

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Pattern Based Usability Testing

Fernando José Dias



Mestrado Integrado em Engenharia Informática e Computação

Orientador: Ana Cristina Ramada Paiva (PhD)

27 de janeiro de 2017

© Fernando Dias, 2017

Pattern Based Usability Testing

Fernando José Dias

Mestrado Integrado em Engenharia Informática e Computação

Aprovado em provas públicas pelo Júri:

Presidente: Raul Fernando de Almeida Moreira Vidal

Vogal Externo: João Miguel Fernandes

Orientador: Ana Cristina Ramada Paiva

27 de janeiro de 2017

Resumo

Hoje em dia é bastante comum a interação com sistemas de *software* através de interfaces gráficas (*GUIs*). E também possível testar aplicações através da interface gráfica de forma manual ou automática simulando as ações do utilizador e avaliando os resultados obtidos através da *GUI*.

PBGT (Pattern Based GUI Testing) é uma abordagem de teste de *software* através da interface gráfica com o utilizador baseada em modelos, que automatiza e sistematiza o processo de teste. O *PBGT* testa comportamento recorrente para o qual possui técnicas genéricas de teste (padrões de teste) capazes de testar diferentes aplicações após um processo de configuração.

O tema desta dissertação consiste na implementação de uma extensão ao *PBGT*, criando uma aplicação semelhante a esta ferramenta/abordagem com a possibilidade de realizar testes de usabilidade. Este trabalho inicia com um levantamento de padrões de usabilidade para os quais se constroem soluções genéricas de testes, com o objetivo de testar diferentes aplicações de *software* e, com isso, detetar problemas de usabilidade num *website*, permitindo avaliar a sua interação com o utilizador. O objetivo é estender a linguagem *PARADIGM* com padrões de teste de usabilidade e estender o ambiente atual do *PBGT*, permitindo a construção de modelos para testes de usabilidade. No fim da implementação, o utilizador poderá construir o seu padrão de teste com as configurações da plataforma *web* que pretender testar, conseguindo obter um resultado para o teste, que poderá detetar problemas de usabilidade.

Um dos aspetos inovadores deste tema são os novos padrões de teste e a ferramenta que os executa. Esta extensão permitirá testar alguns aspetos de usabilidade automaticamente através da interface gráfica do utilizador. Este projeto insere-se na área da Engenharia de Software, mais concretamente, dos Testes de Software.

Palavras-chave: Interfaces do utilizador gráficas, Interfaces do utilizador, Testes de usabilidade, *PBGT*, Engenharia de *Software*, Testes de *Software*.

Abstract

Nowadays it is common the interaction with software systems through graphical user interfaces (GUIs). It is also possible to test applications through the graphical interface, manually or automatically, simulating the actions of the user and evaluating the outcomes obtained through the GUI.

Pattern Based GUI Testing (PBGT) is a software testing approach through the graphical user interface based on models, which automates and systematizes the process of the test. PBGT tests the recurring behavior for which it has testing generic techniques (test patterns) able to test different applications after a process of configuration.

The subject-matter of this dissertation consists on the implementation of an extension to the PBGT, creating a similar tool with the possibility to perform usability tests. This report will begin to explore the usability patterns for which it will find generic testing solutions afterwards to test different software applications and thereby, detect usability problems on a website, allowing to evaluate its interaction with the user.

The main objective is to extend the PARADIGM language with usability testing patterns and to extend the current environment of PBGT, allowing building models to usability tests. After the implementation, the user will be able to build his own testing pattern with the settings of the web platform he plans to test, managing to obtain a result for the test, which can detect usability problems.

One of the innovative aspects of this issue are the new testing patterns and the tool that runs them. This extension will allow the testing of some usability aspects automatically through the graphical user interface. This project is part of Software Engineering's field, more specifically, Software Testing.

Keywords: Graphical User Interface, User Interfaces, Usability Testing, PBGT, Software Engineering, Software Testing.

Agradecimentos

Em primeiro lugar, gostaria de agradecer à minha orientadora, a professora Ana Paiva, por toda a disponibilidade e pelo modo como me acompanhou, encorajou e motivou ao longo do tempo utilizado na realização deste projeto.

A todos os meus amigos e colegas, pela forma como sempre tentaram dar-me motivação e força, mostrando sempre que era possível atingir os objetivos com sucesso.

À minha mãe, Rosa, ao meu pai, Fernando e ao meu irmão, Pedro, que sempre foram pacientes e sempre apoiaram em tudo, acreditando em mim e tentando sempre ajudar-me em tudo o que puderam.

Conteúdo

Introdução.....	1
1.1 Contexto/Enquadramento.....	1
1.2 Projeto	1
1.3 Motivação e Objetivos	2
1.4 Estrutura da Dissertação.....	3
Testes de Usabilidade.....	5
2.1 Plataformas automatizadas.....	6
2.1.1 Análise da interação com o utilizador	6
2.1.1.1 <i>Usaproxy</i>	6
2.1.1.2 <i>m-pathy</i>	8
2.1.1.3 <i>Web Usability Probe (WUP)</i>	8
2.1.1.4 <i>CrazyEgg</i>	10
11	
11	
2.1.1.5 <i>WebTANGO</i>	11
2.1.1.6 <i>EyeTracking</i>	12
2.1.2 <i>A/B Testing</i>	13
2.1.2.1 <i>AttrakDiff</i>	14
2.1.2.2 <i>WaPPU</i>	15
2.1.2.3 <i>Optimizely</i>	15
2.2 Métodos Baseados em Métricas.....	16
2.2.1 <i>USherlock</i>	16
2.2.2 <i>W3TOUCH</i>	16
2.3 Métodos não automatizados	17
2.3.1 Revisões	17
2.3.1.1 Avaliação Heurística	17
2.3.1.2 <i>Cognitive walkthrough evaluation</i>	18
2.3.1.3 Análise de ações	18
2.3.2 Participação do utilizador.....	18
2.3.2.1 Testes do utilizador	18
2.3.2.2 Questionários	19

Conteúdo

2.3.3	Heurísticas de Nielsen.....	19
2.3.4	Metodologia <i>CLEAR</i>	21
2.3.5	<i>INUIT</i>	21
2.4	Boas práticas de usabilidade.....	22
2.5	Resumo.....	24
Abordagem.....		26
3.1	PBGT.....	26
3.2	<i>PBUT</i>	28
3.3	Modelo da linguagem.....	29
3.4	Cenários de utilização.....	30
3.5	Padrão de usabilidade implementado.....	36
3.6	Definição formal de padrão.....	37
3.7	Formato dos padrões.....	38
3.8	<i>Reachability UI Test Pattern</i>	38
3.9	Detalhes de implementação.....	41
3.10	Detalhes de execução.....	44
3.11	Resumo e Conclusões.....	45
Casos de Estudo.....		48
4.1	Ambiente Experimental.....	48
4.2	Formato dos Resultados.....	49
4.3	Caso de Estudo I – Sigarra.....	49
4.3.1	Testes Executados.....	50
4.3.2	Análise aos Resultados.....	51
4.4	Caso de estudo II – <i>University of Oxford</i>	52
4.4.1	Teste Executados.....	53
4.4.2	Análise aos resultados.....	53
4.5	Caso de Estudo III – <i>Website</i> da Samsung Portugal.....	54
4.5.1	Testes executados.....	55
4.5.2	Análise aos Resultados.....	55
4.6	Resumo e Conclusões.....	56
Conclusões e Trabalho Futuro.....		58
5.1	Satisfação dos Objetivos.....	58
5.2	Trabalho Futuro.....	59
Referências.....		60

Lista de Figuras

Figura 1: Mapa do rasto do rato gravado pelo HTTP proxy	7
Figura 2: Display de uma sessão registada em vídeo pelo m-pathy	8
Figura 3: Lista de tarefas de um teste de usabilidade no WUP	9
Figura 4: Linhas do tempo obtidas com o WUP, após a realização do texto	10
Figura 5: <i>ScrollMap</i> obtido através do <i>CrazyEgg</i>	11
Figura 6: <i>Confetti</i> obtido através do <i>CrazyEgg</i>	11
Figura 7: Mapa obtido através do <i>EyeTracking</i>	12
Figura 8: Resultado da avaliação com <i>AttrakDiff</i>	14
Figura 9: Resultados obtidos através do <i>Optimizely</i>	15
Figura 10: Critérios utilizados na avaliação heurística	17
Figura 11: <i>Cognitive walkthrough evaluation</i>	18
Figura 12: <i>Usability guidelines</i> e lista de <i>guidelines</i> de navegação e arquitetura de informação (<i>IA</i>)	23
Figura 13: Fases do processo de teste no PBGT	28
Figura 14: Ambiente da aplicação	29
Figura 15: Meta-modelo da linguagem	30
Figura 16: Casos de uso	31
Figura 17: Exemplo de árvore de navegação	41
Figura 18: Modelo de teste	44
Figura 19: Formulário de preenchimento das variáveis	45
Figura 20: Formato dos Resultados	49
Figura 21: Sigarra na FEUP	50
Figura 22: <i>Website</i> da Universidade de Oxford	52
Figura 23: <i>Website</i> da Samsung Portugal	54

Lista de Tabelas

Tabela 1: Heurísticas de Nielsen	20
Tabela 2: Tabela de severidade	20
Tabela 3: Tabela de frequência	20
Tabela 4: Fatores de usabilidade utilizados no <i>INUIT</i> .	21
Tabela 5: Caso de uso UC01 - criação do projeto	32
Tabela 6: Caso de uso UC02 - criação do nó	33
Tabela 7: Caso de uso UC03 - criação da relação	34
Tabela 8: Caso de uso UC04 - configuração dos dados de teste	35
Tabela 9: Caso de uso UC05 - executar o teste	36
Tabela 10: Descrição da máquina utilizada nos testes	48
Tabela 11 - Resumo dos dados de entrada do caso de estudo I	51
Tabela 12: Resumo dos resultados dos testes do caso de estudo I	51
Tabela 13: Resumo dos dados de entrada do caso de estudo II	53
Tabela 14: Resumo dos resultados do caso de estudo II	53
Tabela 15: Resumo dos dados de entrada do caso de estudo III	55
Tabela 16: Resumo dos resultados do caso de estudo III	55

Abreviaturas e Símbolos

<i>API</i>	<i>Application Programming Interface</i>
<i>CLEAR</i>	<i>Cheap, Loading speed, Efficient, Accurate and Reliable Testing</i>
<i>CPU</i>	<i>Central Processing Unit</i>
<i>FCT</i>	<i>Fundação para a Ciência e Tecnologia</i>
<i>FEUP</i>	<i>Faculdade de Engenharia da Universidade do Porto</i>
<i>GUI</i>	<i>Graphical User Interface</i>
<i>GUIs</i>	<i>Grafical User Interfaces</i>
<i>HTML</i>	<i>HyperText Markup Language</i>
<i>HTTP</i>	<i>HyperText Transfer Protocol</i>
<i>IA</i>	<i>Information Architecture</i>
<i>IDE</i>	<i>Integrated Development Environment</i>
<i>INUIT</i>	<i>The Interface Usability Instrument</i>
<i>MSc</i>	<i>Master of Science</i>
<i>PBGT</i>	<i>Pattern Based GUI Testing</i>
<i>PBUT</i>	<i>Pattern Based Usability Testing</i>
<i>PARADIGM</i>	<i>Domain-Specific Language built for Pattern-Based Graphical User Interface Testing</i>
<i>PARADIGM-ME</i>	<i>PARADIGM Modeling Environment component</i>
<i>PARADIGM-RE</i>	<i>PARADIGM Reverse Engineering component</i>
<i>PARADIGM-TE</i>	<i>PARADIGM Test case Execution component</i>
<i>PARADIGM-TG</i>	<i>PARADIGM automated Test case Generation component</i>
<i>SiFEUP</i>	<i>Sistema de Informação da Faculdade de Engenharia da Universidade do Porto</i>
<i>SIGARRA</i>	<i>Sistema de Informação para Gestão Agregada dos Recursos e dos Registos Académicos</i>
<i>UI</i>	<i>User Interface</i>
<i>UML</i>	<i>Unified Modeling Language</i>
<i>URL</i>	<i>Uniform Resource Locator</i>
<i>WaPPU</i>	<i>“Was that Page Pleasant to Use?”</i>
<i>WUP</i>	<i>Web Usability Probe</i>

Capítulo 1

Introdução

Neste primeiro capítulo é apresentado o contexto do trabalho de investigação, o projeto a desenvolver, o motivo que o faz surgir e quais os objetivos que deve atingir.

1.1 Contexto/Enquadramento

As interfaces gráficas (*GUIs*) têm vindo a ganhar extrema importância na interação entre o utilizador e a plataforma que utiliza. Nas aplicações *web*, é cada vez mais importante que esta interação seja facilitada, isto é, que o utilizador consiga utilizar a aplicação de uma forma intuitiva e simples, sem que seja necessária uma aprendizagem prévia.

A usabilidade do sistema, ou seja, a forma como o utilizador consegue realizar as tarefas no sistema de forma rápida e eficiente, é, portanto, uma característica que assume relevo no desenvolvimento desse *software*. Testar a usabilidade num sistema, é um processo algo demorado e que por vezes envolve a ajuda de terceiros, sendo um processo complicado e bastante custoso. É com base nos testes de usabilidade que surge este projeto, “*Pattern Based Usability Testing*”, permitindo automatizar o processo de teste de usabilidade num *website*.

1.2 Projeto

O projeto “*Pattern Based Usability Testing*”, é uma extensão ao projeto já existente, “*Pattern Based GUI Testing*” (*PBGT*) criando uma ferramenta semelhante que permita realizar testes de usabilidade a uma aplicação *web*.

PBGT é um projeto de investigação liderado pela Faculdade de Engenharia da Universidade do Porto (FEUP) e com financiamento da Fundação para a Ciência e Tecnologia

(FCT). Consiste numa ferramenta de automatização do processo de testes sobre *GUIs*, utilizando a técnica de teste baseado em modelos, visando a construção de *software* de maior qualidade. “*Pattern Based Usability Testing*”, pretende estender as funcionalidades desta ferramenta, construindo uma aplicação que permita realizar testes de usabilidade numa aplicação *web*. O objetivo é construir padrões de teste de usabilidade genéricos, que possam ser aplicados nos *websites* em geral.

1.3 Motivação e Objetivos

Como já foi referido anteriormente, o processo de teste de usabilidade a um *website* é um processo demorado e complicado, sendo que por vezes é necessário a ajuda de pessoas exteriores ao projeto para ajudar na avaliação da interação que a aplicação faz com o utilizador. É, assim, bastante importante automatizar este processo de teste surgindo o projeto “*Pattern Based Usability Testing*”. Tendo em conta que o objetivo deste é o teste de usabilidade dos diversos *websites* e não de um tipo de *website* em específico, então é importante a construção de padrões de teste à usabilidade genéricos e que possam ser aplicados em geral. Assim, serão construídos modelos de teste associados a estes padrões. Este projeto deve construir uma ferramenta com as seguintes funcionalidades:

- Conceção e desenvolvimento de estratégias de teste de usabilidade genéricas para testar qualquer *website*;
- Implementação de um processo de configuração para definir dados de entrada e que permita que a estratégia de teste seja usada em qualquer *website*;
- Execução dos testes.

Tendo como objetivo principal atingir estas funcionalidades, surgem diversas questões, entre as quais:

- “Que padrões de teste deverei implementar para cobrir a maior quantidade possível de problemas de usabilidade frequentes em *websites*?”;
- “Que padrões serão genéricos e possíveis de aplicar em todos os *websites* em geral?”;
- “Que tipos de testes de usabilidade a *websites* são realizados e como transformar esses testes num modelo genérico e aplicável à generalidade dos *websites*?”;
- “Que tipo de parâmetros são variáveis e importantes fornecer na realização dos testes?”.

Com o objetivo de construir a aplicação com as funcionalidades pretendidas, foi criado o padrão de usabilidade “*Reachability Test Pattern*”, o qual permite avaliar a importância das páginas conforme a facilidade de acesso por parte de um utilizador. Este padrão consegue

verificar se uma determinada página de um website é atingida após um número máximo de cliques escolhido pelo utilizador conforme a importância da página que se pretende atingir.

1.4 Estrutura da Dissertação

Para além da introdução, esta dissertação contém mais 4 capítulos. No capítulo 2, é descrito o estado da arte, sendo referidos vários métodos e ferramentas para realizar testes de usabilidade usados na atualidade, sendo apresentados trabalhos relacionados. No capítulo 3, é referida a abordagem, sendo dada uma visão clara do problema e das perspetivas de solução, que além de explicarem mais detalhadamente o problema, fornecem perspetivas e possíveis soluções que poderão resolvê-lo. O capítulo 4 consiste nos casos de estudo, sendo apresentados os resultados da utilização da aplicação, bem como a análise desses resultados, confirmando as potencialidades do projeto implementado. O capítulo 5 consiste nas conclusões e trabalho futuro, sendo apresentadas as conclusões obtidas nesta dissertação, bem como a satisfação dos objetivos propostos e trabalho futuro.

Introdução

Capítulo 2

Testes de Usabilidade

Este capítulo consiste no estudo do estado da arte, tendo sido realizadas pesquisas sobre aplicações e métodos que podem ser usados na realização de testes de usabilidade em aplicações *web*. Foi feita uma divisão dessas aplicações e/ou métodos em 3 tipos: automatizadas, baseadas em métricas e não automatizadas, sendo depois realizada uma análise a estes tipos de ferramentas, fornecendo uma visão geral sobre o funcionamento de cada uma. É também realizada uma pesquisa relativamente a boas práticas de usabilidade, uma vez que o foco deste projeto são os testes de usabilidade. No fim do capítulo, são apresentadas algumas reflexões e conclusões acerca do estudo que foi feito.

Segundo a norma em [1], usabilidade pode ser definida como a “medida em que um sistema, produto ou serviço pode ser utilizado por utilizadores específicos para garantir que os objetivos especificados são atingidos com eficácia, eficiência e satisfação num contexto específico de uso.” Usabilidade trata de garantir a satisfação do utilizador após uso do sistema, fazendo com o que o sistema consiga atingir os objetivos para que foi concebido. Ora, para garantir que um produto consiga realizar as suas tarefas de forma eficaz e eficiente é necessário que este seja “bom” do ponto de vista de usabilidade. Assim, fazer testes de usabilidade a *websites*, permite avaliar se esse tipo de sistema consegue satisfazer os utilizadores e fazer com que estes consigam realizar eficientemente as tarefas que pretendem. Apontando erros comuns de usabilidade, é possível que o sistema possa corrigi-los e, portanto, melhorar a sua utilização e usabilidade.

2.1 Plataformas automatizadas

Processos automatizados são processos que testam usabilidade e que conseguem, de uma forma automática, devolver resultados acerca da avaliação de usabilidade. Existem ferramentas que envolvem a análise das ações do utilizador, sendo recolhidas informações como a velocidade e cliques no rato, cliques no teclado, bem como o tempo demorado na realização de uma tarefa. Há assim uma avaliação da utilização, nomeadamente, do tempo ou do modo como o utilizador movimenta o rato, permitindo inferir confusão, facilidade ou dificuldade na realização de determinada tarefa. Existem também plataformas que realizam “*A/B Testing*”, método que compara duas versões do *website* e permite avaliar qual delas é melhor do ponto de vista da usabilidade. De seguida, apresentam-se exemplos de aplicações automatizadas utilizados em testes de usabilidade.

2.1.1 Análise da interação com o utilizador

Da análise direta da utilização de um *website*, é possível obter informações relacionadas com os movimentos e cliques do rato, cliques no teclado e tempo que o utilizador demora a realizar determinada tarefa. Após recolha dessas informações, é feita uma análise com o objetivo de descobrir problemas de usabilidade associados ao *website* analisado, permitindo que numa próxima versão, este possa melhorar do ponto de vista da usabilidade. Apresentam-se de seguida várias ferramentas que permitem fazer uma análise à interação do utilizador com o *website*.

2.1.1.1 *Usaproxy*

Usaproxy [6] é uma aplicação que permite fazer rastreio da utilização de um *website* usando uma abordagem *HTTP proxy*, que permite registar as atividades do utilizador de forma não intrusiva, isto é, sem que o utilizador consiga perceber que as suas ações estão a ser visualizadas e registadas. Trata-se de um processo de avaliação remoto e que não altera a forma como o utilizador interage com o *website*. É uma aplicação compatível com a generalidade dos *browsers*, que não necessita de ser instalada pelo utilizador e que permite monitorizar e registar as tarefas realizadas pelo utilizador com precisão, resultando uma lista do que o utilizador realmente fez ao realizar as tarefas predefinidas. A partir da lista será possível construir o *website* de forma a melhorar a interação do utilizador com o mesmo numa futura versão. Há 4

tipos de ações que é importante analisar de forma a que a avaliação consiga encontrar problemas de usabilidade, que constituirão melhorias importantes no futuro:

- Métricas de comportamento de navegação, nomeadamente, transição entre páginas no *website* e principais páginas de entrada e saída;
- Métricas baseadas no tempo, como por exemplo, o tempo médio que o utilizador passa numa página, que o cursor do rato passa por determinado elemento, permitindo avaliar hesitações ou facilidade na realização de determinada tarefa e o tempo de utilização de elementos, frequência de cliques e de digitação;
- Ações do utilizador na página *web*, relacionadas com o rato, como posição absoluta do rato, elementos por onde o cursor do rato passou e zonas onde clicou;
- Outras ações como o *scroll* da página, redimensionamento e outras teclas usadas.

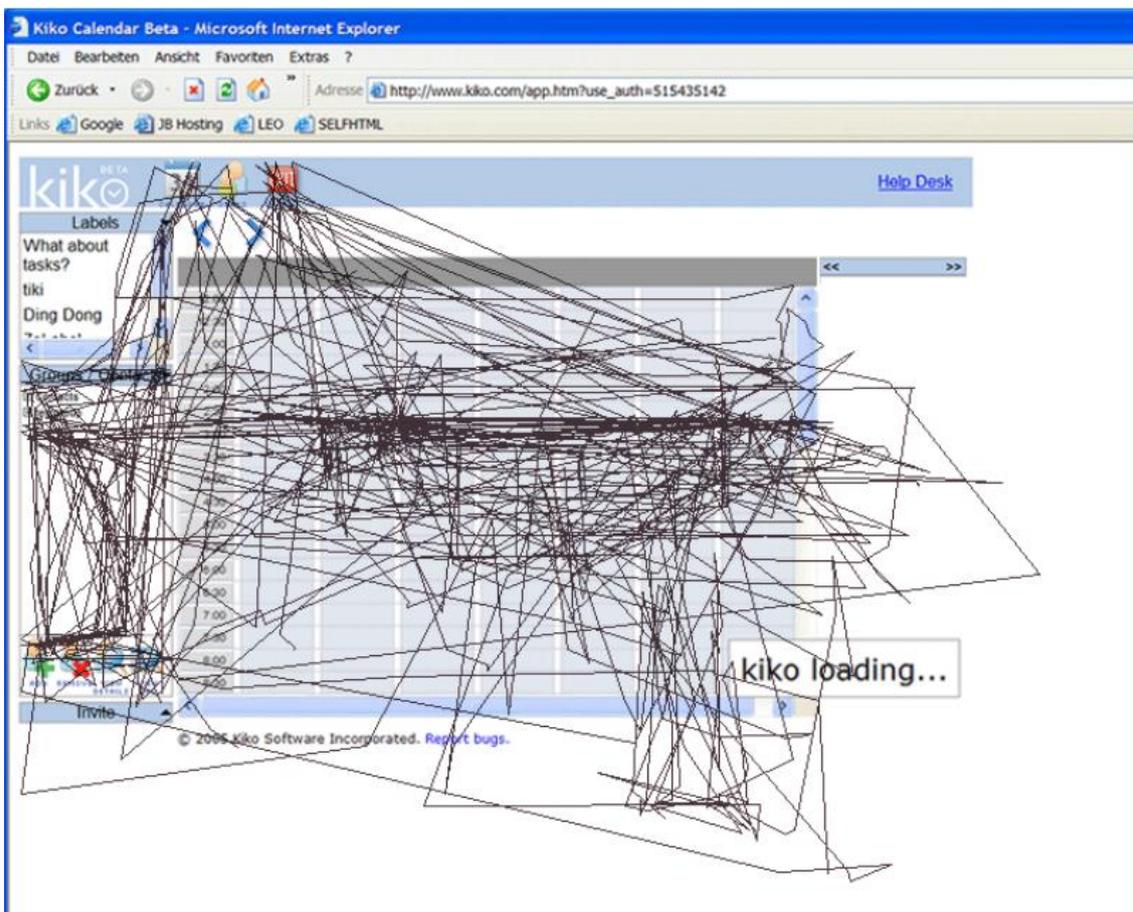


Figura 1: Mapa do rasto do rato gravado pelo HTTP proxy

2.1.1.2 *m-pathy*

*m-pathy*¹ é um ferramenta que utiliza o conceito anterior fazendo uma avaliação qualitativa do comportamento do utilizador. Permite registar os movimentos e cliques do rato, o *scroll* e entrada de teclado, sendo que cada um desses parâmetros pode ser interpretado, posteriormente, através de vídeo ou *heatmap*, ilustrando a interação que foi realizada com o utilizador. Na figura 2, é possível ver o rasto do rato na página *web*, de acordo com os movimentos do rato feitos pelo utilizador.



Figura 2: Display de uma sessão registada em vídeo pelo *m-pathy*

2.1.1.3 *Web Usability Probe (WUP)*

O *Web Usability Probe* [8] consiste numa ferramenta que permite testes de usabilidade automáticos remotamente, que sendo baseada num servidor *proxy*, não necessita de instalação. O servidor *WUP* atua como servidor de usabilidade, permitindo a quem avalia introduzir informações úteis para a realização do teste, como a lista de tarefas e eventos a realizar, entre outros. Após realizarem os testes de usabilidade, os avaliadores podem personalizar as suas representações, a partir dos registos, selecionando que ações devem ser representadas em relatórios, a fim de identificar possíveis problemas de usabilidade. Quando um novo teste de utilizador começa, os responsáveis pela avaliação podem aceder à ferramenta, com o objetivo de fornecer aos utilizadores informações que ajudem na realização do teste. Em seguida, quem avalia pode fazer o papel de utilizador, executando as tarefas de uma forma otimizada e sem erros. A tarefa otimizada é definida durante a fase de planeamento do teste, sendo que deve ser

¹ <http://www.m-pathy.com/cms>

realizada de forma natural e sem ações inúteis. Os utilizadores apenas necessitam de começar a usar o *website* a ser avaliado através do próprio *link* disponível no *proxy*. Quando começam a tarefa, os utilizadores são informados da tarefa que vão realizar, assim como, quando a terminam, devem aceder à caixa de diálogo automaticamente gerada pela realização da tarefa. De seguida, uma nova tarefa será indicada até que seja atingido o fim do teste. Todos os dados registados são enviados de forma assíncrona para o servidor enquanto os utilizadores navegam entre páginas. O desenvolvimento de uma ferramenta em *proxy*, considerando os dados do lado do cliente, leva a diferentes desafios, nomeadamente, a identificação dos elementos com que os utilizadores irão interagir, a gestão da identificação de elementos quando a página está a ser alterada dinamicamente e como gerir os dados quando os utilizadores navegam entre páginas.

São criados para cada teste um modo ótimo de o fazer, que servirá de termo de comparação com a forma como o utilizador realiza a tarefa. Na lista de tarefas de teste, é indicado o nome, a descrição, o avaliador que a criou e o número de tarefas que devem ser realizadas. Para cada tarefa, é indicado o nome, descrição, o URL inicial, se depende de outra tarefa, entre outros parâmetros, como se pode ver na figura 3.

You are here: [Evaluations](#) > [Mobipocket evaluation](#) > Tasks

Task List for Evaluation "Mobipocket evaluation"

<input type="checkbox"/> Name	Description	Starting url	Is skipable	Depends on	Actions
<input type="checkbox"/> Search	Search for books of the author Jules Verne. Then, at the result list, add first Jules Verne's book to the cart.	http://www.mobipocket.com/en/HomePage/default.asp?Language=EN	No		
<input type="checkbox"/> Browsing the authors list	Browse the author list to check books available written by Umberto Eco.		No		
<input type="checkbox"/> Accessing the contact form	Find the contact form in order to send a message to the website. You don't have to send a message. Just reach the contact form.		No		
<input type="checkbox"/> Checking out	Access the cart and proceed to checkout. You don't have to login nor insert credit card information.		Yes		

4 tasks

Delete recording

Figura 3: Lista de tarefas de um teste de usabilidade no WUP

Os eventos suportados pelo *WUP* são eventos padrão, eventos *jQuery*, toque, gestos e eventos de acelerómetro, presentes na *API* do Safari. O conjunto de eventos pode ser agrupado por tipo, por exemplo, acelerómetro, teclado, rato e toque. Podem ser considerados ainda, eventos relacionados com formulários, eventos relacionados com o sistema e eventos personalizados, que podem ser a composição dos vários tipos de eventos simples existentes.

Sempre que o utilizador realize um teste, o avaliador tem acesso a representações gráficas registadas. As linhas do tempo podem ser úteis, havendo uma linha do tempo para cada tarefa,

que constitui uma representação gráfica que servirá de comparação com o comportamento adequado definido pelo avaliador. Assim, a primeira linha do tempo, é a representação do registo da tarefa ótima. Nessas linhas de tempo estão incluídas repetições de um certo evento, nomeadamente, movimentos do rato repetidos ou cliques em zonas “não-clicáveis”.

Quando se visualizam as linhas do tempo, é possível ajustar o *zoom* de forma a visualizar os eventos mais básicos, que estão com cores diferentes conforme o tipo de evento.

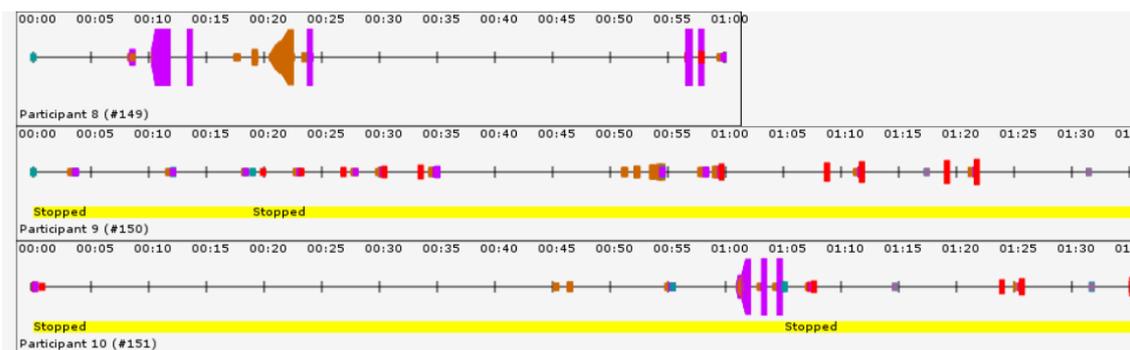


Figura 4: Linhas do tempo obtidas com o WUP, após a realização do texto

2.1.1.4 CrazyEgg

CrazyEgg² é uma plataforma que regista os eventos do rato, permitindo a visualização de mapas das regiões mais populares de cliques numa página. Esta ferramenta contempla 4 tipos de mapas: *HeatMap*, *ScrollMap*, *Overlay* e *Confetti*.

HeatMap é uma representação dos cliques com o rato feitos pelos utilizadores, que permite saber quais as zonas onde devem estar as principais funcionalidades do sistema. *ScrollMap* (figura 5) ajuda a perceber até onde o utilizador faz *scroll*, permitindo conhecer onde o utilizador abandona a página, o que faz com que uma solução possa ser acrescentar nessa zona conteúdos que possam interessar ao utilizador. *Overlay* é um relatório que dá a conhecer os cliques em cada elemento da página, podendo o *website* ter mais cliques que dão mais dinheiro e menos dos que dão menos. *Confetti* (figura 6) é um mapa que permite distinguir todos os cliques do *website*, segmentados por fontes de referência, termos de pesquisa, entre outros. Sabendo onde há um maior tráfego de cliques, é possível rentabilizar o *website* com pouco esforço.

² <https://www.crazyegg.com/overview>



Figura 5: *ScrollMap* obtido através do *CrazyEgg*

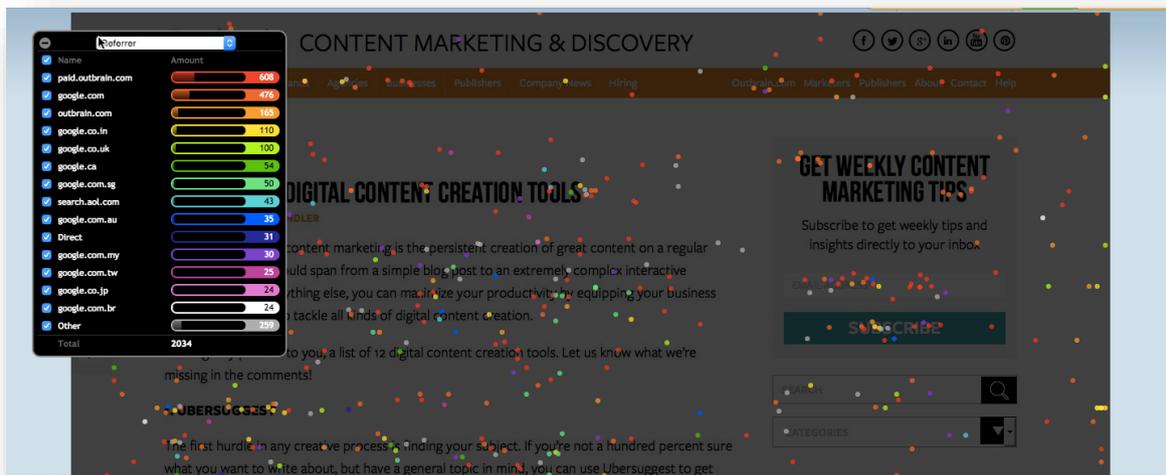


Figura 6: *Confetti* obtido através do *CrazyEgg*

2.1.1.5 *WebTANGO*

*WebTANGO*³ é uma aplicação que aplica a simulação de Monte Carlo e métodos de recuperação de informação para prever o comportamento do utilizador e os percursos de navegação. O objetivo desta metodologia é acelerar e melhorar o processo de *design* de *websites*. Esta ferramenta contempla um conjunto de medidas quantitativas da estrutura da página web, que permite prever se o *site* é considerado rico ou pobre em termos de design, com

³ <http://webtango.berkeley.edu>

alta precisão. Estas medidas são usadas para criar perfis estatísticos e *sites* de qualidade e aplicá-los ao *design* já existente, mostrando como o *design* pode ser melhorado.

2.1.1.6 EyeTracking

*EyeTracking*⁴ é um processo que mede a atividade do olho, bem como a sua reação a estímulos. Os dados são recolhidos com o uso de um *EyeTracker*, ligado ao computador. O *EyeTracker* tem dois componentes comuns, a fonte de luz, geralmente infravermelha, e direcionada diretamente para o olho e a câmara que acompanha o reflexo da fonte e os recursos oculares, nomeadamente, a pupila. Estes dados são usados para conseguir obter informações como rotação do olho, direção do olhar, frequência do “pisar” dos olhos e mudanças no diâmetro da pupila. Os dados, após agregação, são gravados num ficheiro que é compatível com o *software* usado para análise do *EyeTracking*, como por exemplo, o *Eyeworks*.

Através dos mapas criados pelo *EyeTracking*, é possível conhecer as zonas do *website* para as quais os utilizadores mais olham, podendo-se eliminar elementos distratores ou colocar nas zonas mais visíveis funcionalidades mais importantes. Na figura 7, podemos ver um mapa obtido através deste processo.

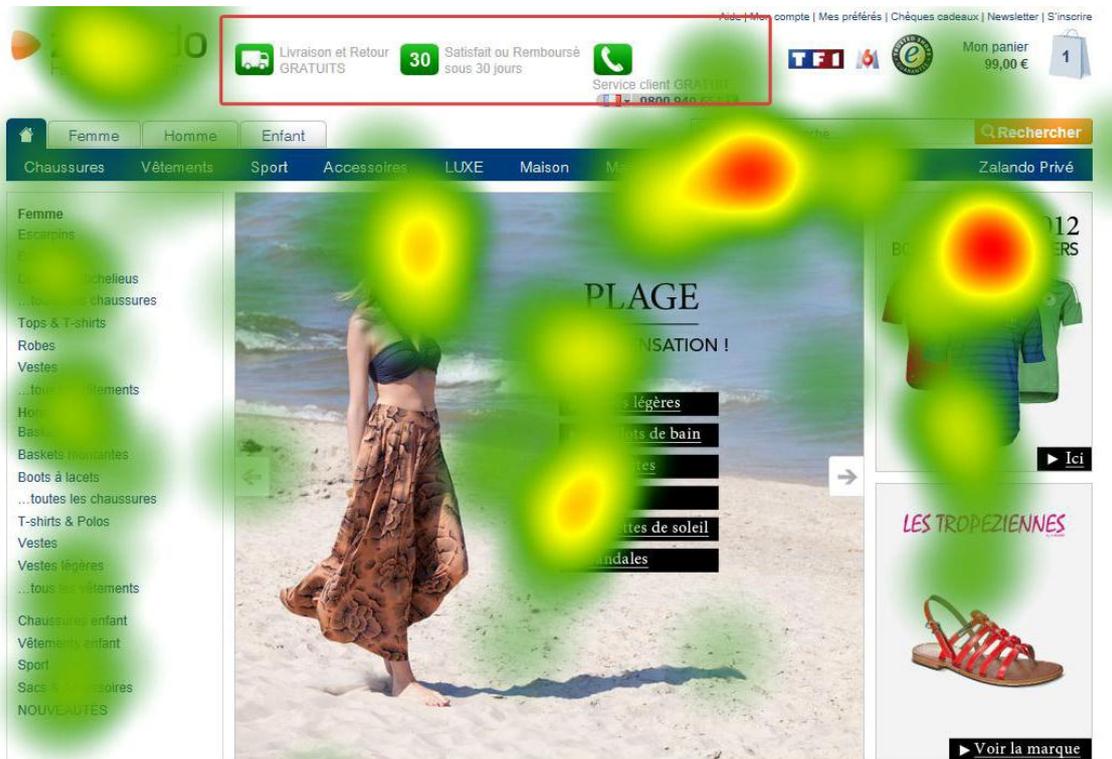


Figura 7: Mapa obtido através do *EyeTracking*

⁴ <http://www.eyetracking.com/About-Us/What-Is-Eye-Tracking>

2.1.2 A/B Testing

*A/B Testing*⁵ é um método que compara duas versões de um mesmo *website* e as compara com o objetivo de determinar qual delas têm um funcionamento melhor. Este método contempla duas versões de uma página *web*, a versão *control*, que é a versão original e a versão *variation*, que é a versão onde foram introduzidas alterações, que podem ser até muito simples, como a alteração de um botão apenas. Então, metade das pessoas vê uma versão e a outra metade vê a outra. O compromisso de cada grupo de pessoas é medido e recolhido para um painel de análise e analisado com um meio estatístico, permitindo avaliar se a mudança teve um efeito positivo, negativo ou não no comportamento do visitante. O teste A/B permite que as equipas de desenvolvimento vão fazendo as alterações que considerem ser benéficas para o *website*, permitindo ir melhorando-o ao longo do tempo, começando a ganhar experiência conforme as melhorias que vão sendo introduzidas e o feedback dos utilizadores.

O processo de teste realiza-se em 6 etapas:

- 1) Recolha de dados – A análise do próprio avaliador, fornece informações sobre por onde começar a otimizar o *website*. Esta permitirá começar com áreas de alto tráfego do *site*, permitindo uma recolha de dados rápida. A procura por páginas com taxas de conversão baixas ou com altas taxas de desistência podem ser páginas que podem ser melhoradas.
- 2) Identificar os objetivos – Os objetivos são as métricas que serão usadas para avaliar se a página de variação será bem-sucedida ou não.
- 3) Gerar hipóteses – Após definição dos objetivos, é possível começar a criar hipóteses que possam vir a melhorar a página *web* e definir prioridades entre as hipóteses conforme a dificuldade de implementação e o impacto que é expectável.
- 4) Criar variações – Utilizando *software* de teste A/B (como o *Optimizely*), introduzir as alterações pretendidas no *website*, mesmo sendo pequenas alterações. Muitas ferramentas de teste A/B permitem a realização dessas alterações de forma simples e rápida.
- 5) Executar a experiência – Iniciando a experiência é necessário esperar visitantes para participar. Os visitantes do *site* vão ser direcionados aleatoriamente para uma das versões. Essa interação é medida e comparada para entender o desempenho dos utilizadores em cada uma das versões.
- 6) Análise de Resultados – Acabada a experiência, é necessário analisar cuidadosamente os resultados. O software regista os dados acerca das duas versões permitindo saber qual delas foi mais bem-sucedida no teste.

De seguida, apresentam-se ferramentas usadas para realização deste tipo de método.

⁵ <https://vwo.com/ab-testing/>

2.1.2.1 *AttrakDiff*

*AttrakDiff*⁶ ajuda a perceber a forma como os utilizadores classificam a usabilidade e *design* de um produto. A ferramenta suporta vários cenários, nomeadamente, uma avaliação individual do *website*, com base na participação de utilizadores, testes A/B, comparando duas interfaces e classificando-as, permitindo saber qual é melhor do ponto de vista de usabilidade e um estudo “antes-depois”, que faz a comparação entre a versão anterior do site e a versão após alterações. No âmbito deste capítulo, importa referir esta ferramenta pela realização dos testes A/B, a qual permite uma avaliação das duas versões, A e B, de forma separada e, depois, por comparação. É dada uma visão geral da perceção com que os utilizadores ficaram de cada versão. Usa um questionário para avaliar as dimensões pragmática e hedónica de ambas as versões do *website*. Na figura 8, é possível ver os resultados de um teste feito com o *AttrakDiff*, permitindo ver onde se situa cada uma das plataformas na avaliação feita por esta ferramenta. Nessa figura, é possível ver que a versão A, está num estado mais desejável do ponto de vista da usabilidade.



Figura 8: Resultado da avaliação com *AttrakDiff*

⁶ <http://attrakdiff.de/>

2.1.2.2 WaPPU

WaPPU [14] é uma ferramenta que suporta teste A/B e usa um total 27 aspetos bem definidos acerca da interação com o utilizador. Esta ferramenta mostra um questionário quando os utilizadores terminam de utilizar uma das plataformas que estava a ser testada e, no caso dos utilizadores aceitarem realizar esse questionário, irão avaliar o *website* segundo os sete itens de usabilidade do método *INUIT*, o qual será referenciado com pormenor mais à frente neste capítulo. Para cada item é dado um valor de -1, 0 ou +1, sendo que no fim o valor de usabilidade final estará entre -7 e +7. Assim, esse valor será considerado a medida quantitativa de usabilidade. Se nenhuma interface possui o questionário, a funcionalidade do *WaPPU* passa a ser a recolha de interações, para usar com as heurísticas de usabilidade. Caso haja uma interface com o questionário, *WaPPU* consegue aprender 7 modelos, um por cada item, baseado nas respostas que os utilizadores deram no questionário, sendo essas respostas guardadas no repositório desta ferramenta.

2.1.2.3 Optimizely

*Optimizely*⁷ é uma ferramenta que realiza testes A/B e que permite a criação das duas versões da página *web* de forma fácil e rápida, fornecendo mecanismos que facilitam a introdução de alterações no *website*. Após a criação de duas versões, consegue de acordo com o feedback do utilizador, organizar os resultados da interação com diversos filtros como, por exemplo, *browser*, dispositivo ou outra personalização segundo as informações do utilizador.

Na figura 9, é possível ver os resultados da análise a duas páginas, sendo que a primeira, a versão de variação, não introduziu melhorias e, na segunda, a variação introduziu melhorias.

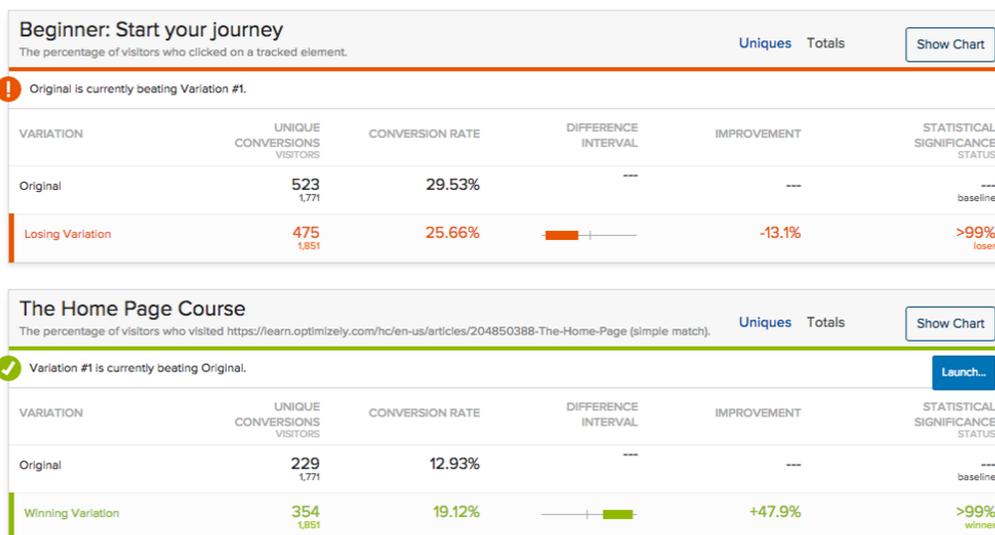


Figura 9: Resultados obtidos através do *Optimizely*

⁷ <https://www.optimizely.com/ab-testing/>

2.2 Métodos Baseados em Métricas

Métodos baseados em métricas fornecem métricas quantitativas para avaliação de páginas *web*. Contemplam ferramentas que fazem uma análise de aspetos espaciais e de *layout*. São métricas estáticas, uma vez que são baseadas no código *HTML*, fazendo uma análise a este, conseguindo obter alguns erros relacionados com o *design* da página. De seguida, são apresentadas três ferramentas que executam este tipo de testes.

2.2.1 *USherlock*

USherlock [16] é uma ferramenta que se baseia na estrutura da interface, analisando o que utilizador visualiza no ecrã. É um sistema automático que reconhece e analisa aspetos estáticos e dinâmicos da interface. O sistema faz uso de várias métricas, nomeadamente, as heurísticas de Nielsen e tem uma lista de 15 propriedades relativas à usabilidade dos aspetos visíveis de uma aplicação interativa, entre eles, a razão entre a altura e a largura do *website*, a relação entre o *background* e os *widgets*, a densidade, margens, alinhamento, tamanho, forma dos *widgets*, a cor do *background*, botões claros e reconhecíveis, entre outros. Para uma análise destes aspetos, é necessário a recolha dos elementos da interface para posterior análise e aplicação dessas propriedades usadas pela ferramenta. O processo de avaliação dá a cada *frame* da interface, uma lista de inconsistências e uma pontuação de 1 a 10, segundo a qualidade dessa *frame*.

2.2.2 *W3TOUCH*

W3TOUCH [17] é uma abordagem baseada em métricas para testar a adaptação de páginas *web* para dispositivos *touch*. Permite fazer uma avaliação das páginas *web* em dispositivos *touch*, havendo um conjunto de métricas específicas, relacionadas com o manuseamento do dispositivo que podem prejudicar o desempenho do *website*, em termos de usabilidade. Esta ferramenta vai permitir ajudar os *web designers* a entender o que poderão fazer para otimizar as páginas *web* nos dispositivos *touch*, uma vez que, essa otimização nem sempre é fácil. É necessário estender as métricas que avaliam as páginas *web* para computador, adaptando-as à realidade deste tipo de dispositivos. Esta adaptação contempla 3 passos: a monitorização da interação com o utilizador e recolha de dados relacionados com essa interação para cada métrica, implementação de técnicas de visualização para inspecionar os dados e a segmentação da interface em componentes críticos, tendo limites bem definidos para cada métrica e a implantação de um catálogo de adaptação, que fixa os problemas de *design* nos diferentes contextos.

2.3 Métodos não automatizados

Métodos não automatizados correspondem a metodologias utilizadas para avaliação de usabilidade de uma página *web*, com base em revisões feitas por *experts* na área da usabilidade ou numa ligação direta com o utilizador, situação que resulta numa interação direta com o utilizador, avaliando presencialmente ou através de questionários a sua experiência no *website*.

De seguida, apresentam-se essas metodologias de uma forma sintética e concisa.

2.3.1 Revisões

Revisões de usabilidade referem-se ao conjunto de técnicas de avaliação que se baseia na experiência dos revisores, os seus últimos projetos e a direção da avaliação que é seguida. O revisor baseia o seu processo numa *checklist*. O processo de avaliação depende de 3 métodos, avaliação heurística, *cognitive walkthrough evaluation* e análise de ações. Cada método tem um grau, sendo que a soma dos graus dos 3 métodos irá dar um resultado de 40 graus atingido dependendo das várias condições que irão ser referidas de seguida.

2.3.1.1 Avaliação Heurística

A avaliação heurística contempla uma equipa de revisores, o *design* da interface baseado na usabilidade e princípios baseados nas experiências passadas dos avaliadores. Na figura 10, é possível visualizar os critérios utilizados, sendo que cada padrão tem 2 pontos, podendo no máximo serem obtidos 20 pontos.

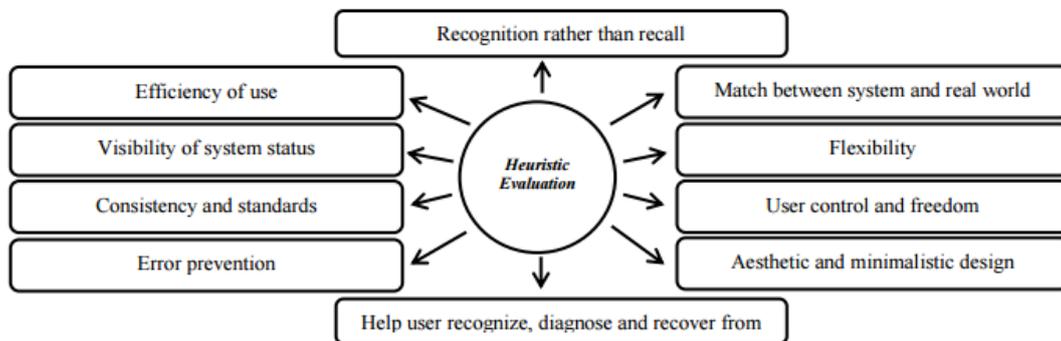


Figura 10: Critérios utilizados na avaliação heurística

2.3.1.2 *Cognitive walkthrough evaluation*

Esta é uma técnica de avaliação do *design* de uma interface, que corresponde a uma análise à facilidade de aprendizagem. Tem 4 passos com 4 questões, que podem ser visíveis na figura 11, sendo que há 4 níveis de avaliação para cada passo, sendo que o total da avaliação resultará em 16 graus, se todas as questões forem classificadas com o nível mais elevado.

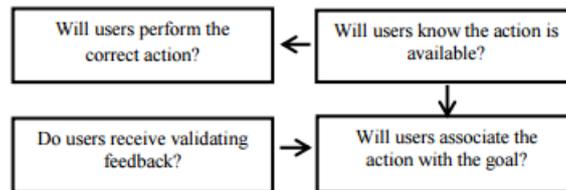


Figura 11: *Cognitive walkthrough evaluation*

2.3.1.3 *Análise de ações*

É uma análise quantitativa das ações para prever o tempo que é necessário para cada tarefa, baseado no tempo. É classificado com 4 graus e depende das 2 fases anteriores. No fim dessas 3 fases, o *website* foi classificado com uma pontuação de um total de 40 pontos.

2.3.2 *Participação do utilizador*

O objetivo da avaliação com intervenção da participação do utilizador é avaliar a facilidade de uso de um *website* para assegurar que o produto atinge nível alto de satisfação por parte do utilizador. Uma vez, que a satisfação do utilizador é o fator que leva ao sucesso de qualquer projeto, então é necessário envolver o utilizador na avaliação das interfaces, com vista a receber *feedback*, verificando se o *site* satisfaz o utilizador ou não. São usados 2 métodos para fazer esta avaliação, os questionários, que contemplam perguntas sobre a facilidade de uso do *website* e os testes do utilizador, em que são dados um conjunto de tarefas e recebido *feedback* do utilizador.

A pontuação total destes 2 métodos é de 40 graus, o que ficará esclarecido nas próximas duas subsecções.

2.3.2.1 *Testes do utilizador*

Testes do utilizador estão relacionados com a observação de comportamentos reais de uma amostra de utilizadores real. Visto que a usabilidade do *website* depende da satisfação do utilizador, a melhor maneira de perceber os índices de satisfação e os problemas de usabilidade

é contactar diretamente com as experiências de uso do utilizador. Durante as experiências, são recolhidos dados empíricos sobre a forma como os utilizadores realizam as tarefas, dados relacionados com o tempo de execução, número de erros e satisfação do utilizador. Depois do teste, esses resultados são interpretados. Este processo contém várias etapas, nomeadamente, a definição dos objetivos de teste, a definição da amostra de utilizadores que irá participar no teste, a identificação de cenário de uso, isto é, as tarefas de teste que irão ser realizadas e a recolha de informação e dados empíricos ao longo da experiência. Este método tem 5 critérios a serem avaliados e classificados, nomeadamente, a inexistência de erros e distrações, segurança, reconhecimento dos passos, habilidade de aprendizagem e velocidade. Cada critério tem 5 graus, perfazendo um total de 25 graus, se o *website* obtiver a classificação mais alta em todos estes parâmetros.

2.3.2.2 Questionários

Uma série de 15 questões é feita ao utilizador, com o objetivo de entender a usabilidade do *website* e a satisfação do utilizador. Após a distribuição dos questionários, os resultados são recolhidos e analisados para determinar os pontos em que a maior parte dos utilizadores não está satisfeito, sendo que a pontuação desta fase são 15 graus, sendo que cada pergunta vale 1 grau, que só é atribuído se a maioria dos utilizadores não encontrou problemas na tarefa abordada por essa questão.

2.3.3 Heurísticas de Nielsen

A avaliação heurística é um método de revisão em que 3 ou 5 especialistas fazem uma avaliação das interfaces seguindo princípios de *design* designados por heurísticas. Nas heurísticas de Nielsen devem ser executadas segundo os seguintes passos:

- 1) Cada avaliador trabalha na análise do *website*, individualmente, para determinar se as interfaces gráficas estão de acordo com as heurísticas de Nielsen (ver tabela 1). Após essa análise, é feita uma lista de problemas de usabilidade, sendo que a cada avaliador terá a sua própria lista.

Heurísticas de Nielsen	
1	Visibilidade do sistema
2	Mapeamento entre o sistema e o mundo real
3	Liberdade e controlo do utilizador
4	Consistência e padrões

Testes de Usabilidade

5	Prevenção de erros
6	Reconhecer ao invés de lembrar
7	Flexibilidade e eficiência de uso
8	<i>Design</i> estético e minimalista
9	Suporte para o utilizador reconhecer, diagnosticar e recuperar erros
10	Ajuda e documentação

Tabela 1: Heurísticas de Nielsen

- 2) Após a avaliação individual, os avaliadores juntam-se para elaborar uma única lista com problemas de usabilidade, obtida através do consenso entre os problemas identificados inicialmente. É necessário identificar a heurística que não é respeitada em cada problema de usabilidade.
- 3) A lista obtida na fase anterior é enviada para cada especialista sendo que, cada um, vai estimar a severidade do problema e a frequência com que ocorre, segundo as tabelas 2 e 3.

Nota	Severidade
0	Não é um problema de usabilidade de todo.
1	Problema estético, apenas deve ser resolvido se houver tempo.
2	Problema de usabilidade menor ao qual deve ser dada uma prioridade baixa.
3	Problema de usabilidade maior, que deve ser resolvido com prioridade alta
4	Problema de usabilidade catastrófico, que deve ser resolvido antes do lançamento do produto.

Tabela 2: Tabela de severidade

Nota	Frequência
0	< 1%
1	1-10%
2	11-50%
3	51-90%
4	> 90%

Tabela 3: Tabela de frequência

- 4) O último passo é calcular o quão o problema é crítico, o que depende da severidade e da frequência de cada problema de usabilidade. O especialista responsável por este passo deve fazer uma média dos resultados de cada um dos avaliadores para chegar a um resultado final.

2.3.4 Metodologia *CLEAR*

CLEAR [18] é uma metodologia usada para testar usabilidade em *websites*. Esta metodologia usa o código da página para julgar a velocidade de carregamento do *website* e usando o relatório dos utilizadores pode encontrar particularidades da interação dos utilizadores com a página *web*. A metodologia seleciona um grupo pequeno de pessoas e visualiza os utilizadores a realizarem as tarefas propostas, registando onde há falhas. Há uma observação direta do que levou o utilizador a falhar em determinada tarefa. Os problemas são reportados aos *designers* e uma nova versão do *website* é testada novamente, mas com outro grupo de pessoas. *CLEAR* é um método que testa se o *website* é preciso, confiável, eficiente e, além disso, ainda testa a velocidade da página. O objetivo é perceber a percepção dos utilizadores acerca da eficiência e facilidade de uso, sendo que os comentários e sugestões dos utilizadores são tidas em conta no resultado final do teste.

2.3.5 *INUIT*

INUIT [19] é um instrumento em que os itens formam uma pontuação de usabilidade. *INUIT* concentra-se em 7 itens, que poderão ser visualizados na tabela 4, e que têm o nível de abstração necessário para refletir o comportamento do utilizador. Este método foi realizado em 2 passos, o primeiro, em que foram revistas mais de 250 regras de usabilidade a partir das quais foram constituídas sob a forma de heurísticas e *checklists* e, o segundo, em que há um conjunto de entrevistas semiestruturadas com *experts*, o que permite avaliar o *website* com base nos parâmetros da tabela.

Fator de usabilidade	Regras relacionadas
Aparência estética	8
Quantidade de distração	6
Densidade de informação	6
Grau de Informação	6
Acessibilidade de conteúdo desejado	4
Legibilidade	5
Compreensibilidade	6

Tabela 4: Fatores de usabilidade utilizados no *INUIT*.

2.4 Boas práticas de usabilidade

Além das aplicações e métodos, que permitem avaliar a usabilidade em aplicações *web*, referidas nas secções anteriores, há que abordar as boas práticas de usabilidade, que permitem conhecer padrões que os *websites* devem adotar para que possam permitir aos seus utilizadores uma boa experiência de usabilidade, isto é, que possam navegar com facilidade e sem que tenha sido necessária uma experiência anterior.

Com o objetivo de implementar padrões de usabilidade genéricos, que possam ser utilizados na maioria das aplicações *web* foi realizada uma pesquisa que permitiu conhecer boas práticas de usabilidade, tendo sido encontradas no *site UserFocus* em [23], boas práticas de usabilidade relacionadas com a usabilidade da página inicial, orientação das tarefas, navegação e arquitetura da informação (*IA*), formulários de entrada de dados, confiança e credibilidade, qualidade do conteúdo escrito, layout e design visual da página, usabilidade da característica de pesquisa e ajuda, *feedback* e tolerância a erros. Assim, são várias as áreas que podem afetar a usabilidade, tendo sido estudados vários princípios a que uma aplicação *web* deve obedecer. E são várias as aplicações referidas em secções anteriores, que usam essas boas práticas como métricas para testar a usabilidade nas interfaces *web*. Na figura 12, é possível visualizar a lista de “*guidelines*” que foi utilizada, sendo possível visualizar as normas de usabilidade relativas à navegação, uma vez que, no contexto do projeto é possível a simulação da navegação, sem que seja necessário a intervenção do utilizador. Será permitindo obter resultados concretos em relação à usabilidade da aplicação *web*, sem que seja necessário avaliar os resultados, tendo em conta os conhecimentos e a facilidade do utilizador em usar os meios informáticos, assim como dificuldades motoras, relacionadas com visão do utilizador. É criada, assim, uma maneira de avaliar a usabilidade do *website* de uma forma concreta e tendo em conta a aplicação *web* e não as características dos diversos utilizadores que realizam os testes, tendo uma intervenção direta.

Web usability guidelines
Home page usability
Task orientation
Navigation and IA
Forms and data entry
Trust and credibility
Writing and content quality
Page layout and visual design
Search usability
Help, feedback and error tolerance

List of navigation and IA usability guidelines

1. There is a convenient and obvious way to move between related pages and sections and it is easy to return to the home page.
2. The information that users are most likely to need is easy to navigate to from most pages.
3. Navigation choices are ordered in the most logical or task-oriented manner.
4. The navigation system is broad and shallow (many items on a menu) rather than deep (many menu levels).
5. The site structure is simple, with a clear conceptual model and no unnecessary levels.
6. The major sections of the site are available from every page (persistent navigation) and there are no dead ends.
7. Navigation tabs are located at the top of the page, and look like clickable versions of real-world tabs.
8. There is a site map that provides an overview of the site's content.
9. The site map is linked to from every page.
10. The site map provides a concise overview of the site, not a rehash of the main navigation or a list of every single topic.
11. Good navigational feedback is provided (e.g. showing where you are in the site).
12. Category labels accurately describe the information in the category.
13. Links and navigation labels contain the "trigger words" that users will look for to achieve their goal.
14. Terminology and conventions (such as link colours) are (approximately) consistent with general web usage.
15. Links look the same in the different sections of the site.
16. Product pages contain links to similar and complementary products to support cross-selling.
17. The terms used for navigation items and hypertext links are unambiguous and jargon-free.
18. Users can sort and filter catalogue pages (e.g. by listing in price order, or showing 'most popular').
19. There is a visible change when the mouse points at something clickable (excluding cursor changes).
20. Important content can be accessed from more than one link (different users may require different link labels).
21. Navigation-only pages (such as the home page) can be viewed without scrolling.
22. Hypertext links that invoke actions (e.g. downloads, new windows) are clearly distinguished from hypertext links that load another page.
23. The site allows the user to control the pace and sequence of the interaction.
24. There are clearly marked exits on every page allowing the user to bale out of the current task without having to go through an extended dialog.
25. The site does not disable the browser's "Back" button and the "Back" button appears on the browser toolbar on every page.
26. Clicking the back button always takes the user back to the page the user came from.
27. A link to both the basket and checkout is clearly visible on every page.
28. If the site spawns new windows, these will not confuse the user (e.g. they are dialog-box sized and can be easily closed).
29. Menu instructions, prompts and messages appear on the same place on each screen.

Figura 12: *Usability guidelines* e lista de *guidelines* de navegação e arquitetura de informação (IA)

Na figura 12, é possível verificar que existem várias categorias, sendo que as mais exploradas foram as de navegação e arquitetura de informação. Dessa lista, constam 29 normas que o sistema deve respeitar, relacionadas com resposta do sistema a situações que vão surgindo à medida que o utilizador vai utilizando o *website*. Estes princípios, descritos na figura, estão relacionados com a simplicidade do sistema, tentando facilitar a interação deste com o utilizador.

2.5 Resumo

O estudo sobre testes de usabilidade, bem como as metodologias e ferramentas aplicadas a estes, permitiu conhecer os aspectos de usabilidade que são importantes testar, para que o *website* contribua para a satisfação do utilizador, evitando a frustração deste e, conseqüentemente, abandono do sistema.

Os vários tipos de métodos estudados e as boas práticas de usabilidade investigadas permitiram conhecer o tipo de testes já existente na área e entender onde o projeto poderá criar valor, criando uma abordagem de testes ainda inexistente.

No caso do estudo das plataformas automatizadas, estas estavam demasiado relacionadas com cliques do rato, movimentos do rato ou visualização, não conseguindo a realização de testes genéricos, tal como, o projeto que se apresenta nesta dissertação. Os métodos baseados em métricas são demasiados limitados, uma vez que se limitam a analisar o código e a forma como as estruturas estão colocadas no *website*. Os processos não automatizados estão demasiado dependentes da presença de uma amostra de pessoas, que nem sempre poderá representar o universo de utilizadores, sendo que as pessoas recrutadas e disponíveis para realizar os testes podem ter habilitações, experiência com o tipo de *sites*, experiência com o computador diferentes do universo geral dos utilizadores. Além disso, esses processos são demasiado demorados e custosos, uma vez que, o estudo envolve recrutamento de pessoas e que as pessoas consigam fornecer a informação que é necessária o mais fiel possível à realidade.

Foi também feito um resumo à pesquisa sobre boas práticas de usabilidade, sendo que foram apresentadas algumas normas relacionadas com a navegação e arquitetura de informação do sistema.

Após a revisão bibliográfica, foi possível entender que o “*Pattern Based Usability Testing*” poderá ser uma abordagem de sucesso e poderá mudar o paradigma dos testes de usabilidade, baseando-se em modelos e construindo padrões de teste genéricos, que poderão ser testados na generalidade dos *websites*. Mais detalhes de implementação serão conhecidos no próximo capítulo.

Capítulo 3

Abordagem

Este capítulo define a abordagem a seguir para uma implementação eficiente do projeto. Define os passos que devem ser seguidos para que projeto possa ser implementado com sucesso, além de introduzir a ferramenta já existente e que irá ser estendida, o *PBGT*.

3.1 PBGT

PBGT (Pattern Based GUI Testing) é uma abordagem de teste de *software*, através da interface gráfica com o utilizador baseada em modelos, que automatiza e sistematiza o processo de teste. O *PBGT* testa comportamento recorrente para o qual possui técnicas genéricas de teste (padrões de teste) capazes de testar diferentes aplicações após um processo de configuração.

Esta ferramenta faz uso de uma linguagem específica de domínio (DSL) denominada “*PARADIGM*”, que é constituída por elementos e conectores, que possibilitam a construção de modelos descrevendo objetivos de teste. Os casos de teste são gerados a partir desses modelos para depois serem executados numa interface gráfica com o utilizador. Os padrões de teste existentes no *PBGT* são genéricos, o que significa que é permitido testar a generalidade dos *websites* e têm em conta a evolução, isto é, podem ser melhorados ao longo do tempo, a fim de atender às necessidades específicas de cada teste. Trata-se de uma ferramenta independente do sistema que está a ser testado e pode ser usado em qualquer fase de desenvolvimento do processo. Os testes têm um objetivo específico, que é o foco desta abordagem, não testando o sistema completo.

Além da linguagem *PARADIGM*, o *PBGT* contém mais 4 componentes:

- *PARADIGM-RE*, que se refere ao componente de engenharia reversa e que permite identificar automaticamente os padrões de teste e, extraí-los para facilitar a construção dos modelos no *PBGT*;

Abordagem

- *PARADIGM-TG*, que é o componente de geração de casos de teste automatizados, construindo casos de teste para o modelo *PARADIGM*;
- *PARADIGM-TE*, que é componente responsável pela execução do caso de teste, retornando relatórios de execução detalhados;
- *PARADIGM-ME*, que é o componente de modelação do ambiente, que permite construção e configuração dos modelos de teste.

O processo de teste no *PBGT* contempla 8 etapas:

- 1) modelação, em que o utilizador selecciona de entre os padrões existentes o modelo que pretende usar para o teste;
- 2) configuração, sendo fornecidos dados necessários, como pré-condições e verificações a fazer;
- 3) geração do caminho de teste, em que o *PARADIGM-TG* gera os casos de teste explorando todos os caminhos possíveis;
- 4) geração do caso de teste, com as configurações anteriormente definidas;
- 5) mapeamento, estabelecido pela pessoa que executa o teste, através do processo de apontar e clicar, isto é, a um parâmetro do teste fazer corresponder a zona da página *web* que lhe corresponde, para que ao longo da execução, haja uma correspondência entre os dados inseridos no *website* e os dados do caso de teste;
- 6) execução, realizado pelo *PARADIGM-TE* e fornecendo os resultados do teste;
- 7) análise de resultados, permitindo verificar se as falhas são do *website* ou do caso de teste, passando neste último caso à próxima etapa;
- 8) atualização do modelo, caso seja necessário, para gerar um teste cujo resultado esteja mais de acordo com a realidade.

A figura que se segue, mostra a sequência entre as etapas do processo.

Abordagem

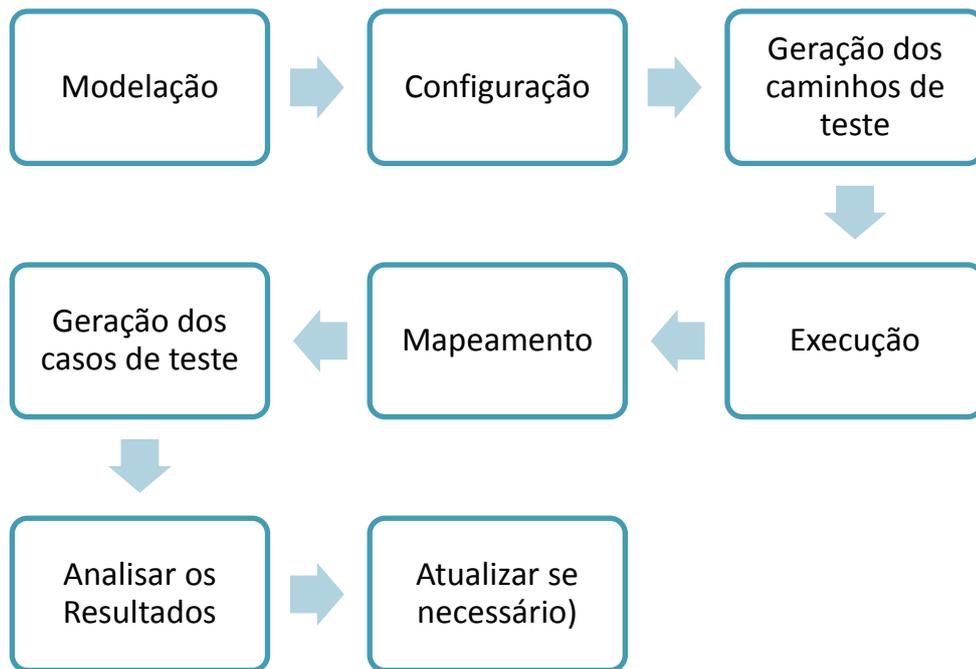


Figura 13: Fases do processo de teste no PBGT

3.2 *PBUT*

Pattern Based Usability Testing é uma aplicação desenvolvida com base no *PBGT*, permitindo a execução de padrões, automatizando o processo de teste de usabilidade a aplicações *web*. Uma vez que deriva do *PBGT*, a forma de implementação é em tudo semelhante, sendo que a estrutura e organização do código se mantêm, alterando o objetivo final a que se propõe, uma vez que a aplicação desenvolvida ao longo desta dissertação consiste na realização de testes de usabilidade, nomeadamente, o teste de *Reachability*, que será abordado mais à frente, em vez de realizar testes às funcionalidades do *website*. O objetivo principal é permitir o teste de usabilidade às interfaces gráficas, conseguindo identificar alguns aspetos que possam ser melhorados, permitindo uma mais fácil adaptação do utilizador à interface que utiliza. Na figura 14, é possível visualizar o ambiente do *PBUT*.

Abordagem

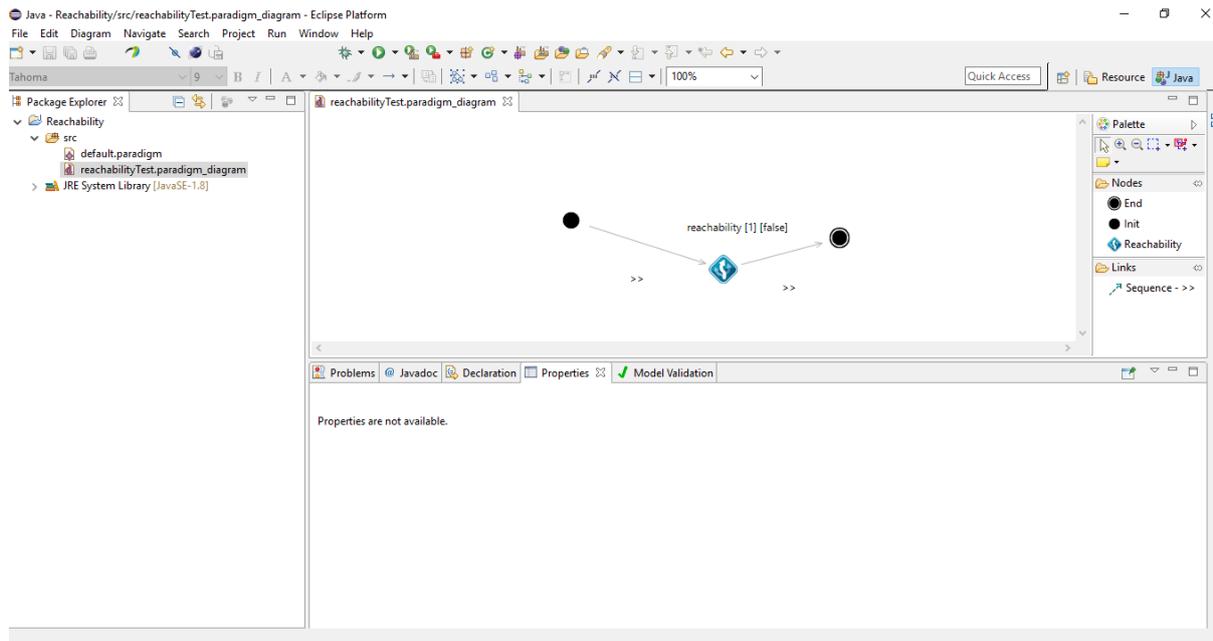


Figura 14: Ambiente da aplicação

3.3 Modelo da linguagem

A linguagem núcleo do modelo representa as abstrações de domínio e especifica as relações entre eles. No contexto do *PBUT*, as abstrações representam conceitos, utilizando a linguagem *PARADIGM*, sendo que essas abstrações podem ser elementos de comportamento (*Behavioural*), que correspondem aos padrões de teste às interfaces do utilizador, que no caso do *PBUT* corresponde ao *Reachability UI Test Pattern* e conectores de linguagem, que no projeto implementado consiste na sequência entre os elementos do modelo de execução.

Na figura 15, encontra-se o meta-modelo da linguagem, no qual podemos distinguir elementos e conectores. Um elemento corresponde a uma entidade abstrata que representa um conceito no contexto do *PBUT*. Existem, portanto, nesta aplicação 3 elementos: *Init*, *Behavioural* e *End*. Um modelo representado em *PARADIGM*, tem como início a entidade *Init* e como fim a entidade *End*, ambas especializações da entidade abstrata *Element*. Os elementos *Behavioural*, correspondem aos padrões de teste implementados, isto é, representam as estratégias de teste. O único conector utilizado pelo *PBUT* é o da sequência, que indica que o a sequência do processo de teste só avança para o destino deste conector, quando o elemento de onde provém finalizar a sua execução.

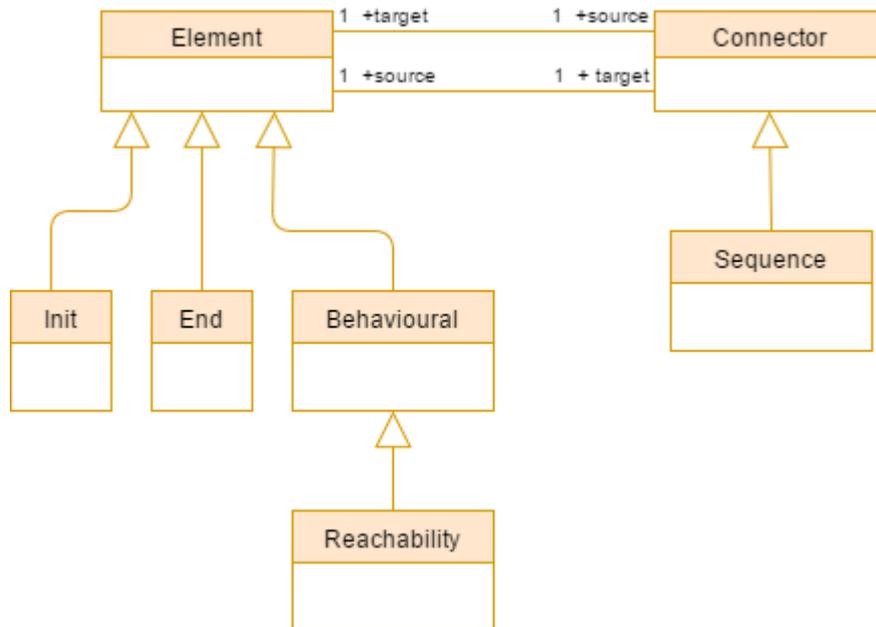


Figura 15: Meta-modelo da linguagem

3.4 Cenários de utilização

Nesta secção é realizada uma descrição dos cenários de utilização do sistema, isto é, são apresentados os diferentes casos de uso. O utilizador inicia criando o projeto, sendo que posteriormente, começa a fase de modelação, criando os nós e o conetor entre eles, sendo que tem que respeitar a regra da aplicação, de que tem que haver um nó *start* no início e um nó *end* no fim. Realiza-se a configuração dos diferentes nós, dando-lhes um nome e um identificador, sendo também adicionados dados de entrada necessários ao teste, neste caso no nó de *reachability*. Neste padrão a fase de mapeamento não existe porque os dados necessários de identificação da aplicação já estão fornecidos nos *inputs*. Na execução do padrão de teste faz-se a exploração da aplicação para a testar e, no final, são apresentados os resultados do teste. Na figura 16, é possível ver todos os casos de uso.

Abordagem

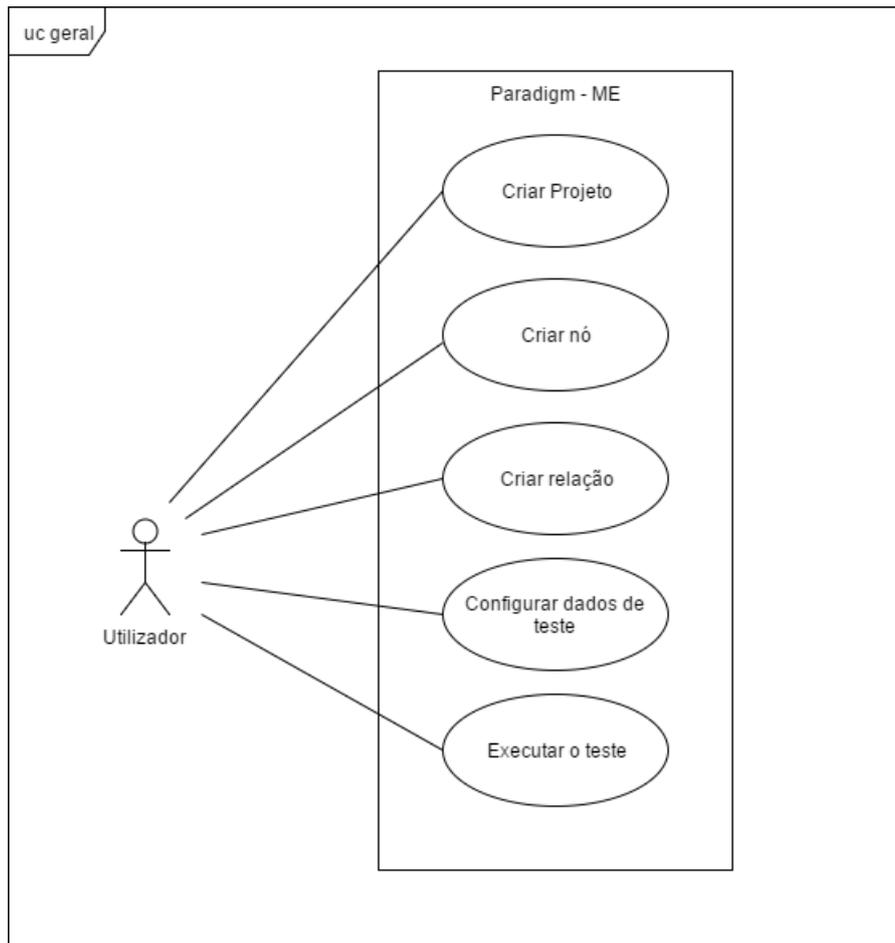


Figura 16: Casos de uso

Na figura 16, é possível visualizar os principais casos de uso da aplicação na construção e execução da aplicação. Além destes casos de uso, são permitidas as funções de eliminar e editar um projeto e um nó ou eliminar uma relação, mas que não são exploradas a fundo, por não se tratarem de funções essenciais à execução do teste.

De seguida, é feita uma descrição mais detalhada de cada um dos casos de uso referidos na figura anterior.

Abordagem

ID	UC01
Nome	Criar projeto
Descrição	Criação de um novo projeto
Fluxo de eventos	<pre> graph TD Inicio((Início)) --> A[Abrir Eclipse e criar novo projeto Java] A --> B[Criar New -> Other... -> Paradigm Diagram] B --> C[Escolher o nome do diagrama] C --> D[Pressionar Finish] D --> Fim(((Fim))) </pre>
Pré-condições	-
Pós-condições	Projeto criado e é apresentado o programa com a <i>palette</i> .

Tabela 5: Caso de uso UC01 - criação do projeto

Este caso de uso consiste na criação do projeto de modelação de interface gráfica. Trata-se da criação de um projeto de uma forma geral no Eclipse, sendo necessário seleccionar o tipo de ficheiro que se pretende criar, neste caso, um *Paradigm Diagram*. Este caso de uso é o ponto de entrada na aplicação, uma vez que a partir do momento em que é pressionado o botão *Finish*, é apresentado ao utilizador o ambiente de modelação, isto é, as funcionalidades da aplicação, nomeadamente a paleta com os nós e os *links* necessários à criação do modelo de teste.

Abordagem

ID	UC02
Nome	Criar nó
Descrição	Criação de um novo nó
Fluxo de eventos	<pre> graph TD Inicio((Início)) --> Selecionar[Selecionar elemento do tipo nó na palette.] Selecionar --> Pressionar[Pressionar o local do diagrama em que se pretende que o nó seja criado.] Pressionar --> Fim(((Fim))) </pre>
Pré-condições	Diagrama do modelo aberto.
Pós-condições	Passa a existir um novo nó do tipo que foi selecionado.

Tabela 6: Caso de uso UC02 - criação do nó

O caso de uso descrito na tabela 6 corresponde ao da criação de um novo nó, correspondendo a uma funcionalidade básica do ambiente de modelação, constituindo o primeiro passo para a criação do modelo de teste. Os nós disponíveis no *PBUT* são 3 e são elementos obrigatórios, uma vez que é necessário que um modelo de teste contenha um nó *Init* e um nó *End*, e também um nó correspondente ao processo de teste, neste caso a aplicação apenas permite o teste de *Reachability*.

Abordagem

ID	UC03
Nome	Criar relação
Descrição	Criação de uma nova relação
Fluxo de eventos	<pre> graph TD Inicio((Início)) --> S1(Selecionar elemento do tipo relação na palette.) S1 --> S2(Pressionar nó de origem) S2 --> S3(Pressionar nó de destino) S3 --> Fim(((Fim))) </pre>
Pré-condições	Diagrama do modelo aberto e existência de um nó de origem.
Pós-condições	Passa a existir uma nova relação entre o nó de origem e nó de destino.

Tabela 7: Caso de uso UC03 - criação da relação

O caso de uso descrito na tabela 7 consiste na criação da relação e constitui outra funcionalidade básica do ambiente de modelação. Para que seja criada uma relação é necessário que exista um nó de origem e um nó de destino, uma vez que a relação é sempre criada entre dois nós do modelo. Após a criação da relação, o utilizador finaliza a criação do seu modelo, faltando verificar a integridade do sistema e configurar os dados de teste.

Abordagem

ID	UC04
Nome	Configurar dados de teste
Descrição	Fornecer os dados de entrada necessários à realização do teste
Fluxo de eventos	
Pré-condições	Diagrama do modelo aberto e existência de um nó descendente de <i>Behavioural</i> , neste caso <i>Reachability</i> .
Pós-condições	O <i>browser</i> abre imediatamente após clicar em “Ok”.

Tabela 8: Caso de uso UC04 - configuração dos dados de teste

Este caso de uso descrito na tabela 8, consiste na descrição da forma de inserir os dados de entrada necessários a que o teste execute com sucesso. Assim, o utilizador necessita de inserir os parâmetros para que a simulação da navegação se inicie.

ID	UC05
Nome	Executar o teste
Descrição	Início da simulação da navegação ativada pelo utilizador
Fluxo de eventos	<pre> graph TD Inicio((Início)) --> A[Clicar com o botão direito do rato no canto superior esquerdo do website] A --> B[Pressionar 2 vezes "Ok" nas 2 caixas de diálogo que aparecerem.] B --> Fim((Fim)) </pre>
Pré-condições	Fornecimento dos dados de entrada e abertura automática do <i>Google Chrome</i> .
Pós-condições	A navegação é iniciada de imediato.

Tabela 9: Caso de uso UC05 - executar o teste

Este último caso de uso, descrito na tabela 9, consiste na fase de execução do teste, em que o utilizador ao clicar com o botão direito do rato no canto superior esquerdo do *website*, faz com que o programa inicie a recolha dos elementos web, sendo posteriormente inseridos na árvore de navegação, entidade que será, posteriormente, explicada na secção 3.9.

3.5 Padrão de usabilidade implementado

Depois de uma análise das boas práticas de usabilidade e de um estudo dessas boas práticas aplicadas à navegação e arquitetura da informação, realizado na secção 2.4 foi escolhido o padrão, que foi considerado adequado implementar no contexto do *PBGT*, o padrão de *reachability*. O padrão refere-se à avaliação da possibilidade de atingir uma página indicada pelo utilizador num número determinado de passos, *n*, também indicado pelo utilizador. Assim, a boa prática implementada foi: “A informação que os utilizadores mais necessitam está acessível de forma fácil e da maior parte das páginas”. É, portanto, possível testar se uma página

é acessível após determinado número de cliques do rato, avaliando a facilidade com que a página pode ser acessível ao utilizador.

3.6 Definição formal de padrão

Uma vez que o objetivo desta dissertação consiste em estender o PBGT, foi necessário estudar a definição formal de padrão, para que houvesse uma escolha correta dos padrões a implementar, para que fosse possível testar usabilidade em *websites*.

Assim, um padrão de teste de uma interface do utilizador consiste num conjunto de estratégias de teste utilizadas para testar comportamento recorrente. No contexto do projeto PBGT, um padrão de teste é um conjunto de estratégias de teste associadas e consiste em $\langle Goal, V, A, C, P \rangle$ como definido em [4] em que:

- *Goal* é o ID do teste;
- *V* é o conjunto do par de variáveis $\{[variável, *dadoInput*]\}$, relacionando os dados de input com as variáveis envolvidas no teste;
- *A* é a sequência de ações que devem ser realizadas durante a execução do caso de teste;
- *C* é o conjunto de verificações possíveis ("*checks*"), a realizar durante a execução do caso de teste;
- *P* é uma expressão booleana, correspondente às pré-condições, definindo as condições a que as variáveis estão sujeitas, sendo possível determinar quando é possível realizar o teste.

Em suma, "*Goal*" é o nome do teste, *A* e o conjunto de variáveis *V* descrevem o que fazer e como executar o teste. *C* descreve a finalidade ou o porquê da realização do teste. *P* define as circunstâncias em que o teste pode ser executado.

Durante a fase de modelação, o utilizador que testa necessita de configurar cada padrão de teste dentro de um modelo. Deve seleccionar os objetivos do teste e para cada um, fornecer as variáveis de *input* (*V*), as verificações (*C*) que devem ser realizadas e definir as pré-condições a que o teste se encontra sujeito. Um mesmo teste pode ser realizado com diferentes configurações de modo a compreender melhor o seu comportamento, perante as alterações que se vão introduzindo.

3.7 Formato dos padrões

Dentro das várias formas de descrever um padrão, é apresentada em seguida, com o objetivo de descrever o padrão, nomeadamente, quando deve ser usado, em que situações pode ser utilizado, como o utilizar e a finalidade do seu uso. Assim, os padrões criados irão ter a seguinte estrutura:

- Nome do padrão, que consiste num identificador único para o padrão;
- Contexto, que é a situação em que o problema ocorre;
- Problema, que descreve o problema tratado pelo teste;
- Forças, que são as reflexões a considerar quando se escolhe a solução;
- Solução, que é a descrição da solução proposta para o padrão;
- Consequências, que podem ser positivas ou negativas e que emergem da solução;
- Candidatos a utilização, que são condições reais (padrões de *UI*) em que o teste pode ser aplicado;
- Usos conhecidos, que consistem nas aplicações *web* em que o teste foi aplicado com sucesso;
- Exemplos de utilização, ou seja, exemplos concretos da aplicabilidade do padrão.

3.8 *Reachability UI Test Pattern*

a) Contexto

Existem vários *websites* que contemplam um grande conjunto de páginas *web* que podem ser acedidas, sendo que existem umas que são mais utilizadas e mais importantes tendo em conta as funcionalidades que permitem ao utilizador. Assim, é importante entender quais as funcionalidades e páginas de uma aplicação *web* são mais importantes ou constituem tarefas mais importantes para o utilizador, sendo necessário facilitar o acesso a estas, para que o utilizador possa descobrir mais rapidamente e de forma intuitiva como chegar às páginas que pretende.

Neste contexto, surge o teste de *Reachability* que foi implementado na aplicação *PBUT*, o qual permite testar se uma determinada página de destino é atingida após um número de cliques do rato indicado pelo utilizador. Assim, caso a página seja atingida após um número de cliques igual ou inferior ao número máximo de cliques que é suposto que seja, o teste passa. Em caso contrário, não passará.

b) Problema

Como definir uma estratégia de teste que permita saber se uma página de destino de um *website* é atingida após um número máximo de cliques definido pelo utilizador?

c) Forças

- Muitas funcionalidades importantes de um *website* podem estar “escondidas” para o utilizador;
- A necessidade de uma boa escolha de um número máximo de cliques, o qual deve ter em conta se a funcionalidade a testar é importante ou não;
- Grande número de páginas de destino que um *website* pode ter, sendo necessário a escolha de uma página que tenha uma importância relacionada com o número de cliques escolhido.

d) Solução

Fornecer uma estratégia de teste genérica para interfaces gráficas de utilizador que permita verificar se a navegação na página *web*, permite atingir determinada página em determinado número de passos. A estratégia de teste consiste no seguinte:

1. Objetivos de teste: “*Reach*” e “*Not_Reach*”.
2. Conjunto de variáveis: {*DestinationURL*, *nClicks*};
3. Sequência de ações: []
4. Conjunto de *Checks*: {*Found page X*, *Not Found page Y*}, onde X foi a página que foi atingida e Y a página que não foi atingida.

e) Consequências

- Após o fornecimento das duas variáveis, *URL* de destino e número máximo de cliques, é simulada a navegação do *website* até ser encontrada. Caso não seja encontrada, o teste tem como resultado “*Not_Reach*”, uma vez que a página não é atingida.
- É permitida a realização deste teste em qualquer *website*, uma vez que é recolhida a lista de links a partir do código *HTML* da página *web*, que se encontra a ser testada, de forma simples.

f) Candidatos a utilização

- *NavigationTabs* [24]

g) Usos conhecidos

- https://sigarra.up.pt/feup/pt/web_page.inicial
- <http://www.ox.ac.uk/>
- <http://www.samsung.com/pt/>

h) Exemplos de utilização

A pessoa que vai testar pretende verificar se determinada tarefa importante para um *website* é atingível de forma rápida e fácil, sem que o utilizador necessite de muito tempo para a atingir. Ao realizar o teste, existem 2 resultados positivos, se a página foi atingida ou não com sucesso após os n cliques definidos pelo utilizador como limite máximo de passos de navegação.

1. Objetivo – *Reach*

V: { [*destinationURL*, https://sigarra.up.pt/feup/en/WEB_PAGE.INICIAL], [*nClicksMax*, 1] }

A: []

C: {*Found*}

P: *true*

Nesta configuração, o utilizador fornece as variáveis necessárias à realização do teste, sendo que o teste tem resultado positivo, uma vez que foi encontrada a página de destino fornecida como valor de *input* na lista de *links* de destino possíveis de aceder após um clique do rato a partir da *homepage*.

2. Objetivo – *Not Reach*

V: { [*destinationURL*, https://sigarra.up.pt/feup/en/web_base.gera_pagina?P_pagina=242378], [*nClicksMax*, 2] }

A: []

C: {*Not_Found*}

P: *true*

Nesta configuração, o utilizador fornece um *website* de uma página, pretendendo atingi-la em 2 cliques. Há, portanto, uma exploração da *homepage*, contemplando todas as páginas que é possível atingir após 2 passos. De entre essas páginas, não é encontrada a página de destino, sendo impossível chegar a essa página após 2 passos. Deste teste, e visto que a página pode ser importante para um estudante estrangeiro que pretenda chegar aos contactos da faculdade, então do ponto de vista de usabilidade, esta página devia estar mais facilmente acessível.

3.9 Detalhes de implementação

O algoritmo utilizado na implementação do padrão descrito na secção anterior consiste na construção de uma árvore (grafo), contendo um conjunto de vértices, ligados por arestas. Cada vértice consiste num *link* de uma página *web* e as arestas correspondem às ligações entre as páginas, por via de um clique em algum *link* existente, isto é, partindo da página inicial, esta vai estar ligada por arestas a todas as páginas que poderão ser atingidas por um clique do rato a partir da *homepage* e, assim, sucessivamente. Uma vez que o utilizador fornece um número máximo de cliques e, para evitar que haja um uso excessivo dos recursos computacionais, o que faria com que o processo de recolha dos *links* de navegação fosse um processo demorado, explora-se a aplicação de forma a construir uma árvore até uma profundidade máxima, indicada pela variável correspondente ao número máximo de cliques (n) definido para o teste. À medida que a árvore de navegação vai sendo explorada, vai sendo verificado se a página de destino pretendida foi atingida. Em caso afirmativo, o teste assume o valor de REACH. Uma vez atingido o n máximo e, caso não seja encontrada a página, o teste termina sendo devolvido como resultado final do teste, NOT_REACH, uma vez que, de entre todas as páginas que podem ser atingidas com n cliques, a página de destino pretendida não corresponde a um vértice da árvore de navegação recolhida. A exploração da aplicação é feita através do algoritmo de procura *Breadth-first*, sendo que os vértices vão sendo adicionados à árvore de acordo com a ordem indicada pela figura que se segue (figura 17).

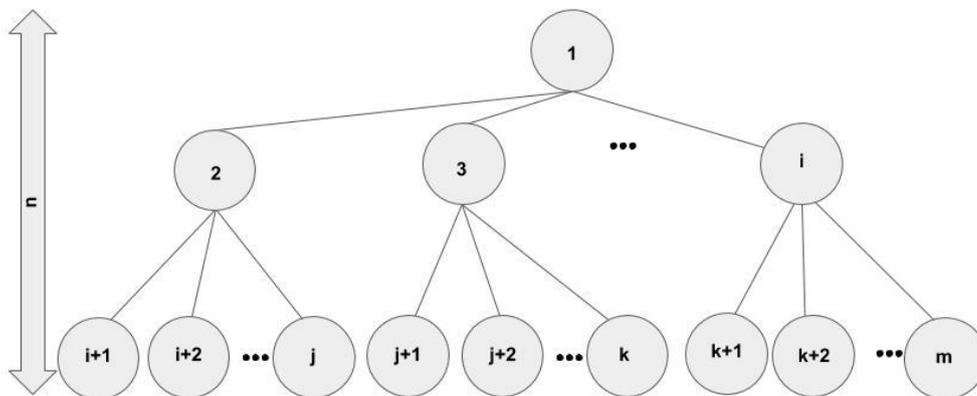


Figura 17: Exemplo de árvore de navegação

Abordagem

Como é possível ver na figura anterior, a árvore é explorada em largura, sendo que são sempre explorados todos os filhos de um vértice antes de explorar os vértices-filho dos seguintes. Assim, e tomando por exemplo o caso em a profundidade máxima de exploração (n) é 1, então a exploração da árvore termina mal acabe a iteração de exploração dos filhos da *homepage*. Este método de pesquisa, facilita a limitação do número máximo de cliques que é fornecido como *input* pelo teste.

De seguida, apresenta-se o pseudocódigo correspondente ao algoritmo da procura:

```
1      adiciona do vértice  $v$ , correspondente à homepage, à lista  $V$ 
2
3      ciclo para percorrer conjunto de links de navegação
4      adiciona-os a uma lista  $L$ 
5
6      ciclo para percorrer  $L$  e se o link não for nulo, cria vértice  $v1$ 
7      cria aresta  $e1$  entre  $v$  e  $v1$ 
8
9      se  $v1$  not in  $V$  &&  $v1$  not extern
10         adicionar  $v1$  a  $V$  e  $e1$  adicionada a lista  $E$ 
11
12      Cria variável finalSize, que é inicializada com o valor do tamanho da lista  $V$ 
13      Cria variáveis countN, que faz a contagem do número de cliques, inicializada com valor 1,
14
15      while  $countN \leq nClicksMax$ {
16         percorre lista  $V$ 
17         navega para cada um dos vértices de  $V$ 
18
19         se o URL atual == URL de destino
20             termina a execução, retorna o resultado que é positivo e guarda o número de
                cliques
21
22         Cria vértice  $v2$  com URL atual
```

Abordagem

```
23
24     se countN < nClicksMax {
25
26         adiciona os elementos web que podem ser acedidos a partir do URL atual a L1
27         percorre L1
28         cria vértice v3 com URL L1, caso link não seja nulo
29         cria aresta e2 entre v2 e v3
30
31         se v3 not in V && v3 not extern
32             adicionar v3 a V e e2 a E
33     }
34     Quando o iterador i que percorre a lista V atinge finalSize - 1
35     finalSize = tamanho de V atual
36     incrementa valor de countN, uma vez que passa para um novo nível de profundidade
37
38     Se chegar ao caso em que o iterador chega ao fim dos links então o ciclo termina
39 }
```

O pseudocódigo apresentado anteriormente corresponde a um fragmento parte do código implementado, que corresponde à simulação da navegação no *website*. Inicialmente, criam-se as listas de vértices (V) e arestas (E), que vão constituir a árvore de navegação. Cria-se o vértice v correspondente à página inicial, adicionando à lista V. Percorre-se uma lista dos *links* que se podem aceder a partir da *homepage*, selecionando os elementos do tipo “<a href= ...” para adicionar a uma lista L. A cada *link* que se vai encontrando é necessário encontrar os *links* externos e excluí-los da árvore de navegação. Percorre-se a lista de elementos e cria-se o vértice com as devidas configurações. Cria-se uma aresta *e1* entre *v* e *v1*. Nesta primeira fase, são apenas explorados os *links* que podem ser acedidos por um único clique na *homepage*. Tanto o vértice *v1* como a aresta *e1* são adicionados às respetivas listas, se o vértice ainda não existir na lista V. Cria-se *finalSize* que vai ser o tamanho do 1º nível de vértices após se terem recolhidos todos os *links* acedidos diretamente a partir da página inicial. Este valor vai variando correspondendo sempre ao índice do último elemento de um nível de profundidade. A variável *countN* inicia-se com valor 1, e faz a contagem de cliques. Depois dá-se o ciclo que será executado enquanto *countN* for menor ou igual que o número de cliques máximo ou então até

encontrar a página de destino pretendida. Percorre-se a lista de vértices verificando sempre se a página de destino foi encontrada. Caso seja encontrada é apresentado o resultado sendo o ciclo terminado de imediato. Caso ainda não seja, vão-se criando os vértices que ainda podem ser atingidos a partir das páginas pelas quais se vai navegando, sendo que os vértices e arestas são criados segundo as mesmas condições. A verificação da linha 24, evita que chegando ao último nível se explorem as páginas que podem ser acedidas a partir desses *links* do último nível. Quando o índice *i* utilizado para percorrer os vértices chega ao fim de um nível de profundidade então, *i* começa nesse índice e o fim passa a ser o tamanho atual dos vértices, após a adição dos “nós filho” das páginas do nível anterior.

A execução deste algoritmo só é possível graças à utilização do *Selenium* [25], ferramenta que permite a automatização de aplicações *web* para objetivos de teste. Esta ferramenta permite que os elementos *web* sejam guardados pela *WebDriver* do *Selenium*, que permite que o *browser* seja aberto, permitindo a recolha dos elementos *web* pretendidos e, posterior, simulação da navegação.

3.10 Detalhes de execução

Como referido em 3.7, um modelo de teste contém um elemento *Init*, um elemento *Behavioural* que define o comportamento da estratégia de teste e, que define todas as variáveis necessárias, terminando com um elemento *End*. Inicialmente, constrói-se o modelo, com os elementos acima referidos, indicando para cada elemento, os parâmetros *Name* e *Number*, ligados por conetores. Um exemplo do modelo construído pode ser visto na figura 18.



Figura 18: Modelo de teste

Após a construção do modelo, são inseridos no botão de *Reachability*, o *mappingURL* e editados os parâmetros em *Field -> Map Fields*, sendo inseridos os parâmetros como pode ser verificável na figura 19.

Field selection

Page URL

Destination Page URL

Number Maximum of Clicks (n)
 1 2 3 4 5

1) Select page's starting point by right-clicking on the top-left corner of the website
 2) Select fields to map

Fields to map	is mapped
<ul style="list-style-type: none"> <input checked="" type="checkbox"/> 1 - reachability <input checked="" type="checkbox"/> reachability 	false

[Select All](#)
[Select None](#)
[Select Unmapped](#)

Figura 19: Formulário de preenchimento das variáveis

Após o preenchimento e confirmação através do botão “ok”, é aberta a página indicada pelo *URL* no *Google Chrome*, sendo necessário um clique com o botão direito do rato no canto superior esquerdo. É então simulada a navegação, seguindo os passos do algoritmo descrito na secção anterior, sendo obtidos os resultados anteriormente descritos.

3.11 Resumo e Conclusões

Neste capítulo, foi realizada uma descrição detalhada do modo de implementação do PBUT, sendo inicialmente introduzido o *PBGT*, aplicação da qual o *PBUT* é estendido. Das boas práticas de usabilidade referidas no capítulo anterior, teve que haver uma seleção para que

Abordagem

fosse possível encontrar o padrão mais adequado a implementar no contexto da aplicação que lhe deu origem. Após este estudo, é realizada uma descrição detalhada da definição e formato dos padrões, segundo os padrões implementados no *PBGT*. Posteriormente, é utilizada essa definição e formato para descrever detalhadamente o padrão de *Reachability*. Por fim, é feita uma exposição dos tópicos relacionados com a aplicação, o modelo de linguagem, e os detalhes de implementação do algoritmo e do modo de execução.

As secções em que este capítulo foi dividido, corresponderam às tarefas que foram realizadas na fase de implementação deste projeto, sendo que ao fim de cada tarefa estavam criadas as condições necessárias para que a próxima tarefa avançasse com sucesso.

Abordagem

Capítulo 4

Casos de Estudo

Neste capítulo são apresentados os casos de estudo, que irão testar a utilidade da aplicação desenvolvida, bem como permitir uma avaliação da usabilidade de uma aplicação *web*, utilizando para isso o padrão que foi implementado pelo *PBUT*. Inicialmente, é feita uma breve descrição da aplicação *web*, é apresentada uma imagem da página inicial, sendo, posteriormente, descritas as condições em que o teste é realizado. Tanto essas condições, como os resultados finais do teste, são sumarizados em tabelas. Depois de apresentados os resultados, é realizada uma análise qualitativa, comentando o resultado e comparando com o que seria de esperar.

4.1 Ambiente Experimental

Especificações	Máquina: Asus X556U
Sistema Operativo	Windows 10 <i>Home</i>
<i>CPU</i>	Intel Core i7-6500U, <i>up to</i> 3.1GHz
Memória	8,00 GB
Conexão à internet	100 Mbps

Tabela 10: Descrição da máquina utilizada nos testes

4.2 Formato dos Resultados

Após a execução da simulação da navegação, são apresentados os resultados numa caixa de diálogo e, posteriormente, na consola. Um exemplo do formato dos resultados encontra-se na figura 20.

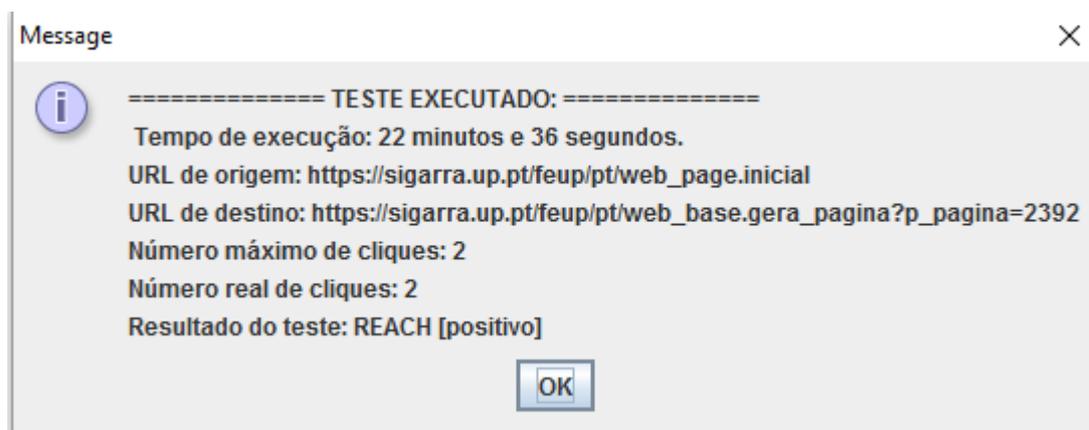


Figura 20: Formato dos Resultados

4.3 Caso de Estudo I – Sigarra

SIGARRA [26] (Sistema de Informação para a Gestão Agregada dos Recursos e dos Registos Académicos) consiste num sistema académico criado para satisfazer as necessidades de uma comunidade universitária a nível de administração, gestão, ensino, investigação e desenvolvimento de atividades das pessoas envolvidas na comunidade. Decorre de uma estratégia consistente que permite o desenvolvimento de componentes informáticas e organizacionais, permitindo satisfazer as necessidades de um Universidade moderna. É atualmente o sistema utilizado pela Universidade do Porto, contemplando cada Faculdade ou organização, uma extensão deste, que permite gerir as atividades académicas do meio académico em que me insiro.

O primeiro caso de estudo corresponde a testes de usabilidade realizados no *SiFEUP*, que é a instância do *SIGARRA* utilizada pela Faculdade de Engenharia da Universidade do Porto (*FEUP*), uma vez que é um *website* utilizado por um universo significativo de utilizadores, contabilizando cerca de 12000 pessoas, incluindo docentes, funcionários e estudantes. Vão ser realizados 5 testes, os quais terão uma descrição detalhada de seguida. Na figura 21, é possível visualizar a página inicial do *SIGARRA*, nomeadamente o *SiFEUP*.

The screenshot shows the website of the Faculty of Engineering at the University of Porto (FEUP). The page features a navigation menu on the left, a main content area with several announcements, and a calendar on the right. The announcements include:

- ALTERAÇÃO À INSCRIÇÃO EM UNIDADES CURRICULARES ADDING/DROPPING COURSES - 2016/17** (23 JAN - 20 FEB, 2017)
- INQUÉRITO À EMPREGABILIDADE 2015/2016**
- CANDIDATURAS / APPLICATIONS 2017/2018**
- CANDIDATOS DO BRASIL (CANDIDATURAS ATÉ 30 DE ABRIL)**
- INQUÉRITOS PEDAGÓGICOS 1º SEMESTRE 2016/17**

The calendar on the right shows the month of January 2017, with a filter set to 'Todos'.

At the bottom of the page, there are 'Últimas notícias' (Latest news) with several links to articles.

Figura 21: Sigarra na FEUP

4.3.1 Testes Executados

O primeiro teste a efetuar no *SiFEUP*⁸ consistiu na tentativa de chegar à página de destino relacionada com a Licenciatura em Ciências da Comunicação: Jornalismo, Assessoria, Multimédia, sendo dado um número máximo de cliques de 3. O segundo teste correspondeu à verificação da possibilidade de atingir o calendário escolar após 2 cliques. O terceiro e quarto testes possuem o mesmo objetivo, sendo a página de destino a mesma, a tentativa de aceder à ementa da cantina. Foram feitas 2 versões do teste, uma com o número máximo de 2 cliques e outra com o número máximo de 5 cliques, permitindo avaliar se o tempo de execução varia ou não consoante o valor dessa variável, permitindo avaliar o comportamento da aplicação em termos do seu tempo de execução. O quinto teste tem como objetivo atingir a página do valor das propinas após 2 cliques. De seguida, apresentam-se as tabelas de dados de entrada (tabela 11) e os resultados (tabela 12).

⁸ https://sigarra.up.pt/feup/pt/web_page.inicial

	Página de destino	Número cliques
1	https://sigarra.up.pt/feup/pt/cur_geral.cur_view?pv_ano_lectivo=2016&pv_origem=CUR&pv_tipo_cur_sigla=L&pv_curso_id=455	3
2	https://sigarra.up.pt/feup/pt/web_base.gera_pagina?p_pagina=2392	2
3	https://sigarra.up.pt/feup/pt/cantina.ementashow	3
4	https://sigarra.up.pt/feup/pt/cantina.ementashow	5
5	https://sigarra.up.pt/feup/pt/web_base.gera_pagina?p_pagina=242391	2

Tabela 11 - Resumo dos dados de entrada do caso de estudo I

	Tempo de execução	Nº cliques	Resultado
1	37 minutos e 59 segundos	2	REACH
2	22 minutos e 36 segundos	2	REACH
3	26 minutos e 38 segundos	2	REACH
4	26 minutos e 4 segundos	2	REACH
5	2 horas, 6 minutos e 48 segundos	-	NOT REACH

Tabela 12: Resumo dos resultados dos testes do caso de estudo I

4.3.2 Análise aos Resultados

Todos os testes a esta plataforma tiveram tempos de execução significativos, uma vez que se trata de um sistema de informação bastante grande, contendo uma elevada quantidade de páginas. Todos os testes realizados são de páginas relevantes, tendo sido todas elas atingidas por 2 cliques, o que faz com que o *website* respeite as métricas de usabilidade que foram usadas para o testar. Além de não ser necessário uma elevada quantidade de cliques para atingir as páginas mais importantes, a página inicial contém menus bem visíveis, facilitando a tarefa de navegação no *website*, por parte dos utilizadores, sendo mais fácil encontrar o que procuram. A maior parte das páginas de maior relevância são facilmente atingíveis pelo utilizador, algo que foi possível verificar pela simulação da navegação, permitindo perceber porque páginas o programa ia passando.

Aceder às páginas relacionadas com um curso da faculdade (experiência 1) ou ao calendário escolar (experiência 2) são tarefas simples e que se podem realizar em 2 cliques, tornando fácil atingi-las. As experiências do teste 3 e 4 pretendiam comprovar que alterando o número máximo de cliques, mas mantendo o *URL* inicial e destino, a performance do programa é a mesma, uma vez que foi obtido o mesmo resultado, sensivelmente após tempos de execução bastante semelhantes, uma vez que o *link* de destino mal é encontrado é logo retornado, parando a execução do programa de imediato. Ora, qualquer que seja o número máximo de cliques, o tempo de execução para este teste que pretende atingir a página da ementa da cantina será sempre próximo dos 26 minutos. A última experiência relacionada com o valor das propinas não foi atingida em 2 cliques, sendo necessários mais do que esses cliques para atingir essa página.

4.4 Caso de estudo II – *University of Oxford*

O *website* da Universidade de Oxford⁹ consiste no sistema de informação utilizado pelos membros da comunidade académica desta Universidade, em Oxford, no Reino Unido, sendo uma das mais importantes Universidades do país e a segunda mais antiga da Europa. O estudo de usabilidade na sua página *web* é útil, uma vez que esta é utilizada por um universo de cerca de 24 000 pessoas, distribuídas por 39 faculdades. Serão realizados 3 testes a este *website*, tentando averiguar possíveis problemas de usabilidade. Apresenta-se de seguida a página inicial desta plataforma (figura 22).

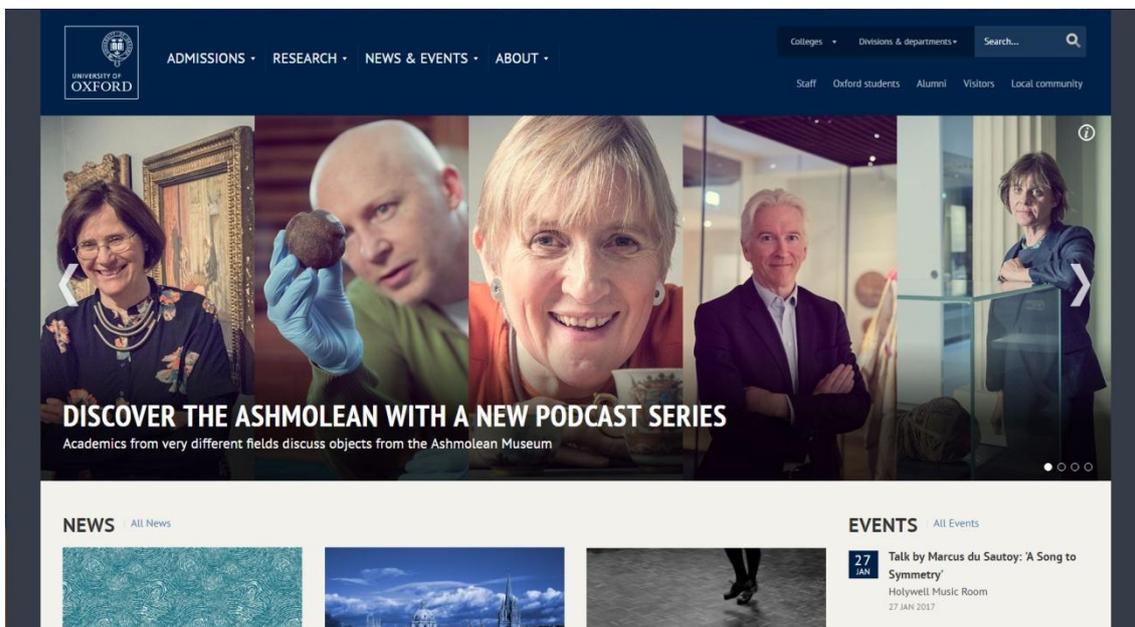


Figura 22: *Website* da Universidade de Oxford

⁹ <http://www.ox.ac.uk>

4.4.1 Teste Executados

O primeiro teste realizado nesta plataforma consistiu em verificar se é possível atingir a página do *MSc in Computer Science*, em 3 passos. O segundo teste correspondia à tentativa de atingir a página relativa aos casos de estudo que tiveram impacto na área da investigação na Universidade, em 4 passos. O terceiro e último teste tem como objetivo chegar à página do *Lincoln College*, com um máximo de 2 cliques. Nas tabelas 13 e 14, é possível verificar as páginas de destino e o número máximo de cliques e os resultados dos testes, respetivamente

	Página de destino	Número cliques
1	https://www.ox.ac.uk/admissions/graduate/courses/msc-computer-science?wssl=1	3
2	http://www.ox.ac.uk/research/research-impact/impact-case-studies	4
3	https://www.ox.ac.uk/admissions/graduate/colleges/lincoln-college?wssl=1	2

Tabela 13: Resumo dos dados de entrada do caso de estudo II

	Tempo de execução	Nº cliques	Resultado
1	56 minutos e 19 segundos	2	REACH
2	1 hora, 8 minutos e 11 segundos	2	REACH
3	2 horas, 58 minutos e 7 segundos	-	NOT REACH

Tabela 14: Resumo dos resultados do caso de estudo II

4.4.2 Análise aos resultados

Os tempos de execução dos testes realizados ao segundo caso de estudo tiveram tempos de execução superiores ao do primeiro, dado que se trata do sistema de informação utilizado por uma Universidade e não por uma Faculdade, além de que, contém uma grande quantidade de páginas relacionadas com todos os cursos desta Universidade. Assim, e comparando com a plataforma do primeiro caso de estudo a quantidade de páginas que se pode aceder é francamente maior, sendo o tempo da simulação da navegação maior. Os 2 primeiros testes executados neste caso tiveram resultado positivo, uma vez que foi possível atingir as páginas

de destino pretendidas com 2 cliques. O terceiro teste pretende chegar à página de uma faculdade da Universidade de Oxford e 2 cliques, algo que não chega a acontecer. Nesta última o tempo de execução é de quase 3 horas, o que significa que para o programa navegar por todos os links até às páginas atingidas após 2 cliques, navega por uma quantidade enorme de páginas, o que significa que a maior parte das páginas deste *website* se encontram acessíveis até ao máximo de 2 cliques. Ao contrário do *SiFEUP* analisado na secção 4.3, o menu acaba por não ser tão extenso e não é visível na primeira visualização do *website*, facilitando a leitura do menu, sendo, no entanto, necessário mover o rato para que sejam apresentadas a lista de páginas que é possível atingir.

4.5 Caso de Estudo III – *Website* da Samsung Portugal

O *website* da Samsung Portugal¹⁰ corresponde a um site meramente informativo, apresentando de forma interativa os produtos da empresa. A Samsung é uma das maiores empresas mundiais do ramo da tecnologia, sendo que os seus produtos são adquiridos por todo o mundo e em elevada quantidade. A página *web* apresenta os produtos da marca, incluindo telemóveis e *tablets*, televisões e produtos áudio e vídeo, eletrodomésticos e produtos informáticos. Na figura 23, apresenta-se a página inicial deste *website*.

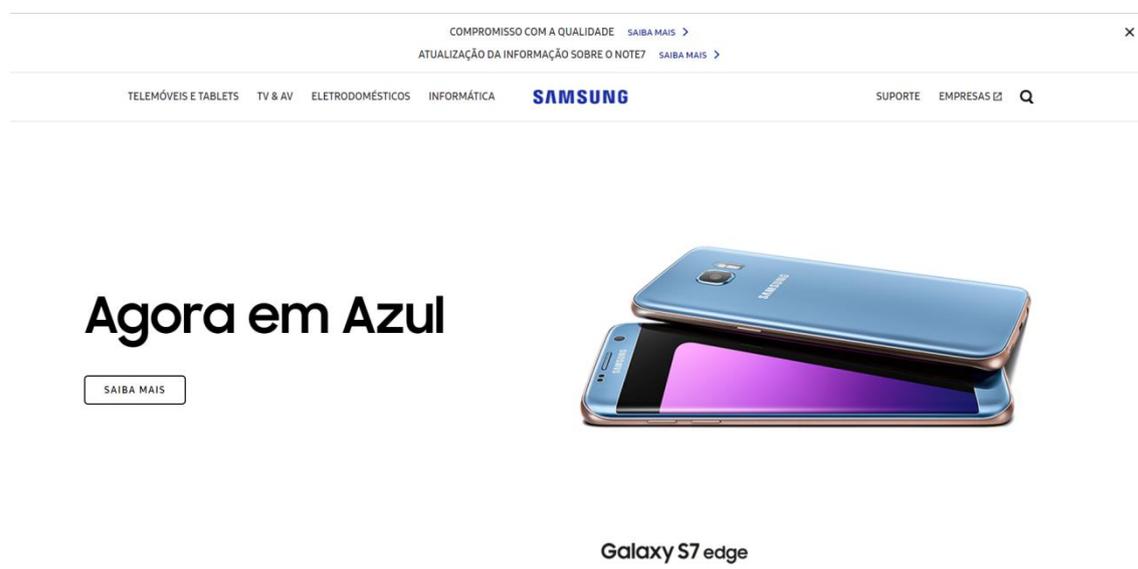


Figura 23: *Website* da Samsung Portugal

¹⁰ <http://www.samsung.com/pt/>

4.5.1 Testes executados

Neste *website* realizaram-se 3 testes, sendo que o primeiro correspondia a atingir a página do “*Samsung Gear VR*” após 2 cliques, o segundo atingir o micro-ondas, o Micro-ondas com grill, 23L, com 3 cliques, e o terceiro atingir o Micro-ondas de Convecção com *Slim Fry*, 28L, após 2 cliques. Os dados de entrada e os resultados estão resumidos nas tabelas 15 e 16, respetivamente.

	Página de destino	Número cliques
1	http://www.samsung.com/pt/wearables/gear-vr-r323/	2
2	http://www.samsung.com/pt/cooking-appliances/microwave-oven-grill-mg23k3515as/	3
3	http://www.samsung.com/pt/cooking-appliances/microwave-oven-convection-mc28h5135ck/	2

Tabela 15: Resumo dos dados de entrada do caso de estudo III

	Tempo de execução	Nº cliques	Resultado
1	4 minutos e 16 segundos	1	REACH
2	7 horas, 13 minutos e 14 segundos	3	REACH
3	18 minutos e 34 segundos	1	REACH

Tabela 16: Resumo dos resultados do caso de estudo III

4.5.2 Análise aos Resultados

Os testes realizados neste caso de estudo permitiram comprovar que o *website* da Samsung tem uma hierarquia bem definida de produtos, algo que pode ser visível comparando os testes 2 e 3, uma vez que a pesquisa por um mesmo tipo de produto, um micro-ondas, leva no caso do teste 2, seja necessário apenas 1 clique para atingir a página desejada, enquanto no teste 3, são necessários 3 cliques. No caso do teste 1, a página do *wearable* é encontrada após 1 cliques. A execução do teste 2, demora bastante tempo, uma vez que, foi necessário percorrer todos os produtos que se encontram a uma distância de 3 cliques da *homepage*, sendo necessário percorrer uma elevada quantidade de páginas. Há, assim, o cuidado de quem concebeu o *site*, de criar uma hierarquia de páginas, estando mais

facilmente acessíveis páginas relacionadas com produtos mais procurados e que poderão ser acedidos de uma forma mais rápida pelo utilizador, sendo que aqueles produtos considerados de menor popularidade, se encontram em níveis de maior profundidade.

4.6 Resumo e Conclusões

Os três casos de estudo realizados permitiram entender e comprovar a utilização do *PBUT*, em várias plataformas *web*, comprovando que o teste implementado é genérico. Foi permitido concluir que todos os *websites* testados, tinham um menu estático, que é acessível à medida que se vai navegando pelas páginas, o que permite que as páginas acedidas a partir deste menu possam ser atingidas em cada página do *website*. Comprova-se então a situação de que, as páginas mais importantes e que, portanto, são mais acedidas são acessíveis a partir da maior parte das páginas da plataforma. Assim, os *websites* analisados estiveram de acordo com a norma de usabilidade testada, tendo havido o cuidado de facilitar em número de cliques o acesso à maior parte das páginas, deixando apenas para segundo plano, páginas que fossem menos acedidas. O estudo permitiu também entender o funcionamento do *PBUT*, bem como entender que pode ser usado em qualquer *website*, tal como era objetivo inicial proposto pela aplicação. A variável de tempo de execução que foi analisada em cada teste de cada caso de estudo foi utilizada para entender e poder comparar os *websites*, em termos do número de páginas que cada um incluía, uma vez que, havendo mais páginas para aceder, o tempo de execução será maior para uma mesma profundidade e permitiu compreender se a maior parte das páginas era acessível até um número máximo de cliques, quando o tempo de execução era elevado após a exploração de todos os *links* até um determinado nível.

Capítulo 5

Conclusões e Trabalho Futuro

Nesta dissertação, foi apresentada, de uma forma detalhada, uma nova abordagem de testes de *software*, no âmbito dos testes de usabilidade automatizados em plataformas *web*. Foi definido um padrão de teste a implementar, para que pudesse realizar o teste na generalidade dos *websites*. O teste simula a navegação no *site*, passando pelas diversas páginas *web*, nele contidas, até encontrar a página pretendida. Este trabalho permitiu construir uma aplicação que realiza testes de usabilidade de uma forma automática, isto é, sem intervenção direta do utilizador, sem que exista uma avaliação da usabilidade do *website*, com base em opiniões das pessoas, que muitas vezes são abstratas e dependem da facilidade com que as pessoas fazem uso dos meios informáticos que utilizam para aceder aos *websites*. Desta forma, será permitido no futuro adicionar mais modelos de teste ou a introdução de pequenas melhorias que otimizem o processo de teste implementado.

Além do projeto que foi desenvolvido, foi publicado um artigo na conferência *A-MOST* 2017, com o título “*Pattern Based Usability Testing*” referido em [30].

De seguida, apresenta-se um contraponto entre o trabalho que foi realizado e o trabalho que era proposto inicialmente, permitindo saber se os objetivos foram satisfeitos ou não, e o trabalho futuro, permitindo conhecer o tipo de funcionalidades que podem ser adicionadas ao *PBUT*.

5.1 Satisfação dos Objetivos

Em relação aos objetivos propostos por esta dissertação, eles incluíam a criação de uma aplicação semelhante ao *PBGT*, que realizasse testes de usabilidade automatizados, servindo

como ponto de partida para uma aplicação que no futuro conseguisse realizar testes de usabilidade aplicando já várias normas de usabilidade.

Nesta dissertação, a aplicação foi criada, com base no *PBGT*, aproveitando dessa aplicação já existente a organização e forma de executar o programa, tendo sido alterada a paleta de botões e a estratégia de teste implementada, uma vez que, a abordagem do *PBUT*, é diferente, tendo objetivos de teste diferentes. Efetivamente, apenas foi implementado um padrão de usabilidade, uma vez, que houve uma certa dificuldade de implementação, para que a aplicação pudesse executar os diversos *websites*. Foi, assim, necessário um estudo aprofundado do código *HTML* de várias aplicações *web*, que permitisse entender certos comportamentos destas, permitindo adaptar o padrão executado pelo *PBUT*, a qualquer uma dessas aplicações, evitando certos erros de execução. Assim, o resultado final acaba por ser adaptado à forma como é construído o *website*, permitindo que o programa tenha o comportamento esperado na execução dos testes.

5.2 Trabalho Futuro

Em termos de trabalho futuro, há algumas melhorias que podem ser inseridas no padrão implementado, mas que não serão muito fáceis, tendo em conta a diversidade de *websites* que existe, e que consiste em, à medida que a navegação for simulada, testar proximidade entre a página atual e a página de destino e conforme a proximidade entre os *URL's* das 2 páginas, tentar navegar sempre para páginas com maior proximidade da página de destino evitando passar por todas as páginas da aplicação. É uma otimização que poderá ser útil e acelerar o processo de teste, mas que poderá ser bastante difícil de implementar, uma vez que nem todos os *websites*, tem um menu principal, e a secção desse menu em que se navega pertence ao *URL*, sendo um trabalho que poderá ser bastante complicado.

Outro das abordagens para o futuro, poderá ser a implementação de mais alguma das boas práticas de usabilidade apresentada em 2.4, no âmbito da navegação, uma vez que a simulação da navegação já é realizada, poderão ser avaliados outros aspetos à medida que o *PBUT* vai executando. Esta já parece uma abordagem mais simples de realizar.

Referências

- [1] ISO. (1998). *International Standard ISO 9241-11*. online em julho de 2016, de *International Organization for Standardization (ISO)*: <https://www.iso.org/obp/ui/#iso:std:iso:9241:-11:ed-1:v1:en>
- [2] Rodrigo Moreira, *PhD Dissertation: "Pattern-Based GUI Testing"*, Faculdade de Engenharia da Universidade do Porto, julho de 2015.
- [3] Rodrigo M. L. Moreira and Ana C. R. Paiva, "*PBGT Tool: An Integrated Modeling and Testing Environment for Pattern-Based GUI Testing*", *29th IEEE/ACM International Conference on Automated Software Engineering (ASE 2014)*. September 15 - 19, Västerås, Sweden, 2014
- [4] Rodrigo Moreira, Ana Paiva and Atif Memon, "*A Pattern-Based Approach for GUI Modeling and Testing*", in *ISSRE'13 - The 24th IEEE International Symposium on Software Reliability Engineering, Pasadena, CA*, 2013
- [5] Tiago Monteiro, *MSc Dissertation: "Ambiente gráfico de modelação para DSL"*, Faculdade de Engenharia da Universidade do Porto, julho de 2012
- [6] Atterer, R., et al. (2006). *Knowing the user's every move: user activity tracking for website usability evaluation and implicit interaction*. *Proceedings of the 15th international conference on World Wide Web*. Edinburgh, Scotland, ACM: 203-212.
- [7] *m-pathy*, online em julho de 2016, <http://www.m-pathy.com/cms>
- [8] Carta, T., et al. (2011). *Web Usability Probe: A Tool for Supporting Remote Usability Evaluation of Web Sites*. *Human-Computer Interaction – INTERACT 2011: 13th IFIP TC 13 International Conference, Lisbon, Portugal, September 5-9, 2011, Proceedings, Part IV*. P. Campos, N. Graham, J. Jorge et al. Berlin, Heidelberg, Springer Berlin Heidelberg: 349-357.

Referências

- [9] *CrazyEgg*, online em julho de 2016, <https://www.crazyegg.com/overview>
- [10] *WebTango*, online em julho de 2016, <http://webtango.berkeley.edu>
- [11] *EyeTracking*, online em julho de 2016, <http://www.eyetracking.com/About-Us/What-Is-Eye-Tracking>
- [12] *A/B Testing*, online em julho de 2016, <https://vwo.com/ab-testing/>
- [13] *AttrakDiff*, online em julho de 2016, <http://attrakdiff.de/>
- [14] Speicher, M., et al. (2014). *Ensuring Web Interface Quality through Usability-Based Split Testing. Web Engineering: 14th International Conference, ICWE 2014, Toulouse, France, July 1-4, 2014. Proceedings. S. Casteleyn, G. Rossi and M. Winckler. Cham, Springer International Publishing: 93-110.*
- [15] *Optimizely*, online em julho de 2016, <https://www.optimizely.com/ab-testing/>
- [16] Cassino, R., et al. (2015). "Empirical validation of an automatic usability evaluation method." *Journal of Visual Languages & Computing* 28: 1-22.
- [17] Nebeling, M., Speicher, M., Norrie, M.C.: *W3Touch: Metrics-based Web Page Adaptation for Touch. In: Proc. CHI (2013).*
- [18] A. Rama and S. Vignesh Dhanraj (2014). *Web Usability Testing Using Clear Methodology. In Middle-East Journal of Scientific Reserach* 20(4): 475-478, 2014. ISSN 1990-9233.
- [19] Speicher, M., et al. (2015). *Inuit: The Interface Usability Instrument. Design, User Experience, and Usability: Design Discourse: 4th International Conference, DUXU 2015, Held as Part of HCI International 2015, Los Angeles, CA, USA, August 2-7, 2015, Proceedings, Part I. A. Marcus. Cham, Springer International Publishing: 256-268.*
- [20] Qadoumi, B. a. and B. a. Al-Shurufat (2015). *Towards Evaluation Method of Usability Engineering for Web Application Sites Using 3D Approach. Proceedings of the International Conference on Intelligent Information Processing, Security and Advanced Communication. Batna, Algeria, ACM: 1-5.*
- [21] <http://www.cmis.brighton.ac.uk/research/patterns/home.html> , online em julho de 2016.
- [22] The Usability Group at the University of Brighton, UK. *The Brighton Usability Pattern Collection.* online em julho de 2016, <http://www.cmis.brighton.ac.uk/research/patterns/home.html>
- [23] *User Focus. Usability Guidelines*, online em julho de 2016, <http://www.userfocus.co.uk/resources/guidelines.html>
- [24] *UI Patterns, Design Patterns*, online em julho de 2016, <http://ui-patterns.com/patterns/>
- [25] *Selenium*, online em janeiro de 2017, <http://www.seleniumhq.org/>
- [26] *Sigarra da Universidade do Porto*, online em janeiro de 2017, https://sigarra.up.pt/up/pt/web_page.inicial
- [27] *SiFEUP*, online em janeiro de 2017, https://sigarra.up.pt/feup/pt/web_page.inicial
- [28] University of Oxford, online em janeiro de 2017, <http://www.ox.ac.uk/>

Referências

- [29] Samsung Portugal, *online* em janeiro de 2017, <http://www.samsung.com/pt/>
- [30] Fernando Dias, Ana C. R. Paiva, “*Pattern Based Usability Testing*”, in *the 13th Workshop on Advances in Model Based Testing (A-MOST), March 17th, 2017, Tokyo, Japan*