

**Gestão de Projeto de Software 2019/2020**

**SmartParking**

**D1.2.2 - Plano de Controlo da Qualidade**

**Autores**

* Ana Filipa Costa Farinha Alves <[analves07@gmail.com](mailto:analves07@gmail.com)>
* Carolina Carmo Abrantes Lopes da Rosa <[carolinalopesrosa@gmail.com](mailto:carolinalopesrosa@gmail.com)>
* Diogo Santos Castelo Branco <[diogoscb7@gmail.com](mailto:diogoscb7@gmail.com)>
* João Pedro Aleixo e Jesus Pereira <[jaleixo1993@gmail.com](mailto:jaleixo1993@gmail.com)>
* Tiago João Cuevas Alves <[tiagoalves0088@gmail.com](mailto:tiagoalves0088@gmail.com)>

**Estado**

* Publicado

**Versões Principais**

* v0.1, 11/10/2019, Diogo Branco, Objetivos da Qualidade.
* v0.2, 12/10/2019, Carolina Rosa, “Introdução”,preenchimento “Relatório de Avaliação de Qualidade”,”Revisões, ”Métricas de Qualidade”.
* v0.3, 12/10/2019, Ana Alves, “Relatório de Revisões” e revisão de capítulo “Revisões” mais “Relatório de Avaliação de Qualidade” com algumas alterações e fornecimento de hiperligação.
* v0.4, 14/10/2019, Ana Farinha Alves, pequena alteração em “Revisões: Orientações” .
* v0.5, 15/10/2019, João Pereira, “Testes”.
* v0.6, 15/10/2019, Diogo Branco, Revisão e Alteração do documento.
* v0.7, 20/10/2019, Ana Farinha Alves e Carolina Rosa, “Gestão de Risco”, “Plano de Risco”, alteração de atributo de qualidade interno e revisão de documento.
* v0.8, 23/10/2019, João Pereira, “Métricas de Qualidade”

**Versões Publicadas**

* v1.0, 29/10/2019, Tiago Alves, Revisto e publicado, [Documento](https://docs.google.com/document/d/e/2PACX-1vQqa1BR9C6TuwjHZnsm2Xv0CQCaivRHpWIQjy_A5BpjPaIvFlvVx70Yp_YjP9Yb1VYqrBTxBrdppyuH/pub)

**Índice**

[1. Introdução](#_rr4988iuksx7)

[2. Objetivos da Qualidade](#_t7p62cjkxd8l)

[3. Revisões](#_n9q4ee3zoyf)

[3.1. Inspeções](#_eldpqfiivn3m)

[3.2. Walkthroughs](#_9e8h8a3nv5wo)

[4. Testes](#_g5r3t2wrod8c)

[4.1. Testes Unitários](#_8wv0gmygydx9)

[4.2. Testes de Integração](#_cfxbl1jty0ph)

[4.3. Testes de Aceitação](#_ltns3ejby4ha)

[5. Gestão do Risco](#_5fqvw8h2695b)

[5.1. Identificação e Avaliação de Riscos](#_ahe6w56uovua)

[5.2. Controlo e Monitorização de Riscos](#_ih3ciiav5xze)

[6. Padrões de Codificação](#_afmo491fgw0v)

[7. Métricas de Qualidade](#_hsn58y5xuikh)

### 

### 

### 

### 

### 

### 

### 

### 1. Introdução

Para uma boa qualidade de um produto de software é importante definir um bom controlo do produto, qual a qualidade pretendida e definir os atributos mais relevantes. Neste documento são definidas as métricas de avaliação e objetivos de qualidade, que ajudam a entender o comportamento e o funcionamento do software para encontrar os seus defeitos e problemas.

Documento: [D.2.3.2 - Relatório de Avaliação de Qualidade](https://docs.google.com/document/d/1wnQfT-NhOBSaqcsYWoVP5enmtwruLlygrH9jVGnV2Zs/edit)

### 2. Objetivos da Qualidade

Assegurando que cada um dos atributos de qualidade deve ser mensurável.

Os atributos de qualidade externos mais importantes que foram identificados, são os seguintes:

* **Segurança**, sem dúvida um dos atributos mais importantes, pois o nosso produto está destinado a armazenar informações pessoais dos utilizadores, bem como a responsabilidade de ceder, ou não, acesso a uma pessoa ao espaço privado do cliente.
* **Usabilidade**, é importante termos uma interface para o administrador e utilizador que seja fácil de utilizar e intuitiva, pois em termos da administração da informação na base de dados é necessário que a informação específica a cada utilizador e veículo esteja completa e bem configurada.

Relativamente aos atributos internos identificamos o seguinte:

* **Complexidade,** indica o nível de quão complexo o produto de software é, mede o grau de dificuldade em entender e compreender a estrutura interna e externa das classes e os seus relacionamentos.

### 

### 3. Revisões

Todos os documentos produzidos têm de ser revistos. As revisões são úteis para encontrar e eliminar defeitos, assim como atingir o consenso da equipa durante o desenvolvimento do projeto. Também desta forma, evita-se que os erros transitem para fases posteriores, causando o incremento de custos na sua correção.

Documento em que constam as revisões oficiais: [D.2.3.2 - Relatório de Avaliação de Qualidade](https://docs.google.com/document/d/1wnQfT-NhOBSaqcsYWoVP5enmtwruLlygrH9jVGnV2Zs/edit).

#### 3.1. Inspeções

A nossa reunião de inspeção tem como objectivo encontrar os defeitos existentes no produto, de forma a corrigi-los. Tem um moderador para guiar o seguimento desta, que se regerá pelos tópicos dados pelo autor.

O moderador terá que certificar que todos os elementos da inspeção estão preparados para reunião. No decorrer da reunião se surgir alguma dúvida a algum elemento da equipa de inspeção este deve solicitar ao leitor que a explique, também no decorrer da reunião o autor não poderá receber críticas da equipa de inspeção devido ao seu propósito ser a correção de erros e não a crítica. Isto é, assumem-se quatro papéis no processo de inspeção:

* **Autor**.
* **Moderador**.
* **Leitor**.
* **Anotador**, responsável por anotar as situações identificadas para correção.

No final será feito um documento a descrever os erros encontrados e o seu grau de severidade. A listagem de inspeções realizadas pode ser consultada através da hiperligação:

[Relatório da Revisão](https://docs.google.com/document/d/11hQNVMvfgf_oCs64CxRYW88njc1aG8tgQ-2Z9sPDfkY/edit)

#### 3.2. Walkthroughs

Os *Walkthroughs* são reuniões informais em que o autor do documento é também o moderador, isto é, solicita a reunião, convida os revisores, pede comentários e assegura que todos os presentes entendem o produto. São reuniões utilizadas em situações nas quais o autor precisa da opinião de pessoas sem os conhecimentos técnicos para rever o documento. Após a reunião, o autor deve manter o contacto com os presentes pois podem necessitar de informações adicionais. Após o *Walkthrough,* o documento original do autor deverá ser corrigido mediante as questões levantadas durante a reunião.

### 

### 4. Testes

#### 4.1. Testes Unitários

Assim que um módulo do produto esteja terminado, este será sujeito a vários testes unitários. Os testes unitários consistem no teste de cada método desenvolvido nas classes que compõe o módulo. Cada módulo será testado pelo programador que esteve responsável pelo o seu desenvolvimento, recorrendo à *framework* JUnit.

#### 4.2. Testes de Integração

Assim que todos os módulos estiverem terminados e tiverem passado pela fase de testes unitários, estes irão ser parcialmente integrados de forma alternada de maneira a encontrar quaisquer incoerências ou incompatibilidades entre eles. Os testes de integração serão realizados por um programador a designar, mediante uma discussão prévia entre a equipa responsável por cada um dos módulos que está a ser testado.

#### 4.3. Testes de Aceitação

Estes testes serão feitos assim que todos os módulos tiverem sido terminados e tiverem sido aprovados após a fase de testes unitários. Serão verificados se todos os requisitos iniciais estão presentes e de acordo com o pretendido pelo cliente. Os testes de aceitação serão realizados por todos os membros da equipa, assim como membros de outras equipa e também, num ambiente controlado, por utilizadores finais da aplicação.

Documento: [Plano de Testes de Aceitação](https://docs.google.com/document/d/1KEE1ysQHofUMPwZb8b12_vYNLEN5p-1DgdQMUOm4Rrc/edit)*.*

### 5. Gestão do Risco

#### 5.1. Identificação e Avaliação de Riscos

A identificação e a avaliação dos riscos será registrada por um dos membros da equipa, onde os todos os membros da equipa irão discutir e tentar identificar os diferentes riscos. Para cada risco deve ser estimado a probabilidade do mesmo acontecer e qual seria o seu impacto no projecto, conduzindo a diferentes probabilidades ao longo das semanas no desenvolvimento do projeto. Seguidamente será elaborado um plano de riscos, conforme consta na hiperligação: [Plano de Riscos](https://docs.google.com/document/d/1uvxkOAbjpW1YEQa10-xUYV1VIPe0SiMNnSXgDzscJoc/edit#)

A cada risco é atribuído um impacto de:

* Tolerável (1);
* Grave (3);
* Catastrófico (5).

A probabilidade de ocorrência atribuída a um impacto pode ser :

* Muito baixa (1);
* Baixa (2);
* Moderada(3);
* Alta (4);
* Muito Alta(5).

#### 5.2. Controlo e Monitorização de Riscos

Os riscos são revistos todas as semanas para determinar se algum se está a tornar provável de acontecer e também controlar se os efeitos desse risco mudam. A lista de riscos será atualizada a cada reunião (hiperligação: [Plano de Riscos](https://docs.google.com/document/d/1uvxkOAbjpW1YEQa10-xUYV1VIPe0SiMNnSXgDzscJoc/edit#)).

Os riscos com “Probabilidade x Impacto >= 15” são potenciais indicadores e suas ações de mitigação têm de ser descritas. Caso se verifique “Probabilidade x Impacto = 25”, os riscos têm de ser mitigados, aplicando o plano previsto anteriormente.

### 6. Padrões de Codificação

Os padrões de codificação que vão ser utilizados pelos programadores durante o desenvolvimento do projeto, encontram-se disponíveis no [Documento Convenções do Código.](https://docs.google.com/document/d/e/2PACX-1vTEHUCRyEgqwS-LrNtOyQcVbx2BXmlYD0WSjkwMxJU5EuHh77bd_uHv4Wg_jCZerjXcv3EaB5HkFkvG/pub)

### 7. Métricas de Qualidade

**Avaliação de Usabilidade**:

* Relaciona o atributo de qualidade externa com a heurística.
* Este processo terá como base 5 entre as 10 heurísticas de Nielsen.
* Cada heurística tem uma pontuação que varia entre 0 a 4 (métrica proposta por Nielsen onde 0 considera que não existe problema e 4 que existe um problema muito grave). Do total dos resultados, é retirado o valor médio, assim como comentários e dificuldades sentidas na avaliação.
* A avaliação heurística permite modificar e melhorar as funcionalidades na interface do utilizador.
* O nosso objectivo neste ponto de qualidade é obter um índice acima dos 70% ou um valor médio inferior a 1.5.

**Avaliação da Segurança:**

Pretende-se auditar o software desenvolvido como meio de averiguar a segurança deste. De forma a facilitar esta auditoria ao código será considerada a noção de superfícies de ataque, que segundo M. Howard *et al*., é algo importante na redução dos riscos de segurança, assim como na melhoria do código que é escrito. Atendendo a estes critérios foi elaborado o gráfico da figura 1. O nosso projeto pretende encontrar-se o mais perto da origem do gráfico que possível.

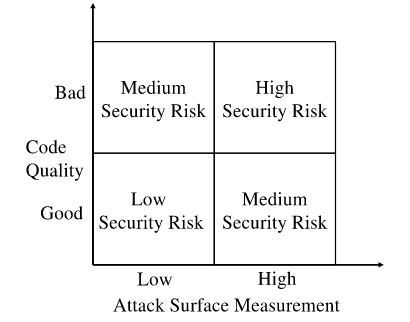


Figura 1 - A superfície de ataque e a qualidade do código são abordagens complementares ao melhoramento da segurança do *software*.

De forma a garantir superfícies de ataque reduzidas ao código a desenvolver será arquitetado de forma a que estas sejam mínimas.

De forma a garantir que a qualidade do código seja a melhor possível serão seguidos os padrões de codificação acima mencionados, assim como será utilizado o SonarLint v4.2 como forma de sinalização de erros de programação, bugs, erros estilísticos e *code smells*. Pretende-se que a soma destes seja inferior a 25 por cada 1000 linhas de código.

Webgrafia dos documentos investigados:

[Superfície de Ataque [EN]](https://en.wikipedia.org/wiki/Attack_surface)

[Uma Métrica de Superfície de Ataque [EN]](https://www.cs.columbia.edu/~wing/publications/ManadhataWing06.pdf)

[Medição de Ataques de Superfície em Software Empresarial [EN]](https://www.cs.columbia.edu/~wing/publications/ManadhataKarabulutWing08.pdf)

[Bugs per Line of Code Ratio [EN]](https://www.mayerdan.com/ruby/2012/11/11/bugs-per-line-of-code-ratio) - Ver McConnel, S. (1993). *Code Complete* Microsoft Press

**Avaliação de Complexidade:**

De forma a avaliar a complexidade do *software* desenvolvido será utilizada o método de análise de complexidade condicional desenvolvido por Thomas J. McCabe. Este método será utilizado com recurso ao *plugin* Metrics Reloaded v1.8. De acordo com o *National Institute of Standards and Technology* (NIST) o resultado por cada módulo não deverá exceder o valor 15.

Webgrafia dos documentos investigados:

[Complexidade Condicional [EN]](https://en.wikipedia.org/wiki/Cyclomatic_complexity)

[Uma Medida de Complexidade [EN]](http://www.literateprogramming.com/mccabe.pdf)

[Uma Metodologia de Testes Utilizando a Métrica Condicional [EN]](http://www.mccabe.com/pdf/mccabe-nist235r.pdf)