|  |  |
| --- | --- |
|  | *Code Inspection Report*  *Anti-Spam Configuration Software Development Project*  BSc/MSc in [ METI]  Academic Year 2017/2018 - 1º Semester  Software Engineering I  Group 47  61829, Tomás Pires, METI  64454, Joana Antunes, METI  65106, Soraia Fernandes, METI  70082, Bruno Mataloto, METI  ISCTE-IUL, Instituto Universitário de Lisboa  1649-026 Lisbon  Portugal  December 20th 2017 |

**Table of Contents**

[Introduction 3](#_Toc498465002)

[Code inspection – Name of the component being inspected 3](#_Toc498465003)

[Code inspection checklist 3](#_Toc498465004)

[Found defects 3](#_Toc498465005)

[Corrective measures 3](#_Toc498465006)

[Conclusions of the inspection process 3](#_Toc498465007)

# IntroductionIntroduction

*O software produzido tem como objetivo configurar um filtro antispam para uma Leisure Box , em que o utilizador insere o ficheiro de regras rules.cf,o ficheiro ham.log e o ficheiro spam.log e o software gera automaticamente uma configuração ideal, possivel de ser editada pelo utilizador. Permite ainda a criação de uma configuração totalmente manual ou aleatória.*

# Code inspection – Filtro AntiSpam

*Este relatório resulta da inspeção ao código desenvolvido pelo grupo.*

|  |  |
| --- | --- |
| *Meeting date:*  *Meeting duration:*  *Moderator:*  *Producer:*  *Inspector:*  *Recorder:* | *20/12/2017*  *40 minutes*  *Joana Antunes*  *Bruno Mataloto*  *Soraia Fernandes*  *Tomás Pires* |
| *Component name (Package/Class/Method):* | *Package : AntiSpamFilter* |
| *Testing coverage achieved:* | *81,4%* |

# Code inspection checklist

The checklist for Java code inspection used in this project is available at <http://www.cs.toronto.edu/~sme/CSC444F/handouts/java_checklist.pdf> and in blackboard ES1 page.

# Found defects

Identify and describe found defects, opinions and suggestions.

|  |  |  |  |
| --- | --- | --- | --- |
| **Found defect Id** | **Package, Class, Method, Line** | **Defect category** | **Description** |
| 1 | Class: Controller | 1 | Atributo String da classe Rule que nomeava a regra, tinha a mesma designação que a própria Rule na classe Controller o que geráva alguam confusão |
| 2 | Class: GUI | 2 | Existem métodos extra desnecessários, que foram criados para permitir testar ActionListeners presentes na classe |
| 3 | Class: Email e Controller | 14 | Loop de pesquisa das regras no email era muito complexo e atrasava a geração da configuração automática, tendo sido adicionado os indices das regras correspondentes a cada email tornando o ciclo extremamente mais simples e rápido |
| 4 | Package : AntiSpamFilter | 10 | A linguagem utilizada para os Java Docs em alguns casos encontra-se em Inglês e Português |
| 5 | Class Controller | 13 | Começámos por utilizar Vetores, mas devido á dificuldade de gestão de espaço para outros ficheiros, optámos por mudar para ArrayLists |

# Corrective measures

*Alguns destes defeitos foram facilmente corrigidos no sprint respetivo, no entanto o problema 3 necessitou de mais tempo pois foi necessário alterar métodos de várias classes. Para a geração de testes JUnit os métodos extra foram necessários para melhorar a cobertura do código, uma vez que possuímos bastante código dentro de ActionListeners que de outra forma nunca seriam testados.*

# Conclusions of the inspection process

*Podemos concluir depois de efetuarmos a análise ao código produzido, que apesar de executar todas as funcionalidades necessárias, existem sempre formas de melhorar a sua organização, tornar a interpretação por parte de outras pessoas mais simples através de código mais claro e conciso. Nem sempre são adotadas todas as medidas necessárias para obter uma boa qualidade geral, no entanto a inspeção do código permite detetar situações que passariam despercebidas.*