The Mars Rover Kata and BDD/ATDD

Ken Pugh ken@kenpugh.com https://kenpugh.com

The Mars Rover Kata is a common TDD kata. The application revolves commanding a rover to move and then checking its position. This article gives a BDD/ATDD approach to the problem. This allows for separating understanding a problem from implementing the solution.

The Problem

The Mars Rover is given an area to rove in, an initial position, and a series of moves. [i] The rover outputs its new position after making the moves. The area looks like the following for one that is a size of 2 in both X and Y directions:

	North		
West	(0,1) (X, Y + 1)	(1,1) (X+1, Y+1)	East
	(0,0)	(1,0) (X+1, Y + 1)	
	South		

One Approach Using BDD/ATDD

There are many ways to approach this problem [ii] This version starts from an overall example and then works down to the details. The physical (e.g. textual) representation of commands and responses are decoupled from the logical representation of them. This permits reasoning about both representations separately.

The Beginning

Here's the overall example that is often used with this kata:

```
Scenario: Series of commands and output for one rover
When commands are
| 5 5         |
| 1 2 N       |
| LMLMLMLMM   |
Then output is
| 1 3 N |
```

The commands represent:

```
Scenario: Commands
* Commands are
| Command     | Meaning              | Notes                    |
| 5 5         | Set the size         | Gives size of map        |
| 1 2 N       | Set starting point   |                          |
| LMLMLMLMM   | Moves                | One or more characters   |
```

Set Size and Starting Point

Let's look at each of these commands in detail. The first command sets the size of the area, which is depicted in the problem section above:

```
Scenario: Command to Set the Size
* Convert command to sizes
| Command  | X Size  | Y Size  | Netes                    |
| 5 5      | 5       | 5       |                          |
```

The next command sets the starting point for the rover:

```
Scenario: Command to Set Starting Point
* Convert command to starting point
| Command  | X  | Y  | Orientation  | Notes                |
| 1 2 N    | 1  | 2  | N            |                      |
```

These two commands create an area and starting point that looks like this:

X Y	0	1	2	3	4
4					
3					
2		Rover -North			
1					
0					

The orientation is a direction, which is defined by:

```
Scenario: Domain Term Direction
* Directions are
| Symbol  | Direction  |
| N       | North      |
| S       | South      |
| E       | East       |
| W       | West       |
```

The two commands could be shown as a scenario with a map of the area. If there were multiple rovers or objects in a situation, the map could be a useful representation. Instead of N for North, an arrow might be used to show the orientation. Note that this scenario is automatable.

```
Scenario: Show the map
Given size is
| X Size  | Y Size  |
| 5       | 5       |
And initial position is
| X  | Y  | Orientation  |
| 1  | 2  | N            |
Then map is now
| X Y  | 0  | 1  | 2  | 3  | 4  |
| 4    |    |    |    |    |    |
| 3    |    |    |    |    |    |
| 2    |    | N  |    |    |    |
| 1    |    |    |    |    |    |
| 0    |    |    |    |    |    |
```

Move The Rover

The third command is to move the rover. This command is a string of characters. It gets split into the separate characters.

```
Scenario: Command to Move
* Convert Command to Moves
| Command     | Moves                          |
| LMLMLMLMM   | L, M, L, M, L, M, L, M, M      |
```

The meaning of each character is:

```
Scenario: Domain Term Move
* Moves are
| Symbol  | Move           |
| M       | Move forward   |
| R       | Turn Right 90  |
| L       | Turn left 90   |
```

Now here's what happens when a move is made.

```
Scenario: Make a single move
Given size is
| X Size  | Y Size  |
| 5       | 5       |
And initial position is
| X  | Y  | Orientation  |
| 1  | 2  | N            |
When move is
| L |
Then new position is
| X  | Y  | Orientation  |
| 1  | 2  | W            |
```

We could have a scenario like the above for each type of move. Alternatively, we could have a tabular format for all the moves. The first three columns give a current position and orientation, and the last three columns are the position and orientation after the move. This does not show a sequence of moves, just how each single move changes things.

```
Scenario: Business Rule Make a move
* Make a move
| X  | Y  | Orientation  | Move  | New X  | New Y  | New Orientation  |
| 2  | 2  | N            | M     | 2      | 3      | N                |
| 2  | 2  | W            | M     | 1      | 2      | W                |
| 2  | 2  | S            | M     | 2      | 1      | S                |
| 2  | 2  | E            | M     | 2      | 3      | E                |
| 2  | 2  | E            | R     | 2      | 2      | S                |
| 2  | 2  | E            | L     | 2      | 2      | N                |
```

A move could also be shown with a map. This could be useful for situations where there were multiple rovers or objects or more complex movements.

```
Scenario: Make a move showing the maps
Then map is
| X Y  | 0  | 1  | 2  | 3  | 4  |
| 4    |    |    |    |    |    |
| 3    |    |    |    |    |    |
| 2    |    | N  |    |    |    |
| 1    |    |    |    |    |    |
| 0    |    |    |    |    |    |
When move is
| L |
Then map is now
| X Y  | 0  | 1  | 2  | 3  | 4  |
| 4    |    |    |    |    |    |
| 3    |    |    |    |    |    |
| 2    |    | W  |    |    |    |
| 1    |    |    |    |    |    |
| 0    |    |    |    |    |    |
```

Now here's what happens when the sequence of moves occurs as described in "Move the Rover" above

```
Scenario: Make a series of moves
Given size is
| X Size  | Y Size  |
| 5       | 5       |
And initial position is
| X  | Y  | Orientation  |
| 1  | 2  | N            |
When moves are
| L |
| M |
| L |
| M |
| L |
| M |
| L |
| M |
| M |
Then positions are
| X  | Y  | Orientation  |
| 1  | 2  | W            |
| 0  | 2  | W            |
| 0  | 2  | S            |
| 0  | 1  | S            |
| 0  | 1  | E            |
| 1  | 1  | E            |
| 1  | 1  | N            |
| 1  | 2  | N            |
| 1  | 3  | N            |
```

This could also be expressed as the following, so the moves and positions are lined up.

```
Scenario: Make a series of moves
```

```
Given size is
| X Size  | Y Size  |
| 5       | 5       |
And initial position is
| X  | Y  | Orientation  |
| 1  | 2  | N            |
* Moves to positions
| Move  | X  | Y  | Orientation  |
| L     | 1  | 2  | E            |
| M     | 0  | 2  | E            |
| L     | 0  | 2  | S            |
| M     | 0  | 1  | S            |
| L     | 0  | 1  | W            |
| M     | 1  | 1  | W            |
| L     | 1  | 1  | N            |
| M     | 1  | 2  | N            |
| M     | 1  | 3  | N            |
```

If you need a more graphic representation of these moves:

X Y	0	1	2	3	4
4					
3		(end)▲			
2	(2)◄(3)▼	(start)▲ (1)◄ (8)▲			
1	(4)▼(5)►	(6)►(7)▲			
0					

Rover Output

The final position needs to be output. The

```
Scenario: Business Rule Position to Text
* Position to Text
| X  | Y  | Orientation  | Text   |
| 1  | 3  | N            | 1 3 N  |
```

If you trace through all of the above scenarios, you'll find the individual pieces that create the solution to the overall scenario that was introduced at the beginning:

```
Scenario: Series of commands and output for one rover
When commands are
| 5 5         |
| 1 2 N       |
| LMLMLMLMM   |
Then output is
| 1 3 N |
```

Discussion

As you create these scenarios, questions may come up that need to be answer by the customer or subject matter expert. Here is an example of questions that might be generated:

- What happens at the edges of the area? Should the rover stop or wrap around or signal an error or some combination?
- What should happen with erroneous input? For example, if the starting position is outside the size?
- What happens if a command is missing (like setting the starting point)? Should there be a default?

Although you might begin implementation without the answers, the implementation will be incomplete until the answers are obtained.

The original example had two rovers receiving a set of commands. It looked like:

```
Scenario: Series of text commands and output for two rovers
When commands are
| 5 5         |
| 1 2 N       |
| LMLMLMLMM   |
| 3 3 E       |
| MMRMMRMRRM  |
Then output is
| 1 3 N   |
| 5 1 E   |
```

The second starting position and moves are for a second rover. What if there were three and you wanted to move the first one twice before moving the third one? This suggests that there might be some rover identifier on the commands.

The Map Revisited

There are variations of this kata which have alternatives for handling the edges of the map. For example, a rover that is moving off the map may be stopped at the edge or it may wraparound to the other edge. Here's a scenario for the latter:

```
Scenario: Make a move that wraps around to the other edge
Then map is
| X Y  | 0  | 1  | 2  | 3  | 4  |
| 4    |    |    |    |    |    |
| 3    |    |    |    |    |    |
| 2    | W  |    |    |    |    |
| 1    |    |    |    |    |    |
| 0    |    |    |    |    |    |
When move is
| M |
Then map is now
| X Y  | 0  | 1  | 2  | 3  | 4  |
| 4    |    |    |    |    |    |
| 3    |    |    |    |    |    |
| 2    |    |    |    |    | W  |
| 1    |    |    |    |    |    |
| 0    |    |    |    |    |    |
```

Summary

All the scenarios listed are automatable in the Gherkin interpreter for your implementation language (Cucumber, SpecFlow, etc.). You could convert them into language dependent XUnit tests. However, keeping them as readable Gherkin allows for easier communication between customers, subject matter experts, developers, and testers. You also decouple working out the logical flow of a solution from the detailed implementation.

Copyright

---

[i] For another description, see https://freesoft.dev/program/82453547.

[ii] Another approach to the kata is at:

https://blog.thesoftwarementor.com/learning-through-example-mars-rover-kata/