Software Testing requires the execution of test cases to validate the functionality required by the customer. The customers requirements are the bases for test cases. Sometimes the customer specifies their requirements in the form of use cases that is the specific scenarios of the many ways any user may interact with the software system. As one can imagine, there may be many different use cases outlining steps a user may take. Example. Frequently, there may be repetition in individual steps or even repetition of a full set of steps. Wouldn’t it be nice if we could reuse some of the code making our test cases easier to maintain. Coding is used in software testing to automate the process make the test job faster and facilitate reporting and for reuse – tests can be run over and over without much effort. The options in software test are huge, many test languages and tools, many scripting languages to be compatible with the tools, and various execution methods (distributed, standalone, record playback, continuous integration). Management of test architecture, tools, and code infrastructure has become an industry-wide problem. Technology has high velocity, changes rapidly but organizations are struggling to keep up with the pace of change.

The purpose of this article is to propose a management methodology for software test automation that facilitates code reuse, makes test repositories easy to maintain, and facilitates a framework architecture that will harness change.

Cucumber is a behavior driven development framework and is gaining wide popularity because it organizes test infrastructure and makes the test cases readable and understandable to stakeholders and business analysts. In Cucumber test case steps are written out as sentences in feature files. These sentences become links to step definitions the coded functions in human readable format. Step definitions are stored separately from the feature files. The trigger to run the test cases is in the TestRunner file. A simple file specifying the feature file to use and the step definition file to use. When all these are compiled and executed, the system will run the appropriate tests.

To facilitate reuse instead of typing the Selenium commands in the step definitions file, wrap up the Selenium commands to execute a step in a separate function (Functional Wrapper) then in the step definitions file place a call to the Functional Wrapper. The advantage the Functional Wrapper gives is that through that wrapper whenever any test case needs access to that particular step all that is needed is a call. Wrappers can be passed parameters such as locators or element types and based on the element type implement the particular test. Lastly, a data repository may be implemented that contains data for a pages locators, type, and test architecture type.