Existing solution:

# Unified Automation Platform (UAP) and ALM:

UAP was Vistaprint’s solution for developing and executing automated testing. It is tightly coupled with the test management system ALM (Application lifecycle management).

The unified automation platform tool executes different test types, and then delegates the test result storage to the ALM tool. This latter feeds several deployed dashboards with the stored test results data. At this point there were no reporting problems. All the data could be visualized at any given time. However, there were a few problems that led the team to migrate towards a new combination.

One of the major problems with this solution is that the UAP/ALM combination is essentially focused on UI tests (System tests), whereas the majority of tests run by the team are Service level tests and Unit tests. Therefore these powerful tools were not really used to the fullest.

Another as important problem is that the ALM tool is a stand-alone application, meaning it is not integrated in the global automation process.

These problems highly demotivated many developers, and this solution was quickly abandoned. The newly adopted combination is Jenkins and Nunit.

# Jenkins Nunit plugin:

As we mentioned in the previous chapter, the newly adopted combination does not support Nunit test results reporting. Jenkins does not naturally support Nunit and thus test results end up overwritten. In order to overcome this problem, the different teams tried to find a quick workaround, which was no other than the Nunit plugin.

The Nunit plugin is a Jenkins plugin that allows the user to publish Nunit test results. In other words, it makes it possible to import Nunit reports from each build into Jenkins so that they are displayed within a trend graph (Internal Jenkins dashboard) and details about which tests have failed.

This adopted method also has two major drawbacks. One of which resides in the way this plugin operates. In fact, what this plugin really does is transforming the **Nunit** test result file into multiple **Junit** test result files so as to be supported by Jenkins, as mentioned above. Unfortunately, this transformation costs some data loss, which is very serious in terms of reporting.

The second drawback is simply the internal Jenkins reporting system. On one hand, Jenkins dashboard is known to be very basic and very inflexible. On the other hand, the gathered information is never stored in a database. The dashboard directly reads from the xml result files, which are poorly structured after the transformation. This hugely prevents the full exploration of the valuable data.

The migration towards Jenkins and Nunit is certainly a very efficient alternative in terms of test automation and easy software development. However, we can’t say as much regarding data reporting, which gave birth to a need of yet another reporting alternative that perfectly integrates in the current system.