Shift-left met early Model Based Testing (eMBT)

Nowadays more and more test activities are being automated, especially when it comes to executing the test cases and checking the results in the system under test. But what about the development of the test cases, the test case design? Are the test cases still developed in a structured way with a certain test coverage based on a well-considered risk? And is the impact on the test cases clear after a change in the test basis? And what about the Shift-left testing approach if we are mainly focusing on automating the test execution and checks? I think, with Model Based Testing (MBT) we already can answer many of these questions. In this blog I would like to give my vision and introduce the testing approach early Model Based Testing (eMBT).

Model Based Testing

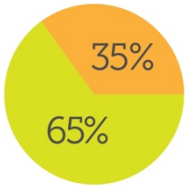
It is well known that Model Based Testing (MBT) has many advantages over other test approaches and that MBT offers a solution to automate the developing of test cases. Very important, because in the present there is no time anymore to manually set up the test cases and work them out using the traditional and formal test specification techniques. Despite the limited time, the goal remains unchanged; develop test cases with a certain test coverage in a structured way for the purpose of test execution.

If we, as testers, want to keep up with the ever-faster development cycles and still want to deliver the same test quality, we also need to automate the preparation of the test cases in the preparation phase. So, it is nice that MBT can help us to automatically generate the test cases from a model. And with any changes to the test basis, the maintenance is minimal. Only the model has to be updated and the test cases can be automatically generated again and again.

Shift-left test approach

Furthermore, MBT fits within the Shift-left test approach, which means that the test activities must start earlier in the Software Development Life Cycle (SDLC). By applying MBT you prepare a model in the test preparation phase and then derive the test cases from that model. With this you Shift all test activity (s) to the left. By start testing earlier, we ultimately ensure that we can provide feedback earlier and that we discover any bugs earlier in the process. Ultimately, any bug we find earlier is cheaper to fix, as also described in the well-known "Law of Boehm".

The question we must ask ourselves now is whether we will start our testing activities early enough, even if we use a Shift-left test approach and / or MBT. Especially when we realize that about 35% of all bugs in production can be traced back to the requirements. To avoid these bugs, we will therefore have to start testing the requirements (static test)! Ultimately, the requirements are also the basis for developing and testing the desired software. So, we need to make sure that this foundation is complete and clear and all stakeholders have the same understanding of it before we even start writing the first lines of code and testing it. Otherwise, we know for sure that the first bugs in the code will be introduced soon.



**Aditi Kulkarni, Global Assets Engineering Lead, Accenture – Software Intelligence Conference 2021. (according to their data based on 1000 projects)*

Early Model Based Testing (eMBT)

As described above, MBT fits perfectly within the Shift-left test approach. However, MBT is often started too late or used in a way without the approach of testing the requirements, but purely to generate automated (executable) test cases. We will therefore have to apply MBT in a specific way, at the earliest possible stage. I call this test approach early Model Based Testing, eMBT for short. By applying eMBT you will break down the requirements at the earliest possible stage by modeling them by means of a so-called eMBT-model. Such an eMBT-model has a high level of abstraction and by drawing it up you will quickly encounter ambiguities, contradictions, open ends, questions, etc. that you can then discuss with the stakeholders. When drawing up an eMBT-model, as a tester you will think differently about the requirements, and you will think more as the final customer. You are currently testing the requirements in an exploratory-like way and already give meaningful and early feedback, namely feedback on the basis. In addition, an eMBT-model is a user-friendly visual representation of the desired situation and is readable for all stakeholders. So not a technical and difficult to read model, as we often encounter within MBT. The user-friendly visual representation of an eMBT-model stimulates the communication and collaboration between all stakeholders with the aim of achieving a shared understanding of what needs to be built and therefore also tested.

Tooling

The eMBT approach does require a tool that supports this approach. For example, the tool must offer the possibility to draw a model with a high level of abstraction, the eMBT-model. This not only increases the readability of the model, but by drawing the eMBT-model you, as a tester, are also forced to think about both the happy and non-happy flow. Furthermore, the tool should offer the possibility to include the questions and comments you have, in the model itself.

Example

Below an example of a type of eMBT-model, based on the following requirements (figure 1).

Requirements calculator app
If you open the calculator app, the calculator standard shows 0.
You can make the following calculations:
1+1= with result 3
1+2= with result 3
1+4= with result 5
You can close the calculator app by clicking on the button 'close'

Figure 1

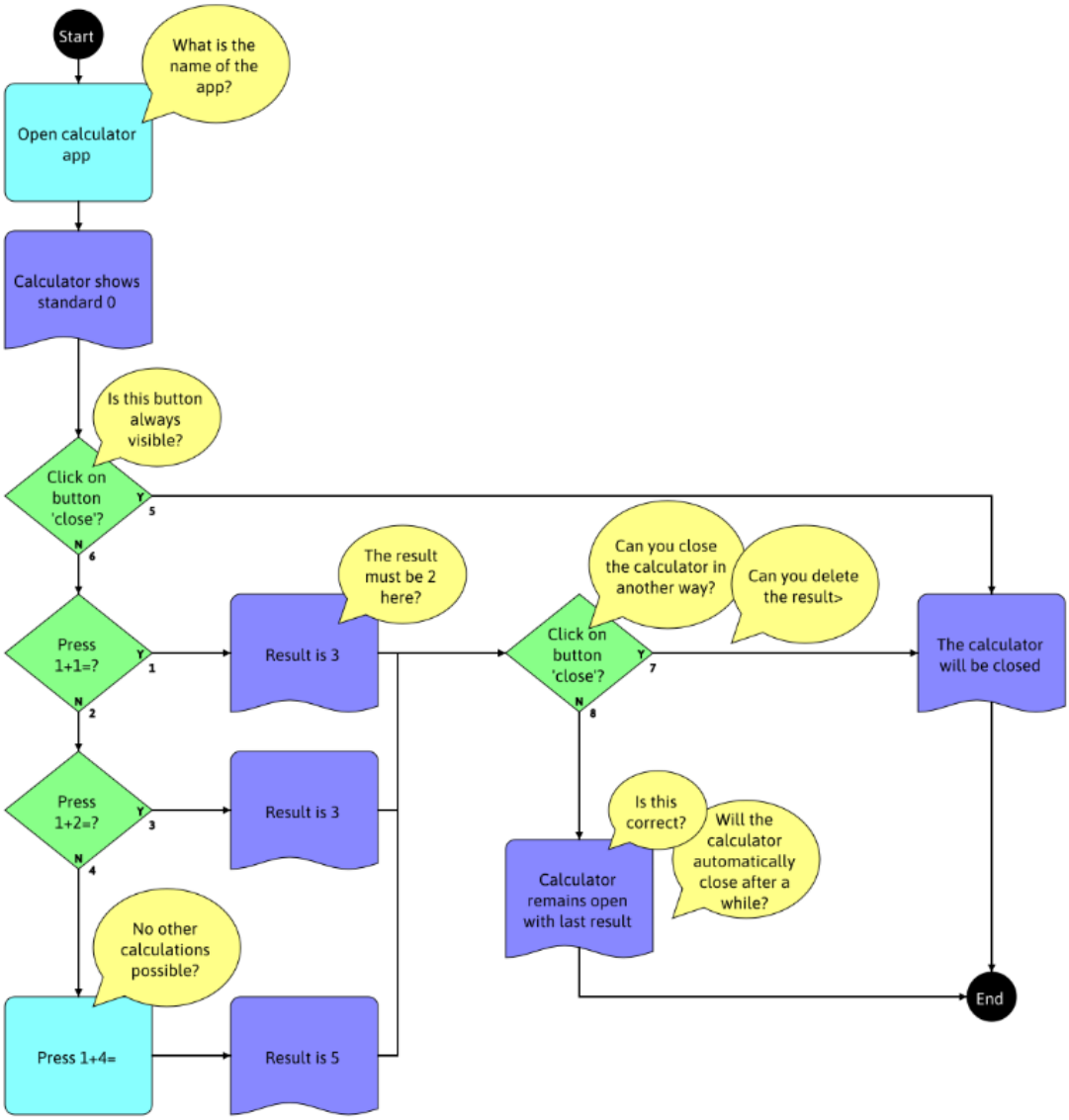


Figure 2

As you can see in figure 2, the eMBT-model is based on the principles of a flowchart, but with some specific conditions. This eMBT-model is not only easy to set up, but also clearly readable for everyone. Only three relevant nodes (action / status, decision and result) are used and furthermore, all questions / uncertainties etc. can be included directly in the eMBT-model via a separate node (balloon). As soon as all questions have been answered, all uncertainties have been removed and all stakeholders have the same understanding of what needs to be built, the eMBT-model is ready and the test cases can be generated automatically based on a pre-selected test coverage. These test cases then can be performed manually or being automated for the automated test.

To summarize

More and more automation is being done within the test process, especially when it comes to test execution and checks. But what about developing the test cases? Are we still doing this in a structured way and should we not automate this in order to keep up with the short development cycles? This is possible with Model Based Testing (MBT). MBT is now a well-known and proven test approach that offers many advantages over other test approaches. In addition, MBT fits within the Shift-left test approach because you can use MBT early in the process. However, we often see that technical models are used to derive the test cases and that the preparation of the model is started too late and therefore not used as a static test on the requirements. The technical model also does not stimulate the communication between the stakeholders, which is precisely so important at the start of the process.

early Model Based Testing (eMBT) can be the solution for this. eMBT is a software testing approach which starts at the very beginning of the SDLC with early feedback as an important goal. It supports a lot of important parts within the test process, such as collaboration, exploratory, study the requirements, determine risks, communication about the test basis and test object, determine test coverage, test case design and modeling. By using the right eMBT approach and tooling to start testing the requirements at the right time, unnecessary bugs are discovered at a very early stage. In addition, with this eMBT approach we quickly achieve a shared understanding of what needs to be built and tested, even before a line of code is written. If the requirements are clear and complete for all stakeholders, with one click you can automatically generate the test cases.

In figure 3 an addition to the well-known test pyramid with the eMBT approach, in which testing starts with the (automated) testing of the requirements, the bottom layer.

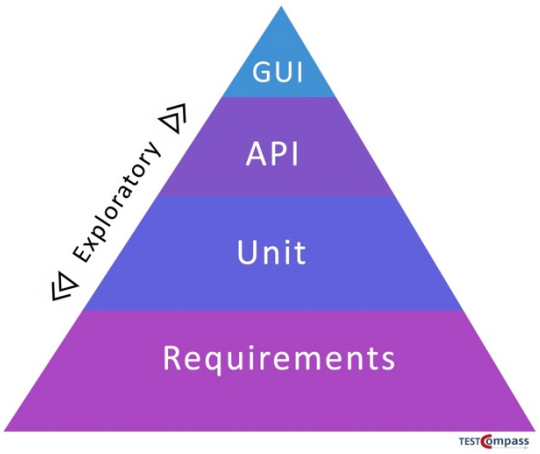


Figure 3

If you would like to know more about the approach of early Model Based Testing (eMBT) or about eMBT tooling, please let me know.

Silvio Cacace
info@compass-testservices.com