Changing Engineering Culture with SDETs

Sheetal Hulloli, Santosh Sahu, Mayur Premi

sheetal_hulloli@mcafee.com, santosh_sahu@mcafee.com ,mayur_premi@mcafee.com

Abstract

Often, the silos of Development and Blackbox Testing groups within an Engineering team result from issues that are more than mere skill differences. There is a much larger cultural aspect that usually needs addressing for an engineering team that has clear cut Development and Blackbox Testing roles but struggles to deliver quality software. While an all-round engineering team must have various skills, we want to highlight the importance and the role of "Software Development Engineers in Test" (SDET) in bringing about a culture change in engineering paradigm. When our own team at McAfee made the conscious decision to arm the team with SDETs, little was known as to what they would do and how their role would be different than the clear-cut demarcations that already existed. As we found out, it was a journey that resulted in greater collaboration, testing of areas unheard of in the team, increased operational efficiencies and a mature engineering team. In this paper we want to share our experience of how SDETs were change agents in the evolution of our engineering team, what challenges we faced along the way and how we were able to overcome barriers as a team. We hope to highlight the following:

1. What a model testing team should be comprised of in skill sets (SDET, Automation, and Product Experts)
2. How the role of SDETs differ from Blackbox/System testing roles
3. Why SDETs are important in fast paced engineering environments (with our own example of a complex product platform)
4. How to hire, train and engage successful SDETs
5. What metrics to introduce to measure success
6. Making role models out of SDETs
7. Seeing measurable improvements in product quality by using SDETs
8. Using SDETs to take the team to the next level

Our paper will cite real examples and we hope that our experience will give engineering leaders an insight into how they can bring about change in their teams but more importantly that we will encourage young engineering professionals to look at becoming future SDETs.

Biography

***Sheetal Hulloli** is a Senior SDET in McAfee with twelve years of industry experience in software testing and quality assurance.*

***Santosh Sahu** is a Senior Software QA Engineer in McAfee with around nine years of industry experience in software development, testing and quality assurance.*

***Mayur Premi** is a Senior Software Engineer in McAfee with around ten years of industry experience in software development.*

Introduction

Software is always evolving and growing in complexity. In a typical environment where multiple security software co-exist depending on a single platform product, the software quality and early availability of the dependent software becomes paramount.

Being a platform product, multiple security solutions (products) had to be integrated with us to make them extremely powerful to fight cyber-attacks. To cater newer business and security needs, we decided to revamp our product architecture and design. As we embraced the new design, technology and implementation, we felt it was also time to re-think on the process that we had been following for our sustenance projects - traditional approach of Dev team and QA (Quality Assurance) team. The process followed being Agile, which required a parallel code and test development.

Thinking beyond the traditional approach

The new design and implementation gave us an opportunity to try out our new idea - test the code before the build with the core modules integrated was ready, so that this could be consumed by our internal security software products. This demanded the team have engineers adopting the new role of software test developers a.k.a. SDET (Software Development Engineer in Test) s.

Being a team that followed the traditional software procedure having virtually two groups - software development and Testing group (QA) despite working for the same product, the first thought was to look outside for engineers to fit this new role in the team. After hiring a few SDETs, we came to realize this was an uphill task as there aren't many who could effectively satisfy our need.

The thought crossed our minds – Why shouldn't we look for our own teammates who could fit this role by following a few transitioning steps to become SDETs? But there were a few storms that had to be weathered – How to change the engineering culture? What are the transition steps? How do we measure and appreciate success?

Challenges – Weather the Storm

We also knew this transition could not be done overnight because it involves

1. Addressing the cultural change in the team.
2. Removing Engineers' stigma to adopting the new role, as they were not sure about the expectation for this role.
3. Coming onboard the required skills and meeting the expectations.

   Considering a team such as the one mentioned under modern approach - having an SDET engineer, who could be a developer willing to test software or a black box QA engineer who could write tools to test the software, will help in early identification of defects. With automation engineers available, automating end-to-end scenarios during early phase of software development would drastically bring down the manual regression time and increases the team's confidence in the software.

Traditional Approach	Modern Approach
1. Developers	1. Developers
2. Black Box QA	2. Software Development Test Engineers (Dev/QA)
3. Process - Water Fall/Agile	3. QA Engineers as Product Champions (PCs)
	4. Automation Engineers (Develop the automation frame work)
	5. Process – Agile

SDETs are Change Agents – How?

The team's composition being this – Developers, SDET, PC (Product Champion) s, and Automation Engineers to bring in early defects and confidence, greater collaboration amongst them was needed.

1. As the product design discussion started, the SDETs and PCs (aspiring to become SDETs) were very much part of the discussion to understand and provide constructive feedback on the design.

2. SDETs worked collaboratively with developers to come up with UT (Unit Test) and WB (White Box) Framework and to add test cases incrementally.

3. After the Design Commit phase, in parallel to the development activity, SDETs were made available with a prototype of the module (in the form of header files) to start test preparation.

4. SDETs also worked with the PCs to help them understand the test frameworks (UT and White Box Testing), provide training on development and testing tools (Visual Studio, WinDbg, Valgrind, BullsEye, Coverity)

5. SDETs and PCs came up with test plan and test cases for modules under development and seek early feedback from developers.

6. Module development was done in phased manner where, upon every module phase completion, the library was available for SDETs to work on the following checks: Unit Testing, Error Conditions, Test for buffer overflow error, Memory Leak errors, Design deviation, False Positive behavior tests, Boundary Value Checks.

7. Being a product that will be integrated by multiple internal security products, conscious decision to test modules used by them was taken as part of SDK (Software Development Kit) testing. As part of this activity, exposed APIs for integration would be tested in a WB framework. The focus was on Argument errors, functional behavior, and scenario-based testing.

8. Based on the results of above tests, the module was either re-worked or made available as part of nightly builds.

The other area of focus was Automation where, automation engineers came up with an automation framework to test E2E scenarios.

1. They worked closely with PCs on the end-to-end scenario identification and get early feedback from the developers.
2. PCs (aspiring to become automation experts) where given training on scripting language, working and execution of automation framework.

Finally, with all the collaborative work, on the nightly build with incrementally developed modules, following validations are done in parallel - UT suite, static Analysis check and automation would be run.

Additionally, these builds were used to measure the code coverage against the available UTs. This code coverage results also facilitated the SDETs to add more UT cases to identify negative scenarios and dead-code. This activity ensured that "the code" is tested to the maximum using WB methodology.

Now comes the million-dollar question – How QA engineers were transformed to SDETs and Automation Experts? What are transition steps? Our long journey on product development provided the answer - in order to adopt the above said new approach we came across the following transition phases in our team.

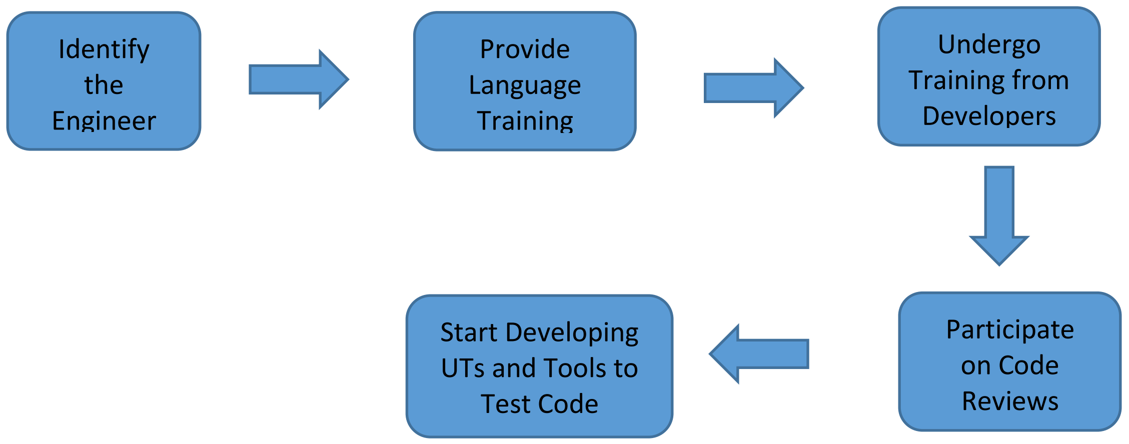
The Transformation Approach

Based on the current role an engineer plays in the software industry, we would like to propose different approaches for transformation to SDET considering the current role of team member.

Approach 1 - Transition of Black Box QA engineer to SDET:

The first approach is for an engineer who currently performs quality assurance and testing using typical black box methodology and aspires to be a SDET

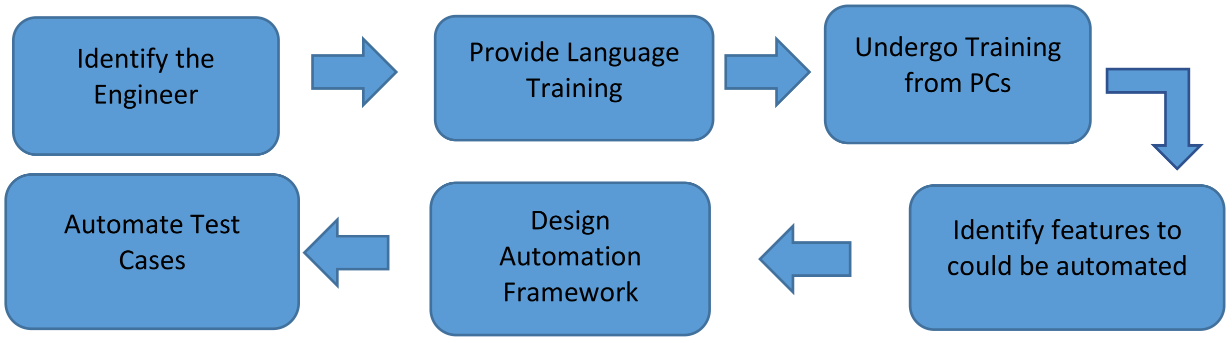
1. Identify a black box QA engineer willing to transform into a SDET.
2. Provide the language training related to software, if required.
3. Undergo training from developers on product implementation and design.
4. Participate on code reviews and design discussions.
5. Start developing UT and tools that tests software modules


Identify the Engineer
Provide Language Training
Undergo Training from Developers
Participate on Code Reviews
Start Developing UTs and Tools to Test Code

Approach 2 - Transition of Black Box QA engineer to Automation Engineer:

The second approach is for an engineer who currently performs quality assurance and testing using typical black box methodology and aspires to be an automation engineer who basically writes code to test the commonly used functionality of the software to provide quick confidence on the software builds.

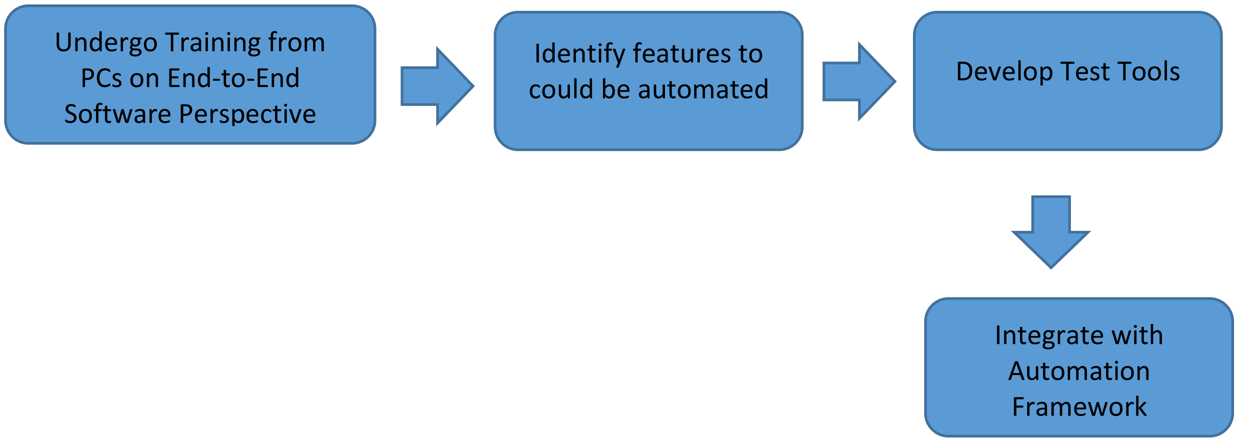
1. Identify a black box QA engineer willing to transform into an automation engineer.
2. Provide the language training related to automation, if required.
3. Undergo training from product champions on software from end-to-end user perspective.
4. Identify features or test cases that could be automated.
5. Design Automation frameworks.
6. Start automating feature test cases.


Identify the Engineer
Provide Language Training
Undergo Training from PCs
Identify features to could be automated
Design Automation Framework
Automate Test Cases

Approach 3 - Transition of a developer to SDET:

The third approach is for an engineer who currently develops software is aspiring to perform the role of an SDET.

1. Undergo training from PC understand the software from end-to-end user perspective.
2. Identify the features that could be tested using test tools.
3. Start developing test tools on the features.
4. Integrate with automation framework for daily run.

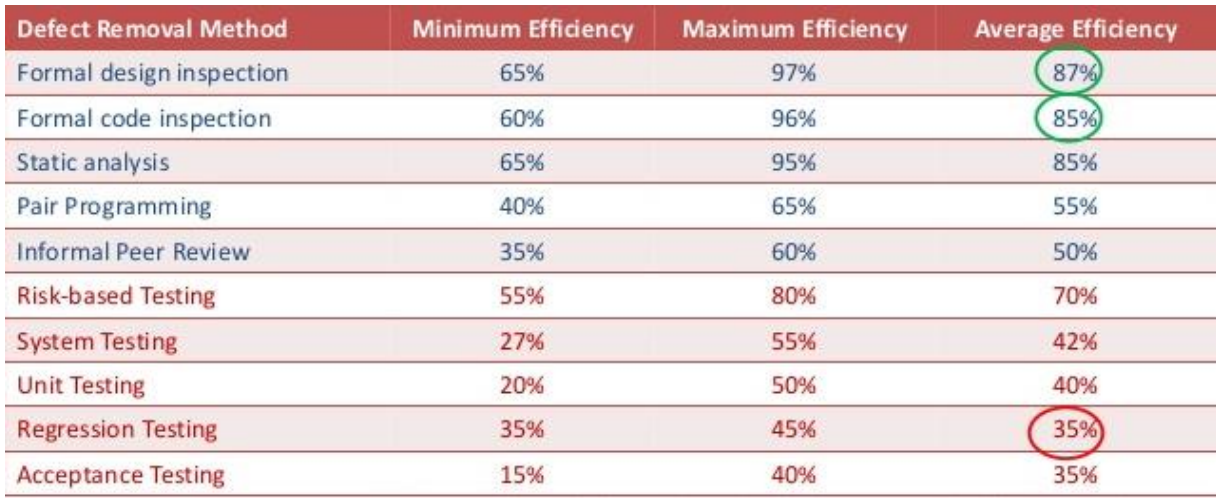

Undergo Training from PCs on End-to-End Software Perspective
Identify features to could be automated
Develop Test Tools
Integrate with Automation Framework

Results

Traditionally team’s release check-list would e – BVT (Basic Verification Test), FVT (Functional Verification Test), Soak, Regression, Open Defects etc.

With the new approach the check-list was modified as – Unit Test, Memory Leak check, Scenario based white box testing, Static Analysis, FVT, BVT, Soak, Automation Results, Number of high severity open defects.

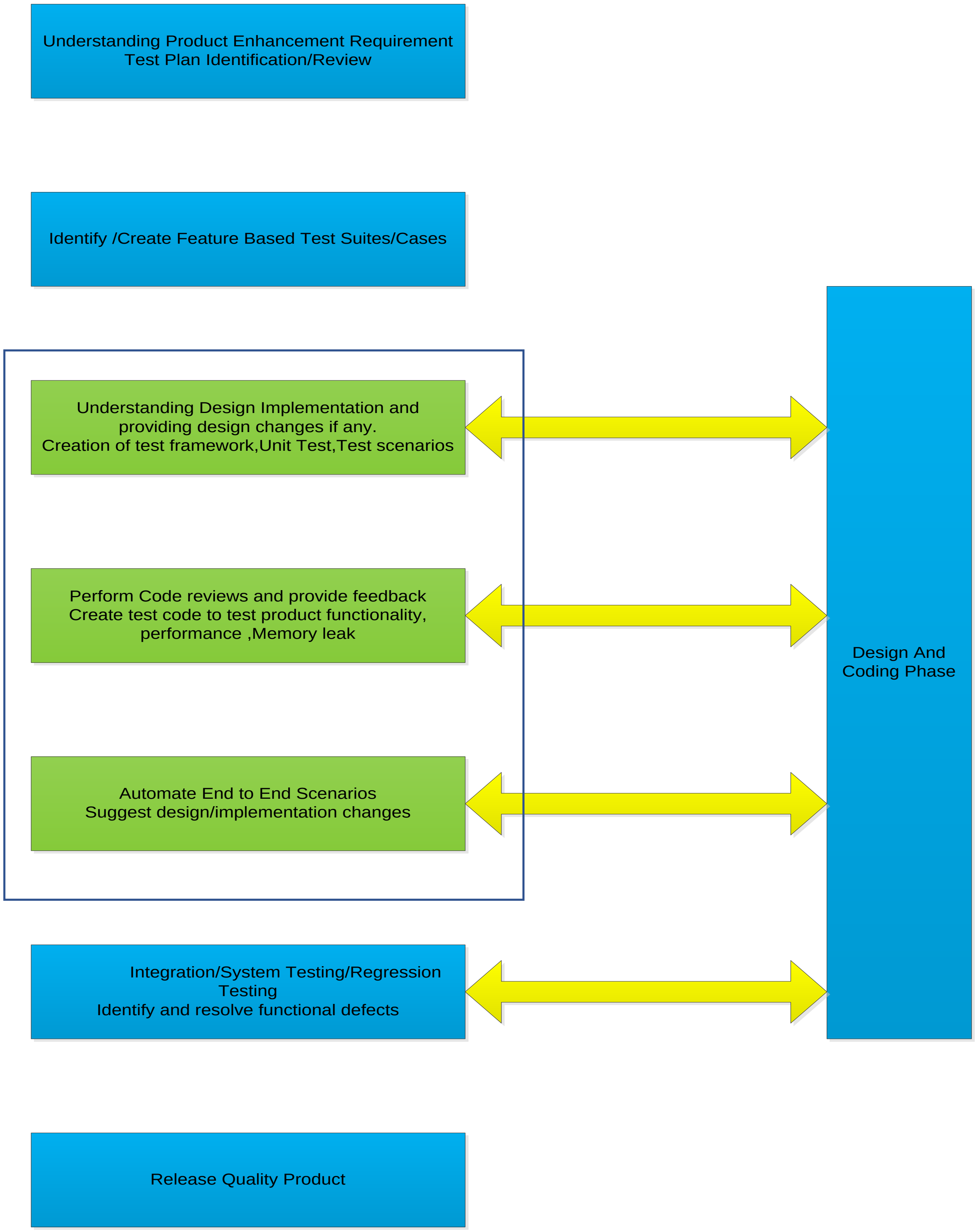
Advantage of a PC (one who is aware of the end-to-end functionality of the product) transitioning as an automation engineer would ensure the presence of better automation framework that suits the product and yield better test coverage.



Defect Removal Method	Minimum Efficiency	Maximum Efficiency	Average Efficiency
Formal design inspection	65%	97%	87%
Formal code inspection	60%	96%	85%
Static analysis	65%	95%	85%
Pair Programming	40%	65%	55%
Informal Peer Review	35%	60%	50%
Risk-based Testing	55%	80%	70%
System Testing	27%	55%	42%
Unit Testing	20%	50%	40%
Regression Testing	35%	45%	35%
Acceptance Testing	15%	40%	35%

Source: The Economics of Software Quality by Capers Jones, Olivier Bonsignour

How gaps in the Testing life cycle were filled by SDET activity's and their early feedback right from design discussion phase to actual end to end scenario testing. This will help to get quality product at the end of the life cycle.


Understanding Product Enhancement Requirement
Test Plan Identification/Review
Identify /Create Feature Based Test Suites/Cases
Understanding Design Implementation and providing design changes if any.
Creation of test framework,Unit Test,Test scenarios
Perform Code reviews and provide feedback
Create test code to test product functionality, performance ,Memory leak
Automate End to End Scenarios
Suggest design/implementation changes
Integration/System Testing/Regression Testing
Identify and resolve functional defects
Release Quality Product
Design And Coding Phase

References.

The Economics of Software Quality by Capers Jones, Olivier Bonsignour

Acronyms

QA	-	Quality Assurance
SDET	-	Software Development Engineer in Test
PC	-	Product Champion
UT	-	Unit Test
WBT	-	White Box Testing
VS	-	Visual Studio
SDK	-	Software Development Kit
FVT	-	Functional Verification Test
BVT	-	Basic Verification Test