MINISTÉRIO DA CIÊNCIA, TECNOLOGIA, INOVAÇÕES E COMUNICAÇÕES
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

TÉCNICAS DE AVALIAÇÃO DE USABILIDADE DE APIS APLICADAS FRAMEWORKS BASEADOS EM METADADOS

RELATÓRIO FINAL DE PROJETO DE INICIAÇÃO CIENTÍFICA
(PIBIC/INPE/CNPq)

Thomas Augusto Teixeira de Albuquerque (Universidade Federal de São Paulo, Bolsista PIBIC/CNPq)
E-mail: tealbthomas@gmail.com

Prof. Dr. Eduardo Martins Guerra (CTE/LAC, Orientador)
E-mail: eduardo.guerra@inpe.br

COLABORADORES

Dr. Tiago Silva da Silva (DCT/UNIFESP)

À minha família, que sempre me dá todo o apoio necessário

Sumário

1 Introdução 15
1.1 Objetivos 15
1.1.1 Objetivo Geral 15
1.1.2 Objetivos Específicos 15

2 Fundamentação Teórica 17
2.1 Usabilidade de APIs 17
2.1.1 Usabilidade 17
2.1.2 Application Programming Interface (API) 18
2.1.3 Usabilidade de APIs 18
2.2 TLD vs TDD 19
2.3 Eye Tracking 19

3 Materiais e Métodos Utilizados 23
3.1 Revisão da Literatura 23
3.1.1 Planejamento 23
3.1.2 Condução 23
3.2 Teste de Usabilidade 24
3.2.1 Tarefa dos participantes 24
3.2.2 Procedimentos e Instrumentação 25

4 Análises e Resultados 27
4.1 Revisão da Literatura 27
4.1.1 Estudos teóricos sobre usabilidade de APIs 29
4.1.2 Estudos que mensuram a usabilidade de uma ferramenta 31
4.1.3 Estudos teóricos sobre Eye Tracking 34
4.1.4 Estudos de uso de Eye Tracking para melhorar algum aspecto de usabilidade 34
4.2 Teste de Usabilidade 35
4.2.1 Gráficos gerados com dados do Eye Tracker - ADI-1 36
4.2.1.1 Fixation Count 36
4.2.1.2 Total Fixation Duration 36
4.2.1.3 Total Visit Duration 37
4.2.2 Linhas do tempo de transições de cada participante - ADI-2 38
4.2.2.1 Primeiro Participante - P01 38
4.2.2.2 Segundo Participante - P02 38

4.2.2.3 Terceiro Participante - P03 39
4.2.2.4 Quarto Participante - P04 39
4.2.2.5 Quinto Participante - P05 39
4.2.2.6 Sexto Participante - P06 40
4.2.2.7 Sétimo Participante - P07 40
4.2.2.8 Oitavo Participante - P08 40
4.2.3 Média do número de transições para todos participantes - ADI-2 41
4.2.4 Tempo de duração de cada caso de teste para cada participante 44

5 Conclusões 49

Referências 51

Lista de ilustrações

Figura 1 – Eye tracker fixado na parte inferior de um monitor comum de computador experimento. 20
Figura 2 – Estrutura básica do olho humano. 20
Figura 3 – Olho humano o qual é possível ver a posição da reflexão da luz ambiente pela córnea (ponto branco), e a posição da pupila (região preta no centro do olho). 21

Figura 4 – Criação de tarefas no UEMan. 33
Figura 5 – Alterando o fator de avaliação da tarefa pelo UEMan. 33
Figura 6 – Gráfico de quantidade de fixações para cada participante. 36
Figura 7 – Gráfico de duração total das fixações para cada participante. 37
Figura 8 – Gráfico de duração total das visitas para cada participante. 38
Figura 9 – Linha do tempo das transições do primeiro participante - P01. 38
Figura 10 – Linha do tempo das transições do segundo participante - P02. 39
Figura 11 – Linha do tempo das transições do terceiro participante - P03. 39
Figura 12 – Linha do tempo das transições do quarto participante - P04. 39
Figura 13 – Linha do tempo das transições do quinto participante - P05. 40
Figura 14 – Linha do tempo das transições do sexto participante - P06. 40
Figura 15 – Linha do tempo das transições do sétimo participante - P07. 40
Figura 16 – Linha do tempo das transições do oitavo participante - P08. 41
Figura 17 – Média das transições TDD 41
Figura 18 – Média das transições TLD 42
Figura 19 – Média das transições TDD vs TLD 43
Figura 20 – Total de Transições para Todas as Seções 44
Figura 21 – Duration Time of Test Case 1 for each participant 45
Figura 22 – Duration Time of Test Case 2 for each participant 46
Figura 23 – Duration Time of Test Case 3 for each participant 47
Figura 24 – Duration Time of Test Case 4 for each participant 48

Lista de tabelas

Tabela 1 – Tabela com os termos de busca referentes a cada tópico abordado. 24
Tabela 2 – Entradas e Saídas esperadas para os casos de teste do método de conversão CamelCase 25
Tabela 3 – Artigos selecionados para composição da revisão sistemática 28
Tabela 4 – Tabela com o número de artigos referentes a cada abordagem 29

Lista de abreviaturas e siglas

IDE	*Integrated Development Environment. Tradução: Ambiente de Desenvolvimento Integrado*
API	*Application Programming Interface*
TCC	*Trabalho de Conclusão de Curso*
UNIFESP	*Universidade Federal de São Paulo*
UCD	*User-Centered Design. Tradução: Design Centrado no Usuário*
GUI	*Graphical User Interface. Tradução: Interface Gráfica do Usuário. São interfaces gráficas usadas para interação do usuário com a aplicação.*
TDD	*Test Driven Development. Tradução: Desenvolvimento Dirigido a Teste.*
TLD	*Test-Last Development. Tradução: Desenvolvimento "Teste por Último".*

Resumo

A usabilidade de APIs (Application Programming Interface) é vista como um tema de extrema relevância na computação, pois busca melhorar a interação do usuário com esse componente computacional. Dessa forma, torna-se importante que soluções novas sejam elaboradas para que haja ferramentas que busquem melhorias na usabilidade de APIs. Estas soluções são pertinentes de serem pesquisadas pela comunidade científica. Entretanto, pouco tem sido pesquisado em relação a este tema. Este trabalho foca no estudo de Usabilidade de APIs e sua avaliação via Eye Tracking, através de testes de usabilidade controlados com usuários. Neste trabalho foi aplicado um teste de usabilidade em programadores com a finalidade de comparar a eficiência no desenvolvimento de métodos e testes de unidade na linguagem Java, utilizando duas abordagens: TDD e TLD. Os usuários codificaram os testes de unidade utilizando a API JUnit 5.

Palavras-Chaves: Usabilidade de API, Eye Tracking, TDD, TLD, API, Usabilidade, Teste de Usabilidade.

1 Introdução

A reutilização de código-fonte em diferentes domínios é um processo chave para acelerar a produtividade de equipes de desenvolvimento. Nesse sentido, a abstração e a modularidade estimularam o desenvolvimento de APIs e frameworks, que são utilizados para simplificar a elaboração de aplicações. Nesse contexto, foram surgindo diversas API's e frameworks ao longo do tempo, entre elas as relacionadas com testes de software, uma área de grande importância no cenário da computação em geral.

Junto com as API's de testes, surgiram também diferentes abordagens de como realizar os testes. Cada técnica ganha adeptos que buscam através delas melhorarem seus processos de trabalho, buscando eficácia e, principalmente, eficiência.

As técnicas de desenvolvimento com testes podem variar de diversas formas, como a ordem em que o programador escreve os casos de teste, podendo ser antes, durante ou depois da implementação dos métodos do programa.

Neste projeto foram analisados, através de um teste de usabilidade, os desempenhos de 8 participantes que foram propostos a escrever os mesmos métodos e casos de teste, porém metade deles deveria escrevê-los utilizando uma técnica, e a outra metade utilizando outra técnica. A análise possui a finalidade de de revelar resultados relevantes sobre TDD e TLD, comparadas levando em consideração seus desempenhos.

1.1 Objetivos

1.1.1 Objetivo Geral

O principal objetivo dessa Iniciação Científica será realizar o estudo sobre a usabilidade de APIs e frameworks e comparar duas abordagens de teste de software tendo como base um teste de usabilidade em uma API de testes de unidade.

1.1.2 Objetivos Específicos

Realizar uma revisão da literatura acerca de Usabilidade de APIs e Frameworks, e técnicas de avaliação da experiência do usuário.

Aplicar um teste de usabilidade a programadores a fim de comparar o desempenho das técnicas de desenvolvimento de software orientadas a teste: TDD e TLD. O teste de usabilidade terá como base a API de testes de unidade JUnit.

As técnicas envolvem a análise com base nas atitudes do usuário, que são medidas

através de expressões faciais e também da trajetória do olhar do usuário. Esta será medida através de um equipamento específico para isso, denominado eye tracker.

2 Fundamentação Teórica

Serão apresentados neste capítulo conceitos referentes a assuntos que serão abordados neste Trabalho. Os assuntos foram selecionados com base nos objetivos específicos, e são expostos neste capítulo de maneira que o leitor tenha conhecimento teórico suficiente para situar-se adequadamente no tema.

2.1 Usabilidade de APIs

2.1.1 Usabilidade

O termo usabilidade refere-se à experiência do usuário em relação ao uso prático de alguma aplicação, que pode ser um site, aplicativo móvel, software, etc. A experiência é medida através da aplicação de testes de usabilidade, onde são utilizadas métricas pré-definidas com o intuito de analisar o comportamento do usuário perante o sistema. As técnicas de análise podem ser várias, como o tempo necessário para um usuário finalizar uma determinada tarefa a ele imposta, bem como suas dificuldades na realização das mesmas captadas por entrevista após a realização do teste, ou mesmo medindo para onde o usuário olhou em cada momento. Com estas análises, é possível promover a melhoria da interação do usuário com a aplicação.

A seguir estão alguns aspectos a serem considerados na avaliação da usabilidade de um sistema (USABILITY, 2017). Boas práticas de usabilidade buscam melhorá-lo nesses aspectos.

- **Utilidade**: Ele deve ser original e cumprir uma necessidade;
- **Usável**: Deve ser fácil de usar;
- **Desejável**: Imagem, identidade, marca e outros elementos de design são usados para causar emoções específicas e apreciação;
- **Achável**: Conteúdo deve ser navegável e localizável dentro e fora do sistema;
- **Acessível**: Conteúdo deve ser acessível para pessoas com deficiências e
- **Acreditável**: Usuários devem confiar no que o sistema mostra a eles.

Observa-se que cada aspecto a ser avaliado em uma aplicação deve levar em consideração os perfis dos possíveis tipos de usuários daquela aplicação, que são usualmente denominados **personas**. Por exemplo, uma API pode ser projetada para abranger usuários com nível básico de uma linguagem de programação, e também para usuários avançados da mesma linguagem. Neste caso, a API foi projetada para duas personas.

2.1.2 Application Programming Interface (API)

Uma API se consiste em um conjunto de sub-rotinas, protocolos e ferramentas que facilitam a construção de softwares. Ela geralmente engloba funcionalidades que pertencem a um mesmo contexto, para que o programador a use para uma finalidade específica dentro de sua aplicação.

Esta possibilidade de reuso é muito benéfica para a comunidade em geral, pois permite que códigos úteis gerados por outros programadores sejam "empacotados" , sendo utilizados muitas vezes somente com chamadas simples das rotinas da API. Dessa forma, o programador usuário não precisa saber detalhes da implementação da API em si, mas somente seu funcionamento.

2.1.3 Usabilidade de APIs

A usabilidade se estende também às APIs, que apesar de não se caracterizarem como softwares para usuários comuns - que costumam ter interfaces gráficas (GUI's), caso em que a organização visual se torna o fator principal na usabilidade -, elas podem ser avaliadas em termos de usabilidade pois também são planejadas para usuários, que neste caso são os próprios desenvolvedores.

De acordo com (CLARKE, 2004), a usabilidade da API impacta na qualidade do desenvolvimento do software, pois o programador não quer perder tempo desnecessário usando uma ferramenta que já deveria estar funcionando de forma eficaz e eficiente, inclusive no quesito usabilidade. Os desenvolvedores da API devem modelá-la para que o usuário consiga utilizar intuitivamente a API e, dada uma persona específica, conseguir atingir o resultado da melhor maneira possível, e da forma com que o usuário espera que seja atingido. O projetista deve garantir que está ciente das características dos usuários e como elas impactam na maneira como eles esperam que a API funcione.

Pode-se notar também ao avaliar a usabilidade de APIs que características simples, como nomes para funcionalidades, podem aumentar o tempo de aprendizagem da API, algo que necessita ser evitado. Isso revela uma particularidade, que é o fato de que muitas deficiências podem passar despercebidas na usabilidade da API, já que muitos dos problemas estão em sintaxes de código, o que torna mais difícil de se analisar em comparação com interfaces gráficas, com botões, cores, fontes diferentes, etc. De forma geral, o design da API deve fazer com que operações simples sejam simples, e operações complexas sejam possíveis (PICCIONI, 2013).

2.2 TLD vs TDD

Práticas de teste tem ganhado destaque na última década, devido principalmente ao aumento da cobrança de eficiência dos produtos, que será aumentada caso seja devidamente testado. E o aumento desta cobrança vem dos próprios clientes, pois com cada vez mais aplicações no mercado, características do software como seu desempenho são cruciais para que usuários optem ou não por ele.

São apresentadas, então, duas técnicas de aplicação de teste mais usadas no desenvolvimento de um software:

1. **TLD**: Test-Last Development. Tradução: Desenvolvimento "Teste por Último".
2. **TDD**: Test Driven Development. Tradução: Desenvolvimento Dirigido a Teste.

O TLD se consiste em primeiro escrever o código normalmente, até que o programador acredite que ele esteja correto. Em seguida, ele escreve os casos de teste (entradas e saídas esperadas) e os executa, a fim de verificar se o programa faz o que o programador espera que fizesse. Esta técnica é a mais comum e surgiu junto com o conceito geral de teste. Já a técnica TDD requer que o programador primeiro escreva os casos de teste, e depois faz a implementação dos métodos para passar naquele teste.

Muitos especialistas em teste de software afirmam que o TDD se sobressai positivamente em relação ao TLD no sentido de que, uma vez que se o programador escreve os testes antes, ele consegue abstrair melhor como deverá ficar o método implementado. Já outros afimam que o TDD é mais demorado, pois requer que o programador gaste muito tempo planejando, e no final acaba gastando o mesmo tempo para escrever o método que alguém que tenha feito por TLD.

Este panorama geral é apresentado por (MUNIR, 2014), que em seu estudo foca em aplicar um teste de usabilidade com profissionais para comparar qual das duas técnicas é mais eficiente. E seus resultados mostram que em termos de teste - cobertura das linhas de código, número de métodos implementados por horas trabalhadas e quantidade de casos de teste que passaram - o TLD e o TDD apresentam resultados bem semelhantes. Entretanto, aqueles que utilizaram TDD apresentaram códigos mais limpos e organizados - o que corresponde com a ideia de que ao utilizar TDD, o programador abstrai melhor o problema e elabora mentalmente soluções organizadas. O autor conclui também que a maioria dos desenvolvedores preferem o TLD, dado o menor nível de esforço necessário para aplicar o TDD comparado com o TLD.

2.3 Eye Tracking

Um eye tracker é um dispositivo que permite rastrear a movimentação do olhar de um usuário em uma tela digital. O dispositivo deve ficar de frente ao usuário, enquanto ele realiza

alguma atividade olhando uma tela, que geralmente é um monitor comum de computador de mesa ou notebook. A Figura 1 abaixo mostra um eye tracker fixado a uma monitor.

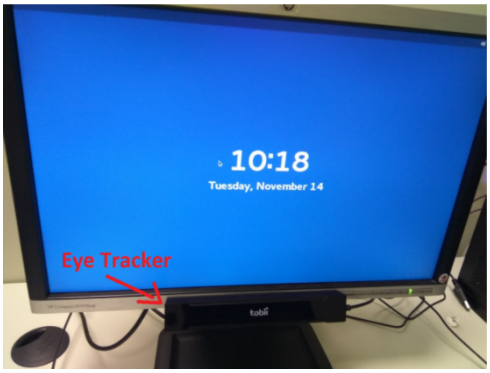

Figura 1 – Eye tracker fixado na parte inferior de um monitor comum de computador experimento.

Toda a movimentação dos olhos é captada através de ondas infravermelha emitidas pelo eye tracker em direção aos olhos do indivíduo, que são refletidas e captadas de volta pelo aparelho. Dessa maneira, consegue-se saber o posicionamento da pupila e da córnea de cada olho do usuário. A Figura 2 abaixo mostra a estrutura básica de um olho humano, tirada de (BOJKO, 2013).

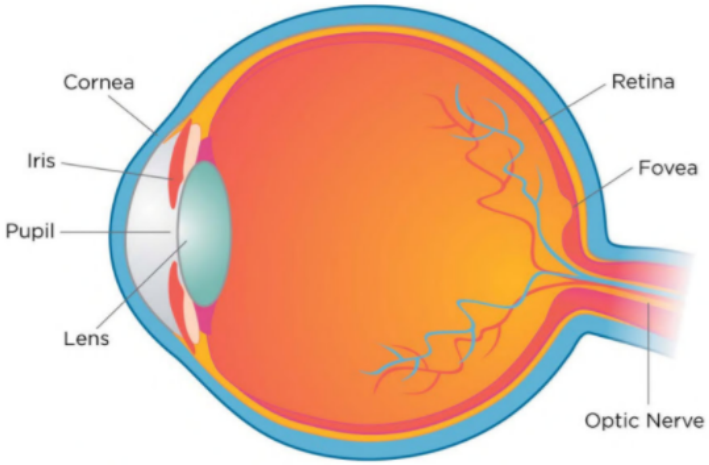

Figura 2 – Estrutura básica do olho humano.

Como pode-se observar, a pupila corresponde ao “centro” do olho, e está sempre apontada para o ponto exato onde o indivíduo está olhando, pois é por ela que todo o conteúdo visual externo é captado pelo olho. Já a córnea corresponde à parte transparente da frente do olho. E o motivo do eye tracker captar essas duas partes do olho é que o cálculo da direção do olhar, ou seja, da posição da pupila, é feito em relação à posição da reflexão da luz ambiente que incide sobre a córnea, como pode-se observar pelo ponto branco na Figura 3 abaixo retirada também de (BOJKO, 2013).

Figura 3 – Olho humano o qual é possível ver a posição da reflexão da luz ambiente pela córnea (ponto branco), e a posição da pupila (região preta no centro do olho).

O cálculo da direção do olhar comparando a posição da pupila em relação à reflexão da luz ambiente na córnea também é interessante pois o eye tracker consegue corrigir eventuais mudanças da posição da cabeça da pessoa, já que quando a cabeça é movimentada, a reflexão na córnea não muda significativamente - isso para movimentos naturais e suaves da cabeça da pessoa.

Estas características fazem com o que os eye trackers sejam robustos e muito funcionais, se tornando uma poderosa ferramenta de avaliação do comportamento do usuário. A sua utilidade dentro deste projeto será considerável, já que permitirá rastrear o olhar dos usuários sobre os códigos das APIs e tirar conclusões relevantes sobre estes resultados.

3 Materiais e Métodos Utilizados

3.1 Revisão da Literatura

Este Capítulo visa mostrar como foi feita a Revisão da Literatura no contexto deste Projeto. O Planejamento e a Condução foram elaborados conforme as orientações sugeridas por (KITCHENHAM, 2004).

3.1.1 Planejamento

Foi planejado para este trabalho a realização de uma pesquisa na literatura sobre Usabilidade de APIs, área vista como relevante no contexto computacional, e que ainda foi pouco explorada pela comunidade científica.

A seguir estão os três tópicos principais aos quais foram planejados para fazer a Revisão.

- Usabilidade de APIs
- Eye Tracking e usabilidade
- TDD vs TLD

3.1.2 Condução

Após concluído o planejamento, com os objetivos definidos e um caminho traçado para atingi-los, iniciou-se a condução da Revisão da Literatura. A condução visa selecionar os artigos dentro das áreas estabelecidas, analisá-los, extrair dados e sintetizar novos, conforme foi citado na Seção 1.3. Nesta Seção será apresentada a metodologia de extração dos artigos. O restante será explanado no capítulo referente aos resultados.

Para a extração dos artigos, primeiramente foi definida a biblioteca de pesquisa de documentos científicos para que fosse feita a busca dos mesmos. Foi definido que iria ser usada a biblioteca Scholar do Google (SCHOLAR, 2017), por ser gratuita e vasta, já que outras grandes bibliotecas, incluindo a ACM e a IEEE Xplore, são indexadas pela Scholar, ou seja, é possível encontrar artigos nestas bibliotecas por meio da Scholar. Todos os artigos foram encontrados usando a Scholar, porém extraídos de diversas bibliotecas.

Para definir quais seriam os termos usados nas buscas, foram considerados os três tópicos apresentados na Seção anterior (3.1). A Tabela 1 mostra os termos de busca para cada tópico.

Tópico abordado	*Termo de Busca*
Usabilidade de APIs	api usability
Eye Tracking e Usabilidade	eye tracking in software engineering eye tracking in software traceability
TDD vs TLD	test driven development test last development test driven development vs test last development

Tabela 1 – Tabela com os termos de busca referentes a cada tópico abordado.

Após a definição dos termos de busca referentes a cada tópico abordado, foram elaborados critérios de inclusão dos artigos levantados pela busca, a fim de selecionar apenas artigos relevantes no contexto do trabalho. Os critérios são os numerados abaixo.

1. Contém algum dos termos de busca
2. Artigos com abordagem julgada relevante a este projeto
3. Artigos escritos em inglês ou português
4. Número de citações - considerado porém não como critério excludente
5. Quantidade de artigos já selecionados - buscou-se escolher uma quantidade adequada de artigos. Atingida essa quantidade, parou-se a busca.

3.2 Teste de Usabilidade

Os participantes foram propostos a escrever um programa na linguagem Java, a fim de avaliar as diferenças entre os dois tipos de desenvolvimento explicados previamente. Para isso, utilizou-se de alguns procedimentos e equipamentos específicos. Foram escolhidos 8 programadores da disciplina de Desenvolvimento Ágil do INPE para a participação do teste. Abaixo está detalhada a tarefa que foi pedida aos participantes, a também os procedimentos e instrumentações.

3.2.1 Tarefa dos participantes

Os participantes deveriam escrever um programa que fizesse uma conversão de strings. As palavras dadas como entrada deveriam ser necessariamente do tipo CamelCase, e a saída deveria ser uma lista de instâncias de palavras comuns em português. Os métodos na implementação deveriam aplicar a conversão de cada um dos casos de entrada/saída mostrados na Tabela X abaixo. Metade do grupo realizou as tarefas utilizando TLD, e a outra metade utilizando TDD. Cada um foi previamente selecionado e orientado a utilizar uma das duas formas

de desenvolvimento guiado a testes, e foi planejado a todos terem um tempo de no máximo 1 hora e 15 minutos para a fazer a implementação e os testes de unidade do método.

A Tabela 2 contém exemplos de entrada e saída do método.

Entradas e Saídas

Entrada em CamelCase	*Saída em lista de Strings*
nome	“nome”
Nome	“nome”
nomeComposto	“nome”, “composto”
recupera10Primeiros	“recupera”, “10”, “primeiros”

Tabela 2 – Entradas e Saídas esperadas para os casos de teste do método de conversão CamelCase

3.2.2 Procedimentos e Instrumentação

Para coletar as interações dos participantes, foi necessário uma configuração que proporcionasse uma visualização remota pelo administrador, através de uma parede de vidro. O administrador, então, assistiu os 8 participantes em tempo real através de um computador - CA (Computador do administrador) - em um lado da parede de vidro, enquanto cada participante estava realizando a tarefa em dois outros computadores no outro lado. Cada computador usado pelo participante tinha uma funcionalidade, como é mostrado abaixo.

1. Computador 1 do Participante (CP1): Usado para escrever métodos em Java e testes de unidade. A classe que deveriam conter os métodos foram propositalmente posicionadas no lado esquerdo do monitor, enquanto a classe que deveria conter os testes de unidade foram posicionadas no lado direito.

2. Computador 2 do Participante (CP2): O CP2 estava disponível para os participantes verem a especificação do problema, e também para pesquisarem na web, se fosse necessário, alguma informação referente a sintaxes Java ou sobre fundamentos de computação - desde que não fosse nada sobre a lógica do programa requisitado.

Toda a captura de vídeo dos participantes e transmissão para o CA foi feita através do software Morae©, que capturava a imagem por uma webcam instalada no CP1. Outro software utilizado foi o Tobii Studio©, instalado no CP1, que tem diversas funcionalidades de coleta e análise de dados de captura do eye tracker, instalado também no CP1.

4 Análises e Resultados

Neste capítulo são apresentados os resultados decorrentes da Revisão da Literatura, e do teste de usabilidade.

4.1 Revisão da Literatura

Foram selecionados os seguintes artigos mostrados na Tabela 3 com base nos termos de busca e critérios de seleção apresentados na Seção 3.2.

ID	*Título*	*Autores*	*Ano*
A1	A Case Study of API Redesign for Improved Usability (STYLOS, 2008)	Jeffrey Stylos, Benjamin Graf, Daniela K. Busse, Carsten Ziegle, Ralf Ehre, Jan Karstens	2008
A2	An Empirical Study Assessing the Effect of SeeIT 3D on Comprehension (SHARIF, 2013)	Bonita Sharif, Grace Jetty, Jairo Aponte, Esteban Parra	2013
A3	An Empirical Study of API Usability (PICCIONI, 2013)	Marco Piccioni, Carlo A. Furia, Bertrand Meyer	2013
A4	An experimental evaluation of TDD vs TLD (MUNIR, 2014)	Hussan Munir, Krzysztof Wnuk, Kai Petersen, Misagh Moayyed	2014
A5	API Peer Reviews: A Method for Evaluating Usability of Application Programming Interfaces (FAROOQ, 2010)	Umer Farooq, Dieter Zirkler	2010
A6	Automatic Evaluation of API Usability using Complexity Metrics and Visualizations (SOUZA, 2009)	Cleidson R. B. de Souza, David L. M. Bentolila	2009
A7	Describing and Measuring API Usability with the Cognitive Dimensions (CLARKE, 2006)	Steven Clarke	2006
A8	Eye-tracking Metrics in Software Engineering (SHARAFI, 2015)	Zohreh Sharafi, Timothy Shaffer, Bonita Sharif, Yann-Gaël Guéhéneuc	2015

ID	*Título*	*Autores*	*Ano*
A9	Improving Automated Source Code Summarization via an eye tracker study (RODEGHERO, 2014)	Paige Rodeghero, Collin McMillan, Paul W. McBurney, Nigel Bosch, Sidney D'Mello	2014
A10	Improving the Usability of Eclipse for Novice Programmers (STOREY, 2003)	Margaret-Anne Storey, Jeff Michaud, Marcellus Mindel, Mary Sanseverino, Daniela Damian, Del Myers, Daniel German, Elizabeth Hargreaves	2003
A11	Measuring API Usability (CLARKE, 2004)	Steven Clarke	2004
A12	On the Use of Eye Tracking in Software Traceability (SHARIF, 2011)	Bonita Sharif, Huzefa Kagdi	2011
A13	Padrões de Projeto para Frameworks e Componentes Baseados em Metadados (GUERRA, 2011)	Eduardo M. Guerra, Fernando Pavão, Clóvis T. Fernandes	2008
A14	Towards Automating Fixation Correction for Source Code (PALMER, 2016)	Christopher Palmer, Bonita Sharif	2016
A15	UEMan A Tool to Manage User Evaluation in Development Environments (HUMAYOUN, 2009)	Shah Rukh Humayoun, Yael Dubinsky, Tiziana Catarci	2009

Tabela 3 – Artigos selecionados para composição da revisão sistemática

Para a realização da análise dos artigos, primeiro classificou-se os mesmos em categorias, dispostas de acordo com a abordagem de cada um. Nota-se que esta divisão difere da divisão feita em tópicos anteriores, já que aqui estamos avaliando o conteúdo dos artigos. A Tabela 4 mostra essas abordagens consideradas.

A primeira categoria definida por "Estudos teóricos sobre usabilidade de APIs" contém os artigos em que o autor disserta sobre a usabilidade de APIs, mostrando evidências que comprovam sua relevância. São apresentadas métricas de usabilidade de APIs (CLARKE, 2006) e (SOUZA, 2009), as quais os autores expressam com alguma evidência que suas métricas propostas são eficientes para se medir o quanto uma API é fácil de ser usada. (PICCIONI, 2013) aborda o assunto com um estudo empírico, neste caso um experimento. Ele direciona o planejamento e as métricas de avaliação do experimento para seu objetivo principal, que é dissertar sobre usabilidade de APIs em um contexto geral. Já (FAROOQ, 2010) disserta sobre um método de avaliação de usabilidade baseado em discussões em grupo e em entrevistas com especialistas

Abordagens dos artigos

Abordagem de cada artigo	*Número de artigos*
Estudos teóricos sobre usabilidade de APIs	5
Estudos que mensuram a usabilidade de uma ferramenta	6
Estudos teóricos sobre Eye Tracking	2
Estudos de uso de Eye Tracking para melhorar algum aspecto de usabilidade	1
TDD vs TLD	1

Tabela 4 – Tabela com o número de artigos referentes a cada abordagem

na API e em APIs correlatas. E (GUERRA, 2011) foca em frameworks e componentes baseados em metadados, que têm suas particularidades de uso de APIs.

A segunda categoria definida como "Estudos que mensuram a usabilidade de uma ferramenta" contém artigos sobre análise de usabilidade de um software específico, e como melhorá-la. (SHARIF, 2013) descreve um estudo de usabilidade completo de um plugin Eclipse chamado SeeIT 3D utilizando Eye Tracking. O artigo de (HUMAYOUN, 2009) e o de (DUBINSKY, 2008) propõem um plugin Eclipse para avaliar a usabilidade de um software enquanto ele está sendo desenvolvido. (STOREY, 2003) conduz uma análise da usabilidade da IDE Eclipse para o aprendizado de programadores iniciantes, e apresenta uma ferramenta derivada do Eclipse para atingir essa melhoria no aprendizado. Já (STYLOS, 2008) mede a usabilidade de uma API de desenvolvimento de plataformas de negócio já consolidada no mercado, porém que começou a ter problemas de usabilidade no momento em que desenvolvedores de aplicações genéricas começaram a utilizá-la.

A terceira categoria definida como "Estudos teóricos sobre Eye Tracking contém artigos referentes" ao uso de dispositivo eye tracker na avaliação de softwares (SHARIF, 2011), e métricas de análise de dados gerados por eye trackers (SHARAFI, 2015).

A quarta categoria definida como "Estudos de uso de Eye Tracking para melhorar algum aspecto de usabilidade" contém artigos sobre: automatizar o processo de resumir o funcionamento de um método (RODEGHERO, 2014), e automatizar a correção de posições da tela onde o usuário olhou, captadas por um eye tracker (PALMER, 2016).

A quinta categoria definida como "TDD vs TLD" contém um artigo (MUNIR, 2014), que mostra uma comparação entre as duas formas de desenvolvimento guiado a testes.

4.1.1 Estudos teóricos sobre usabilidade de APIs

(CLARKE, 2004) disserta sobre o tema Usabilidade de APIs de maneira muito abrangente, ou seja, tenta abordar o tema como um todo. Seu texto baseia-se principalmente no

conceito de Dimensões Cognitivas, que são um conjunto de fatores que individualmente ou coletivamente tem um impacto na maneira que os desenvolvedores trabalham com a API, e como eles esperam que a API trabalhe. O ideal é que haja o casamento entre o que o usuário espera que a API faça, com o que ela realmente faça. Nota-se nesse aspecto que existem diversos tipos de personas para uma determinada API, o que torna o desenvolvimento de uma API mais complexa, pois ela precisa atender estas diferentes personas. Abaixo estão algumas dimensões cognitivas apresentadas pelo autor.

- **Nível de abstração**: Os níveis de abstração de uma API comparados com os níveis de abstração usáveis por um desenvolvedor específico.
- **Avaliação progressiva**: Em que medida o código parcialmente concluído pode ser executado para obter feedback sobre o comportamento do código.
- **Expressividade funcional**: grau de dificuldade para interpretar cada seção de código e de saber qual método ou classe de uma API usar para escrever um código.

Experimentos elaborados por (CLARKE, 2006) revelaram que usuários gastavam muito tempo procurando por informações e trechos de código prontos na documentação. E segundo ele, isso é um sinal de que a usabilidade da API está comprometida, já que boas APIs são autoexplicativas, ou seja, o usuário aprende a usá-la na prática.

Já Souza, C. et al propõe a avaliação de APIs baseada em métricas de complexidade, pois afirma que quanto menos complexa é uma API, mais fácil de usar ela é. São aplicadas estas métricas em toda uma classe, gerando um valor de complexidade. As métricas usadas por ele são as seguintes:

- **Interface size**: medida de complexidade baseada no número de parâmetros um método tem, e se estes parâmetros são objetos. Quanto maior estes números, mais complexos são os métodos.
- **Interaction level**: medida do quanto informações fluem para dentro e fora da classe, ou a quantidade de interação potencial direta que pode ocorrer em um sistema, classe ou método.
- **Operation argument complexity**: medida que toma o somatório da complexidade de todos os tipos de parâmetros de um método. Por exemplo, um booleano tem zero de complexidade, um inteiro possui complexidade, etc.

Por mais que Clarke, C. et al e Souza, C. et al usem métricas diferentes, eles se concordam em vários pontos, como por exemplo, que o uso da API para qualquer persona deve ser facilmente aprendido com base principalmente na parte prática, sem ter que voltar tanto para a documentação.

(PICCIONI, 2013) toma disserta sobre usabilidade de APIs a partir de um experimento, que se consistia em um teste de usabilidade aplicado em programadores com diferentes níveis de programação. Suas métricas de avaliação foram as seguintes:

- Entrevista com cada participante após a execução das tarefas
- Análise das expressões manifestadas pelos usuário, chamadas de tokens

Ambas basearam-se nas Dimensões Cognitivas propostas por (CLARKE, 2004) citadas anteriormente. Os tokens eram diferenciados de acordo com cada tipo de expressão manifestada pelo usuário, como expressão de surpresa, de incorreção, etc.

Com essas variáveis e técnicas, foram obtidos resultados qualitativos, referentes às entrevistas, e quantitativos, referentes aos tokens. Os níveis diferentes dos programadores também foi relevante para que a usabilidade da API fosse mensurada com diferentes personas.

Ao final do teste, foram encontradas as seguintes questões referentes à usabilidade da API ABEL:

- Dificuldade em associar nomes de classes, métodos e atributos às funcionalidades;
- Dificuldade em descobrir relação entre tipos (de classes, por exemplo); pelos usuário, chamadas de tokens
- Documentação da API incompleta; e
- Flexibilidade garantida pela abstração bem aceita pelos programadores experientes, porém mal aceita pelos menos experientes.

Indo um pouco contra as técnicas comuns de avaliação de usabilidade que se baseiam em testes de usabilidade, (FAROOQ, 2010) discute em seu artigo um método de avaliação por revisões feitas por usuários da API, que se reúnem e discutem os pontos principais que cada um pensa que deveriam ser melhorados na API. Com o debate de opiniões de usuários, são revelados muitos problemas em diversas Dimensões Cognitivas da API (CLARKE, 2006), que por sua vez não foram reveladas por testes de usabilidade. Geralmente acontecem reuniões entre os projetistas e desenvolvedores da API, mas são poucas as empresas que investem neste tipo de reunião com usuários.

4.1.2 Estudos que mensuram a usabilidade de uma ferramenta

(SHARIF, 2013) teve como objetivo avaliar a usabilidade de um plugin Eclipse de visualização 3D de código, chamado SeeIT 3D. Ela tem a função de mostrar elementos do código, principalmente na forma de containers. A performance dos usuários foi medida através de um questionário individual com os participantes após realizarem todas as tarefas, e também através do rastreamento da direção do olhar dos participantes durante a realização das tarefas, através de

um equipamento denominado eye-tracker. O teste buscou avaliar o quão eficiente é a ferramenta em propor solução em 3 cenários diferentes:

- **Overview**: Usuário precisa ter a visão geral das classes e métodos de seu programa;
- **Bug Fixing**: Usuário deseja eliminar um bug; e
- **New Feature**: Usuário deseja implementar uma nova funcionalidade ao software.

Após os testes de usabilidade, Sharif, B. et al concluiu que a ferramenta é adequada somente para o primeiro cenário, pois sua visualização gráfica facilita a visão geral dos containers. Mas caso o usuário queira alterar partes do código. SeeIT 3D não se mostra eficiente.

(STYLOS, 2008) avalia uma API chamada "Business Rules Framework Plus" API (BRFplus), que é uma das APIs que compõem um programa chamado SAP, voltado para o desenvolvimento de plataformas de negócio. O autor conduziu seu estudo nessa API principalmente pelo fato de que ela estava crescentemente sendo usada por programadores de aplicações genéricas - que não sejam do ramo de negócio - , e que eles estavam tendo problemas voltados a usabilidade. Dessa forma, o autor focou em analisar a diferença do uso da API por essas duas personas.

As análises do estudo foram guiadas por entrevistas com os usuários especialistas que usam a API frequentemente e com os "novos usuários", que se introduziram recentemente no uso da API para o desenvolvimento de aplicações. Os resultados revelam que os "novos usuários" tendem a ser mais pragmáticos, e querem entender o código na medida em que seja produtivo. E no geral, a usabilidade da API se torna mais difícil de ser adaptada para usuários não especialistas e genéricos, pois aumenta-se muito o número de personas diferentes.

Algo parecido ocorreu em (STOREY, 2003), que mostra como foi feito um estudo de adaptação da IDE Eclipse para usuários novatos em Java - aqui os "novos usuários" não só não são especialistas no uso da ferramenta, como também estão no processo de aprendizagem da linguagem de programação requerida pela ferramenta. O estudo se baseou em um tipo de simplificação da IDE Eclipse para que fosse mais fácil usuários novatos aprenderem a usá-la. Foram escondidas funcionalidades desnecessárias para os alunos no momento, e também destacadas aquelas mais importantes. Esta classificação de funcionalidades foi feita com base na opinião de diversos professores da área.

(HUMAYOUN, 2009) afirma em seu estudo que muitas equipes de desenvolvimento testam a usabilidade do software desenvolvido apenas no final do processo, após todas as funcionalidades estarem implementadas. Isto acaba tornando custoso o processo de alterar os códigos, pois o software se encontra muito complexo. Pensando nisso, o autor considera ideal que sejam aplicados testes de usabilidade durante o desenvolvimento do software. Para isso sua solução é uma ferramenta chamada UEMan, a quel é um plugin para a IDE Eclipse. Esta ferra-

menta foi desenvolvida com a finalidade de automatizar o processo de desenvolvimento voltado para o usuário, a partir da execução de tarefas do software.

O avaliador configura pelo plugin quais serão as tarefas a serem executadas (Figura 4), e qual será a formas de avaliação (Figura 5). A Figura X mostra quatro exemplos de tarefas para um software de reprodução de música qualquer. Nessa versão da ferramenta, o autor apresentou 3 fatores para avaliação: o número de clicks do mouse; o número de vezes que alguma tecla foi pressionada; e o tempo gasto pelo usuário para executar determinada tarefa, que são mostradas pela Figura 6.

Figura 4 – Criação de tarefas no UEMan.

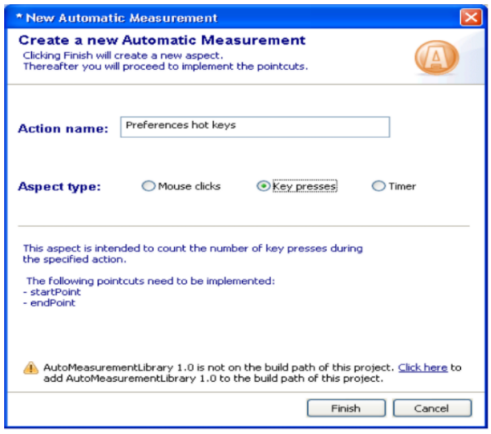

Figura 5 – Alterando o fator de avaliação da tarefa pelo UEMan.

Foram aplicados testes de usabilidade em um navegador web, avaliando o tempo de execução das tarefas pelos usuários e aplicando um questionário pós teste para cada um. O autor concluiu que o plugin UEMan apresentou resultados satisfatórios, pois conseguiu mapear corretamente, a partir dos experimentos e, principalmente, do questionário, que a ferramenta se mostra útil para avaliar a usabilidade do software enquanto está em desenvolvimento. Entretanto, foi detectado que o software precisa estar se encaminhando para sua fase final de produção. Caso contrário, torna-se mais complicado criar tarefas de teste, devido ao fato de que podem existir funcionalidades ainda com muitos defeitos.

4.1.3 Estudos teóricos sobre Eye Tracking

(SHARIF, 2011) disserta em seu estudo sobre uma avaliação de software guiada por Eye Tracking, em que o uso de suas métricas podem contribuir de diversas formas para a avaliação desenvolvimento de uma aplicação, e não só para experimentos comuns de usabilidade. São apresentadas duas formas de visualização de dados de software bastante usadas que podem servir como objetos de estudo de Eye tracking, que são os diagramas UML e os próprios códigos de implementação. Neste estudo, Sharif, B. et al usa estes exemplos de visualização de dados para dizer que pode haver um padrão na sequência do olhar do usuário enquanto escreve a implementação e olha ao diagrama UML. Por exemplo, caso o caminho dos olhares de diversos programadores sejam parecidos para a implementação de um método de implementação com base em um diagrama UML, pode-se haver um padrão de ligação entre o método com um elemento do diagrama UML.

(SHARAFI, 2015) apresenta diversas métricas que utilizam os dados do olhar do usuário gerados pelo eye tracker, e que são representadas representadas por fórmulas matemáticas. Todas essas fórmulas foram elaboradas por diferentes autores, e reunidas neste artigo, que tem como objetivo resumir todas essas métricas como forma de padronização, já que as fórmulas foram usadas pelos autores em diferentes contextos e com diferentes nomes, mesmo que usadas muitas vezes com o mesmo objetivo. A terceira categoria definida como “Estudos teóricos sobre Eye Tracking contém artigos referentes” ao uso de dispositivo eye tracker na avaliação de softwares (SHARIF, 2011), e métricas de análise de dados gerados por eye trackers (SHARAFI, 2015).

4.1.4 Estudos de uso de Eye Tracking para melhorar algum aspecto de usabilidade

Um eye tracker salva todo o caminho no olhar do usuário, sendo este composto por saccades e fixations. As fixations são os pontos onde o usuário fixou os olhos por um tempo maior que 100 milissegundos, e as saccades representam o caminho entre esses pontos de fixação. Porém, muitas vezes ocorrem algumas variações nas posições das fixations, por diversos motivos de erro de captação do eye tracker. Nesse sentido, (PALMER, 2016) disserta sobre a correção dessas variações, e apresenta um algoritmo para isso. O algoritmo se baseia em corrigir as posições das fixations com agrupando-as nas palavras mais próximas na implementação, e pequenas variações momentâneas que não variam significativamente a posição das fixações são desconsideradas. No estudo o autor ainda comparou os resultados do algoritmo com correções feitas manualmente por programadores, chegando a um valor de 89 por cento de semelhança.

(RODEGHERO, 2014) disserta sobre a necessidade de ferramentas de resumo de métodos, pois muitas vezes um código é lido por diversos programadores, e toma-se muito tempo deles para pegar a ideia geral do método. Dessa forma, o autor conduz dois experimentos:

1. Pede aos participantes para resumirem métodos dados, e checa-se se eles reparam mais nas assinaturas dos métodos, nas instruções de controle (if, for e while), ou nas invocações dos métodos em outras classes

2. Pede aos participantes escolherem palavras-chave dos métodos

Abaixo está a ordem, do maior para o menor, da frequência de vezes em que os usuários olharam para cada um dos três setores definidos no experimento 1.

1. Assinaturas dos métodos

2. Invocação dos métodos

3. Instruções de controle

No segundo experimento, as palavras chave escolhidas são classificadas de acordo com pesos, definidos de acordo com a ordem dos três setores acima. Ou seja, uma palavra-chave que aparece na assinatura do método tem peso maior que uma que aparece na invocação do método. Com isso, obteve-se uma evidência de um caminho para a execução do resumo de um método. O autor trabalha o artigo todo em torno destes experimentos, explicando cada passo e exibindo valores referentes aos valores obtidos pelo eye tracker. Ele propõe que os resultados obtidos serão de muito valor para trabalhos futuros que visem elaborar ferramentas para automatização da realização de resumo dos métodos.

4.2 Teste de Usabilidade

Abaixo são mostrados alguns gráficos interessantes que representam diferentes análises de dados gerados sobre os usuários através do Eye Tracker e do software Mora. Alguns gráficos foram gerados dentro do software Tobii Studio, que usou os dados das posições dos olhares do usuário, capiturados pelo Eye Tracker, considerando duas Áreas de Interesse para cada usuário: Implementation and Tests (Implementação e Testes). Mais à frente neste capítulo são mostradas linhas do tempo, uma para cada participante, que representam as transições entre Áreas de Interesse também, mas dessa vez as Áreas de Interesse consideradas foram: Implementation, Tests, Specification (Especificação) e Web Search (Pesquisa na Web). São mostrados também gráficos com análises quantitativas das linhas do tempo, agrupadas em TDD e TLD. Para saber exatamente quando os participantes trocaram o olhar entre as quatro áreas, as gravações feitas através do software Morae foram assistidas.

No restante deste documento, serão chamadas como Conjunto de Áreas de Interesse 1 (ADI-1) o primeiro conjunto de áreas de interesse, que contém as áreas Implementation e Tests. E serão chamadas como Conjunto de Áreas de Interesse 2 (ADI-2) o segundo conjunto mencionado, que contém as áreas Implementation, Tests, Specification e Web Search.

4.2.1 Gráficos gerados com dados do Eye Tracker - ADI-1

4.2.1.1 Fixation Count

O gráfico de Fixation Count representa a quantidade de fixações que cada usuário realizou em cada Área de Interesse. Uma fixação representa que o participante fixou o olhar por 100 milissegundos ou mais. Visualmente, pode-se notar que nos participantes que utilizaram TLD existe uma diferença maior de quantidade de fixações entre implementação e testes, tendo bem mais fixações na área Implementation do que na área Tests. Para os participantes que utilizaram TDD, essa diferença foi menor, com exceção ao participante 4. Esta leitura significa que os participantes que usaram TDD foram mais efetivos, pois gastaram menos tempo na implementação.

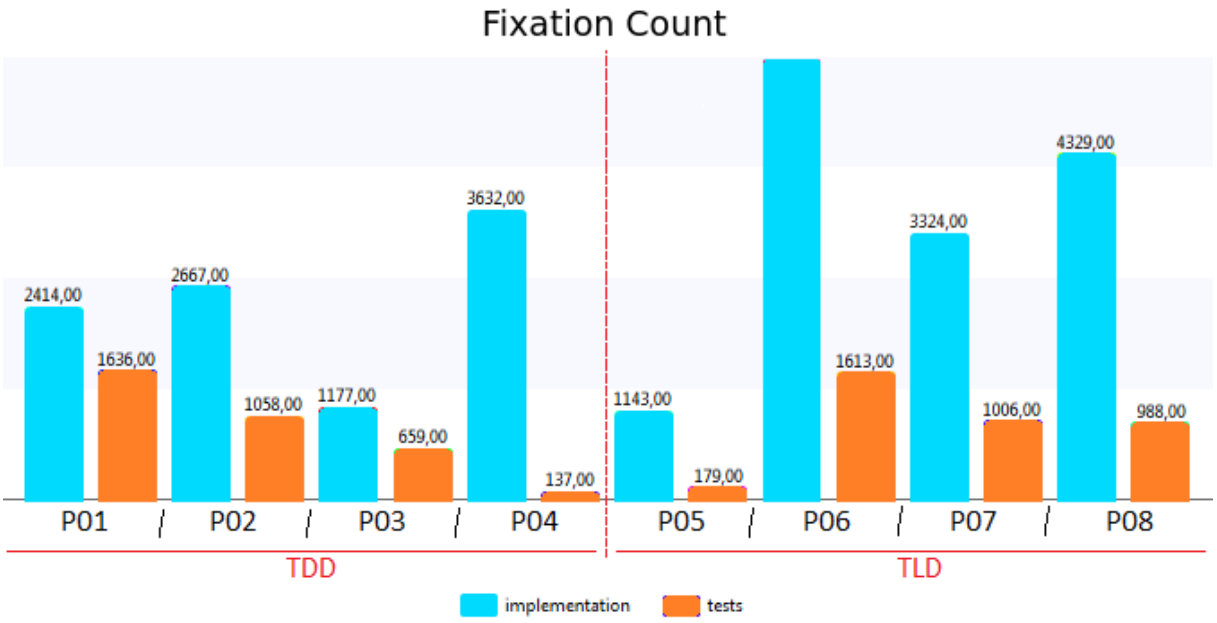

Figura 6 – Gráfico de quantidade de fixações para cada participante.

4.2.1.2 Total Fixation Duration

O gráfico de duração total das fixações (Total Fixation Duration) tem um resultado parecido visualmente com o de quantidade de fixações, mostrado acima. Isso se deve pois ambas as métricas analisam "o quanto cada usuário olhou para cada área". Sendo assim, pode-se fazer a mesma análise que a do gráfico anterior.

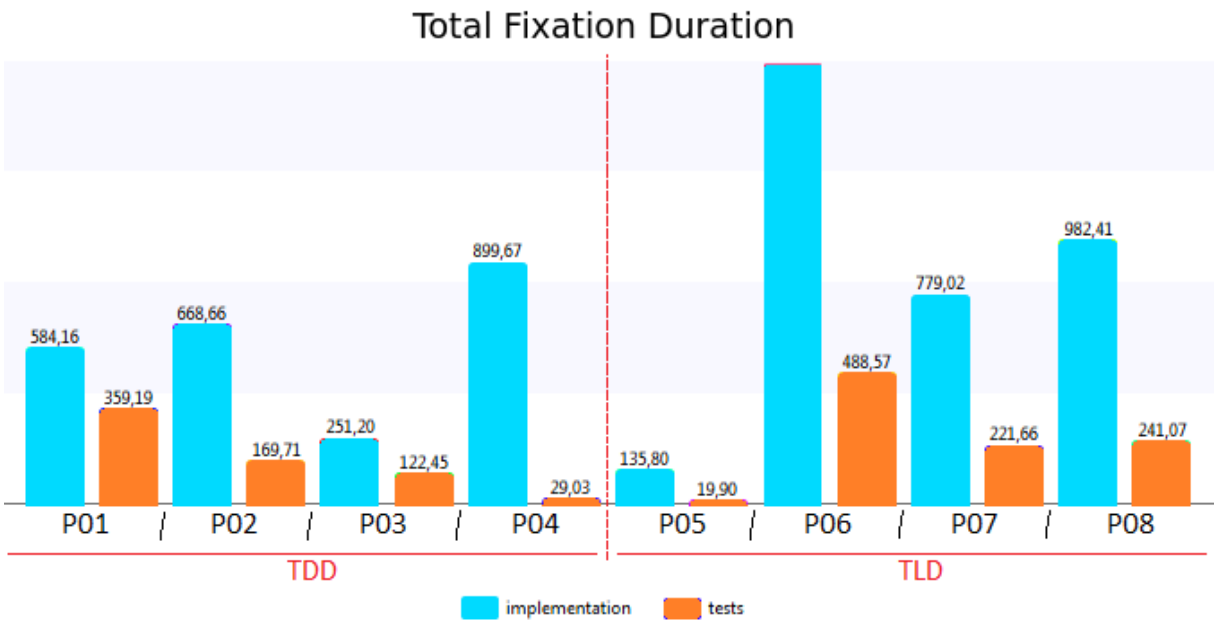

Figura 7 – Gráfico de duração total das fixações para cada participante.

4.2.1.3 Total Visit Duration

Uma visita é considerada desde o momento que o usuário "entra"em uma área de interesse, até o momento em que ele "sai". Por exemplo, se um usuário trocou o olhar para uma área de interesse, e contabilizou-se 20 fixações naquela área e depois saiu, foi contabilizada uma visita - ela é indiferente de quantas fixações ocorreram naquela Área de Interesse. A duração total das visitas (Total Visit Duration) de cada área soma os tempos de duração de cada visita, dando também um resultado de "quanto o usuário olhou"para cada Área de Interesse. Logo, o mesmo tipo de análise pode ser feita como nos gráficos acima.

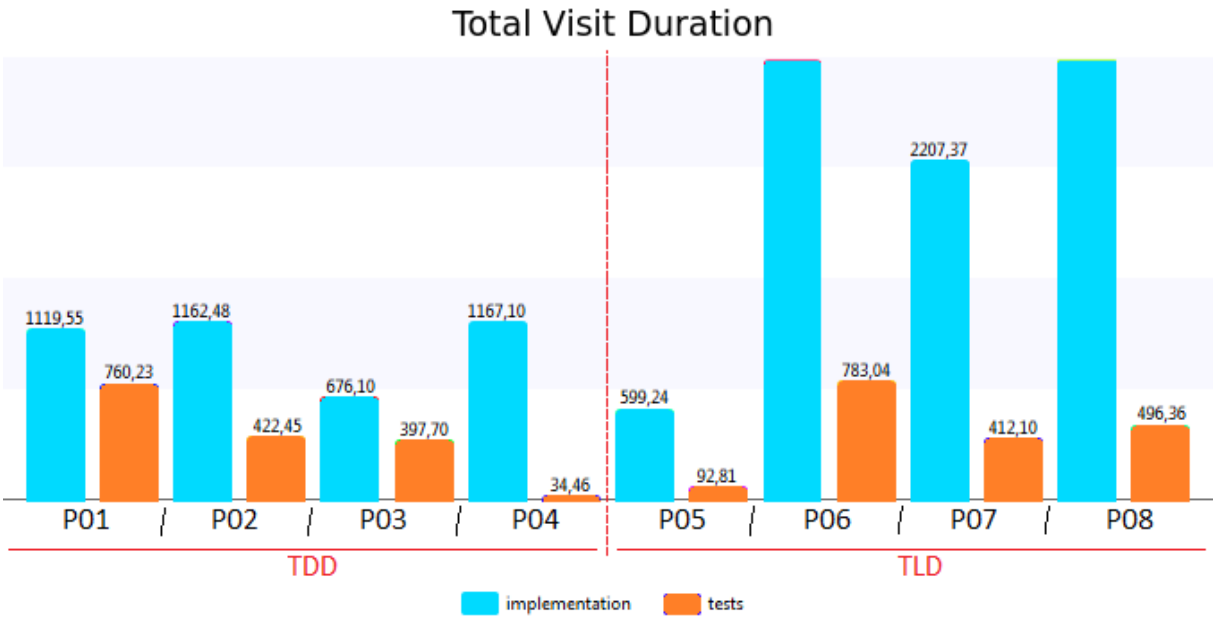

Figura 8 – Gráfico de duração total das visitas para cada participante.

4.2.2 Linhas do tempo de transições de cada participante - ADI-2

Os gráficos das linhas do tempo de cada participante mostram onde olhou cada participante em cada momento. Com estes dados, pode-se notar nos gráficos abaixo que os participantes que utilizaram TDD variaram bastante o olhar entre as regiões Implementation e Tests. Já nos que usaram TLD, houve transições mais desuniformes entre as áreas, variando bastante entre todas as áreas menos a área Tests.

4.2.2.1 Primeiro Participante - P01

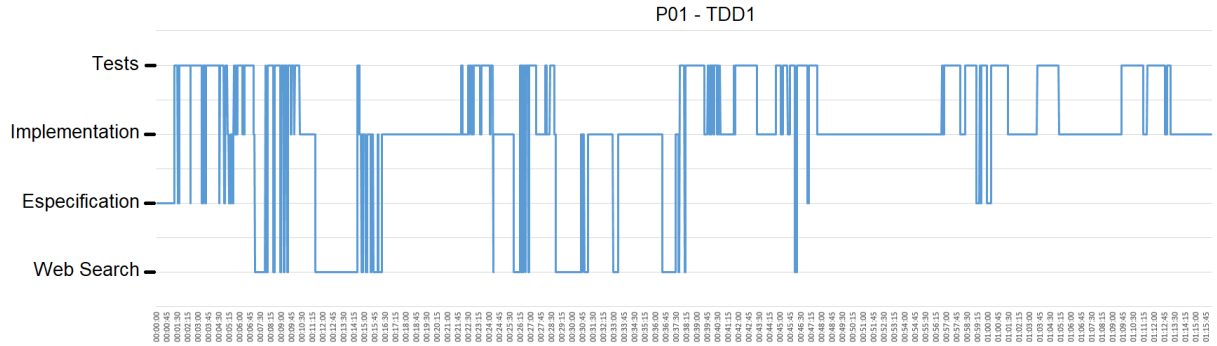

Figura 9 – Linha do tempo das transições do primeiro participante - P01.

4.2.2.2 Segundo Participante - P02

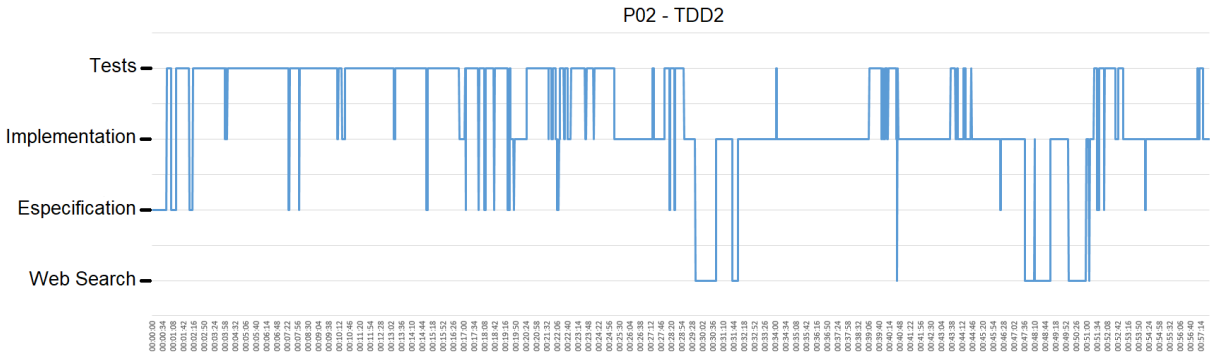

Figura 10 – Linha do tempo das transições do segundo participante - P02.

4.2.2.3 Terceiro Participante - P03

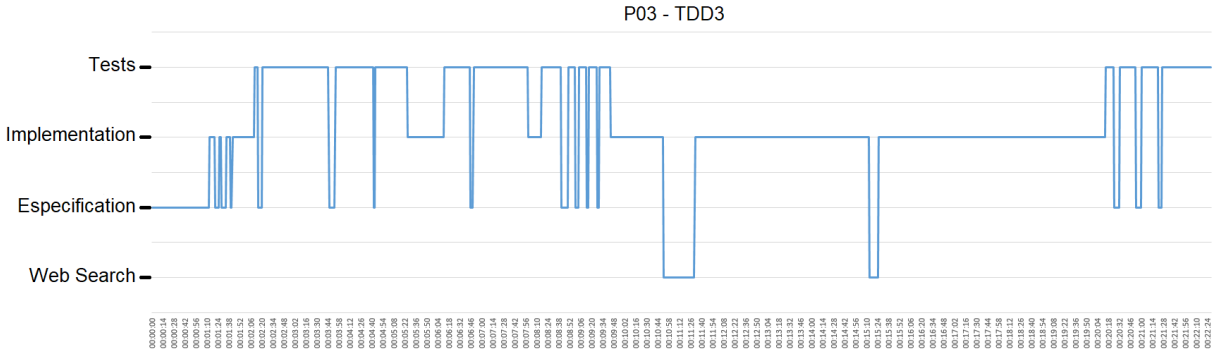

Figura 11 – Linha do tempo das transições do terceiro participante - P03.

4.2.2.4 Quarto Participante - P04

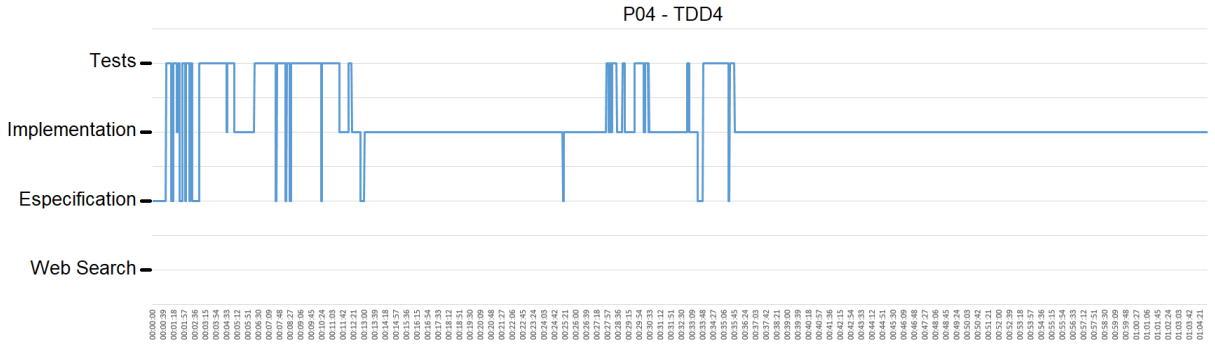

Figura 12 – Linha do tempo das transições do quarto participante - P04.

4.2.2.5 Quinto Participante - P05

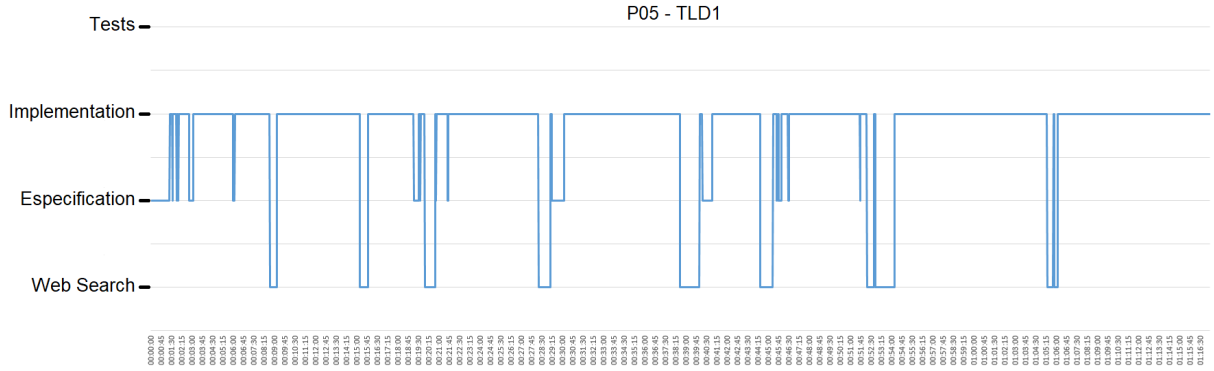

Figura 13 – Linha do tempo das transições do quinto participante - P05.

4.2.2.6 Sexto Participante - P06

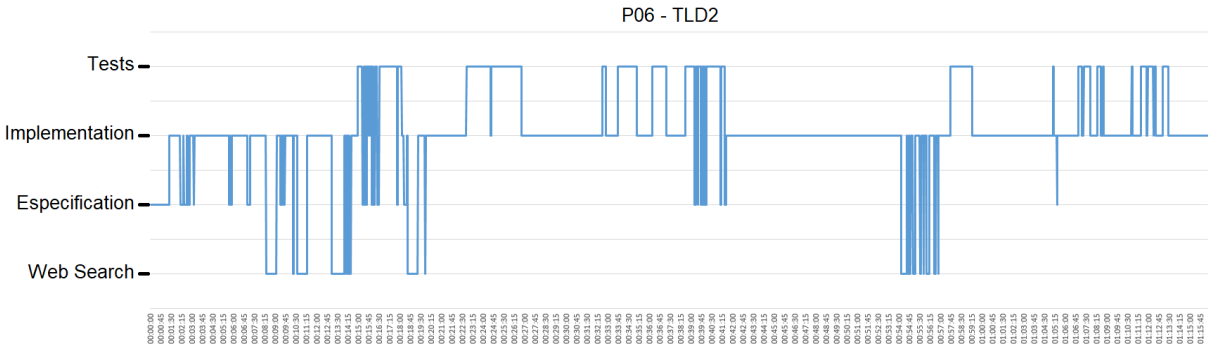

Figura 14 – Linha do tempo das transições do sexto participante - P06.

4.2.2.7 Sétimo Participante - P07

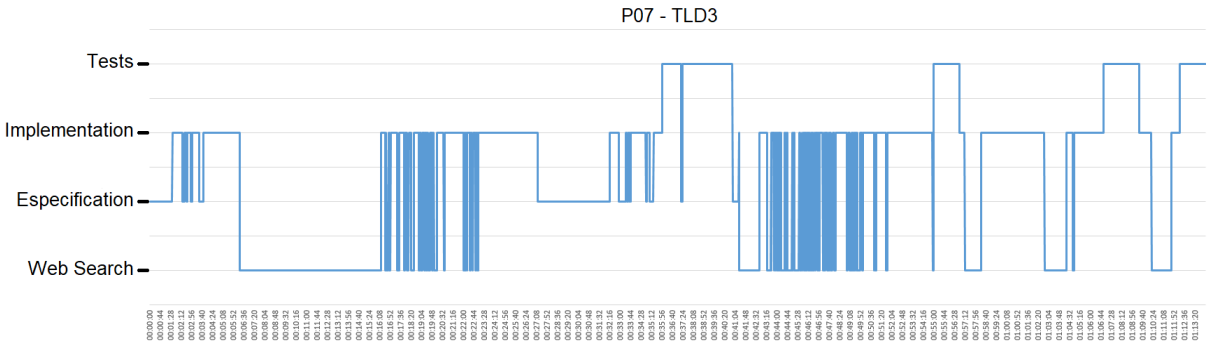

Figura 15 – Linha do tempo das transições do sétimo participante - P07.

4.2.2.8 Oitavo Participante - P08

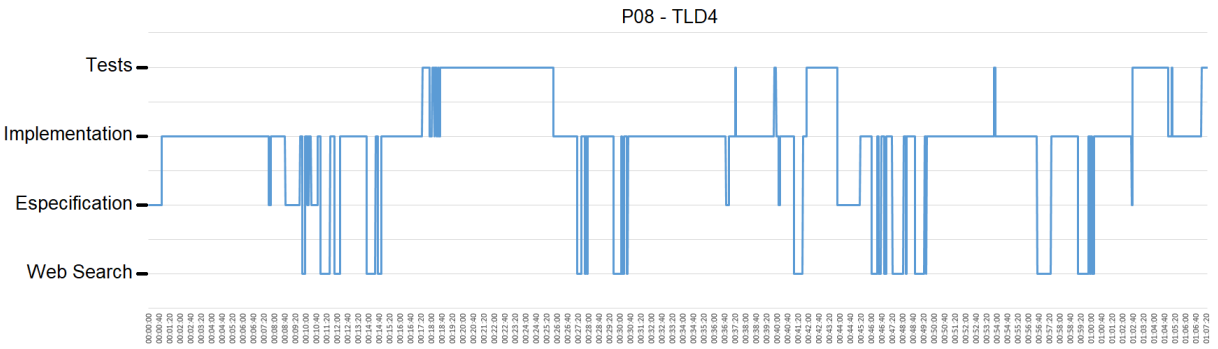

Figura 16 – Linha do tempo das transições do oitavo participante - P08.

4.2.3 Média do número de transições para todos participantes - ADI-2

A Figura 17 mostra a média do número de transições nos participantes que usaram TDD.

Transitions Mean.png

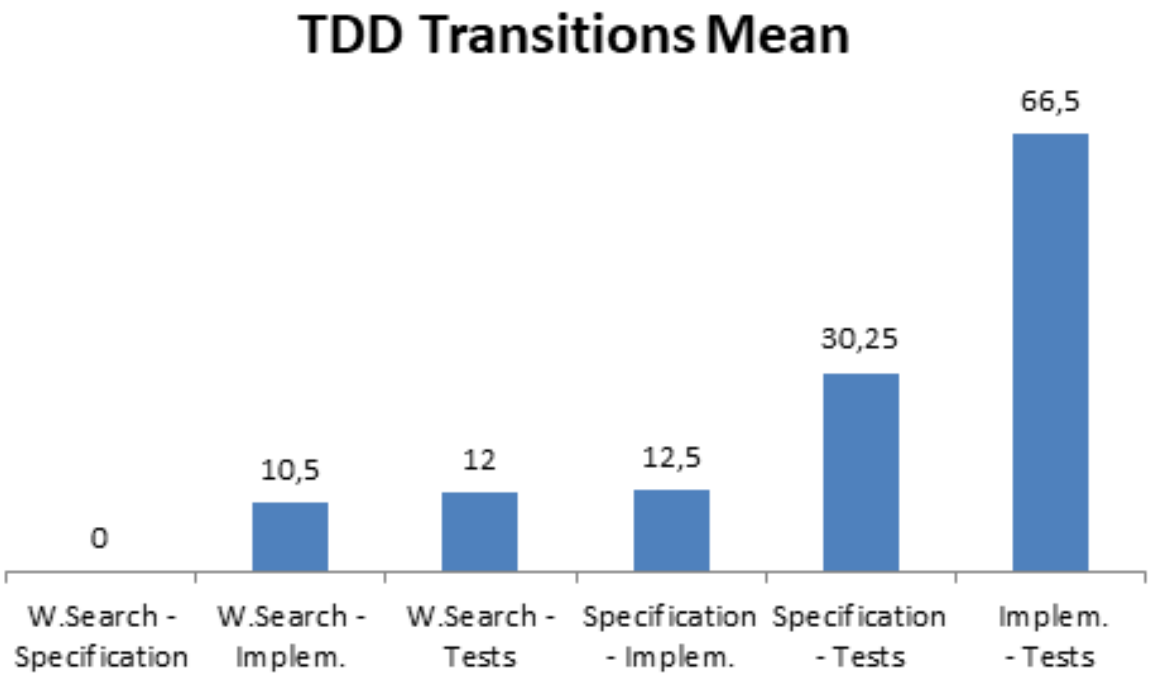

Figura 17 – Média das transições TDD

A Figura 18 mostra a média do número de transições nos participantes que usaram TLD.

Transitions Mean.png

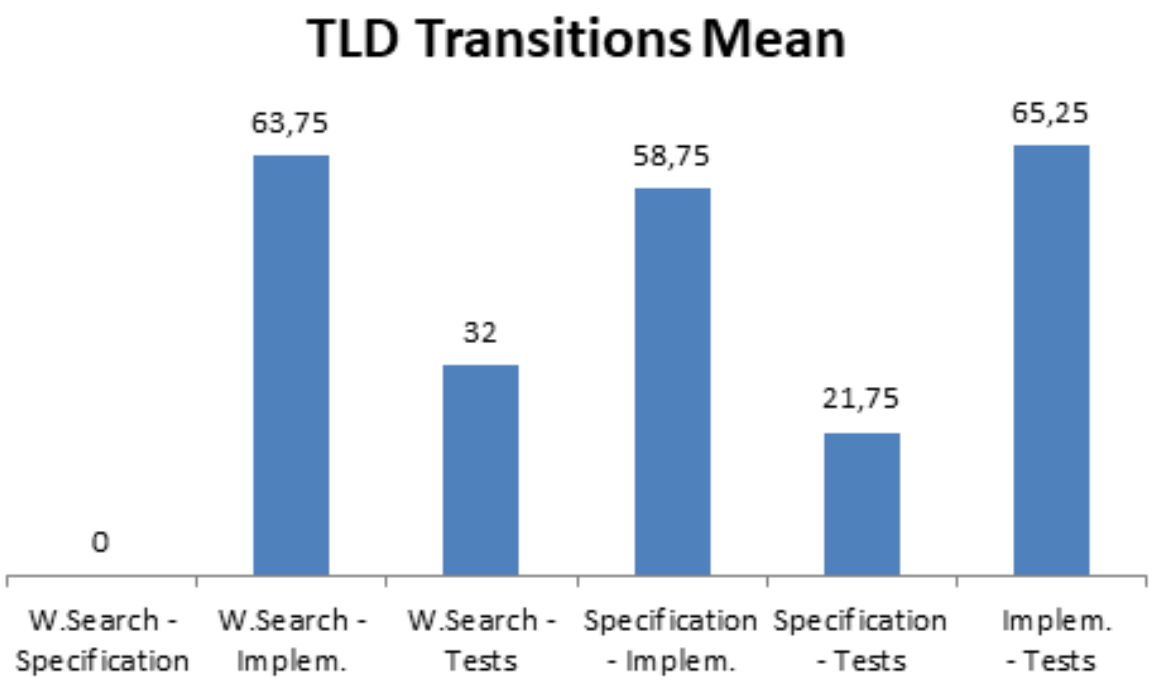

Figura 18 – Média das transições TLD

A Figura 19 mostra a comparação das médias de transições de TDD e TLD. Pode-se notar que no TDD, os participantes trocaram mais de olhar entre espeficicação e testes, e entre implementação e testes. Já no TLD, houve um foco grande em todas as áreas, com destaque na Pesquisa na Web-Implementação, Especificação-Implementação, e Implementação-Tests. Analisando estes resultados, pode-se notar que os participantes do TDD ficaram mais focados nos testes, e os do TLD variaram bastante entre as demais áreas.

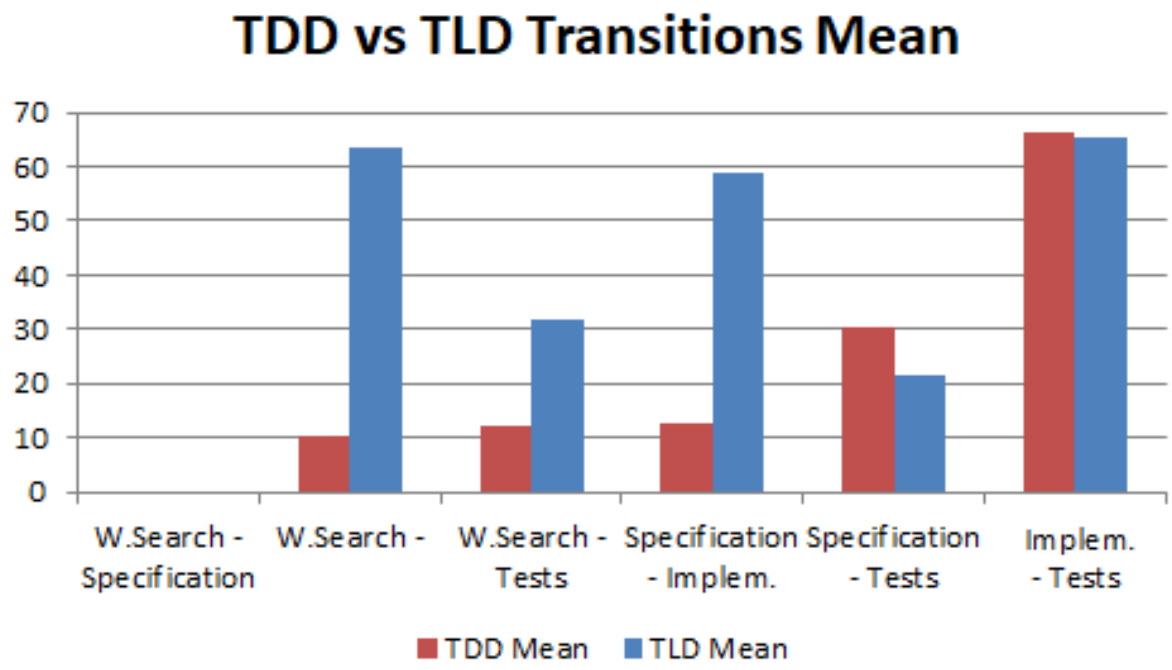

Figura 19 – Média das transições TDD vs TLD

A Figura 20 mostra o total de transições para todas as seções. Pode-se notar que houve mais transições para os participantes de TLD. Considerando que um número grande de transições é um resultado negativo, pois siginifica perda de foco do participante, os participantes do TDD se mantiveram mais focados.

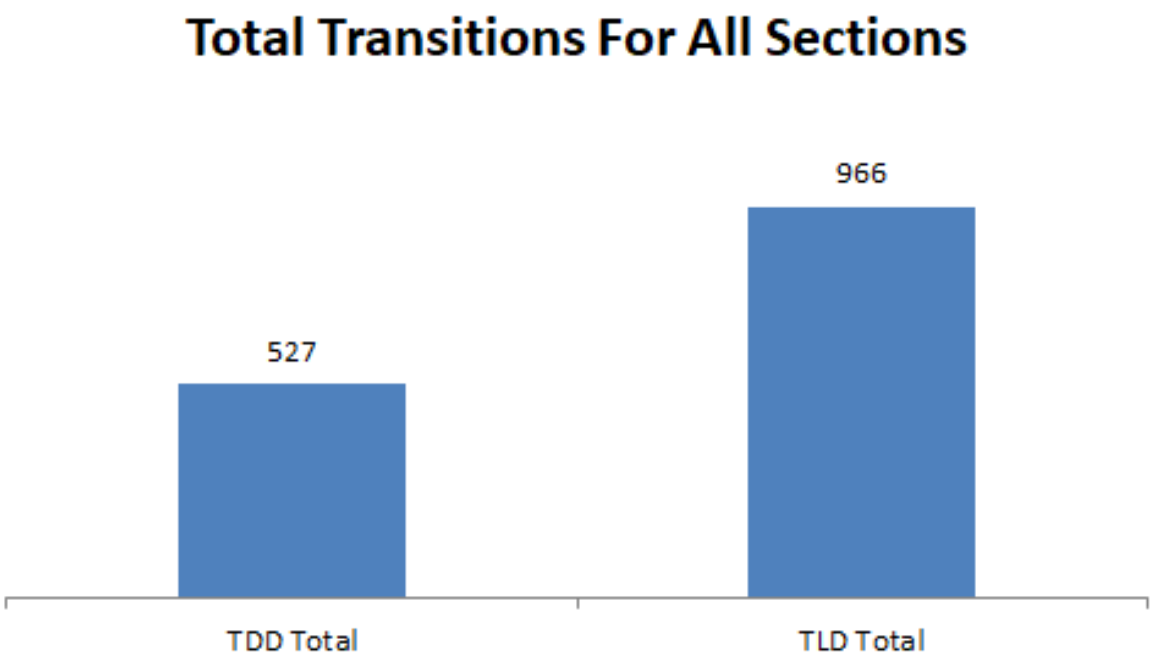

Figura 20 – Total de Transições para Todas as Seções

4.2.4 Tempo de duração de cada caso de teste para cada participante

As Figuras 21, 22, 23 e 24 mostram os tempos de duração de cada caso de teste para cada participante, com os participantes TDD na esquerda e os TLD na direita. As barras em azul mostram o tempo gasto pelo participante para finalizar com sucesso o caso de teste. E as barras em branco contornadas em laranja mostram o tempo gasto pelo participante para finalizar sem sucesso o caso de teste. Espaços sem parras (com tempo de duração zero) representam que o participante nem começou o caso de teste.

A Figura 21 mostra que os participantes TDD foram bem mais efetivos, conseguindo todos finalizar o caso de teste 1. Três participantes TLD iniciaram o caso de teste, mas somente dois finalizaram com sucesso, sendo seus tempos gastos maiores que qualquer participante TDD.

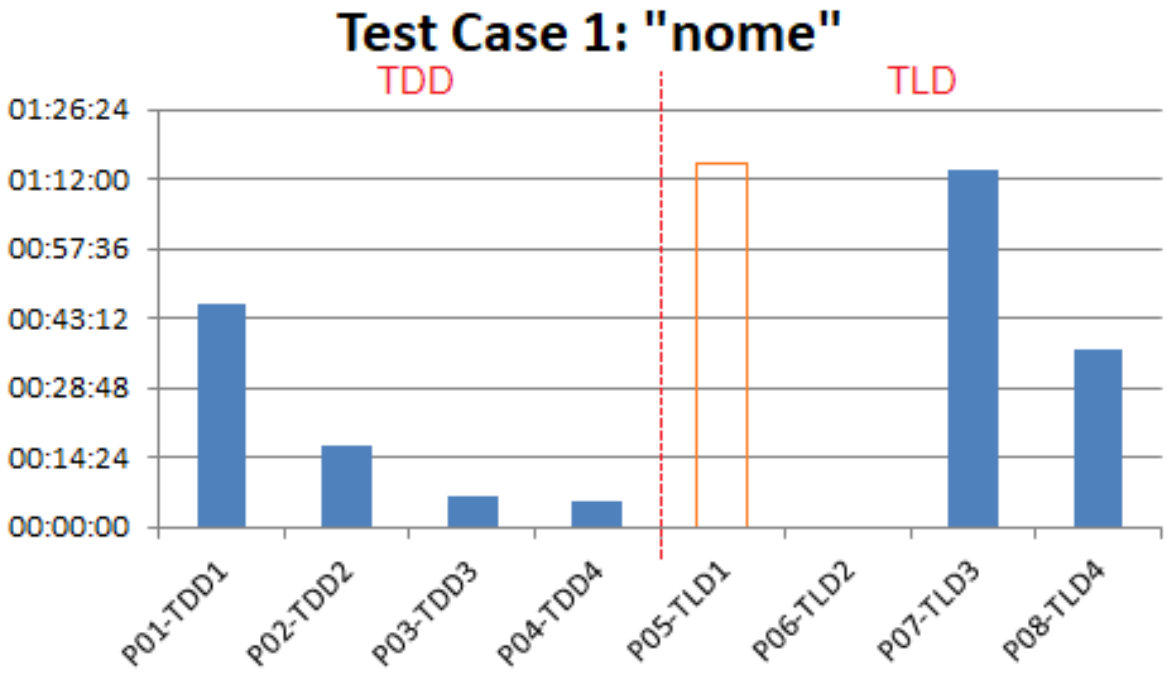

Figura 21 – Duration Time of Test Case 1 for each participant

O caso de teste 2 foi iniciado por poucos participantes, possivelmente por ser bem parecido com o caso de teste 1. Um participante de cada grupo finalizou com sucesso, e o participante TDD teve melhor tempo.

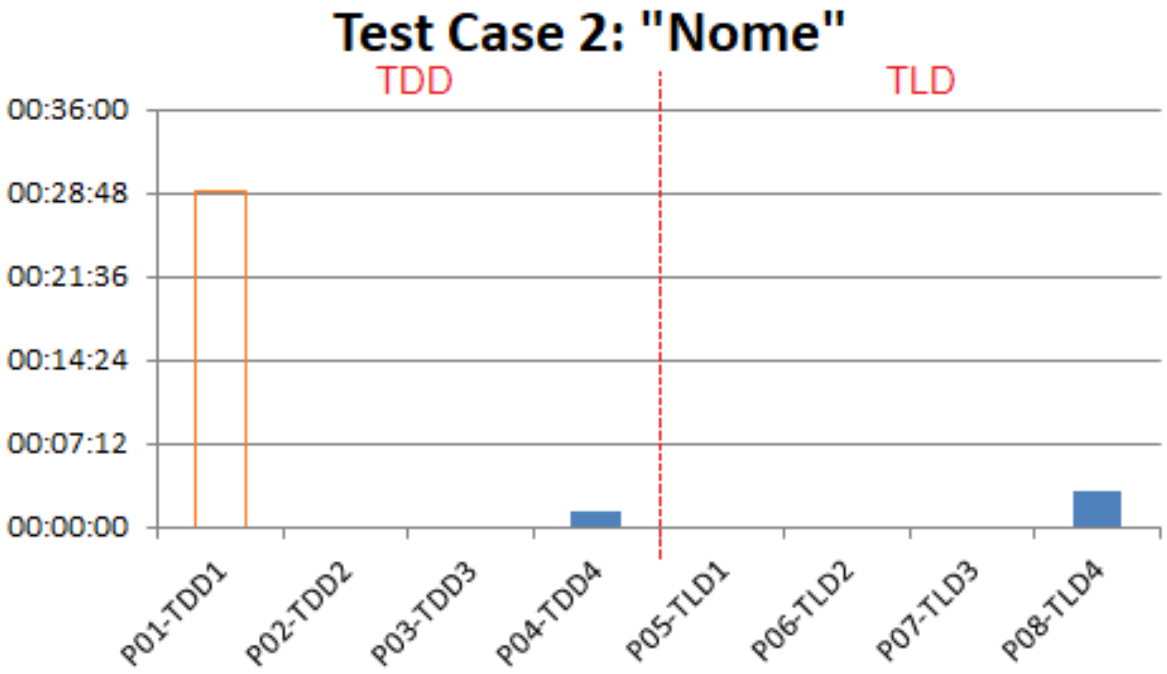

Figura 22 – Duration Time of Test Case 2 for each participant

O caso de teste 3 foi iniciado e finalizado com sucesso com três participantes TDD, já no TLD o mesmo ocorreu com dois participantes, mostrando novamente um resultado melhor para o TDD.

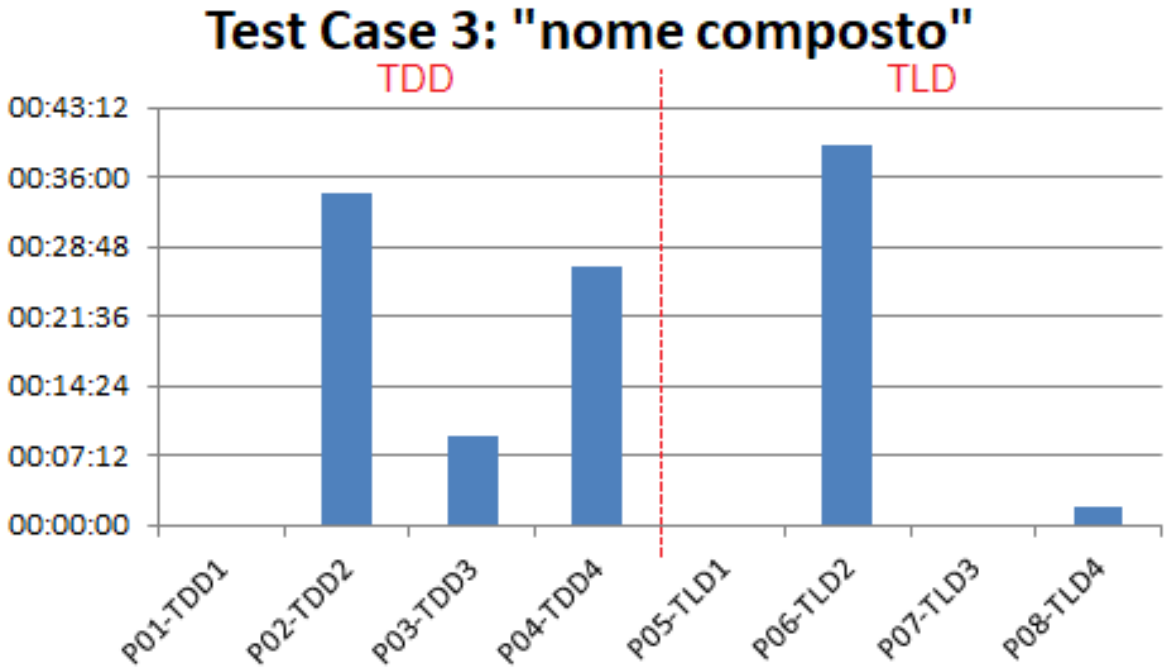

Figura 23 – Duration Time of Test Case 3 for each participant

Três participantes TDD fizeram o caso de teste 4, e os três conseguiram terminá-lo com sucesso. Um participante TLD finalizou com sucesso e outro sem sucesso. Os participantes TDD foram melhor neste caso de teste também.

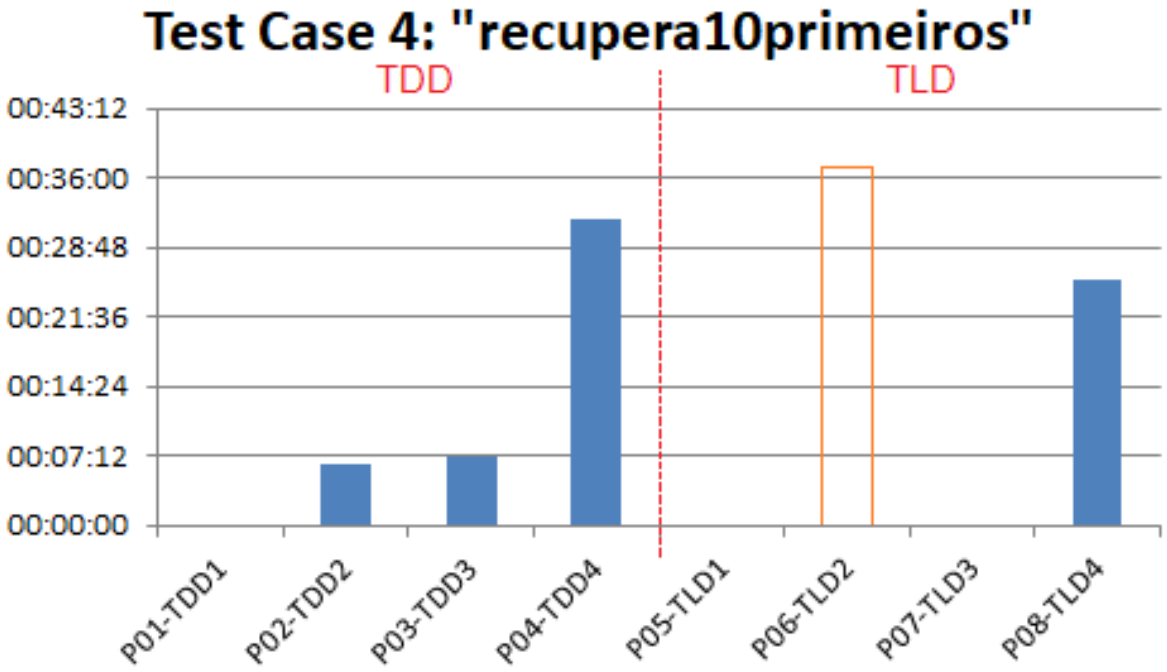

Figura 24 – Duration Time of Test Case 4 for each participant

5 Conclusões

Foram selecionados 16 artigos da biblioteca Scholar do Google para a realização deste TCC, e envolveram os principais assuntos definidos pelo planejamento: Usabilidade de APIs e Eye Tracking e Usabilidade. Após a realização da Revisão destes artigos, foi possível obter uma boa base de conhecimento esperada por uma boa Revisão. Os conhecimentos adquiridos foram importantes para a realização do teste de usabilidade, que requiriu bastante planejamento para a elaboração da tarefa, procedimentos e instrumentação, e também para analisar os resultados obtidos.

No geral, o participantes que utilizaram TDD se sairam melhor, pois pelos resultados eles se mantiveram mais focados, e também conseguiram realizar com sucesso mais casos de teste e em menos tempo.

Referências

BOJKO, A. *Eye Tracking the User Experience: A Practical Guide to Research*. 1st. ed. New York, NY, USA: Rosenfeld Media, 2013. 6–8 p. Citado na página 20.

CLARKE, S. Measuring API Usability. p. 1–5, 2004. Disponível em: <http://www-.researchgate.net/publication/200086234_Measuring_API_usability>. Citado 4 vezes nas páginas 18, 28, 29 e 31.

CLARKE, S. Describing and Measuring API Usability with the Cognitive Dimensions. p. 1–4, 2006. Disponível em: <http://www.researchgate.net/publication-/200086113_Describing_and_Measuring_API_Usability_with_the_Cognitive_Dimensions>. Citado 4 vezes nas páginas 27, 28, 30 e 31.

DUBINSKY, Y. Eclipse Plug-in to Manage User Centered Design. *Proceedings of the International Workshop on: Interplay between Usability Evaluation and Software Development*, p. 41–46, 2008. Citado na página 29.

FAROOQ, U. API Peer Reviews: A Method for Evaluating Usability of Application Programming Interfaces. *Proceedings of the 2010 ACM conference on Computer supported cooperative work*, p. 207–210, 2010. Citado 3 vezes nas páginas 27, 28 e 31.

GUERRA, E. Padrões de Projeto para Frameworks e Componentes Baseados em Metadados. *Preliminary Proceedings on the 7th Latin American Conference on Pattern Languages of Programming*, p. 3–24, 2011. Citado 2 vezes nas páginas 28 e 29.

HUMAYOUN, S. UEMan A Tool to Manage User Evaluation in Development Environments. *IEEE 31st International Conference on Software Engineering, 2009. ICSE 2009*, p. 65–68, 2009. Citado 3 vezes nas páginas 28, 29 e 32.

KITCHENHAM, B. Procedures for performing systematic reviews. *Keele, UK, Keele University*, v. 33, n. 2004, p. 1–26, 2004. Citado na página 23.

MUNIR, H. An Experimental Evaluation of Test Driven Development vs. Test-Last Development with Industry Professionals. *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, 2014. Citado 3 vezes nas páginas 19, 27 e 29.

PALMER, C. Towards Automating Fixation Correction for Source Code. *Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research Applications*, p. 65–68, 2016. Citado 3 vezes nas páginas 28, 29 e 34.

PICCIONI, M. An Empirical Study of API Usability. *ACM / IEEE International Symposium on Empirical Software Engineering and Measurement*, p. 5–14, 2013. Citado 4 vezes nas páginas 18, 27, 28 e 31.

RODEGHERO, P. Improving Automated Source Code Summarization via an eye tracker study. *Proceedings of the 36th International Conference on Software Engineering*, p. 390–401, 2014. Citado 3 vezes nas páginas 28, 29 e 34.

SCHOLAR. Google Scholar. 2017. Disponível em: <http://scholar.google.com.br/>. Citado na página 23.

SHARAFI, Z. Eye-tracking Metrics in Software Engineering. *Software Engineering Conference (APSEC), 2015 Asia-Pacific*, p. 1–8, 2015. Citado 3 vezes nas páginas 27, 29 e 34.

SHARIF, B. On the Use of Eye Tracking in Software Traceability. *Proceedings of the 6th International Workshop on Traceability in Emerging Forms of Software Engineering*, p. 67–70, 2011. Citado 3 vezes nas páginas 28, 29 e 34.

SHARIF, B. An Empirical Study Assessing the Effect of SeeIT 3D on Comprehension. *First IEEE Working Conference on Software Visualization (VISSOFT)*, 2013. Citado 3 vezes nas páginas 27, 29 e 31.

SOUZA, C. Automatic Evaluation of API Usability using Complexity Metrics and Visualizations. *31st International Conference on Software Engineering - Companion Volume, 2009. ICSE-Companion 2009*, p. 299–302, 2009. Citado 2 vezes nas páginas 27 e 28.

STOREY, M.-A. Improving the Usability of Eclipse for Novice Programmers. *Proceedings of the 2003 OOPSLA workshop on eclipse technology eXchange*, p. 35–39, 2003. Citado 3 vezes nas páginas 28, 29 e 32.

STYLOS, J. A Case Study of API Redesign for Improved Usability. *IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, p. 189–192, 2008. Citado 3 vezes nas páginas 27, 29 e 32.

USABILITY. User Experience Basics. 2017. Disponível em: <http://www.usability.gov/what-and-why/user-experience.html>. Citado na página 17.