**How to review codes**

Code review is one of the very important processes of software development, it involves analysing the source codes either manually or using automation tools to review codes with the aim of improving the quality of codes and check for errors and bugs.

* In carrying out an effective code review, it is essential to first set out the necessary goals and standards. Companies or organizations involved in software development have set goals and standards, by setting out these goals, it is possible meet sustainable coding standards and avoid unexplicit goals. setting out the goals will greatly improve the speed of the review.
* It is also vital for the reviewing team to have very effective means of communicating the set goals. When there is strong communication amongst team members and every member of the team knows what is expected from him/her, then there is guarantee of arriving at accepted results.
* Putting out a well outlined code review follows in the process, and it helps members focus on task. This can be enhanced by introducing a review checklist.
* Clear Illustrations (annotation with comment) should be employed in codes to aid the entire (SDLC).
* A well streamlined correctional strategy can be adopted. When defects are discovered, there should be a well organised way of making corrections.
* As humans we are susceptible to errors when the mind is weary. Hence it is important to use minimum time range when carrying out reviews. Prolonged review exercise can cause stress to team members which will in turn lead to even omitted errors.
* It is vital to automate the code review process with tools like Teamcity, PregoBot etc. these tools not only save time but greatly enhance the code review process.

**How to enforce coding standards**

Software development is not just all about writing codes, it also entails writing clear, readable appreciated codes, which is why it is important to adhere to accepted coding standards such as C & C++ standards.

Adopting these standards guarantees the codes do not change regardless the environment it is deployed, it can be sustained/maintained, it can be tested, it is highly impregnable or fortified, it can not cause harm and it is dependable.

Having a coding standard is one thing but adherence is a totally different thing.

To enforce standards, the static code analysis is used. Example Klocwork & Helix QAC

This can be achieved by creating a Helix QAC project.

Setting up the Klocwork Desktop also comes in handy in this regard. It has the environment in which developer can look at issues while coding.

**How do you plan the approach to test automation?**

In planning what steps to take in test automation, it is important to know that the main goal of automation is to reduce the tests done manually and not to stop manual testing all together. The first thing to consider the test cases involved.

At the planning stage the team members are available together with the product owner, scrum master or coach, depending on what model to be adopted AGILE or Waterfall.

Deciding what to automate and what not to. Such as test cases are time consuming, those that are to be executed repeatably, those that are a bit challenging etc.

Once the test cases are well defined the technology to be used follows i.e., tools to be used such as Selenium, robot framework, QTP. To know what tool to be used depends on the base or fundamental technology used in building the application. E.g., application built with AngularJs may be difficult to test with robot framework except with protractor.

In choosing the right tool for the right automation testing tasks it is important to have in mind how the tool can support multiple testing frameworks,

How adaptable is the tool when used for different types of tests such as functional testing, regression testing, mobile testing etc.

It is widely accepted that tool selection is one of the most difficult tasks in automation. It is very important to choosing the right tool.

Once the requirements are known, the different tools and their versatilities are known, targets or expectations for each tool are set, then it becomes easy.

Developing the plan and design then follows.

After this the steps are executed. The prepared scripts are executed at this time.

The scripts are constantly reviewed, and adjustments are made to give way to new functionalities where necessary since there are always development in software technology. This is where maintenance comes in for each release cycle.

Through maintenance the efficacy of automated scripts is sustained

**In summary**

* It is important to know the scope of automation thoroughly in details before delving into it.
* Picking the right automation tool is important.
* The right framework best fit for the task should be chosen.
* Scripting must be done with great adherence to universally accepted standards.
* It is important to be able to know just how much of help (productivity)each automation has been, the efficacy when compared to the deliverables e.g., how many bugs was discovered etc.
* Not to forget the team itself. Team building is also vital in automation.

**Code testability, how to enforce it.**

* It is better to keep the units, function, blog, class of codes as short as possible.
* Data modules should be separated, some functionalities that could connect to network should be separated from these used inside the application.
* Units of code should be executing a single or very few tasks. e.g., the Linux command line will do one thing and it will do it exactly right. Just few tasks
* For a code to be testable it must not violate the dependency inversion principles. “High level modules should not depend on low level modules” codes should depend on abstractions.
* Any confusing texts should be avoided so as not to cause more confusion.
* It is best to have a clear description of the problems to be solves.
* The rules to implement should be such that it applies to the project as a whole.