UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
UNIDADE ACADÊMICA DE SISTEMAS E COMPUTAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

OLIVEIROS CAVALCANTI DE OLIVEIRA NETO

UM RELATO DE EXPERIÊNCIA SOBRE A INSERÇÃO DE PRÁTICAS DE QUALIDADE DE SOFTWARE NO AMBIENTE DE DESENVOLVIMENTO DA STARTUP PROJ4ME

CAMPINA GRANDE - PB

2020

OLIVEIROS CAVALCANTI DE OLIVEIRA NETO

UM RELATO DE EXPERIÊNCIA SOBRE A INSERÇÃO DE PRÁTICAS DE QUALIDADE DE SOFTWARE NO AMBIENTE DE DESENVOLVIMENTO DA STARTUP PROJ4ME

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Professor Dr. Wilkerson de Lucena Andrade.

CAMPINA GRANDE - PB

2020

O48r Oliveira Neto, Oliveiros Cavalcanti de.
Um relato de experiência sobre a inserção de práticas de qualidade de software no ambiente de desenvolvimento da Startup PROJ4ME. / Oliveiros Cavalcanti de Oliveira Neto. – 2020.

9 f.

Orientador: Prof. Dr. Wilkerson de Lucena Andrade.
Trabalho de Conclusão de Curso - Artigo (Curso de Bacharelado em Ciência da Computação) - Universidade Federal de Campina Grande; Centro de Engenharia Elétrica e Informática.

1. Qualidade de software. 2. Desenvolvimento de software. 3. Startup. 4. Estudo de caso. 5. Incidentes reportados pelos usuários – softwares. 6. Boas práticas – qualidade de software. I. Andrade, Wilkerson de Lucena. II. Título.

CDU:004(045)

Elaboração da Ficha Catalográfica:

Johnny Rodrigues Barbosa
Bibliotecário-Documentalista
CRB-15/626

OLIVEIROS CAVALCANTI DE OLIVEIRA NETO

UM RELATO DE EXPERIÊNCIA SOBRE A INSERÇÃO DE PRÁTICAS DE QUALIDADE DE SOFTWARE NO AMBIENTE DE DESENVOLVIMENTO DA STARTUP PROJ4ME

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA:

Professor Dr. Wilkerson de Oliveira Neto
Orientador – UASC/CEEI/UFCG

Professora Dra. Melina Mongiovi Cunha Lima Sabino
Examinador – UASC/CEEI/UFCG

Professor Dr. Tiago Lima Massoni
Disciplina TCC – UASC/CEEI/UFCG

Trabalho aprovado em: 2020.

CAMPINA GRANDE - PB

UM RELATO DE EXPERIÊNCIA SOBRE A INSERÇÃO DE PRÁTICAS DE QUALIDADE DE SOFTWARE NO AMBIENTE DE DESENVOLVIMENTO DA STARTUP PROJ4ME

Trabalho de Conclusão de Curso

Oliveiros Cavalcanti de Oliveira Neto
Unidade Acadêmica de Sistemas e Computação,
Universidade Federal de Campina Grande,
Campina Grande, Paraíba, Brasil,
oliveiros.neto@ccc.ufcg.edu.br

Wilkerson de Lucena Andrade
Unidade Acadêmica de Sistemas e Computação,
Universidade Federal de Campina Grande,
Campina Grande, Paraíba, Brasil,
wilkerson@computacao.ufcg.edu.br

Resumo

Práticas relacionadas à qualidade de software são bem vistas, principalmente por grandes empresas, pois agregam valor ao produto final. Entretanto, o cenário em pequenas empresas e startups tende a ser diferente. Empresas menores ou recém criadas, em muitos casos, precisam lançar seus produtos rapidamente e em tempo hábil para se estabelecer no mercado. Do outro lado da balança observamos que programas com níveis de qualidade superior conseguem mais espaço entre os consumidores, logo é necessário existir um equilíbrio entre agilidade e qualidade. Neste trabalho conduzirei estudo de caso sobre a implantação dessas práticas em uma dessas empresas de pequeno porte, inserindo padrões comuns da área de qualidade ao processo de desenvolvimento da empresa buscando aumentar a confiabilidade da solução desenvolvida e diminuir as taxas de incidentes que são reportadas pelos clientes da mesma.

Palavras Chaves

Qualidade de software, processos, teste.

1. Introdução

Qualidade de software pode ser considerada um campo dentro da computação, olhando puramente pela visão popular temos em geral que o termo se refere a algo abstrato, um sinônimo da perfeição em seu estado da arte, entretanto no campo profissional da Engenharia de Software sabemos que o termo se refere a conformidade de uma aplicação para com seus requisitos e adequação ao uso dos seus usuários.

Outro ponto importante de observação é que a área de Qualidade de Software já é por si só muito vasta, contando com diversas técnicas e ferramentas que podem ser usadas no auxílio dos times de desenvolvimento para agregar valor ao seu produto final.

Porém, apesar de ser um agregador de valor ao produto final, a adoção de boas práticas no processo de desenvolvimento normalmente acarreta em um aumento linear do custo final do produto, assim como também tende naturalmente a aumentar o tempo total do desenvolvimento, já que temos uma adição de etapas de verificação na esteira produtiva.

Sendo assim, as equipes de Qualidade de Software enfrentam o desafio contínuo de traduzir o custo da qualidade em termos de dólares e centavos. Eles estão percebendo que essa reinterpretação da qualidade é essencial para obter o envolvimento e o apoio da alta administração [1].

Logo, um dos maiores desafios que empresas vislumbram é ter uma ideia dos impactos causados por uma transformação baseada em qualidade. E esse cenário é ainda mais problemático em empresas de tamanho pequeno ou recém criadas, onde a necessidade de colocação rápida dos seus produtos no mercado é gigante.

Iremos neste trabalho observar a inserção de práticas básicas de qualidade dentro de uma pequena e recém criada empresa fabricante de software, com o intuito de mensurar quanto essa mudança de paradigma agregou de valor ao produto final. Faremos isto falando um pouco sobre trabalhos relacionados que já foram realizados, apresentaremos nossos objetivos gerais e específicos, discutiremos o processo pré transformação e pós transformação, descrevendo cada mudança realizada e por fim avaliamos o impacto na seção dos resultados.

---

2 Trabalhos relacionados

Em 1988, Tarek K. Abdel-Hamid [2] fez um estudo sobre o custo da inserção de padrões de qualidade dentro de um projeto, investigando o impacto financeiro acarretado por uma melhoria baseada em qualidade do produto final e chegou a conclusão que essa inserção tem impacto considerável, por volta de 15% do orçamento total.

Saindo um pouco da área de custos e entrando na visão de desenvolvimento ágil, em 2004, Ming Huo, June Verner, Liming Zhu e Muhammad Ali Babar [3] fizeram uma análise sobre a inserção de qualidade dentro de uma equipe ágil, concluindo que passos de QA devem ser inseridos nas tarefas rotineiras dos desenvolvedores assim como devem ser inseridas etapas a mais no processo, a frequência de testes é maior em times ágeis e essas práticas devem ser inseridas desde a concepção e estágios iniciais das aplicações.

Em 2017, Megan Sumrell [4] avaliou o impacto da mudança de paradigma, do clássico waterfall para o ágil colocado num projeto real e comentou que a mudança não foi simples, vários pontos focais foram listados, como por exemplo, buscar o auxílio de pessoas que são mais experientes e passaram por este processo de disrupção antes, não assumir que regras e responsabilidades devem ser rotuladas eternamente a certos membros e que todos os participantes devem ser considerados donos da qualidade do produto.

Edinson J. Santos, Rita Suzana Pitangueira Maciel e Cláudio Sant'Anna [5] realizaram um levantamento de *bad smells* que podem ser aplicados para processos, levantando alguns sinais interessantes de processos que podem estar sendo maléficos para o produto final e sobre como realizar melhorias em cada um desses *smells*.

E finalmente, Renata Sousa, Karla Malta, Roselana Silva, Paulo Masiero, Eduardo Almeida e Ivan Machado [6] em 2019 fizeram o estudo sobre como as startups lidam com processos de desenvolvimento, chegando a conclusão que a maioria delas não utiliza nenhum processo definido claramente em sua esteira de produção, se modificando e tentando realizar melhorias unicamente por meio de feedback de seus usuários.

3. Metodologia

Nesta seção apresentaremos a forma que o trabalho foi conduzido, apresentando nossos objetivos e depois se debruçando sobre o processo realizado pela equipe de construção do software antes e após a transformação baseada em ideais da Qualidade de Software.

3.1 Objetivos

Nesta seção explicitamos nosso objetivo geral e nosso objetivo específico para que o leitor entenda qual o nossa proposta inicial para o trabalho.

3.1.1 Objetivo Geral

Neste estudo de caso analisaremos o impacto de inserir técnicas de qualidade dentro do processo de desenvolvimento de uma aplicação, a fim de mensurar o quanto foi ganho em valor ao produto final da empresa.

Nosso critério básico de análise da qualidade do software é a quantidade de incidentes reportados por usuários reais da aplicação, então temos como objetivo geral analisar o impacto quantitativo provocado neste número após uma série de mudanças realizadas no processo de produção de novas partes do sistema.

3.1.2 Objetivos Específicos

Mostraremos os resultados obtidos em um caso de transformação de processo de desenvolvimento da startup Proj4Me, saindo de um início com padrões arcaicos de codificação para o que é considerado atualmente o padrão a ser seguido de qualidade, medindo o impacto que esta transformação causou na equipe e no produto.

3.2 Base de dados

Os dados foram obtidos a partir da coleta manual de um agente de qualidade que foi inserido na equipe de desenvolvimento da startup. Os dados dizem respeito a detalhes de desenvolvimento do time (histórias desenvolvidas, esforço em horas para a realização de cada história, linhas de código adicionadas ou alteradas por história, incidentes reportados durante o processo de desenvolvimento de cada história e incidentes reportados por usuários finais em ambiente de produção). De acordo com Steve McConnell, em seu livro "Code Complete", temos entre 15 e 50 erros a cada mil linhas de código entregues [7], com isso como base, apenas consideramos histórias válidas para a pesquisa demandas que geraram e/ou alteraram pelo menos mil linhas de código do repositório.

3.3 Cenário pré-transformação

Por se tratar de uma startup em passos iniciais de existência dentro do mercado, os donos do produto logo se preocuparam em deixar disponível um *MVP* [8] o mais rápido possível. Para isto a equipe consistia de três desenvolvedores que realizavam o processo de desenvolvimento no tradicional modelo de cascata [9], como podemos ver a seguir:

Imagem 1. Processo de desenvolvimento inicial em cascata

Durante as semanas iniciais de desenvolvimento da solução, a equipe seguiu esta metodologia. Seguem os dados obtidos a partir da coleta das métricas:

História	Horas	Linhas	A0*	A1*
Feature 1	42 horas	2031	-	17
Feature 2	35 horas	1848	-	13
Feature 3	22 horas	1222	-	8
Feature 4	17 horas	1089	-	6

***A0** - Incidente reportado durante o processo de desenvolvimento da história.

***A1** - Incidente reportado por usuário final em ambiente de produção até 30 dias após a liberação da feature.

Tabela 1. Dados coletados relativos às primeiras semanas de desenvolvimento

Como observado na tabela, os desenvolvedores não encontraram ou não registraram incidentes durante o processo de desenvolvimento das histórias iniciais da aplicação. Chama atenção também o alto número de incidentes que foram reportados apenas por usuários da aplicação em ambiente de produção, número este que fez os donos do produto repensarem as estratégias e métodos usados na concepção da solução, já que cada incidente reportado representa pelo menos um cliente insatisfeito com o programa usado.

3.4 Adoção de práticas de qualidade de software

O alto número de incidentes vazados até os usuários finais preocupou a direção da startup, que via a quantidade de clientes do sistema aumentar porém não contava com a capacidade de desenvolver novas partes do sistema num bom nível de confiança.

Então para conseguir entregar melhorias em tempo hábil, a direção decidiu contratar dois novos desenvolvedores além de um agente de qualidade (primeiro autor do artigo), que seria a pessoa responsável por indicar as alterações que precisavam ser realizadas para que o time de desenvolvedores pudesse efetuar entregas em um alto padrão de confiabilidade.

Nesta seção falaremos um pouco sobre cada mudança proposta pelo analista de testes e explicaremos cada novo passo adicionado a esteira produtiva da empresa.

3.4.1 Alteração no processo

Com a inserção do agente de qualidade, além da adoção de ciclos de entrega com a duração de quinze dias, foi visto que o processo deveria ser revisado e algumas etapas deveriam ser adicionadas a ele, de forma a garantir a fiscalização, verificação e consistência dos entregáveis, o novo processo pode ser visto na seguinte imagem:

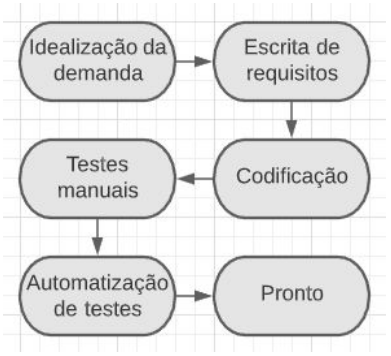

Imagem 2. Processo de desenvolvimento após inserção de etapas de verificação

3.4.1.1 Idealização da demanda

A fase de idealização da demanda foi a única que permaneceu inalterada durante a transformação de qualidade. Os donos da solução

sugerem a concepção de novas features a partir de necessidades apontadas pelos usuários finais.

3.4.1.2 Escrita dos requisitos

A fase de escrita e refinamento de requisitos foi adicionada com o intuito de deixar claro aos programadores os comportamentos necessários das demandas [10]. Essa escrita e refinamento foi realizada a partir de entrevistas do agente de qualidade com os idealizadores da funcionalidade.

Nesta entrevista o agente de qualidade procurou entender os motivos que fizeram a demanda surgir, os efeitos colaterais que essa demanda específica pode causar ao produto como um todo e as necessidades especiais dos usuários relativas a esta nova parte do sistema.

Ao final deste ciclo de entrevistas, é gerado um relatório onde temos uma prototipação das novas telas a serem criadas, o fluxo principal e fluxos adjacentes que podem acontecer nessa nova parte do sistema e uma descrição detalhada de cada componente que faz parte do entregável, facilitando assim a visão do time de desenvolvimento e mitigando possíveis maus entendimentos sobre o que deve ser feito, além de otimizar o tempo de construção já que a maiorias das dúvidas já são sanadas pela documentação construída previamente.

3.4.1.3 Codificação

A fase de codificação também permaneceu no fluxo, entretanto esta fase recebeu o incremento de adição de boas práticas de qualidade pelo time de desenvolvimento.

Um workshop sobre os assuntos *Clean Code e* refatoração de código [11] foi apresentado à equipe para que todos os participantes do projeto tivessem uma base teórica para colocar em prática as informações absorvidas.

Além disso, foi convencionado que o time de desenvolvimento seria responsável pela escrita de testes unitários para todo e qualquer código que escrevesse e/ou alterasse. Apesar de ser uma métrica bastante questionada pela comunidade [12] por fornecer dados muitas vezes fantasiosos sobre o nível de qualidade de um trecho de código, foi acordado entre todos os membros da equipe que apenas seria aceito código no repositório que tivesse pelo menos 90% de cobertura por testes unitários.

3.4.1.4 Testes manuais

Os testes manuais foram vistos como necessários pelo agente de qualidade e foram adicionados como uma etapa fundamental da concepção de uma nova parte do sistema.

O agente de qualidade realizou o levantamento de cenários de teste a partir da documentação de requisitos criados no começo do processo e utilizou as mais diversas técnicas de teste existentes, entre elas, partição de equivalência e teorema de valores limites, com o objetivo de estressar a aplicação e forçar erro. Os testes neste ponto eram realizados usando a técnica de caixa preta [13] e em ambiente de desenvolvimento.

Ao final de cada bateria de testes, os resultados eram computados e incidentes catalogados para correção pelo time de desenvolvimento antes da liberação da demanda em ambiente produtivo de fato.

3.4.1.5 Automação dos testes

Após a aprovação do agente de qualidade na etapa de testes manuais, os principais cenários eram automatizados utilizando a ferramenta *Cypress.io,* com o objetivo de evitar futuros danos colaterais que pudessem vir a surgir em novas implementações ou novas visitas a esta parte específica do sistema. Esta etapa poderia ser realizada tanto pelo agente de qualidade ou como também por um dos desenvolvedores do time, porém era convencionado que o desenvolvedor que trabalhou na criação da demanda não seria o desenvolvedor que trabalharia na codificação dos cenários automatizados.

3.4.1.6 Definição de pronto

Tendo a nova parte do sistema passado corretamente por todas as etapas de desenvolvimento e apresentado bons resultados em todas elas, esta parte seria disponibilizada de maneira segura em ambiente produtivo para que os usuários finais da aplicação tivessem acesso e pudessem desfrutar de uma nova funcionalidade em sua aplicação.

3.4.2 Ferramentas de análise estática de código

Além da mudança no processo a ser seguido durante o desenvolvimento, a equipe sentiu a necessidade de utilizar ferramentas que gerassem relatórios sobre a saúde do código em aspectos básicos como padronização, estilo e complexidade ciclomática. Para isso o time passou a fazer varreduras semanais do código com o uso das ferramentas open source Checkstyle, FindBugs e PMD.

4. Resultados

As primeiras três sprints do projeto foram consideradas experimentais, com todo o time aprendendo a colocar em prática as mudanças propostas pelo agente de qualidade e ainda sem todo o aparato testamental funcionando a pleno vapor.

A partir da sprint 4, já com todos os passos da esteira de desenvolvimento sendo devidamente realizados e com as ferramentas de análise estático de código realizando varreduras constantes, o time obteve os seguintes resultados:

História	Horas	Linhas	A0*	A1*
Feature 5	55 horas	3004	12	4
Feature 6	53 horas	2459	7	2
Feature 7	49 horas	2555	11	0
Feature 8	33 horas	1236	5	5
Feature 9	18 horas	1645	7	1
Feature 10	12 horas	1268	2	0

***A0** - Incidente reportado durante o processo de desenvolvimento da história.

***A1** - Incidente reportado por usuário final em ambiente de produção até 30 dias após a liberação da feature.

Tabela 2. Dados coletados relativos às sprints 4, 5 e 6

Observa-se uma substancial diminuição do número de incidentes vazados até o usuário final do produto. Nas quatro features desenvolvidas inicialmente, tivemos em média um incidente reportado por um usuário final a cada aproximadamente 140,68 linhas de código, já nas seis features desenvolvidas após a mudança do processo tivemos a média de um incidente reportado por usuários finais a cada aproximadamente 873,81 linhas de código. Uma melhoria de aproximadamente 521,13% no número.

Olhando para a esfera de produtividade, sem usar o processo de qualidade bem definido, tivemos em média uma demanda entregue a cada 29 horas e utilizando o processo bem definido, tivemos em média uma demanda entregue a cada aproximadamente 36,66 horas. Um aumento de aproximadamente 26,41% em média no tempo de entrega das histórias.

4.1 Resultados refletidos nos custos

Realizando uma análise em termos puramente financeiros, levaremos em conta dados observados no *glassdor* (site especializado em calcular médias salariais) do ano de 2019 (ano de realização do estudo de caso) do salário mensal médio de um desenvolvedor júnior no município de Campina Grande (local onde a startup está localizada geograficamente) que é de dois mil e sessenta reais mensais (80 horas de trabalho por mês). Sendo assim a hora de trabalho por desenvolvedor custa em média R$ 25,75.

Antes da transformação do processo baseado nos princípios de Qualidade de Software tivemos um custo médio por feature de R$ 746,75. Após a série de medidas com o objetivo de melhorar a entrega final do produto tivemos em média um custo por feature de R$ 944,17.

Nota-se um aumento de R$ 197,42 no custo médio de cada feature implementada, porém a diretoria da startup decidiu manter o processo estabelecido pelo agente de qualidade por achar que o valor acrescido é pequeno quando comparado com o valor agregado ao produto final, dada a diminuição dos incidentes reportados pelos usuários nas features fabricadas seguindo as novas diretivas de produção.

Além disto, se formos mais a fundo nas entregas do time, vemos que antes da implantação do novo processo tínhamos um custo de R$ 0,48 por linha de código escrita pela equipe de desenvolvimento e após a implantação, este número caiu para R$ 0,35. Sendo assim, observamos que o time demorou mais para construir as entregas, entretanto as entregas foram compostas por mais linhas de código que as iniciais, o que pode ser explicado pelo acréscimo das etapas de codificação de testes unitários e de testes automatizados (que foram contabilizados como parte da feature).

5. Conclusão e trabalhos futuros

Neste artigo conduzimos um estudo de caso sobre os impactos que a adoção de processos bem definidos de qualidade podem trazer a um projeto. Usamos dados coletados diretamente da fonte de um projeto real recém iniciado que passou pelo dilema de escolher entre agilidade pura e agilidade casada com qualidade.

A avaliação dos resultados mostrou um grande avanço no valor do produto final concebido pelo time de desenvolvimento, entretanto com alguma perda de produtividade em seu sentido básico da palavra e aumento no custo médio.

O maior desafio encontrado na realização deste trabalho foi sem sombra de dúvidas realizar uma mudança de paradigma na mentalidade das pessoas que faziam parte da startup, fazer os colaboradores saírem do mundo onde entregas rápidas significam entregas boas para um mundo onde a realização de etapas de planejamento e verificação são tão importantes e agregadoras de valor quanto a fase de codificação do entregável em si.

As disciplinas de Engenharia de Software e Projeto em Computação foram essenciais para que as ações necessárias para uma melhoria no processo de desenvolvimentos fossem executadas com sucesso. Além delas, podemos citar a disciplina de Validação e Verificação como um norte nas ações tomadas nas etapas de testes manuais e testes automatizados.

Para possíveis trabalhos futuros sugerimos a utilização de processos de qualidade que envolvam TDD e/ou BDD para realização de medições do impacto causado por essas técnicas de testes, que atualmente são consideradas as mais sofisticadas pela comunidade. Além disso, um bom ponto de enfoque pode ser na questão monetária, fazendo um estudo mais aprofundado sobre o impacto financeiro causado em razão do valor agregado ao produto final, levando em conta por exemplo, dados de quantidade de clientes impactados por incidentes que são vazados ao produto final e como isso gera atrito no cofres das empresas.

Referências

[1] Riggs, H.E. Managing High-Technology Companies, Lifetime Learning Publications, Belmont, CA, 1983.

[2] Tarek K. Abdel-Hamid. The Economics of Software Quality Assurance: A Simulation-Based Case Study, Management Information Systems Research Center, University of Minnesota, 1988.

[3] Ming Huo, June Verner, Liming Zhu, Muhammad Ali Babar. Software Quality and Agile Methods, Proceedings of the 28th Annual International Computer Software and Applications Conference, 2004. COMPSAC 2004.

[4] Megan Sumrell. From Waterfall to Agile - How does a QA Team Transition?, Agile 2007 (AGILE 2007).

[5] Santos, E. J., Maciel, R. S. P., & Sant'Anna, C. (2018). A Catalogue of Bad Smells for Software Process. Proceedings of the 17th Brazilian Symposium on Software Quality - SBQS.

[6] Souza, R., Malta, K., Silva, R., Masiero, P., Almeida, E., & Machado, I. (2019). A Case Study about Startups' Software Development Practices. Proceedings of the XVIII Brazilian Symposium on Software Quality - SBQS'19.

[7] Steve McConnell. Code Complete: A practical handbook of software construction, Second Edition, 2004.

[8] Valentina Lenarduzzi, Davide Taibi. MVP Explained: A Systematic Mapping Study on the Definitions of Minimal Viable Product, Euromicro SEAA, 2016.

[9] Kai Petersen, Clais Wohlin, Dejan Baca. The Waterfall Model in Large-Scale Development, International Conference on Product-Focused Software Process Improvement, 2009.

[10] Manju Geogy, Andhe Dharani. A Scrutiny of the Software Requirement Engineering process, Global Colloquium in Recent Advancement and Effectual Researches in Engineering, Science and Technology (RAEREST 2016).

[11] Amandeep Kaur, Manpreet Kaur. Analysis of Code Refactoring Impact on Software Quality, MATEC Web of Conferences, 2016.

[12] Vardy Antinyan, Jesper Derehag, Anna Sandberg, Miroslaw Staron. Mythical Unit Test Coverage, IEEE Software, 2018.

[13] Mohd. Emer Can. Different Approaches To Black Box Testing Technique For Finding Errors, International Journal of Software Engineering & Applications, 2011.