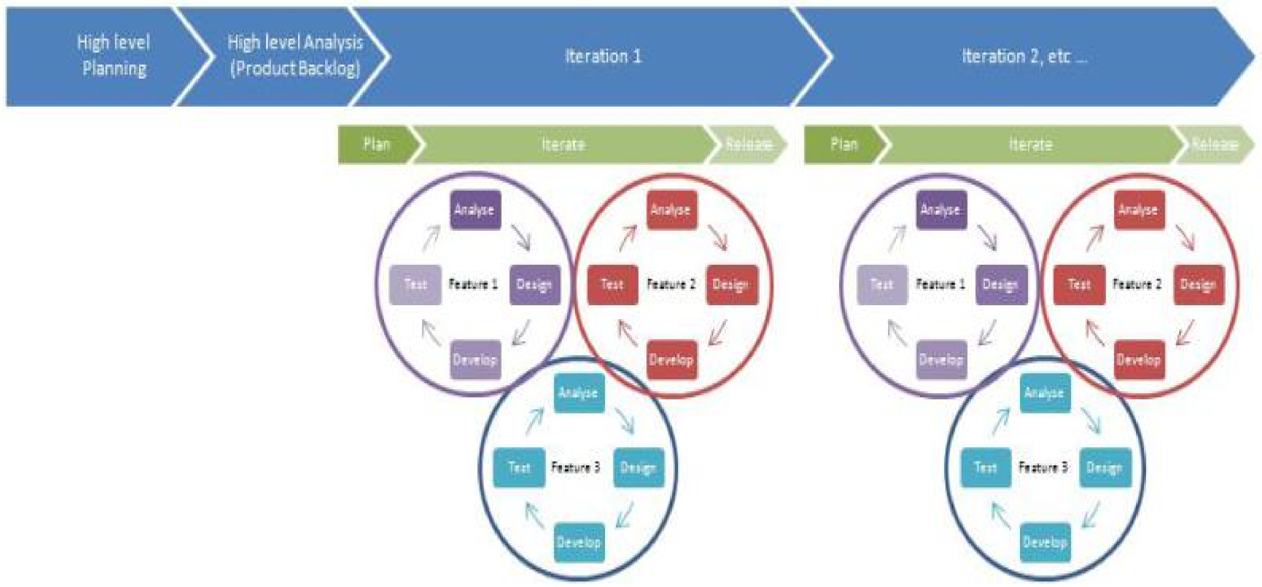
# TEST AUTOMATION IN AGILE AND WHY IT FAILS

It's fairly safe to say that quite a lot of test automation efforts fail. It is also very safe to say that without test automation an agile team fails. So how can you make sure that while doing agile your test automation will not fail and thus your agile team will not fail? One of the ways to answer this question is by looking at why test automation often fails within agile environments.

When I am talking about test automation within this post, I am referring to testing that is done to reduce the amount of manual regression work, the so called functional test automation or automatic regression testing.

## Moving target

Test automation quite often does not receive the attention it needs and deserves, also in agile teams. Quite some test automation efforts start off too late and without the appropriate preparation, resulting in organic test automation driven by a moving target. The moving target is the system under test which, in agile, is constantly in flux. Each sprint new features are added, bugs are fixed and quite often it is not clear at the start of a project where it is going to end up. Writing automated scripts against such a flexible environment which will stand the test of time, is difficult. It is even more difficult when the base on which automation is done is weak.



Quite often test automation runs behind on what is being delivered within an iteration, this is somewhat logical, considering that it is difficult to test, let alone automatically test what has not been built yet. Ideally while manually testing the new feature(s) as a tester, you're already pondering how to automate it so that you do not have to do the tedious work more than once. Given enough time within your iteration, you actually might be able to automate some of the features, from what I have seen thus far, generally not all features will be covered in test automation within one iteration. So if these tests are not all automated, what happens to them in the next iteration? Are they omitted? Are they picked up and automated retrospectively?

If you do not keep track of what has been automated during an iteration for both your current iteration and your previous iteration, how can you rely on your test automation? You can't be sure what exactly it is going through, so a bug can easily get through the net of your automated tests.

This moving target you are testing needs to be traced and tested solidly, repeatedly and in a trust-worthy way!

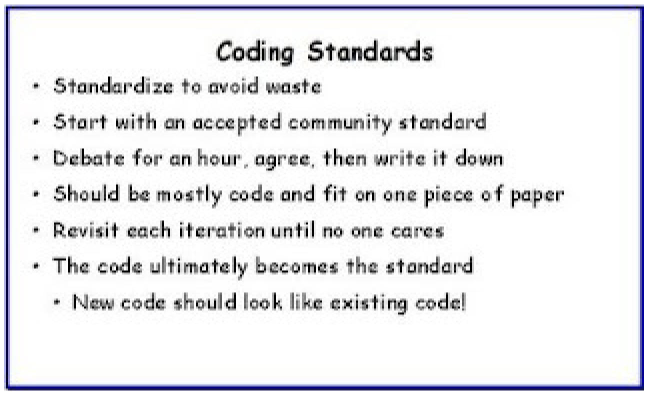
## Definition of Done

In the majority of the DoD's I have seen, one of the items is something referring to "tests automated". The thing I have thus far not seen however, is the team adding as much value to the automation code as they do to the production code. Quite a lot of DoD's refer to certain coding standards, however these standards often seem to not apply to functional test automation. Isn't your functional automation *code* also just *code*? If so, why then should this not be covered in code reviews, be written according to some useful guidelines and standards and hopefully use a framework to make the code sustainable?

## Test automation is just writing code

I have seen several automation efforts going on within agile teams where test automation was done without proper thinking having been put into it. A tool was chosen, based on what exactly other than members of the team having heard of it or having had good experiences with the tool. No base or framework to keep the code clean chosen. Since you are writing code, you should follow the same rules as the rest of the software developers. Don't think your code, since they are merely tests, should not be hooked up to some form of framework. If you want to make your tests survive a few iterations, considering reuse of your code would be logical.

By the way, *coding standards* do not need to be too complicated. In 2009 "Agile in a flash" came up with a coding standard that could work for all languages and for most environments:


**Coding Standards**

- Standardize to avoid waste
- Start with an accepted community standard
- Debate for an hour, agree, then write it down
- Should be mostly code and fit on one piece of paper
- Revisit each iteration until no one cares
- The code ultimately becomes the standard
  - New code should look like existing code!

All of the above mentioned points are “logical” when writing an application which is supposed to go into production. However when looking at a lot of (agile) projects, these logical “best practices” seem to be totally forgotten when it comes to test automation.

## Succeed in test automation

So, how do you succeed in your test automation? How do you make it work? The answer seems clear to me: test automation is not *like* writing code, it is **equal** to writing code. Since it is the same, treat it the same way!

Do your code reviews, follow a form of a standard, use a (simple) framework to make life easier in writing tests, create reusable modules in your automation code. In other words, treat your functional test automation with the same respect as your production grade code. Who knows, you might want to run your tests against your production environment some day! In setting up your initial test automation environment and framework, don’t be shy and ask the developers in your team for tips, tricks and suggestions. They quite likely have gone through those setup steps more often than you have, so use their knowledge. Asking them for their insights and ideas not only helps you, it also helps them feel more responsible for doing their 5 pennies worth on the test automation side. They will get a clearer idea of what you intend to achieve, so they might also be more willing to help out keeping their code testable, they might even enjoy helping you write the testscripts!

**Resources**

Some informational resources where you can find some ideas on how to setup the test automation framework:

- UI Test automation
- Page object model
- Automating a legacy system

Source : http://martijndevrieze.net/2012/06/18/test-automation-in-agile-and-why-it-fails/