Acceptance tests (“ATs”) and why they are important

First of all, what exactly is an acceptance test?

An acceptance test ("**AT**" from now on) is, according to the ISTQB (why come up with my own definition when they have said it better), "*formal testing with respect to user needs, requirements, and business processes conducted to determine whether or not a system satisfies the acceptance criteria and to enable the user, customers or other authorized entity to determine whether or not to accept the system*".

So, basically, they are some steps that need to be followed, and an expected result that should be obtained, in order to verify some functionality is working properly.

Why is an AT needed when a bug is first reported?

When a bug is reported an AT should be provided, also specifying what's the actual result as opposed to the expected result (why the behavior is considered "buggy"). This helps in two main stages of a bug report:

- For Developers team to be able to reproduce and assess the bug,
- For QA team to make sure the fixes devs introduced are valid (i.e.: the bug is actually "fixed").

Can't devs and QE figure out ATs from some description, link or screenshots?

No. In fact, they could, but they shouldn't need to. In addition to spending extra time analyzing and finding the needed information, sometimes it's actually impossible to know for sure if the steps being performed are the exact steps that led to the bug in the first place. So in that case all that's left is to find an environment with a build of the application where the bug was present and test until the exact steps are found, or ask whoever reported it and beg them to make an effort to remember how it was reproduced. The reporter is always the best person to provide a detailed AT and it's best if they do it when they still have a good idea and context of the whole use case scenario.

How many ATs are needed in a bug report?

As long as they provide some information, all needed ATs should be added. Most of the times, reporters will only provide one AT. But sometimes the exact same issue is experienced with slightly different steps, in different use case scenarios, so more ATs might be added by the reporter or even by the developer (so the bug is well documented and the QE that is assigned to validate the fix knows what other cases to verify).

Testing is a task that may potentially never have an end, as there's always something else that can be tried, some small configuration changes that can be made, some edge case that could be hit by a user. But we also need to be practical and think of what leads to be bug being consistently reproduced, fixed and then validated.

Who can write ATs?

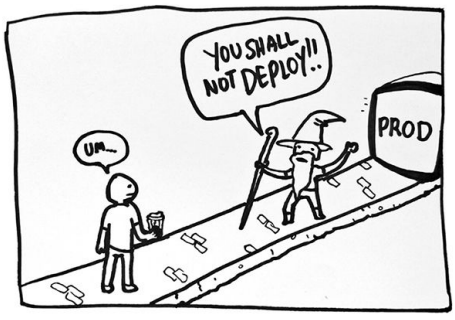
When validating a fix, QEs will perform all the ATs to verify the bug is not present anymore. They might also perform extra ATs they come up with, if they consider it necessary to cover all possible use cases and scenarios. But it's always good to have the developer's point of view as well.

- The first AT is always (or most of the times, unless it's a poor report) added by the bug **reporter**. This is what devs use to find the bug in the code and fix it. As explained above, a reporter might add more than one AT.

- When a **developer** fixes a bug, they can also add extra ATs (depending on the company procedures, this might even be a requirement). In this case, the goal is to uncover any regressions or other functionality that might have got "broken" because of the fix and will leave a well-documented bug report for future reference. The dev who fixed the bug is the person who knows better about what other areas could have been impacted by the changes they introduced in the code.

A developer AT is helpful to find out if any **changes introduced in the code fix** affected something else. *For instance, if a library that's being used was updated, they would probably know of other parts of the application that also make use of it and would be wise to test.* Sometimes it's not obvious. Sometimes the affected part is some obscure functionality that is triggered by a weird setting nobody heard of. Also, a developer would also know of any **settings** that could enable functionality that is not present by default and that makes use of the feature that was fixed, or something that is not the most common use case. *E.g.: the fix is about something related to some user profile field visibility, so it might be good to test if everything works as expected when anonymous users attempt to view a registered user's profile where some fields should not be visible to them.* Only devs know the real depths of the code they work on.

What is *not* an adequate AT?



An AT needs to have specific steps to reproduce the bug and then make sure the issue is no longer reproduced once it's fixed. An AT is **not** good if:

- Steps are the same as another AT already provided in the bug report, only with different wording (why duplicate stuff?)
- Information is missing. E.g.: configurations that need to be set up, or preconditions (things that need to happen before the actual steps are performed).
- Depends on links or information that needs to be gathered somewhere else. It's ok to provide a link for clarification, but an AT can't just say "read the discussion here" and

provide a link for a thread with 35 replies that has to be re-read and analyzed every time the AT has to be performed.

Why should I care enough to write proper ATs?

Let's say you are best friends with the developer that is assigned to fix the issue you're reporting, so you can discuss it over some coffee or beer (or a strawberry daikiri, which would definitely be my choice), so you don't feel the need to explain every little detail in the report. However, bug reports are normally used to generate release logs, or read by people who write the application documentation, or even used to add more test cases to a test suite. And sometimes (especially in big teams) the QE who validates the fix is not the same that first reported the bug. Or, with constantly changing teams, there might be new people assigned to work on a bug report and they might not know the application as well as old-timers. Not to mention customers that use the application and report bugs they find (it's always better to have found it ourselves, isn't it?). Last but not least, we never know when in the future the AT could be needed (maybe a similar bug was found, or the functionality is not well documented and your bug report helps find the correct settings for it to work).

After all, we should always give our best and aim for the best quality at all times :)

Disclaimer

I don't own any of the images here and they are for illustrative purposes only. If any of the copyright owners wish to request their removal, they should contact me through LinkedIn private messaging.