Tactics to support quality assurance:

Quality Assurance should be responsible for preventing defects, not merely for finding them. For this to happen, QA should be moved up to the front of the development cycle. Since we’ve chosen to use Github to keep our repository of code this could not be easier with the use of pull requests, versioning and branches! Another way to safeguard against application breaking bugs is by enforcing the ***efficient*** use of Unit, Integration and E2E(end to end) tests.

Pull Requests initiate discussion about your commits. Anyone can see exactly what changes would be merged if they accept your request. Each pull request should be reviewed by at least two other developers before the code can be successfully merged into the master branch. Once a Pull Request has been opened, the developer reviewing your changes may have questions or comments. Perhaps the coding style doesn't match the rest of the project, the change is missing unit or integration tests, or maybe everything looks great and kudos are in order. Pull Requests are designed to encourage and capture this type of conversation. Please go to <https://github.com/SAP/BUILD/pull/51> to see an example of a good productive pull request between an architect and three senior devs discussing an infrastructure issue.

In the case of some small refactoring – a quick read through of the code might be sufficient but otherwise it’s really easy to pull down the branch that the changes were made on and test it locally on your own machine without any need to bother the person who requested the PR.

With the use of versioning, it’s easy to revert to an older version of a module. Perhaps a “Very High” or “Epic” bug is introduced to the system. All of the commits can be easily reverted to the latest version of the application that worked. Of course with the correct use of Unit, Integration and E2E tests this should be unlikely to happen.

Automated testing is a huge part of quality assurance. And the most important thing to do when creating tests is to emulate an end user – so you might be tempted to say add more End to End test right? And to an extent you are correct but end to end tests are expensive. They take a lot of time and when they do fail they are not as efficient at pinpointing the source of the bugs as integration and unit tests. Just read this article slating the use of E2E tests by google <http://googletesting.blogspot.ie/2015/04/just-say-no-to-more-end-to-end-tests.html> . It’s much easier to have a developer who understands their own code write basic integration tests to test the API that will run in seconds as opposed to starting your E2E tests (That an automation tester wrote) going to get a cup of coffee while they run and come back to it having not passed and still not understanding why.