is a flow from external requirements to internal tests. Requirement Driven Development (RDD) (aka Behavior Driven Development / Acceptance Test Driven Development / Specification by Example) focuses on the external behavior. Test Driven Development focuses on the internal behavior that creates the external behavior. There is an overlap between the two. TDD can go all the way up to external behavior. RDD can differentiate into more detailed behavior. This article takes the example from Kent Beck's *Test Driven Development: By Example* and shows how it emerges with RDD using Gherkin. RDD presents a form of application-level tests that Kent referenced in his book. It gives a different perspective on the problem and its solution. These tests focus on readable and structure-insensitive aspects of Kent's Test Desiderata (https://kentbeck.github.io/TestDesiderata/), but they also address some of the other aspects.

The requirement comes from Ward Cunningham's WyCash example. Specifically, there is a set of investments in different currencies and a total in US dollars is desired. There are many other aspects to this requirement, but I'll keep it in the same detail as the book.

The Requirement

The original requirement as stated in Gherkin looks like the following: (I show the same values as in Kent's book, so you can make a comparison.)

```
Scenario: Customer Story
Given holdings are:
| Instrument | Shares | Price   | Total     |
| IBM        | 1000   | 25 USD  | 25000 USD |
| Novartis   | 400    | 150 CHF | 60000 CHF |
When totaled
Then result is:
| Overall Total  |
| 65000 USD      |
```

In my ATDD/BDD workshops, I use a scenario like this as a starting point to understand the problem. A guideline I suggest is to trace how every value in the Then statement comes from the Given, the When, or a business rule. If you cannot identify where these values come from, then you are missing part of the requirement. In the above scenario, there is a conversion from CHF to USD. The data for that conversion is missing. A discussion between the Triad (Customer, Developer, Tester) could suggest that these values come from a currency exchange. So that item can be added to the Given.

```
Scenario: Customer Story with More Detail
Given holdings are:
| Instrument | Shares | Price   | Total     |
| IBM        | 1000   | 25 USD  | 25000 USD |
| Novartis   | 400    | 150 CHF | 60000 CHF |
And exchange rate is
| From  | To   | Rate  |
| CHF    | USD  | 1.5    |
Then totals in USD
| Instrument | Shares | Price   | Total     | Total in USD  |
| IBM        | 1000   | 25 USD  | 25000 USD | 25000 USD     |
| Novartis   | 400    | 150 CHF | 60000 CHF | 40000 USD     |
And result is
| Overall Total  |
| 65000 USD      |
```

Scenarios usually do not exist in isolation. In general, each item in the Given does not magically appear. They come from the Then part of some other scenarios. You may be aware of those scenarios, responsible for them, or have no knowledge of them. If it is the last, then you should do an exploration to discover those scenarios to understand what may already be implemented. Now this could lead to a discussion at the appropriate time as to what currency exchange to use, what time to obtain the exchange values, and so forth.

There also appear some ubiquitous domain terms – currency, that is part of an amount and instrument, which is something is held as an investment. You can write scenarios that documents these terms:

```
Scenario: Domain Term Money (used in Price, Totals, Overall Total)
* Money is Amount and Currency
| Amount | Currency  |
| 25000  | USD       |
| 60000  | CHF       |

Scenario: Domain Term Amount
* Amount part of Money
| Data    |
| 25000   |
| 60000   |

Scenario: Domain Term Currency
* Currencies
| CHF |
| USD |
| USD |

Scenario: Domain Term Instrument
* Instruments are held as investments
| IBM      |
| Novartis |
```

You can add to each of these terms additional examples that show acceptable data (e.g. EUR for Currency) and unacceptable data (e.g. ZZZ). Now you could add a few more overall scenarios to further understand the context. For example, you could have a set of investments that have more than one investment in a currency or an investment in two currencies. You might have one complicated scenario that represents a real set of investments. That works as an overall test of the requirement. Here's an expanded story that has some of those variations, as well as others:

```
Scenario: Customer Story Expanded
Given holdings are:
| Instrument  | Shares  | Price      | Total            |
| IBM         | 1000.1  | 25.20 USD  | 25,202.52 USD    |
| IBM         | 100     |  40 CHF    | 4000 CHF         |
| Novartis    | 400     | 150 CHF    | 60000 CHF        |
And exchange rate is
```

```
| From   | To   | Rate   |
| CHF    | USD  | 1.52   |
When totaled
Then result is:
| Overall Total   |
| 67307.78 USD    |
```

Scenario Breakdown

The overall scenario can get differentiated into smaller scenarios, each of which deals with a part of the behavior. The smaller scenarios are part of the external behavior. In these examples, the scenarios include the addition of two amounts and multiplication of a money by a quantity.

```
Scenario:   Business Rule Calculations with currencies
* additions are:
| Addend1      | Addend2      | Sum          | Notes            |
| 25000 USD    | 40000 USD    | 65000 USD    | Same currency    |
| 60000 CHF    | 37250 CHF    | 97250 CHF    | Same currency    |
* multiplications are:
| Quantity   | Money     | Result      |
| 1000       | 25 USD    | 25000 USD   |
| 400        | 150 CHF   | 60000 CHF   |
```

Also, there is a conversion between two currencies.

```
Scenario:  Business Rule Calculations with currencies and conversion
Given exchange rate is
| From   | To   | Rate   |
| CHF    | USD  | 1.5    |
* conversions are:
| From         | Result       |
| 60000 CHF    | 40000 USD    |
| 40000 USD    | 60000 CHF    |
* additions are
| Addend 1     | Addend 2     | Output   | Result       | Notes                      |
| 25000 USD    | 60000 CHF    | USD      | 65000 USD    |                            |
| 25000 USD    | 60000 CHF    | CHF      | 92500 CHF    | Will it always be USD?     |
```

The values in these scenarios came from the overall scenario. Obviously, these must pass so that the overall one will. (These represent behavior which “You Are Going to Need It”). Additional cases which cover the boundary conditions or other interesting cases can be added to these smaller scenarios.

At this level, you may discover details which need to be resolved. The overall scenario consisted of whole numbers for conversions. What if there were decimal numbers for amounts and conversion rates, as shown in the previous expanded scenario? Then you can have a discussion in your triad as to what types of roundoffs should apply and when they should be applied. Business rules such as this are an area where it can be very beneficial to work out the details before implementation.

You might use these scenarios as the tests of methods or convert them into the implementation language. Keeping them in Gherkin allows for non-programmers to read and understand the problem that is being solved. You can add a method to convert the format in the scenario to an implementation’s internal values and vice versa. The tests for this conversion method could look like:

```
Scenario:  External to internal
* display to internal
| Display     | Amount | Currency   | Notes                           |
| 25000 USD  | 25000  | USD       |                                  |
| 60000 CHF  | 60000  | CHF       |                                  |
| 25000 XXX  | 0      | INV       | invalid currency                 |
| 25000      | 25000  | USD       | USD assumed if missing currency? |

Scenario:  Internal to external
* internal to display
| Amount | Currency  | Display   | Notes                |
| 25000  | USD       | 25000 USD |                      |
| 60000  | CHF       | 60000 CHF |                      |
| 0      | INV       | 25000 INV | Invalid currency     |
| 25000  | USD       | 25000 USD |                      |
```

The invalid tests will help to detect errors in creating larger scale scenarios.

Passing the Tests

At this point, the Triad has a shared understanding of the behavior and its differentiated behaviors. It's now time to implement it.   At the beginning, each of the scenarios will show up as red, since there is no implementation. You can tag those scenarios with @UnderDevelopment, so they aren't executed during the build.   As you make a scenario pass, you can remove the tag. When all the scenarios pass, you've implemented the requirement correctly (as defined by the tests) and that part of the doneness criteria is met.

You Try It

Here's a slightly more complicated story along the same lines of the previous one. In it, you need to compute the current investments based on a series of transactions and profits / losses that has been realized for those transactions. You'll need to compute those final values.   See If you can break it down into smaller scenarios.

```
Scenario: Customer Story
Given initial investments are nothing
And transactions have been
| Instrument   | Shares  | Price       | Transaction  | Date       |
| IBM          | 1000    | 25 USD      | Buy          | 1/1/2021   |
| IBM          | 100     | 40 CHF      | Buy          | 2/2/2021   |
| IBM          | 50      | 30 USD      | Sell         | 3/1/2021   |
| IBM          | 150     | 35 CHF      | Sell         | 4/1/2021   |
And exchange rate are
| From   | To    | Rate   | Date       |
| CHF    | USD   | 1.5    | 1/1/2021   |
| CHF    | USD   | 1.4    | 2/2/2021   |
| CHF    | USD   | 1.3    | 3/3/2021   |
| CHF    | USD   | 1.6    | 4/1/2021   |
| CHF    | USD   | 1.2    | 5/1/2021   |
When totaled on
| Date       |
| 5/1/2021   |
Then result is:
| Overall Total Investments   | Total Profit    | Profit Due to Currency   |
| ??      USD                 | ?? USD          | ?? USD                   |
```

Summary

When requirements involve lots of business rules, such as financial applications, you could try writing scenarios in a customer readable format (e.g. Gherkin) to create a shared understanding. You can decompose these scenarios into smaller ones that represent more detailed portions of the requirements.   The smaller scenarios can form the acceptance tests for the implementation.