Tap

Development process

In this section, we will have a look into the development process used for the project during the semester. This will take into consideration the entire project, including MS01, MS02 and MS03, with different subsections per milestone.

MS01

The first milestone was developed under the Test Driven Development approach (TDD). Since this was the beginning of the project, every domain model concept was still under development, and the code base was non existent.

In order to ensure a good development and preventing the maximum amount of bugs since the first moment, we first wrote unit tests that would ensure the correctness of our domain model. Only after have the tests written, we started the development of the first come first served algorithm.

This approach helped us to prevent for some bug that might have shown up later in the project.

Writing the unit tests first definitely helped to clarify what we wanted from the code to do.

Having a reasonable amount of unit tests present in the project would also prevent collateral changes to damage the domain model, as a failure when running the unit tests would signal a defect in the latest changes.

MS02

The second milestone was dedicated to Property Based Testing (PBT).

In this milestone, the focus was to think of properties to include in our project and help to validate the code correctness.

Unit tests written during MS01 are a great way to have fast feedback about our code, but they don’t test every possible scenario. As a software developer, we try to include the best edge cases possible but since it’s a manual process, we might not even know or think about all the edge cases that we need to consider.

This is where Property Based Testing comes into place.

The unit tests from the previous iteration were a great case to cover domain concepts, but they lacked in the interaction between these concepts. The First Come First Served algorithm was barely tested via any kind of tests. The only tests we had were the control files that helped to ensure correctness however, those tests have fixed input. Property based testing are great because they really force difference kinds of input and will try to brute force fail the test.

A great example of it’s usage, in our case, was that it detected a defect related to Allen’s Algebra. Although we had covered some scenarios related to the overlapped time periods, we had not covered all the cases, even though we thought we had.

Because these tests are based in properties, they are harder to implement, and therefore should only be applied in parts of the code that are not easily written via a unit test.

To summarize, these tests are great to implements when we want to test to many moving parts at the same time.

As a side effect of development, when considering writing PBT for certain scenarios, we understood that these were easily written as a unit test, so those cases were also taken into consideration and included in the code test base for the project.

MS03